Multilinear algebra applied to restoration and image recognition

Asmaa Khouia

Thèse de Doctorat

Mention: Mathématiques et leurs interactions

présentée à ***l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)***
de l'Université du Littoral Côte d'Opale
et
au Centre D'Études Doctorales Sciences de l'ingénieur
de l'Université l'Université Cadi Ayyad

par

Asmaa Khouia

pour obtenir le grade de Docteur de l'Université du Littoral Côte d'Opale

ALGÈBRE MULTILINÉAIRE APPLIQUÉE À LA RESTAURATION ET À LA RECONNAISSANCE D'IMAGES

Soutenue le 26 Décembre 2022, après avis des rapporteurs,
devant le jury d'examen:

Nour Eddine ALAA,	**Professeur, Université Cadi Ayyad, Marrakech**	**Président**
Abdellah BNOUHACHEM,	**Professeur, Université Ibn Zohr, Agadir**	**Rapporteur**
Hassan SAFOUHI ,	**Professeur, University of Alberta, Canada**	**Rapporteur**
Otmane *SOUHAR*,	**PH, Université Chouaib Doukkali, El Jadida**	**Rapporteur**
Hassane SADOK,	**Professeur, Université du Littoral Côte d'Opale**	**Directeur**
Abdeslem Hafid BENTBIB,	***Professeur, Université Cadi Ayyad, Marrakech***	***Directeur***

Thèse de Doctorat

Mention: Mathématiques et leurs interactions

présentée à ***l'Ecole Doctorale en Sciences Technologie et Santé (ED 585)***
de l'Université du Littoral Côte d'Opale
et
au Centre D'Etudes Doctorales Sciences de l'ingénieur
de l'Université l'Université Cadi Ayyad

par

Asmaa Khouia

pour obtenir le grade de Docteur de l'Université du Littoral Côte d'Opale

MULTILINEAR ALGEBRA APPLIED TO RESTORATION AND IMAGE RECOGNITION

Soutenue le 26 Décembre 20222, après avis des rapporteurs,
devant le jury d'examen:

Nour Eddine ALAA,	**Professeur, Université Cadi Ayyad, Marrakech**	**Président**
Abdellah BNOUHACHEM,	**Professeur, Université Ibn Zohr, Agadir**	**Rapporteur**
Hassan SAFOUHI ,	**Professeur, University of Alberta, Canada**	**Rapporteur**
Otmane *SOUHAR*,	**PH, Université Chouaib Doukkali, El Jadida**	**Rapporteur**
Hassane SADOK,	**Professeur, Université du Littoral Côte d'Opale**	**Directeur**
Abdeslem Hafid BENTBIB,	***Professeur, Université Cadi Ayyad, Marrakech***	***Directeur***

Cette thèse à été préparée à:

Laboratoire de Mathématiques Pures et Appliquées

Centre Universitaire de la Mi-Voix

Maison de la Recherche Blaise Pascal

50, Rue Ferdinand Buisson

CS 80699

Calais Cedex, France.

Fax: (33) 03 21 46 55 86.

Site: www.lmpa.univ-littoral.fr

Laboratoire de Mathématiques Appliquées et Informatique

Faculté des Sciences et Techniques Marrakech

B.P 549, Av.Abdelkarim Elkhattabi Guéliz

Marrakech, Maroc

Fax: (+212) 524 43 31 70.

Site: www.fstg-marrakech.ac.ma

Résumé

Les tenseurs sont des tableaux multidimensionnels. Ils sont une généralisation des matrices et des vecteurs. Ils fournissent un moyen naturel de représenter les données dans différents domaines, ce qui fait le domaine tensoriel un cadre idéal pour formuler et résoudre de nombreux problèmes dans différents domaines. L'une des applications les plus importantes des tenseurs se trouve dans le domaine du traitement des images, comme la restauration d'images et la reconnaissance des visages, où les images couleur (images RGB) sont présentées comme des tenseurs d'ordre 3 et une vidéo composée d'images couleur est un tenseur d'ordre 4. Mais lorsque nous travaillons dans des espaces de dimension supérieure pour résoudre des problèmes d'ordre supérieur, un ensemble de défis se pose, comme un problème connu sous le nom de "the curse of dimensionality". En effet, lorsque la dimension augmente, les problèmes d'ordre supérieur deviennent plus difficiles (besoin en calcul et en mémoire) car la taille des données d'un tenseur augmente exponentiellement avec l'augmentation de la dimensionnalité du tenseur. Par conséquent, les calculs tensoriels deviennent très coûteux. Dans cette thèse, nous nous sommes concentrés sur la résolution de certains problèmes tensoriels. Les algorithmes proposés sont obtenus en combinant des méthodes itératives telles que la méthode LSQR et des décompositions d'ordre supérieur pour surmonter le problème de la dimensionnalité. Les approches que nous proposons sont appliquées à la restauration d'images et de vidéos.

De plus, cette thèse étudie les méthodes de reconnaissance des visages basées sur le format tensoriel, où des outils d'algèbre multi-linéaire tels que la $HOSVD$ (Higher- Order Singular Value Decomposition) ont été utilisés. Nous proposons un nouvel algorithme qui peut être appliqué à une base de données d'images représentées par un tenseur

d’ordre 3 ou 4.

Mots clés: HOSVD, decomposition CP, LSQR, T-product, produit cosine , traitement des images et des vidéos, reconnaissance faciale.

Abstract

Tensors are multi-dimensional arrays. They are a generalization of matrices and vectors. They provide a natural way to represent the data in different fields, which make tensor field a great framework for formulating and solving many problems in different areas. One of the most important applications of tensors is in the field of image processing such as image restoration and face recognition, where color images (RGB images) are presented as third-order tensors and a video composed of color images is a fourth-order tensor. But when we work in higher dimension spaces to solve higher-order problems, a set of challenges arise such as a problem known as "the curse of dimensionality ". In fact, when the dimension increase the higher-order problems become harder (computation and memory requirement) since the data size of a tensor increases exponentially with the increase of the dimensionality of the tensor. As consequence tensor computations become much expensive. In this thesis, we have focused on solving some tensor problems. The suggested algorithms are obtained by combining iterative methods such as LSQR method and higher-order decompositions to overcome the problem of dimensionality. Our suggested approaches are applied to the restoration of images and videos.

In addition, this thesis studies the face recognition methods based on the tensor format, where multi-linear algebra tools such as the $HOSVD$ (Higher- Order Singular Value Decomposition) have been used. We suggest a new algorithm that can be applied to a database of images represented by a third or fourth-order tensor.

Keywords: HOSVD, CP decomposition, color image restoration, video restoration, LSQR, face recognition, t-product, cosine product.

Acknowledgements

First of all, I would like to thank Mr. **Hassane SADOK**, and Mr. **Abdeslem Hafid BENTBIB**, who have supervised me throughout this thesis and who have shared with me their brilliant intuitions. I would like to thank them for their kindness, their availability despite their numerous duties and for the numerous encouragements they have given me.

I would like to thank all the members of my jury: Mr. **Nour Eddine ALAA**, Mr. **Abdellah BNOUHACHEM**, Mr. **Hassan SAFOUHI**, and Mr. **Otmane SOUHAR**, for the honor they have given me by agreeing to be the referees of this thesis.

I address all my gratitude to all my friends and to all the people who helped me in the realization of this work. My sincere thanks also goes to **Abdelilah HAKIM** the director of the laboratory LAMAI-FSTG, and **Carole ROSIER** the director of the laboratory LMPA-UCLO as well as **Khalide JBILOU** and **Abderrahman BOUHAMIDI** to have welcomed me in the research unit and to have allowed me to work under good conditions.

My thanks would be incomplete if I did not mention my dear professors and educators who have accompanied me from kindergarten to my PhD.

Last but not the least, I would like to thank my family: my parents, my sister, and my brother for supporting me throughout writing this thesis and my life in general.

Contents

List of Tables

List of Figures

List of Acronyms

PSF Point Spread Function

ALS Alternating Least Squares

KPA Kronecker Product Approximation

PCA Principal Component Analysis

SVD Singular Value Decomposition

CP CANDECOMP decomposition

HOSVD Higher-order SVD

GKB Golub–Kahan bidiagonalization

GMRES Generalized minimal residual method

LSQR Least-squares method

GL-LSQR Global LSQR

RGB Red, Green, Blue

DFT Discrete Fourier Transformation

FFT Fast Fourier Transform

DCT Discrete Cosine Transformation

Publications and talks

Publications and submitted papers

- Bentbib, Abdeslem H., Asmaa Khouia, and Hassane Sadok. "The LSQR method for solving tensor least-squares problems." Electronic Transactions on Numerical Analysis 55 (2022): 92-111.
- Bentbib, Abdeslem H., Asmaa Khouia, and Hassane Sadok. "Color image and video restoration using tensor CP decomposition." BIT Numerical Mathematics (2022): 1-22.
- Asmaa khouia, "Higher-Order Tensor Decomposition Applied To Face Recognition" (submited).

List of talks

We presented our works in:

- Numerical Methods for Large Scale Problems which took place: 6-10 June 2022 in Belgrade
- Francophone Computer Algebra Days, national days of formal calculation which took place: February 28 - March 4, 2022 in Marseille.
- AHI EVRAN International Conference on Scientific Research, which took place: November 30 - December 1-2, 2021.

- 9th International Conference (online) on Applied Analysis and Mathematical Modeling, held: June 11-13, 2021.
- Scientific and Technical Days (JST 2021), which took place: March 22-25, 2021.
- International Hybrid Conference on Numerical Analysis and Optimization Days (JANO'13), which took place: 22-24 February 2021.
- Inter-laboratory research day 2020 JRIL, December 24, 2020.
- Doctoral Days of the Ecole Normale Supérieure of Rabat, December 24 and 25, 2019.

Chapter 1

Introduction

Tensors are multidimensional array, and they generalize matrices to higher dimensions. The order of a tensor is the number of dimensions, also known as ways or modes. Scalars can therefore be interpreted as zeroth-order tensors, vectors as first-order tensors, and matrices as second-order tensors. The tensors of order three or higher are refereed as higher-order tensors. The following figure shows how we can move from scalars to tensors.

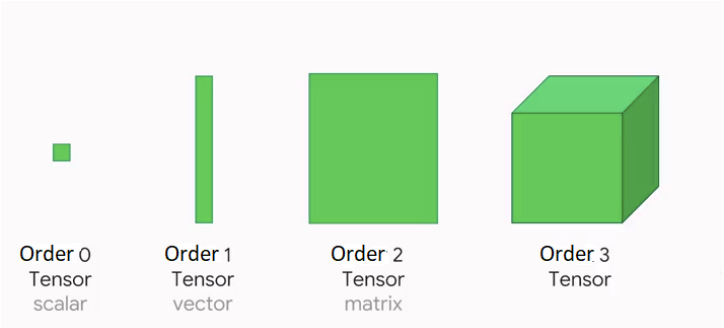


Figure 1.1: *0D, 1D, 2D and 3D tensors.*

Tensors and their decompositions originally appeared in 1927 [43], but have remained untouched by the computer science community until the late 20th century. They have since then spread to numerous other disciplines, including machine learning. In the last decade, the field of tensors has gained a huge interest in many scientific areas. Examples include computer vision [77, 78], signal processing [23, 21], numerical analysis [11, 12], neuroscience [4, 60] and other more fields. Just as matrices are used to

represent linear transformations, tensors can be used to represent more general types of transformations. Tensors are also a natural way of representing multidimensional data, such as images, where grayscale images can be considered as second-order tensors, color images (RGB images) are presented as third-order tensors, and a video composed of color images is a fourth-order tensor.

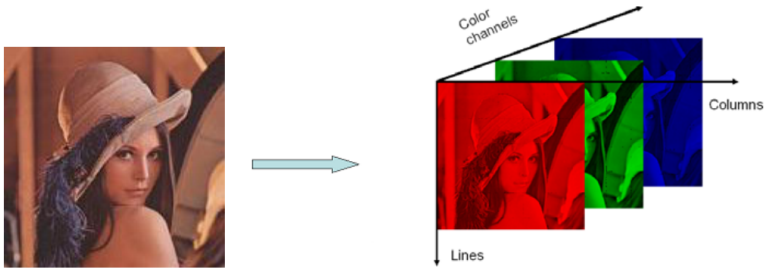


Figure 1.2: *A color image representation as a third-order tensor .*

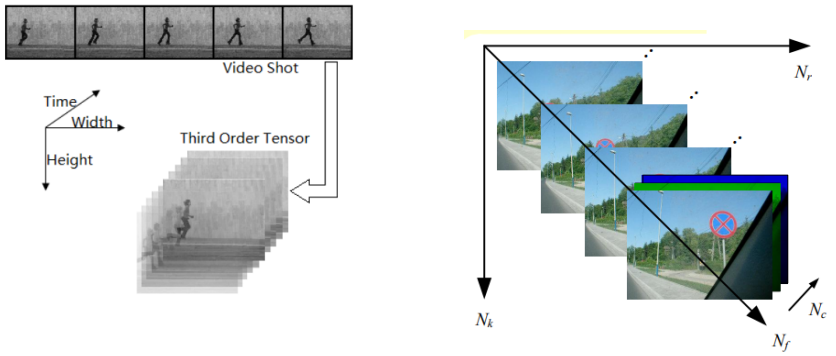


Figure 1.3: *gray-scale and color videos representation as a third-order and fourth-order tensors.*

Many of the datasets we deal with today come in a tensor format, such as in medical imaging, where different modalities of medical images of a given organ of a patient are captured in order to make a medical decision. Those images are often structured into a tensor.

Figure 1.4: *A multi-modal MRI dataset of a patient.*

In face recognition, since several factors such as expression, view angle, and illumination can significantly affect an image of a given person, in order to consider those factors, multi-linear algebra tools, such as high-order tensors, are being used. The following figure shows how a database that contains images of different people captured under different expressions, different angles of view, and different illuminations is stored as a fifth-order tensor.

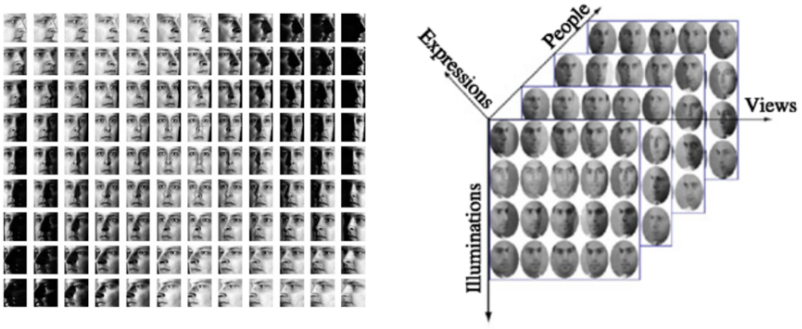


Figure 1.5: *Tensor Representation of a database.*

One of the reasons for the importance of tensors comes from the fact that they can be used to efficiently represent very high-dimensional data. For example, a tensor with 10^6 elements can represent a million-dimensional vector. Thus, tensors represent an essential tool for handling large-scale datasets. In order to handle large size data stored as tensors, higher-order factorizations (originated with Hitchcock in 1927[43, 44]) are proved as an important tool to extract useful information out of data. Rather than

flattening multi-dimensional arrays into matrices and using matrix factorization techniques to preserve the multi-way nature of the data and extract the underlying factors in each dimension, unlike matrix factorizations, the uniqueness of some tensor factorizations can be guaranteed under mild conditions.

One of the most important tensor factorizations are the CANDECOMP/PARAFAC (CP) decomposition and the Tucker decomposition. Those decompositions can be considered to be higher-order extensions of the matrix singular value decomposition. The CP decomposition of multi-dimensional arrays was first introduced by Hitchcock in 1927 [43, 44]. It decomposes a tensor as a sum of rank-one order tensors. It has received a lot of attention in the different areas of science and engineering, such as signal processing, data science, and machine learning. One of the famous methods for computing the CP decomposition is the alternating least squares (ALS). It was first proposed by Carroll and Chang [17]. There are multiple approaches to obtaining the Tucker decomposition, but the most widely used is the one known as the higher-order singular value decomposition (HOSVD) [24]. The applications of Tucker decompositions are varied, such as face recognition, human motion, data compression, etc.

1.1 Image Restoration

Images are produced in order to record or display useful information. They play a very important role in many aspects of our lives. A gray-scale image is stored and represented numerically as a matrix.

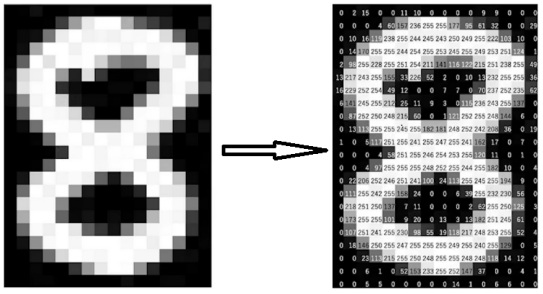


Figure 1.6: *Gray-scale image representation as a matrix.*

Each element of the matrix is called a pixel, which means that the dimension of an image is the number of pixels across the height and the width of the image. Each of these pixels contains a numerical value called pixel values. These pixel values represent the intensity of the pixel and they vary from 0 to 255. The numbers closer to zero represent the darker shades, while the numbers closer to 255 represent the lighter or the white shade. The matrix of pixel values is known as the channel, and in the case of a gray-scale image, there is only one channel. In the case of color images, they can be generated from three primary colors: red, green, and blue, which means a color image is composed of three channels: red channel, green channel, and blue channel.

Due to imperfections in the image formation process, the recorded image often represents a degraded version of the original scene. We can distinguish two sources of degradation: the process of image formation and the process of image recording. The degradation due to the process of image formation is usually denoted by blurring It can be caused by relative motion between the camera and the original scene, or by the optical system. The degradation introduced by the recording process is usually denoted by noise and is due to measurement errors. Even with the most recent technologies, however, there are many situations where the gradations of the original scene are too important. This can be, for example, due to the difficult acquisition conditions encountered and sometimes for reasons of cost.

The field of image restoration is concerned with the problem of undoing the effects of imperfections in the image formation process. More specifically, the goal of image restoration is to estimate the proprieties of the imperfect imaging system (blur) from the observed degraded image. In many fields, images represent a key source of data, making high quality imaging systems very essential. As a consequence, the applications of image restoration are various, such as medical imaging [68, 69], astronomical imaging [3, 10], surveillance [63] etc.

In medical imaging, a number of imaging techniques and devices are being invented. As in any other imaging system, medical image acquisition devices also introduce degradation to the images. Image restoration methods play an important role in improving the quality of images obtained from medical imaging devices.

One of the most common and important applications of image restoration is in the field of astronomy, where the images obtained from space telescopes are subject to many degradations. They were a result of atmospheric turbulence, aberrations of the optical system, relative motion between the camera and the object, and other reasons due to the enormous expense required to obtain such images. The loss of information due to the degradation of astronomical images could be devastating, which makes astronomical imaging an important application of image restoration.

The linear model of the image restoration problem is described by the Fredholm integral equation of the first kind:

$$g(x,y) = \int_{\mathbb{R}^2} k(x,y;s,t) f(s,t) \mathrm{d}s\mathrm{d}t + \eta(x,y) \tag{1.1}$$

where f is the true image, g is the observed image, and η is additive noise. The kernel function k models the blurring operation and is called the point spread function (PSF). In many situations, the blur is assumed to be spatially invariant, which means the kernel operator k satisfies $k(x,y;s,t) = k(x-s, y-t)$. In this case, the degradation model given in (1.1) is formulated as a convolution operation:

$$g(x,y) = (k \star f)(x,y) + \eta(x,y)$$

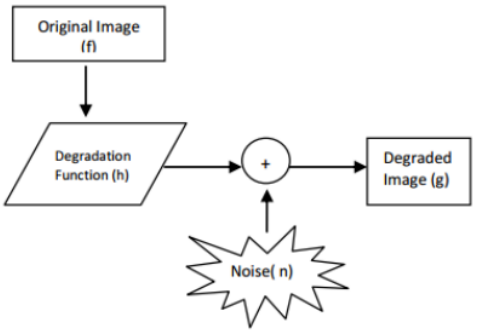


Figure 1.7: *Degradation Model*

And thus, the image restoration problem is often referred to as deconvolution. The symbol " $\star$ " indicates 2-dimensional convolution. The objective of restoration is to

obtain an estimate $\hat{f}$ of the original image f.

Unfortunately, the model (1.1) is not very useful for image restoration because of the complexity implied by the possibility of having a different PSF $k(x, y; s, t)$ at each coordinate (s, t) of the image, and it is unrealistic to assume that one might be able to estimate a different PSF for each location in the image. In addition, we do not have a precise function definition for g because the observed image is recorded digitally, and thus is known only at discrete values. Moreover, in many cases, it is necessary to estimate k from measured data. Therefore, it is natural to consider the digital image restoration problem

$$g = Kf + \eta \tag{1.2}$$

which is obtained from the equation (1.1) by discretizing the functions and approximating integration with a quadrature rule. K is a matrix that represents the blurring operation, and it can be constructed via the point spread function (PSF) and the boundary conditions [39] since edges are important structures of the true image and they should be preserved during image restoration. In the case of spatially invariant blur, the structure of the matrix Kdepends on the imposed boundary conditions. For example, if we impose zero boundary conditions by assuming that outside the borders of the image is black everywhere, the matrix K is a block Toeplitz with Toeplitz blocks (BTTB). This assumption is useful when dealing with astronomical images, since most of the time it is possible to assume that the outside borders of the image are black.

If the images are assumed to be matrices of size $m \times n$. Then the blurring matrix K is of size $N \times N$, with $N = mn$ the number of pixels of the image, and g, f, and η are vectors of size N. The problem of image restoration has now been reduced to the problem of solving the equation (1.2). There are a large number of approaches providing solutions to the image restoration problem [1, 2, 8]. A classical approach for solving Eq (1.2) is to calculate its least squares solution

$$\min_f \|Kf - g\| \tag{1.3}$$

But there are many aspects that make solving the problem (1.3) very challenging. In

fact, the dimensions of the matrix K can be extremely large since if the observed image is of size 512×512, then the matrix K is of size 262144×262144. Thus, the problem (1.3) is large-scale, which makes solving it computationally expensive since these computations usually involve matrix vector products. A successful technique for overcoming this problem is to exploit the structure of the matrix K. In particular, when the blur is separable, which means the horizontal and vertical components of the blur can be separated, in this case, the blur kernel satisfies $k(x, y; s, t) = k_1(x, s)k_2(y, t)$, and so the matrix K can be represented as Kronecker product of two matrices K_r and K_c,

$$K = K_r \otimes K_c$$

If the observed images have $m \times n$ pixels, then K_r and K_c are matrices of size $n \times n$ and $m \times m$ respectively. This Kronecker decomposition of the matrix K reduces the dimension of the problem (1.2) from $mn \times mn$ to $m \times n$. In this case, the blurring model can be formulated in this form,

$$K_c F K_r^T = G,$$

where $G, F \in \mathbb{R}^{m \times n}$, and K_r^T is the transpose of K_r. In the non-separable case, one can approximate the matrix K by solving the Kronecker product approximation (KPA) problem [76].

$$(\hat{K}_r, \hat{K}_c) = \underset{K_r, K_c}{\mathbf{arg\,min}} \|K - K_r \otimes K_c\| . \tag{1.4}$$

The second challenging problem of the image restoration problem is that the matrix K is severely ill-conditioned, with singular values decaying to and clustering at 0. This means that the problem (1.3) is sensitive to any perturbation. Regularization is needed to avoid computing solutions that are corrupted by noise. There are several regularization techniques that can be used to regularize the problem (1.2). The most popular regularization approach in the field of image restoration research is the Tikhonov regularization. The Tikhonov seeks to determine a useful approximation of f by replacing

the minimization problem (1.3) by the problem of the form:

$$\min_f \|Kf - g\|_2^2 + \lambda \|Lf\|_2^2 \tag{1.5}$$

where $\lambda > 0$ is the regularization parameter that need to be chosen, and L is the regularization matrix. A common choice of L is the identity matrix. In this case, we obtain Tikhonov regularization in standard form:

$$\min_f \|Kf - g\|_2^2 + \lambda \|f\|_2^2 \tag{1.6}$$

The problem (1.6) is equivalent to the problem:

$$\min_f \left\|\hat{K}f - \hat{g}\right\|^2$$

where $\hat{g} = \begin{pmatrix} g \\ 0 \end{pmatrix}$, $\hat{K} = \begin{pmatrix} K \\ \sqrt{\lambda} I \end{pmatrix}$, which satisfy the normal equation:

$$(K^T K + \lambda I) f = K^T g.$$

For the choice of the regularization parameter. There exist numerous methods [16, 34, 29, 55], such as generalized cross validation (GCV) [34]. The GCV parameter λ_{GCV} is computed to minimize the GCV function,

$$GCV(\lambda) = \frac{||Kf_\lambda - g||_2^2}{(trace(I - KK_\lambda^{-1}K^T))^2},$$

where f_λ is solution of $K_\lambda f = K^T g$, and $K_\lambda = K^T K + \lambda L^T L$.

1.2 Face recognition

In the past several decades, biometrics has gained a lot of attention, and it has been growing rapidly. A biometric system is a pattern recognition system that recognizes a person based on their physiological characteristics (fingerprints, face, hand contour,

... etc.), or behavioral characteristics (signature, ... etc.). These characteristics are called biometric modalities. The following figure shows the different types of biometric modalities,

Figure 1.8: *Example of biometric modalities*

Among all the biometric modalities used for people recognition, face is one of the commonly acceptable biometrics used to recognize a person due to the huge developments in the field of image processing and machine learning, beyond its scientific interest, compared to other biometrics, the face offers many advantages, for instance, instead of asking users to place their hand or fingers on a reader or to accurately position their eyes in front of a scanner, face data can be easily captured via digital cameras, PC cameras or even smart phones camera (Cameras are available all around us). Another reason that makes face recognition very important is that most biometric modalities require professional equipment to implement.

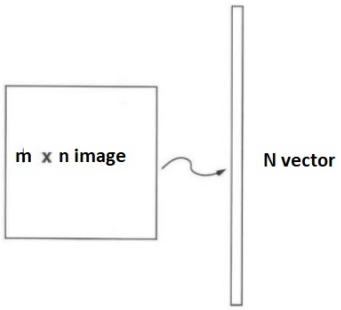
Recently, face recognition technology has become an increasingly important part of people's daily lives in the form of relevant applications. Face recognition can be applied in different areas, such as information security, access control, law enforcement, smart cards, and surveillance systems. Face recognition is an interdisciplinary science in which techniques from computer science, statistics, data analysis, optimization, linear algebra, and most recently, multi-linear algebra are used.

There are different approaches and algorithms that have been developed for face recognition. Principal Component Analysis (PCA)[70, 75] and Linear Discriminant Analysis (LDA) [6] are well known techniques commonly exploited in the field of face recognition. The PCA approach, known as the eigenface method, is one of the popular methods for feature selection and dimension reduction. It was introduced by Matthew Turk

and Alex Pentland in 1991. Numerous extensions to the standard PCA method have been developed, such as multi-linear PCA [40]. Next, we will give a brief description of the eigenfaces algorithm.

1.2.1 Eigenfaces (PCA)

Eigenface is a set of features obtained by principal component analysis (PCA) building on singular value decomposition (SVD) to project the higher-dimensional face-image space to a lower dimension. Let's consider a set of l images $(X_1, \ldots, X_l)$ of dimension $m \times n$. The first step is to convert these images into vectors $(x_1, \ldots, x_l)$ of size N, with $N = mn$.



Then, we calculate the mean of all the face vectors $\bar{x} = \frac{1}{l}\sum_{i=1}^{l} x_i$, and subtract it from each vector x_i, $i = 1, \ldots, l$

$$a_i = x_i - \bar{x}.$$

We define the matrix A of size $N \times l$ as follows,

$$A = [a_1\ a_2\ \cdots a_l]$$

The covariance matrix is given by

$$C = \frac{1}{l-1} AA^T$$

where A^T is the transpose of A, C is symmetric semi-definite and is orthogonally diagonalizable, which can be written as $C = UDU^T$, where D is a diagonal matrix that contain the eigenvalues of C, U is an orthogonal matrix, where each column u_i of U is

an eigenvector of C. The eigenvectors of the matrix C are referred to as the principal components. After computing the eigenvectors and the eigenvalues of the covariance matrix C, we chose the k eigenvectors of C corresponding to the k largest eigenvalues, with $k < l$. Using the first k eigenvectors, we can approximate each of the normalized face vectors $a_i$as a linear combination of $(u_1, \ldots, u_k)$.

$$a_i = \sum_{j=1}^{k} \alpha_j^i u_j \tag{1.7}$$

These u_j for $j = 1, \ldots, k$ are called eigenfaces, since each eigenface can be viewed as a feature. Using (1.7), we can represent each vector a_i by its coordinates $\alpha^i = (\alpha_1, \ldots, \alpha_k)$ with respect to the k principal components. We point out that, for numerical reasons, the covariance matrix C and its eigenvectors are not computed. Instead, given the SVD of the matrix $A = USV^T$, we have

$$C = \frac{1}{l-1} US^2 U^T,$$

with $D = \frac{1}{l-1} S^2$, and U contains the eigenvectors of C.

Given an unknown face $Y \in \mathbb{R}^{m \times n}$ which we want to classify. The first step is to subtract the face from the mean,

$$\bar{y} = y - \bar{x},$$

with $y = vec(Y)$. Then we project the normalized vector $\bar{y}$ into the eigenspace $U_k\ \bar{y}$. At the end, we classify the person in the image Y as the person $X_{\bar{l}}$ in the training set, with $\bar{l}$ satisfy

$$\bar{l} = \underset{i}{argmin} \|\alpha - \alpha_i\|$$

with α is the coordinate of $\bar{y}$ with respect to the k principal components. Recently, multiple extensions of the eigenfaces method have been developed, such as the t-SVD approach [40] defined via the t-product. This approach differ from the traditional eigenfaces method is that the data is represented as third-order tensor of size $m \times l \times n$, which mean in this approach the training images are not vectorized, and similarly

to the traditional method the covariance tensor and its associated are not computed. Instead, a generalization of the matrix SVD via the t-product is used.

The performance of face recognition algorithms such as eigenfaces and Fisherfaces (LDA based approach) is good when the only variable that counts for image formation is the identity of the person, which means that the faces are captured under controlled conditions. In reality, there are several factors that can affect the image of a given person, such as illumination, that have a great influence on the appearance of the face in the image. Illumination conditions are unavoidable in the real world, especially when views are collected at different times. Other factors that can affect a face image are view angle, expression... Another problem with using face recognition algorithms such as eigenfaces is that in different fields, a natural representation of images is a third-order tensor rather than a simple matrix of vectorized images. Multiple approaches have been developed to overcome these problems. Among the solutions proposed is using a tensor representation of the data. For example, we can represent the face database of subjects photographed in different poses under different illuminations and different face expressions as a fifth-order tensor when the images are represented as matrices, or we can represent the face database as a sixth-order tensor when the images are represented as third-order tensors. In order to manipulate the data stored as a tensor, different multi-linear algebra tools are used. For instance, in [26, 77] the High-Order SVD (HOSVD, see [24, 74]) is used to classify the image of an unknown person, and in [14] the authors explore the use of the Tensor-Train decomposition (for TT-decomposition, see [64]) for multi-feature recognition strategies.

1.3 Chapter-by-chapter overview

The thesis can be divided into chapters, a brief summary of every chapter is given as follows:

Chapter 3 is the preliminaries chapter in which we present the basics of multi-linear algebra that will be useful in the development of the techniques presented in this thesis and to help the readers understand multi-linear concepts. We introduce some basic

tensor and matrix operations together with their properties, and also give a brief introduction to some of the basic tensor decompositions used in later chapters.

In chapter 4, we propose a new approach to image and video restoration. This approach constructs a degradation model based on a tensor representation, where a color image is represented by a third-order tensor, and a video composed of color images is a fourth-order tensor. Applying tensor CP decomposition to our original problem leads to three subproblems. To solve those subproblems, we apply the global LSQR algorithm and a new algorithm based on Golub Kahan bidiagonalization.

In chapter 5, we are interested in finding an approximate solution $\hat{\mathcal{X}}$ of the tensor least squares minimization problem $\min_{\mathcal{X}} \left\| \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} - \mathcal{G} \right\|$ where $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and $A^{(i)} \in \mathbb{R}^{J_i \times I_i}$ $(i = 1, \ldots, N)$ are known, and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the unknown tensor to be approximated. Our approach is based on two steps. Firstly, we apply the CP or HOSVD decomposition to the right-hand side tensor $\mathcal{G}$. Secondly, we perform the well-known Golub-Kahan bidiagonalization on each coefficient matrix $A^{(i)}(i = 1, \ldots, N)$ to obtain a reduced tensor least squares minimization problem. This type of equation may appear in color image and video restorations.

In chapter 6, we explore the use of the t-product, the cosine product, and the outer product, applied to face recognition. The proposed approaches are based on using tensor decompositions of an arrangement of images in a database when we add a factor such as illumination, view angle, or expression. Our algorithms can be applied to a database of images represented by a third or fourth-order tensor.

Chapter 2

Introduction en français

Les tenseurs sont des tableaux multidimensionnels, et ils généralisent les matrices à des dimensions supérieures. L'ordre d'un tenseur est le nombre de dimensions, également appelées voies ou modes. Les scalaires peuvent donc être interprétés comme des tenseurs d'ordre zéro. les vecteurs comme des tenseurs de premier ordre et les matrices comme des tenseurs de second ordre. Les tenseurs d'ordre trois ou plus sont appelés tenseurs d'ordre supérieur. La figure suivante montre comment on peut passer des scalaires aux tenseurs.

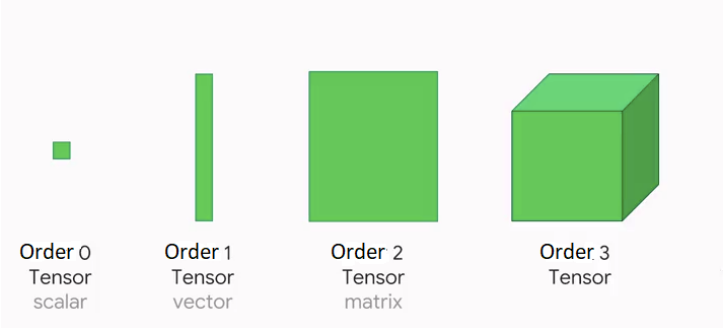


Figure 2.1: *Tenseurs 0D, 1D, 2D et 3D*

.

Les tenseurs et leurs décompositions sont apparus à l'origine en 1927 [43], mais sont restés intouchés par la communauté informatique jusqu'à la fin du 20e siècle. Elles Depuis lors, ils se sont utilisés à de nombreuses autres disciplines, dont l'apprentissage automatique. Au cours de la dernière décennie, le domaine des tenseurs a suscité un

intérêt considérable dans de nombreux domaines scientifiques. Citons par exemple la vision par ordinateur [77, 78], le traitement du signal [23, 21], l'analyse numérique [11, 12], les neurosciences [4, 60] et d'autres domaines encore. Tout comme les matrices sont utilisées pour représenter les transformations linéaires, les tenseurs peuvent être utilisés pour représenter des types de transformations plus généraux. Les tenseurs sont également un moyen naturel de représenter des données multidimensionnelles, comme les images, où les images en niveaux de gris peuvent être considérées comme des tenseurs de deuxième ordre, les images en couleurs (images RVB) sont présentées comme des tenseurs de troisième ordre, et une vidéo composée d'images en couleurs est un tenseur de quatrième ordre.

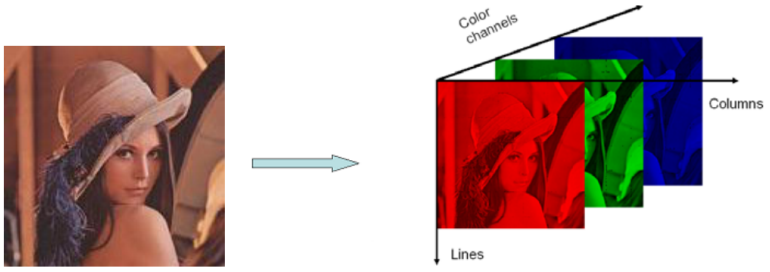


Figure 2.2: *Une représentation d'une image couleur sous forme de tenseur du troisième ordre .*

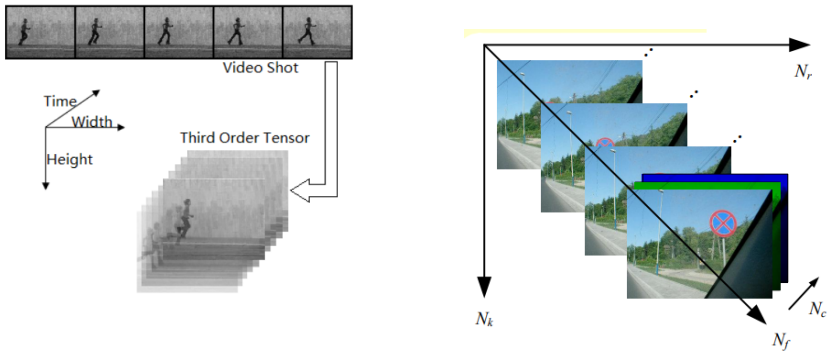


Figure 2.3: *représentation des vidéos en gris et en couleur sous forme de tenseurs d'ordre 3 et 4*

.

De nombreux ensembles de données que nous traitons aujourd'hui se présentent sous

la forme d'un tenseur, comme dans le cas de l'imagerie médicale, où différentes modalités d'images médicales d'un organe donné d'un patient sont capturées afin de prendre une décision médicale. Ces images sont souvent structurées en un tenseur.

Figure 2.4: *Ensemble de données IRM multimodales d'un patient*

.

Dans le domaine de la reconnaissance des visages, étant donné que plusieurs facteurs tels que l'expression, l'angle de vue et l'éclairage peuvent affecter de manière significative l'image d'une personne donnée, des outils d'algèbre multi-linéaire, tels que les tenseurs d'ordre élevé, sont utilisés afin de prendre en compte ces facteurs. La figure suivante montre comment une base de données contenant des images de différentes personnes capturées sous différentes expressions, différents angles de vue et différents éclairages est stockée comme un tenseur d'ordre 5.

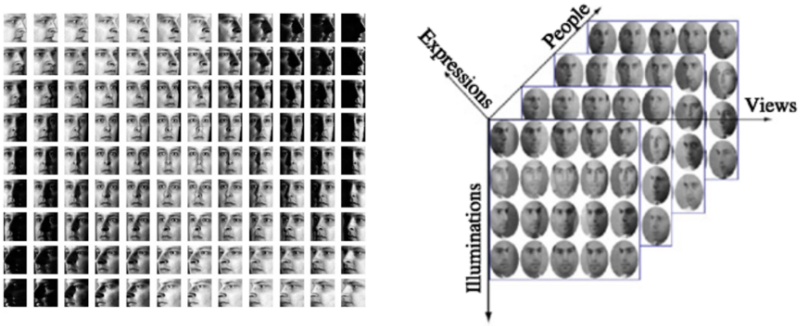


Figure 2.5: *Représentation tensorielle d'une base de données*

.

L'une des raisons de l'importance des tenseurs vient du fait qu'ils peuvent être utilisés pour représenter efficacement des données à très haute dimension. Par exemple, un tenseur comportant 10^6 éléments peut représenter un vecteur à un million de dimensions. Ainsi, les tenseurs représentent un outil essentiel pour traiter des ensembles de données à grande échelle. Afin de traiter les données de grande taille stockées sous forme de tenseurs, les factorisations d'ordre supérieur (dont l'origine remonte à Hitchcock en 1927) s'avèrent être un outil important pour extraire des informations utiles des données. Plutôt que de transformer des tableaux multidimensionnels en matrices et d'utiliser des techniques de factorisation matricielle pour préserver la nature multidimensionnelle des données et extraire les facteurs sous-jacents dans chaque dimension, contrairement aux factorisations matricielles, l'unicité de certaines factorisations tensorielles peut être garantie sous certaines conditions.

L'une des factorisations tensorielles les plus importantes est la décomposition CANDECOMP/PARAFAC (CP) et la décomposition de Tucker. Ces décompositions peuvent être considérées comme des extensions d'ordre supérieur de la décomposition de la valeur singulière des matrices. La décomposition CP des tableaux multidimensionnels a été introduite pour la première fois par Hitchcock en 1927 [43, 44]. Il décompose un tenseur en une somme de tenseurs d'ordre un. Elle a reçu beaucoup d'attention dans les différents domaines de la science et de l'ingénierie, tels que le traitement du signal, la science des données et l'apprentissage automatique. L'une des méthodes les plus connues pour calculer la décomposition CP est la méthode des moindres carrés alternatifs (ALS). Elle a été proposée pour la première fois par Carroll et Chang [17]. Il existe de multiples approches pour obtenir la décomposition de Tucker, mais la plus utilisée est celle connue sous le nom de décomposition en valeurs singulières d'ordre supérieur (HOSVD) [24]. Les applications des décompositions de Tucker sont variées, telles que la reconnaissance des visages, le mouvement humain, la compression de données, etc.

2.1 Restauration d'images

Les images sont produites afin d'enregistrer ou d'afficher des informations utiles. Elles jouent un rôle très important dans de nombreux aspects de notre vie. Une image en niveaux de gris est stockée et représentée numériquement sous forme de matrice.

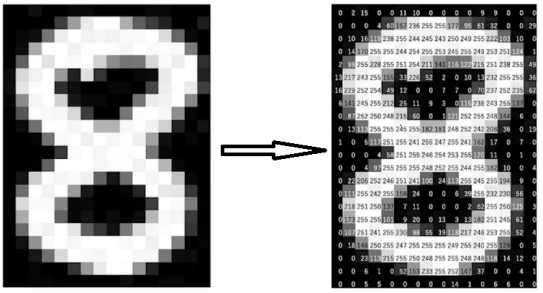


Figure 2.6: *Représentation d'une image à échelle de gris sous forme de matrice*

.

Chaque élément de la matrice est appelé pixel, ce qui signifie que la dimension d'une image est le nombre de pixels sur la hauteur et la largeur de l'image. Chacun de ces pixels contient une valeur numérique appelée valeur du pixel. Ces valeurs de pixel représentent l'intensité du pixel et elles varient entre 0 et 255. Les nombres plus proches de zéro représentent les nuances les plus sombres, tandis que les nombres plus proches de 255 représentent les nuances les plus claires ou le blanc. La matrice des valeurs des pixels est appelée canal, et dans le cas d'une image en niveaux de gris, il n'y a qu'un seul canal. Dans le cas des images en couleur, elles peuvent être générées à partir de trois couleurs primaires : le rouge, le vert et le bleu, ce qui signifie qu'une image en couleur est composée de trois canaux : le canal rouge, le canal vert et le canal bleu.
En raison des imperfections du processus de formation de l'image, l'image enregistrée représente souvent une version dégradée de la scène originale. On peut distinguer deux sources de dégradation : le processus de formation de l'image et le processus d'enregistrement de l'image. La dégradation due au processus de formation de l'image est généralement désignée par le terme "flou". Elle peut être causée par le mouvement relatif entre la caméra et la scène originale, ou par le système optique. La dégradation introduite par le processus d'enregistrement est généralement appelée bruit et est due

à des erreurs de mesure. Cependant, même avec les technologies les plus récentes, il existe de nombreuses situations où les gradations de la scène originale sont trop importantes. Cela peut être dû, par exemple, aux conditions d'acquisition difficiles rencontrées et parfois pour des raisons de coût.

Le domaine de la restauration d'images s'intéresse au problème de l'annulation des effets des imperfections dans le processus de formation de l'image. Plus précisément, le but de la restauration d'image est d'estimer les propriétés du système d'imagerie imparfait (flou) à partir de l'image dégradée observée. Dans de nombreux domaines, les images représentent une source de données essentielle, ce qui rend les systèmes d'imagerie de haute qualité très indispensables. Par conséquent, les applications de la restauration d'images sont variées, telles que l'imagerie médicale [68, 69], l'imagerie astronomique [3, 10], la surveillance [63] etc.

En imagerie médicale, un certain nombre de techniques et de dispositifs d'imagerie sont inventés. Comme dans tout autre système d'imagerie, les dispositifs d'acquisition d'images médicales introduisent également des dégradations dans les images. Les méthodes de restauration d'images jouent un rôle important dans l'amélioration de la qualité des images obtenues à partir de dispositifs d'imagerie médicale.

L'une des applications les plus courantes et les plus importantes de la restauration d'images se trouve dans le domaine de l'astronomie, où les images obtenues à partir de télescopes spatiaux sont soumises à de nombreuses dégradations. Elles sont le résultat de turbulences atmosphériques, d'aberrations du système optique, de mouvements relatifs entre la caméra et l'objet, et d'autres raisons dues à l'énorme dépense nécessaire pour obtenir de telles images. La perte d'informations due à la dégradation des images astronomiques pourrait être dévastatrice, ce qui fait de l'imagerie astronomique une application importante de la restauration d'images.

Le modèle linéaire du problème de restauration d'images est décrit par l'équation intégrale de Fredholm:

$$g(x,y) = \int_{\mathbb{R}^2} k(x,y;s,t) f(s,t) \mathrm{d}s\mathrm{d}t + \eta(x,y) \tag{2.1}$$

où f est l'image réelle, g est l'image observée, et η est un bruit additif. La fonction noyau k modélise l'opération de flou et est appelée fonction d'étalement du point (PSF). Dans de nombreuses situations, le flou est supposé être spatialement invariant, ce qui signifie que l'opérateur de noyau k satisfait à $k(x, y; s, t) = k(x - s, y - t)$. Dans ce cas, le modèle de dégradation donné dans (2.1) est formulé comme une opération de convolution :

$$g(x, y) = (k \star f)(x, y) + \eta(x, y)$$

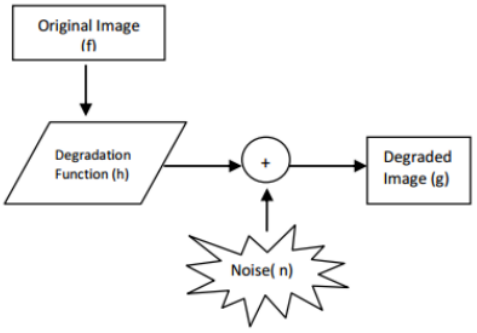


Figure 2.7: *Modèle de dégradation*

Ainsi, le problème de restauration d'image est souvent appelé déconvolution. Le symbole " $\star$ " indique une convolution bidimensionnelle. L'objectif de la restauration est d'obtenir une estimation $\hat{f}$ de l'image originale f.

Malheureusement, le modèle (2.1) n'est pas très utile pour la restauration d'images en raison de la complexité impliquée par la possibilité d'avoir une PSF différente $k(x, y; s, t)$ à chaque coordonnée (s, t) de l'image, et il n'est pas réaliste de supposer que l'on puisse estimer une PSF différente pour chaque emplacement dans l'image. De plus, nous ne disposons pas d'une définition précise de la fonction pour g car l'image observée est enregistrée numériquement, et n'est donc connue que pour des valeurs discrètes. De plus, dans de nombreux cas, il est nécessaire d'estimer k à partir de données mesurées. Par conséquent, il est naturel de considérer le problème de restauration d'images numériques suivant

$$g = Kf + \eta \tag{2.2}$$

qui est obtenue à partir de l'équation (2.1) en discrétisant les fonctions et en approchant

l'intégration avec une règle de quadrature. K est une matrice qui représente l'opération de flou, et elle peut être construite via la fonction d'étalement du point (PSF) et les conditions aux limites [39] puisque les bords sont des structures importantes de l'image réelle et qu'ils doivent être préservés pendant la restauration de l'image. Dans le cas d'un flou spatialement invariant, la structure de la matrice K dépend des conditions aux limites imposées. Par exemple, si nous imposons des conditions aux limites nulles en supposant qu'en dehors des frontières de l'image, tout est noir, la matrice K est un bloc Toeplitz avec des blocs Toeplitz (BTTB). Cette hypothèse est utile lorsqu'on traite des images astronomiques, car la plupart du temps, il est possible de supposer que les bords extérieurs de l'image sont noirs.

Si l'on suppose que les images sont des matrices de taille $m \times n$. Alors la matrice de flou K est de taille $N \times N$, avec $N = mn$ le nombre de pixels de l'image, et g, f, et η sont des vecteurs de taille N. Le problème de la restauration d'image est maintenant réduit au problème de la résolution de l'équation (2.2). Il existe un grand nombre d'approches fournissant des solutions au problème de la restauration d'images [1, 2, 8]. Une approche classique pour résoudre l'équation (2.2) consiste à calculer sa solution en résolvant ce problème moindres carrés

$$\min_f \|Kf - g\| \tag{2.3}$$

Mais de nombreux aspects rendent la résolution du problème (2.3) très difficile. En effet, les dimensions de la matrice K peuvent être extrêmement grandes puisque si l'image observée a une taille de 512×512, alors la matrice K a une taille de 262144×262144. Ainsi, le problème (2.3) est à grande échelle, ce qui rend sa résolution coûteuse en termes de calcul puisque ces calculs impliquent généralement des produits vectoriels matriciels. Une technique efficace pour surmonter ce problème consiste à exploiter la structure de la matrice K. En particulier, lorsque le flou est séparable, ce qui signifie que les composantes horizontale et verticale du flou peuvent être séparées, dans ce cas, le noyau du flou satisfait $k(x, y; s, t) = k_1(x, s)k_2(y, t)$, et donc la matrice K

peut être représentée comme produit de Kronecker de deux matrices K_r et K_c,

$$K = K_r \otimes K_c$$

Si les images observées ont $m \times n$ pixels, alors K_r et K_c sont des matrices de taille $n \times n$ et $m \times m$ respectivement. Cette décomposition de Kronecker de la matrice K réduit la dimension du problème (2.2) de $mn \times mn$ à $m \times n$. Dans ce cas, le modèle de dégradation peut être formulé sous cette forme,

$$K_c F K_r^T = G,$$

où $G, F \in \mathbb{R}^{m \times n}$, et K_r^T est le transposé de K_r. Dans le cas non-séparable, on peut approximer la matrice K en résolvant le problème de l'approximation du produit de Kronecker (KPA) [76].

$$(\hat{K}_r, \hat{K}_c) = \underset{K_r, K_c}{\textbf{arg min}} \left\| K - K_r \otimes K_c \right\|. \tag{2.4}$$

Le deuxième défi du problème de restauration d'image est que la matrice K est sévèrement mal conditionnée, avec des valeurs singulières décroissant vers et se regroupant à 0. Cela signifie que le problème (2.3) est sensible à toute perturbation. Une régularisation est nécessaire pour éviter de calculer des solutions qui sont corrompues par le bruit. Il existe plusieurs techniques de régularisation qui peuvent être utilisées pour régulariser le problème (2.2). L'approche de régularisation la plus populaire dans le domaine de la recherche sur la restauration d'images est la régularisation de Tikhonov. La Tikhonov cherche à déterminer une approximation utile de f en remplaçant le problème de minimisation (2.3) par le problème de la forme :

$$\min_f \|Kf - g\|_2^2 + \lambda \|Lf\|_2^2 \tag{2.5}$$

où $\lambda > 0$ est le paramètre de régularisation qui doit être choisi, et L est la matrice de régularisation. Un choix courant de L est la matrice d'identité. Dans ce cas, nous

obtenons une régularisation de Tikhonov sous forme standard :

$$\min_f \|Kf - g\|_2^2 + \lambda \|f\|_2^2 \tag{2.6}$$

Le problème (2.6) est équivalent au problème :

$$\min_f \left\|\hat{K}f - \hat{g}\right\|^2$$

où $\hat{g} = \begin{pmatrix} g \\ 0 \end{pmatrix}$, $\hat{K} = \begin{pmatrix} K \\ \sqrt{\lambda} I \end{pmatrix}$, qui satisfont l'équation normale :

$$(K^T K + \lambda I) f = K^T g.$$

Pour le choix du paramètre de régularisation. Il existe de nombreuses méthodes [16, 34, 29, 55], comme la validation croisée généralisée (GCV) [34]. Le paramètre GCV λ_{GCV} est calculé en minimisant la fonction GCV,

$$GCV(\lambda) = \frac{||Kf_\lambda - g||_2^2}{(trace(I - KK_\lambda^{-1}K^T))^2},$$

où f_λ est la solution de $K_\lambda f = K^T g$, et $K_\lambda = K^T K + \lambda L^T L$.

2.2 Reconnaissance des visages

Au cours des dernières décennies, la biométrie a fait l'objet d'une grande attention et a connu une croissance rapide. Un système biométrique est un système de reconnaissance de formes qui reconnaît une personne sur la base de ses caractéristiques physiologiques (empreintes digitales, visage, contour de la main, ... etc.), ou comportementales (signature, ... etc.). Ces caractéristiques sont appelées modalités biométriques. La figure suivante montre les différents types de modalités biométriques, Parmi toutes les modalités biométriques utilisées dans la reconnaissance des personnes, le visage est l'une des biométries les plus couramment utilisées pour reconnaître une personne en raison des énormes développements dans le domaine du traitement d'images et de

Figure 2.8: *Exemple de modalités biométriques*

l'apprentissage automatique. Au-delà de son intérêt scientifique, comparé aux autres biométries, le visage offre de nombreux avantages, par exemple, au lieu de demander aux utilisateurs de placer leur main ou leurs doigts sur un lecteur ou de positionner précisément leurs yeux devant un scanner, les données du visage peuvent être facilement capturées par des caméras numériques, des caméras de PC ou même des caméras de téléphones intelligents (les caméras sont disponibles partout autour de nous). Une autre raison qui rend la reconnaissance faciale très importante est que la plupart des modalités biométriques nécessitent un équipement professionnel pour être mises en œuvre.

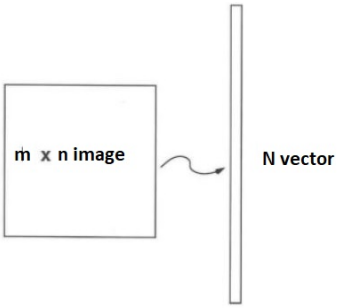
Récemment, la technologie de reconnaissance du visage est devenue une partie de plus en plus importante de la vie quotidienne des gens sous la forme d'applications pertinentes. La reconnaissance faciale peut être appliquée dans différents domaines, tels que la sécurité de l'information, le contrôle d'accès, l'application de la loi, les cartes à puce et les systèmes de surveillance. La reconnaissance des visages est une science interdisciplinaire dans laquelle sont utilisées des techniques issues de l'informatique, des statistiques, de l'analyse des données, de l'optimisation, de l'algèbre linéaire et, plus récemment, de l'algèbre multi-linéaire.

Il existe différentes approches et algorithmes qui ont été développés pour la reconnaissance des visages. L'analyse en composantes principales ou PCA [70, 75] et l'analyse discriminante linéaire ou LDA [6] sont des techniques bien connues et couramment exploitées dans le domaine de la reconnaissance des visages. L'approche PCA, connue sous le nom de méthode des faces propres ou eigenfaces, est l'une des méthodes populaires pour la sélection des caractéristiques et la réduction de la dimension. Elle a été

introduite par Matthew Turk et Alex Pentland en 1991. De nombreuses extensions de la méthode PCA standard ont été développées, telles que le PCA multi-linéaire [40]. Ensuite, nous donnerons une brève description de l'algorithme des faces propres.

2.2.1 Faces propres (PCA)

Eigenface est un ensemble de caractéristiques obtenues par l'analyse en composantes principales (PCA) en s'appuyant sur la décomposition en valeurs singulières (SVD) pour projeter l'espace de l'image du visage de dimension supérieure à une dimension inférieure. Considérons un ensemble de l images $(X_1, \ldots, X_l)$ de dimension $m \times n$. La première étape consiste à convertir ces images en vecteurs $(x_1, \ldots, x_l)$ de taille N, avec $N = mn$.



Ensuite, nous calculons la moyenne de tous les vecteurs de visage $\bar{x} = \frac{1}{l}\sum_{i=1}^{l} x_i$, et on la soustrait de chaque vecteur x_i, $i = 1, \ldots, l$.

$$a_i = x_i - \bar{x}.$$

On définit la matrice A de taille $N \times l$ comme suit,

$$A = [a_1\ a_2\ \cdots a_l]$$

La matrice de covariance est donnée par

$$C = \frac{1}{l-1} A A^T$$

où A^T est le transposé de A, C est symétrique, semi-définie et diagonalisable orthogonalement, ce qui peut s'écrire comme suit : $C = UDU^T$, où D est une matrice diagonale qui contient les valeurs propres de C, U est une matrice orthogonale, où chaque colonne u_i de U est un vecteur propre de C. Les vecteurs propres de la matrice C sont appelés les composantes principales. Après avoir calculé les vecteurs propres et les valeurs propres de la matrice de covariance C, nous choisissons les k vecteurs propres de C correspondant aux k plus grandes valeurs propres, avec $k < l$. En utilisant les k premiers vecteurs propres, nous pouvons approximer chacun des vecteurs de visage normalisés a_i comme une combinaison linéaire de $(u_1, \ldots, u_k)$.

$$a_i = \sum_{j=1}^{k} \alpha_j^i u_j \tag{2.7}$$

Ces u_j pour $j = 1, \ldots, k$ sont appelés faces propres, puisque chaque face propre peut être considérée comme une caractéristique. En utilisant (2.7), nous pouvons représenter chaque vecteur a_i par ses coordonnées $\alpha^i = (\alpha_1, \ldots, \alpha_k)$ par rapport aux k composantes principales. Nous soulignons que, pour des raisons numériques, la matrice de covariance C et ses vecteurs propres ne sont pas calculés. Au lieu de cela, étant donné la SVD de la matrice $A = USV^T$, nous avons

$$C = \frac{1}{l-1} US^2 U^T,$$

avec $D = \frac{1}{l-1} S^2$, et U contient les vecteurs propres de C.

Étant donné un visage inconnu $Y \in \mathbb{R}^{m \times n}$ que l'on veut classer. La première étape consiste à soustraire le visage de la moyenne,

$$\bar{y} = y - \bar{x},$$

Ensuite, nous projetons le vecteur normalisé $\bar{y}$ dans l'espace propre $U_k \bar{y}$. Enfin, nous classons la personne dans l'image Y comme la personne $X_{\hat{l}}$ dans l'ensemble de données, avec $\bar{l}$ satisfaisant à

$$\bar{l} = \underset{i}{argmin} \|\alpha - \alpha_i\|$$

avec α est la coordonnée de $\bar{y}$ par rapport aux k composantes principales. Récemment, de multiples extensions de la méthode des faces propres ont été développées, telles que l'approche t-SVD [40] définie via le t-produit. Cette approche diffère de la méthode traditionnelle des faces propres car les données sont représentées sous la forme d'un tenseur d'ordre 3 de taille $m \times l \times n$, ce qui signifie que dans cette approche, les images utilisées ne sont pas vectorisées et, comme dans la méthode traditionnelle, le tenseur de covariance et son associé ne sont pas calculés. Au lieu de cela, une généralisation de la SVD matricielle via le t-produit est utilisée. Les performances des algorithmes de reconnaissance de visages tels que les eigenfaces et Fisherfaces (approche basée sur LDA) sont bonnes lorsque la seule variable qui compte dans la formation de l'image est l'identité de la personne, ce qui signifie que les visages sont capturés dans des conditions contrôlées. En réalité, il existe plusieurs facteurs qui peuvent affecter l'image d'une personne donnée, comme l'illumination, qui ont une grande influence sur l'apparence du visage dans l'image. Les conditions d'illumination sont inévitables dans le monde réel, en particulier lorsque les vues sont collectées à différents moments. D'autres facteurs qui peuvent affecter l'image d'un visage sont l'angle de vue, l'expression... Un autre problème lié à l'utilisation d'algorithmes de reconnaissance des visages tels que les visages propres est que dans différents domaines, une représentation naturelle des images est un tenseur du troisième ordre plutôt qu'une simple matrice d'images vectorisées. De multiples approches ont été développées pour surmonter ces problèmes. Parmi les solutions proposées, l'utilisation d'une représentation tensorielle des données. Par exemple, nous pouvons représenter la base de données de visages de sujets photographiés dans différentes poses sous différents éclairages et différentes expressions du visage comme un tenseur d'ordre 5. Par exemple, nous pouvons représenter la base de données de visages de sujets photographiés dans différentes poses sous différents éclairages et différentes expressions du visage comme un tenseur d'ordre 5 lorsque les images sont représentées comme des matrices, ou nous pouvons représenter la base de données de visages comme un tenseur d'ordre 6 lorsque les images sont représentées comme des tenseurs d'ordre 3. Afin de manipuler les données stockées sous forme de tenseur, différents outils d'algèbre multi-linéaire

sont utilisés. Par exemple, dans [26, 77] le High-Order SVD (HOSVD, voir [24, 74]) est utilisé pour classer l'image d'une personne inconnue, et dans [14], les auteurs explorent l'utilisation de la décomposition Tensor-Train (pour la décomposition TT, voir [64]) pour les stratégies de reconnaissance multi-fonctions.

2.3 Aperçu chapitre par chapitre

La thèse peut être divisée en chapitres, un bref résumé de chaque chapitre est donné comme suit :

Le chapitre 3 est le chapitre des préliminaires dans lequel nous présentons les bases de l'algèbre multi-linéaire qui seront utiles dans le développement des techniques présentées dans cette thèse et pour aider les lecteurs à comprendre les concepts multi-linéaires. Nous introduisons certaines opérations tensorielles et matricielles de base ainsi que leurs propriétés, et nous donnons également une brève introduction à certaines des décompositions tensorielles de base utilisées dans les chapitres suivants.

Dans le chapitre 4, nous proposons une nouvelle approche de la restauration d'images et de vidéos. Cette approche construit un modèle de dégradation basé sur une représentation tensorielle, où une image couleur est représentée par un tenseur de troisième ordre, et une vidéo composée d'images couleur est un tenseur de quatrième ordre. L'application de la décomposition CP tensorielle à notre problème original conduit à trois sous-problèmes. Pour résoudre ces sous-problèmes, nous appliquons l'algorithme global LSQR et un nouvel algorithme basé sur la bidiagonalisation de Golub Kahan.

Dans le chapitre 5, nous nous intéressons à la recherche d'une solution approximative $\hat{\mathcal{X}}$ du problème de minimisation des moindres carrés du tenseur $\min_{\mathcal{X}} \left\| \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} - \mathcal{G} \right\|$ où $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ et $A^{(i)} \in \mathbb{R}^{J_i \times I_i}$ $(i = 1, \ldots, N)$ sont connus, et $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ est le tenseur inconnu à approximer. Notre approche est basée sur deux étapes. Premièrement, nous appliquons la décomposition CP ou HOSVD au tenseur de droite $\mathcal{G}$. Deuxièmement, nous effectuons la bidiagonalisation bien connue de Golub-Kahan sur chaque matrice de coefficient $A^{(i)} (i = 1, \ldots, N)$

pour obtenir un problème réduit de minimisation des moindres carrés du tenseur. Ce type d'équation peut apparaître dans les restaurations d'images et de vidéos en couleur. Dans le chapitre 6, nous explorons l'utilisation du produit t, du produit cosinus et du produit extérieur, appliqués à la reconnaissance des visages. Les approches proposées sont basées sur l'utilisation de décompositions tensorielles d'un arrangement d'images dans une base de données lorsque nous ajoutons un facteur tel que l'illumination, l'angle de vue ou l'expression. Nos algorithmes peuvent être appliqués à une base de données d'images représentées par un tenseur d'ordre 3 ou 4.

Chapter 3

Preliminaries

In this chapter we present the basics of multi-linear algebra that will be useful in the development of the techniques presented in this thesis and to help the readers understand multi-linear concepts. We introduce some basic tensor and matrix operations together with their properties, and also give a brief introduction to some of the basic tensor decompositions used in later chapters.

3.1 Notation

Throughout this thesis we use the following notation, real numbers or scalars are denoted as lowercase letters $x, y \in \mathbb{R}$, vectors as boldface lowercase letters $\mathbf{x} = (x_i) \in \mathbb{R}^n$ and matrices as boldface capital letters $\mathbf{A} = \left(a_{ij}\right) \in \mathbb{R}^{m \times n}$. Tensors are denoted by boldface Euler script letters $\mathcal{X} = \left(x_{i_1 i_2 \cdots i_N}\right) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. The columns and rows of a matrix are written, respectively, as

$$\mathbf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_n] \quad \text{and} \quad \mathbf{A} = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix}$$

3.2 Matrix analysis

Definition 3.1.

Let A and B be two matrices of the same size, the inner product of A, B is given by

$$\langle A, B \rangle = tr(A^T B) \tag{3.1}$$

where $tr(A)$ denotes the trace of A, the sum of its diagonal elements, and the corresponding norm of (3.1) is the Frobenius norm defined as

$$\|A\| = \sqrt{\langle A, A \rangle}$$

Properties 3.2.1.

Let A and B two matrices of appropriate size, then we have

- $\left\|A^T\right\| = \|A\|$
- $\|AB\| \leq \|A\| \, \|B\|$
- $\|A - B\|^2 = \|A\|^2 + \|B\|^2 - \langle A, B \rangle$

3.2.1 Matrix products

Apart from the standard matrix product, there are multiple other matrix products that play an important role when working with tensors.

Definition 3.2. (Kronecker product)

The Kronecker product of two matrices $A \in R^{I \times J}$ and $B \in R^{T \times R}$ is a matrix denoted as $A \otimes B \in R^{IT \times JR}$ and defined as

$$A \otimes B = \begin{vmatrix} a_{11}B & \dots & \dots & {}_{1J}B \\ \vdots & & & \\ a_{I1}B & \dots & \dots & a_{IJ}B \end{vmatrix}$$

Theorem 3.3.

For matrices and vectors of appropriate size, the following properties hold:

1. $(\mathrm{A} \otimes \mathrm{B})^T = \mathrm{A}^T \otimes B^T$,

2. $(\mathrm{A} \otimes \mathrm{B})(\mathrm{C} \otimes \mathrm{D}) = \mathrm{AC} \otimes \mathrm{BD}$,

3. A *and* B *orthogonal* $\Rightarrow$ $\mathrm{A} \otimes \mathrm{B}$ *orthogonal*,

4. $(\mathbf{A} \otimes \mathbf{B})\mathbf{v} = \mathrm{vec}\left(BVA^T\right), \quad v = \mathrm{vec}(V)$,

For a matrix of size $m \times n$, vec($\mathbf{A}$) is a vector of size mn, obtained by stacking the columns of the matrix on top of one another.

Proof.

1. Fallows immediately from Definition 3.2.

2. For $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{p\times q}, C \in \mathbb{R}^{n\times r}, D \in \mathbb{R}^{q\times s}$,

$$\begin{aligned}\underbrace{(A \otimes B)}_{mp\times nq}\underbrace{(C \otimes D)}_{nq\times rs} &= \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}\begin{bmatrix} c_{11}D & \cdots & c_{1r}D \\ \vdots & \ddots & \vdots \\ c_{n1}D & \cdots & c_{nr}D \end{bmatrix} \\ &= \begin{bmatrix} \sum_{k=1}^{n} a_{1k}c_{k1}BD & \cdots & \sum_{k=1}^{n} a_{1k}c_{kr}BD \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} a_{mk}c_{k1}BD & \cdots & \sum_{k=1}^{n} a_{mk}c_{kr}BD \end{bmatrix} \\ &= AC \otimes BD \in \mathbb{R}^{mp\times rs}.\end{aligned}$$

3. Assuming that $\mathbf{A} \in \mathbb{R}^{m\times m}$ and $\mathbf{B} \in \mathbb{R}^{n\times n}$ are orthogonal, and using the properties (1) and (2) of Theorem 3.3, we have

$$\begin{aligned}(\mathbf{A} \otimes \mathbf{B})^T(\mathbf{A} \otimes \mathbf{B}) &= \left(\mathbf{A}^T \otimes \mathbf{B}^T\right)(\mathbf{A} \otimes \mathbf{B}) = \mathbf{A}^T\mathbf{A} \otimes \mathbf{B}^T\mathbf{B} = \mathbf{I}_m \otimes \mathbf{I}_n \\ &= \mathbf{A}\mathbf{A}^T \otimes \mathbf{B}\mathbf{B}^T = (\mathbf{A} \otimes \mathbf{B})\left(\mathbf{A}^T \otimes \mathbf{B}^T\right) = (\mathbf{A} \otimes \mathbf{B})(\mathbf{A} \otimes \mathbf{B})^T,\end{aligned}$$

where $\mathbf{I}_m$ and $\mathbf{I}_n$ denote identity matrices of order m and n, respectively. From the definition of Kronecker product, obviously $\mathbf{I}_m \otimes \mathbf{I}_n = \mathbf{I}_{mn}$.

4. For $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{p\times q}$ and $v \in \mathbb{R}^{nq}$,

$$
\begin{aligned}
(A \otimes B)v &= \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_{nq} \end{bmatrix} \\
&= \begin{bmatrix} a_{11}\mathrm{B}v_{1:q} + a_{12}\mathrm{B}v_{q+1:2q} + \cdots + a_{1n}\mathrm{B}v_{(n-1)q:nq} \\ \vdots \\ a_{m1}\mathrm{B}v_{1:q} + a_{m2}\mathbf{B}\mathbf{v}_{q+1:2q} + \cdots + a_{mn}\mathrm{B}v_{(n-1)q:nq} \end{bmatrix} \\
&= vec\left(\begin{bmatrix} \mathrm{B}v_{1:q} & \mathrm{B}v_{q+1:2q} & \cdots & \mathrm{B}v_{(n-1)q:nq} \end{bmatrix} \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} \right) \\
&= \mathrm{vec}(\mathbf{B} \underbrace{\begin{bmatrix} \mathbf{v}_{1:q} & \mathbf{v}_{q+1:2q} & \cdots & \mathbf{v}_{(n-1)q:nq} \end{bmatrix}}_{\mathbf{V}} \mathbf{A}^T)
\end{aligned}
$$

□

Definition 3.4. (Khatri-Rao product)

The Khatri-Rao product (KRP), denoted by $\odot$, of two matrices $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{B} \in \mathbb{R}^{p\times r}$ is defined as the "column-wise Kronecker product" given by

$$\mathrm{A} \odot \mathrm{B} = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \ldots & a_r \otimes b_r \end{bmatrix},$$

where a_j, b_j are the j^{th} columns of A and B respectively. By definition A and B have the same number of columns, and $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{mp\times r}$.

Theorem 3.5.

For matrices and vectors of appropriate size, the following properties hold:

1. $(\mathrm{A} \otimes \mathrm{B})(\mathrm{C} \odot \mathrm{D}) = \mathrm{AC} \odot \mathrm{BD}$,
2. $(\mathrm{A} \odot \mathrm{B})\mathbf{v} = \mathrm{vec}\left(B\,\mathrm{diag}(v)A^T\right)$,

If $v \in \mathbb{R}^n$, diag $(\mathbf{v})$ denotes diagonal $n \times n$ matrix with elements $v_1, v_2 \ldots, v_n$ on the diagonal.

Proof.

1. For $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{p\times q}, \mathbf{C} \in \mathbb{R}^{n\times r}, \mathbf{D} \in \mathbb{R}^{q\times r}$, using property (2) of Theorem 3.3, we have

$$\begin{aligned}
\underbrace{(\mathrm{A} \otimes \mathrm{B})}_{mp\times nq}\underbrace{(\mathrm{C} \odot \mathrm{D})}_{nq\times r} &= \begin{pmatrix} \mathbf{A} \otimes \mathrm{B}) \end{pmatrix} \begin{bmatrix} \mathbf{c}_1 \otimes \mathrm{d}_1 & \cdots & \mathbf{c}_r \otimes \mathrm{d}_r \end{bmatrix} \\
&= \begin{bmatrix} (\mathrm{A} \otimes \mathrm{B})(\mathbf{c}_1 \otimes \mathrm{d}_1) & \cdots & (\mathrm{A} \otimes \mathrm{B})(\mathbf{c}_r \otimes \mathbf{d}_r) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}\mathbf{c}_1 \otimes \mathrm{Bd}_1 & \cdots & \mathrm{Ac}_r \otimes \mathrm{Bd}_r \end{bmatrix} \\
&= \mathrm{AC} \odot \mathrm{BD}.
\end{aligned}$$

2. For $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{p\times n}$ and $\mathbf{v} \in \mathbb{R}^n$, using property (4) of Theorem 3.3, we have

$$\begin{aligned}
(\mathbf{A} \odot \mathbf{B})\mathbf{v} &= \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_n \otimes \mathbf{b}_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \\
&= (\mathbf{a}_1 \otimes \mathbf{b}_1) v_1 + (\mathbf{a}_2 \otimes \mathbf{b}_2) v_2 + \cdots + (\mathbf{a}_n \otimes \mathbf{b}_n) v_n \\
&= \mathrm{vec}\left(\mathbf{b}_1 v_1 \mathbf{a}_1^T\right) + \mathrm{vec}\left(\mathbf{b}_2 v_2 \mathbf{a}_2^T\right) + \cdots + \mathrm{vec}\left(\mathbf{b}_n v_n \mathbf{a}_n^T\right) \\
&= \mathrm{vec}\left(\mathbf{b}_1 v_1 \mathbf{a}_1^T + \mathbf{b}_2 v_2 \mathbf{a}_2^T + \cdots + \mathbf{b}_n v_n \mathbf{a}_n^T\right) \\
&= vec\left(\begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_n \end{bmatrix} \begin{bmatrix} v_1 & & & \\ & v_2 & & \\ & & \ddots & \\ & & & v_n \end{bmatrix} \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix}\right).
\end{aligned}$$

□

Definition 3.6. (Hadamard product)

The Hadamard product of two equal-size matrices $A, B \in \mathbb{R}^{m \times n}$ is the element-wise product denoted by $*$ and defined as

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Proposition 3.7. *For matrices of appropriate size, the following properties hold:*

1. $A * B = B * A$
2. $(A * B)^T = A^T * B^T$
3. $A * (B * C) = (A * B) * C$
4. $A * (B + C) = A * B + A * C$
5. $(A \odot B)^T (A \odot B) = A^T A * B^T B$

3.2.2 Singular-Value Decomposition

The most widely known and widely used matrix decomposition is the Singular-Value Decomposition (SVD). It has numerous applications in machine learning and statistics. The SVD can be used in the calculation of matrix operations, such as matrix inverse and data reduction. It can be used in least-squares linear regression, image compression, restoration, and denoising data.

Theorem 3.8. *(SVD) Any $m \times n$ matrix A, with $m \geq n$, can be factorized*

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal, and $\Sigma \in \mathbb{R}^{n \times n}$ is diagonal,

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$$

Proof. See [26]. □

The columns $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ of $\boldsymbol{U}$, which form an orthonormal set, are called left singular vectors.. The columns $v_1, \ldots, v_n$ of $\boldsymbol{V}$, which also form an orthonormal set, are called right singular vectors. The SVD yields a decomposition of $\boldsymbol{A}$ as a sum of n rank-one matrices:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top \tag{3.2}$$

The factorization (3.2) can equivalently be expressed by the equations $\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$ for $i = 1, \ldots, n$. In a general a matrix may have many different SVDs. However the following proposition shows that all SVDs involve the same singular values. Thus we may speak of the singular values of a matrix A.

Proposition 3.9. *Given any SVD of $\boldsymbol{A}$, the singular values are the square roots of the nonzero eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ or $\boldsymbol{A}\boldsymbol{A}^\top$ (these matrices have the same eigenvalues).*

Proof. We show the result for $\boldsymbol{A}^\top\boldsymbol{A}$. Given a SVD $A = U\Sigma\boldsymbol{V}^\top$, we have

$$\begin{aligned} A^\top\boldsymbol{A}\boldsymbol{V} &= \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\right)^\top \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\right)\boldsymbol{V} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^2 \end{aligned}$$

It follows that for $i = 1, \ldots, r$ each right singular vector $\boldsymbol{v}_i$ of $\boldsymbol{A}$ is an eigenvector of $\boldsymbol{A}^\top\boldsymbol{A}$ with non-zero eigenvalue σ_i^2. The remaining columns of $\boldsymbol{V}$ span the eigenspace of $\boldsymbol{A}^\top\boldsymbol{A}$ corresponding to the eigenvalue zero.
Similarly we can show that the left singular values of $\boldsymbol{A}$ are eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$. □

3.2.3 QR Decomposition

Another matrix decomposition is QR decomposition, which is a factorization of a matrix into a product of an orthogonal matrix and a triangular matrix.

Theorem 3.10. *(QR decomposition)*

Any matrix $A \in \mathbb{R}^{m\times n}, m \geq n$, can be transformed to upper triangular form by an orthogonal matrix. The transformation is equivalent to a decomposition

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular. If the columns of A are linearly independent, then R is nonsingular.

Proof. See [26]. □

The QR decomposition can be symbolically illustrated as follows

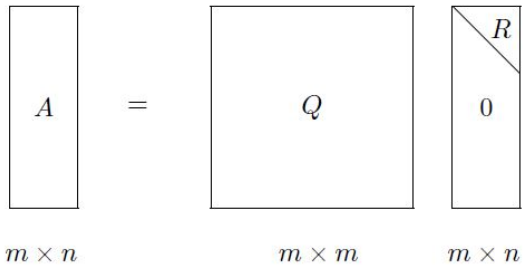


Figure 3.1: *Symbolic illustration of the QR decomposition.*

3.3 Tensor Computation

Tensors are multi-dimensional arrays, which are higher-order generalizations of vectors (first-order tensors) and matrices (second-order tensors). The order of a tensor is the number of dimensions, also known as ways or modes. Tensor can be formally defined as

$$\mathcal{A} = (a_{i_1 i_2 \cdots i_N}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}.$$

If $I_1 = I_2 = \cdots = I_N$, then the tensor $\mathcal{A}$ is called cubic (cubical). An N-th order cubical tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is diagonal if its elements $y_{i_1,i_2,\ldots,i_N} \neq 0$ only if $i_1 = i_2 = \cdots = i_N$. We use $\mathcal{I}$ to denote the cubical identity tensor with ones on the superdiagonal and zeros elsewhere.

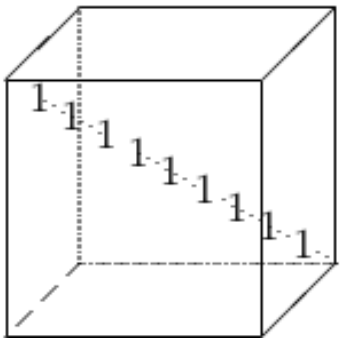


Figure 3.2: *Special forms of third-order tensors: identity tensor.*

A tensor fiber is a one-dimensional fragment of a tensor, obtained by fixing all indices except for one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column, row, and tube fibers, denoted by $\mathcal{A}(:, j, k)$, $\mathcal{A}(i, :, k)$ and $\mathcal{A}(i, j, :)$ respectively (using Matlab notations).

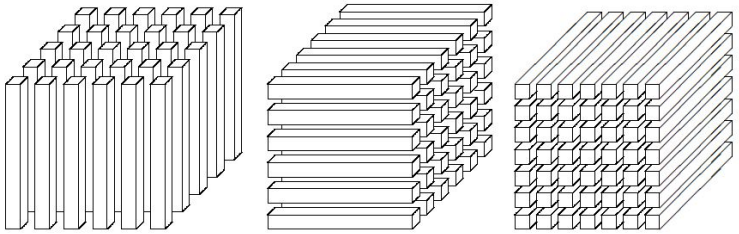


Figure 3.3: *Fibers of a third-order tensor.*

A tensor slice is a two-dimensional section of a tensor, obtained by fixing all indices except for two indices. For example in case of a third-order tensor, we use $\mathcal{A}(i, :, :)$, $\mathcal{A}(:, j, :)$ and $\mathcal{A}(:, :, k)$ to denote horizontal, lateral and frontal slices respectively. $\mathcal{A}(:, :, k)$ is more often denoted as $A^{(k)}$.

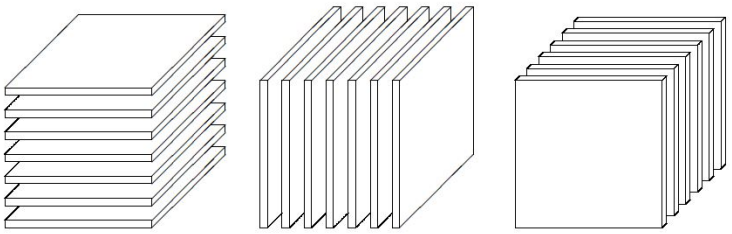


Figure 3.4: *Slices of a third-order tensor.*

Definition 3.11. The inner product of two same size tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is given by

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} \mathcal{A}_{i_1 \cdots i_N} \mathcal{B}_{i_1 \cdots i_N}.$$

It follows immediately that

$$\langle \mathcal{A}, \mathcal{A} \rangle = \| \mathcal{A} \|^2 = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} \mathcal{A}_{i_1 \ldots i_N}^2.$$

This is analogous to the matrix Frobenius norm.

3.3.1 Matricization

Matricization, also known as unfolding or flattening, is the process of reordering the elements of an N-way array into a matrix. For instance, a $2 \times 3 \times 4$ tensor can be arranged as a 6×4 matrix or a 2×8 matrix, etc. A special case of matricization is n-mode matrix defined as follow,

Definition 3.12 ([20, 56])**.**
The n-mode matrix of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathcal{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ and arranges the mode-n fibers into columns of a matrix. More specifically, we have

$$\mathcal{A}_{(n)}(i_n, j) = \mathcal{A}(i_1, i_2, ..., i_N),$$

where $j = 1 + \sum_{k=1, k \neq n}^{N} (i_k - 1) J_k$, and $J_k = \prod_{m=1, m \neq n}^{k-1} I_m$.

Example 3.3.1. Let $X \in \mathbb{R}^{4 \times 3 \times 2}$ then the three n-mode matrix are

$$X_{(1)} = \begin{pmatrix} x_{111} & x_{121} & x_{131} & x_{112} & x_{122} & x_{132} \\ x_{211} & x_{221} & x_{231} & x_{212} & x_{222} & x_{232} \\ x_{311} & x_{321} & x_{331} & x_{312} & x_{322} & x_{332} \\ x_{411} & x_{421} & x_{431} & x_{412} & x_{422} & x_{432} \end{pmatrix}$$

$$X_{(2)} = \begin{pmatrix} x_{111} & x_{211} & x_{311} & x_{411} & x_{112} & x_{212} & x_{312} & x_{412} \\ x_{121} & x_{221} & x_{321} & x_{421} & x_{122} & x_{222} & x_{322} & x_{422} \\ x_{131} & x_{231} & x_{331} & x_{431} & x_{132} & x_{232} & x_{332} & x_{432} \end{pmatrix}$$

and

$$X_{(3)} = \begin{pmatrix} x_{111} & x_{211} & x_{311} & x_{411} & \dots & x_{131} & x_{231} & x_{331} & x_{431} \\ \\ x_{112} & x_{212} & x_{312} & x_{412} & \dots & x_{132} & x_{232} & x_{332} & x_{432} \end{pmatrix}$$

Definition 3.13. The vectorization of matrix $Y \in \mathbb{R}^{I \times T}$ is defined as

$$y = vec(Y) = \left[Y(:,1)^T, Y(:,2)^T, \dots, Y(:,T)^T\right]^T \in \mathbb{R}^{IT}.$$

Analogously, the vectorization of a tensor $\mathcal{Y}$ is defined as the vectorization of the associated 1-mode unfolded matrix $\mathcal{Y}_{(1)}$:

$$vec(\mathcal{Y}) = vec(\mathcal{Y}_{(1)}).$$

3.3.2 Tensor products

Definition 3.14. (n-mode product)
Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ be an N^{th} order tensor and $U \in \mathbb{R}^{J \times I_n}$ be a matrix. Then the n-mode product of $\mathcal{X}$ by U, denoted by $\mathcal{X} \times_n U$, is a tensor of size $\mathrm{I}_1 \times I_2 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ whose entries are given by:

$$(\mathcal{X} \times_n U)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} \mathcal{X}_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} U_{j i_n}$$

The idea can also be expressed in terms of the mode-n matrix:

$$\mathcal{B} = \mathcal{X} \times_n U \Longleftrightarrow \mathcal{B}_{(n)} = U \mathcal{X}_{(n)}$$

Example 3.3.2. Let $\mathcal{X} \in \mathbb{R}^{4\times3\times2}$ be defined by its frontal slices

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} 13 & 17 & 21 \\ 14 & 18 & 22 \\ 15 & 19 & 23 \\ 16 & 20 & 24 \end{bmatrix},$$

And let $\mathbf{U} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$. Then the 2-mode product of $\mathcal{X}$ and $\mathbf{U}$ is

$$\boldsymbol{y} = \mathcal{X} \times_2 \mathrm{U} \in \mathbb{R}^{4\times2\times2}$$

with frontal slices

$$\mathbf{Y}_1 = \begin{bmatrix} 61 & 76 \\ 70 & 88 \\ 79 & 100 \\ 88 & 112 \end{bmatrix}, \quad \mathbf{Y}_2 = \begin{bmatrix} 169 & 220 \\ 178 & 232 \\ 187 & 244 \\ 196 & 256 \end{bmatrix}$$

Proposition 3.15. *For tensors and matrices of appropriate size, the following properties hold:*

1. $\|\mathcal{X}\| = \left\|\mathcal{X}_{(n)}\right\|$
2. $\mathcal{X} \times_n (\mathrm{U} + \mathrm{V}) = \mathcal{X} \times_n \mathrm{U} + \mathcal{X} \times_n \mathrm{V}$
3. *For distinct modes in a series of multiplication, the order of the multiplication is irrelevant, i.e*

   $$\mathcal{X} \times_m U \times_n V = \mathcal{X} \times_n v \times_m U$$

   If the modes are the same, then:

   $$\mathcal{X} \times_n U \times_n V = \mathcal{X} \times_n VU$$

4. $\langle \mathcal{X}, \mathcal{Y} \times_n \mathrm{A} \rangle = \left\langle \mathcal{X} \times_n \mathrm{A}^T, \mathcal{Y} \right\rangle$,
5. *If* U *is an orthonormal matrix, then*

(a) $\mathcal{Y} = X \times {}_n\mathrm{U} \Rightarrow \mathcal{X} = \mathcal{Y} \times {}_n\mathrm{U}^T$,

(b) $\|\mathcal{X}\| = \|\mathcal{X} \times_n \mathrm{U}\|$,

6. $\mathcal{Y} = \mathcal{X} \times_1 \mathrm{A}^{(1)} \times_2 \mathrm{A}^{(2)} \times_3 \cdots \times_N \mathrm{A}^{(N)} \Leftrightarrow$

$$\mathcal{Y}_{(n)} = \mathbf{A}^{(n)} \mathcal{X}_{(n)} \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)} \right)^T.$$

7. $\left\| \mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)} \right\| \leq \|\mathcal{X}\| \left\| \mathbf{A}^{(1)} \right\| \left\| \mathbf{A}^{(2)} \right\| \cdots \left\| \mathbf{A}^{(N)} \right\|$.

Proof.

1. Follows immediately from Definition 3.11.

2. Let $\mathcal{Y} = \mathcal{X} \times_n (\mathrm{U} + \mathrm{V})$, we have

$$\mathcal{Y} = \mathcal{X} \times_n (\mathrm{U} + \mathrm{V}) \Longleftrightarrow \mathcal{Y}_{(n)} = (\mathrm{U} + \mathrm{V}) \mathcal{X}_{(n)} = \mathrm{U}\mathcal{X}_{(n)} + \mathrm{V}\mathcal{X}_{(n)},$$

   which lead to $\mathcal{Y} = \mathcal{X} \times {}_n \mathrm{U} + \mathcal{X} \times {}_n \mathrm{V}$.

3. For distinct n-mode product, we have

$$\begin{aligned}
\mathcal{X} \times_m U \times_n V &= \sum_{i_m} \left(\sum_{i_n} \mathcal{X}_{i_1 \ldots i_n \ldots i_m \ldots i_N} U_{j i_n} \right) V_{k i_m} = \\
&= \sum_{i_n i_m} \mathcal{X}_{i_1 \ldots i_n \ldots i_m \ldots i_N} U_{j i_n} V_{k i_m} = \sum_{i_m i_n} A_{i_1 \ldots i_n \ldots i_m \ldots i_N} V_{k i_m} U_{j i_n} = \\
&= \sum_{i_n} \left(\sum_{i_m} \mathcal{X}_{i_1 \ldots i_n \ldots i_m \ldots i_N} V_{k i_m} \right) U_{j i_n} \\
&= \mathcal{X} \times_n V \times_m U
\end{aligned}$$

   If the modes are the same, we have

$$\begin{aligned}
\mathcal{X} \times_n U \times_n V &= \sum_{i'_n} \left(\sum_{i_n} A_{i_1 \ldots i_n \ldots i_N} U_{i'_n i_n} \right) V_{k i'_{n n}} = \\
&= \sum_{i_n} A_{i_1 \ldots i_n \ldots i_N} \sum_{i'_n} U_{i'_n i_n} V_{k i'_n} = \sum_{i_n} A_{i_1 \ldots i_n \ldots i_N} \sum_{i'_n n} V_{k i'_n} U_{i'_n i_n} = \\
&= \sum_{i_n} A_{i_1 \ldots i_n \ldots i_N} W_{k i_n} = B_{i_1 \ldots k \ldots i_N} \\
&= \mathcal{X} \times_n VU
\end{aligned}$$

4. Follows immediately from Definition 3.11.

5. Let $\mathcal{X}$ be a N-order tensor of size $I_1 \times \cdots \times I_N$ and U is an orthonormal matrix of size $J \times I_n$. Multiplying $\mathcal{Y} = \mathcal{X} \times_n \mathrm{U}$ by U^T, we have

$$\begin{aligned} \mathcal{Y} \times_n U^T &= \mathcal{X} \times_n \mathrm{U} \times_n U^T \\ &= \mathcal{X} \times_n U^T \mathrm{U} \end{aligned}$$

U is orthonormal ($U^T U = I$), which shows (a). On the other hand using orthogonal invariance of matrix Frobenius, we have

$$\|\mathcal{X} \times_n \mathbf{U}\|_F = \left\|\mathbf{U}\mathcal{X}_{(n)}\right\|_F = \left\|\mathcal{X}_{(n)}\right\|_F = \|\mathcal{X}\|_F$$

6. Let $\mathcal{X}$ be $I_1 \times \cdots \times I_N$ tensor and $\mathbf{A}^{(n)} J_n \times I_n$ matrices. By Definition 3.12, each $y_{j_1 \cdots j_N}$ element of $J_1 \times \cdots \times J_N$ tensor $\boldsymbol{y}$ is mapped to $y_{j_n k}$ element of matrix $\mathbf{Y}_{(n)}$, with

$$k = 1 + \sum_{\substack{l=1 \\ l \neq n}}^{N} (j_l - 1) \prod_{\substack{m=1 \\ m \neq n}}^{l-1} J_m$$

We will prove the statement by showing that every element

$$\left(\mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}\right)_{j_1 \cdots j_N}$$

maps to element

$$\left[\mathbf{A}^{(n)} \mathcal{X}_{(n)} \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right)^T\right]_{j_n k},$$

with k as stated. From Definition 3.14, we have

$$\left(\mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}\right)_{j_1 \cdots j_N} = \sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \cdots i_N} a^{(1)}_{j_1 i_1} \cdots a^{(N)}_{j_N i_N}$$

On the other hand, by denoting $\mathbf{M}_n = \left(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}\right)^T$, we have

$$\begin{aligned}\left(\mathbf{A}^{(n)}\mathcal{X}_{(n)}\mathbf{M}_n\right)_{j_n k} &= \mathbf{A}^{(n)}\left[j_n,:\right]\left(\mathcal{X}_{(n)}\mathbf{M}_n\right)[:,k] \\ &= \sum_{i_n=1}^{I_n} a^{(n)}_{j_n i_n}\left(\mathcal{X}_{(n)}\mathbf{M}_n\right)[i_n,k] \\ &= \sum_{i_n=1}^{I_n} a^{(n)}_{j_n i_n}\sum_{i=1}^{\hat{I}_n}\mathcal{X}_{(n)}[i_n,i]\,\mathbf{M}_n[i,k]\end{aligned}$$

with $\hat{I}_n = I_1 \cdots I_{n-1} I_{n+1} \cdots I_N$. Now, $\mathcal{X}_{(n)}[i_n,i] = x_{i_1 \cdots i_N}$, with

$$i = 1 + \sum_{\substack{l=1 \\ l\neq n}}^{N}(i_l - 1)\prod_{\substack{m=1 \\ m\neq n}}^{l-1} I_m.$$

From the definition of Kronecker product follows that the same i stands in

$$\mathbf{M}_n[i,k] = \tilde{a}^{(N)}_{i_N j_N} \cdots \tilde{a}^{(n+1)}_{i_{n+1} j_{n+1}} \tilde{a}^{(n+1)}_{i_{n-1} j_{n-1}} \cdots \tilde{a}^{(1)}_{i_1 j_1},$$

with $\tilde{a}^{(m)}_{i_m j_m}$ denoting an element of $\mathbf{A}^{(m)^T}$. Using these conclusions, we can rewrite (3.6) as

$$\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1 \cdots i_N} a^{(1)}_{j_1 i_1} \cdots a^{(N)}_{j_N i_N},$$

which completes the proof.

7. Follows directly from the previous property.

□

Definition 3.16. (n-mode tensor-vector product)

The n-mode multiplication of a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by a vector $a \in \mathbb{R}^{I_n}$ is denoted by $\mathcal{Y}\overline{\times}_n a$ and has dimension $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$, that is,

$$\mathcal{Z} = \mathcal{Y}\overline{\times}_n a \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N},$$

Element-wise, we have

$$z_{i_1, i_2, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_n} y_{i_1, i_2, \ldots, i_N} a_{i_n}$$

Definition 3.17. (Outer product)

The outer product of the tensors $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{X} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ is given by

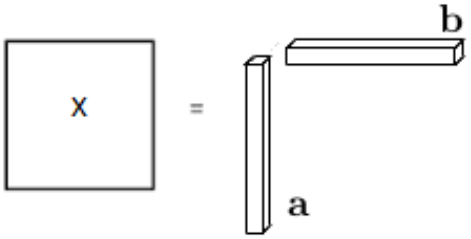
$$\mathcal{Z} = \mathcal{Y} \circ \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \times \cdots \times J_M},$$

with

$$\mathcal{Z}_{i_1,i_2,\ldots,i_N,j_1,j_2,\ldots,j_M} = \mathcal{Y}_{i_1,i_2,\ldots,i_N} \mathcal{X}_{j_1,j_2,\ldots,j_M}.$$

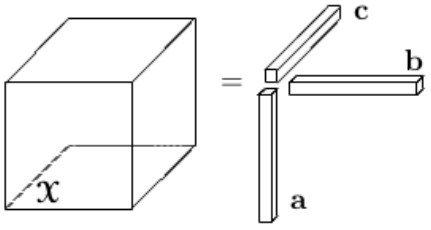
As special cases, the outer product of two vectors $a \in \mathbb{R}^I$ and $b \in \mathbb{R}^J$ yields a rank-one matrix

$$X = a \circ b = ab^T \in \mathbb{R}^{I \times J},$$



and the outer product of three vectors: $a \in \mathbb{R}^I$, $b \in \mathbb{R}^J$, and $c \in \mathbb{R}^Q$ yields a third-order rank-one tensor:

$$\mathcal{Z} = a \circ b \circ c \in \mathbb{R}^{I \times J \times Q}, \quad \text{with} \quad z_{i,j,k} = a_i b_j c_k.$$



Definition 3.18. (Kronecker product)

The Kronecker product of two tensors $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{X} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ is given by

$$\mathcal{Z} = \mathcal{Y} \otimes \mathcal{X} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2 \times \cdots \times I_N J_N},$$

with

$$\mathcal{Z}_{k_1,\ldots,k_N} = \mathcal{Y}_{i_1,\ldots,i_N}\mathcal{X}_{j_1,\ldots,j_N}, \quad k_n = j_n + (i_n - 1)J_n, \quad n = 1,\ldots,N.$$

3.3.3 Notion of rank for tensors

The rank of a matrix is the number of linearly independent column vectors, or, equivalently, the number of non-zero singular values. However, this definition of the matrix rank is not directly extensible to tensors.

Definition 3.19. (Rank-one tensor) An N^{th} order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is rank 1 if it can be written as the outer product of N vectors:

$$\mathcal{A} = v_1 \circ v_2 \circ \cdots \circ v_N \text{ with } v_k \in \mathbb{R}^{i_k}$$

This means that:

$$\mathcal{A}(i_1, i_2, \ldots, i_N) = \sum_{k=1}^{N} v_k(i_k) \text{ for all } 1 \leq i_k \leq I_k$$

$v_k(i_k)$ denotes the i_k^{th} element of vector v_k

Definition 3.20. (Rank of a tensor)

The rank of a tensor is the minimum number of rank one tensors we need to add up to get it.

Definition 3.21. (Tensor n-rank)

The n-rank is defined as the number of linearly independent n-mode fibres of a tensor $\mathcal{A}$:

$$\text{rank}_n(\mathcal{A}) = \text{rank}\left(\mathcal{A}_{(n)}\right)$$

Definition 3.22. (Multi-linear rank)

The multi-linear rank of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a vector, noticed $\text{rank}_{\mathcal{A}}$ and is given by:

$$\text{rank}_{\mathcal{A}} = \left(\text{rank}\left(\mathcal{A}_{(1)}\right), \ldots, \text{rank}\left(\mathcal{A}_{(N)}\right)\right)$$

Where $\text{rank}\left(\mathcal{A}_{(n)}\right)$, for $1 \leq n \leq N$, is the rank of the unfolding matrix $\mathcal{A}_{(n)}$

3.3.4 Tensor decompositions

In this section, we give a brief introduction to higher order decompositions. In particular, we focus on two tensor decompositions, which are CP decomposition that approximate a tensor as sum of rank one tensors, and higher order SVD (HOSVD) decomposition.

3.3.4.1 CP decomposition

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an N^{th}-order tensor. The CP decomposition [20, 41, 51, 56] of $\mathcal{A}$ is given by

$$\mathcal{A} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)},$$

where $a_r^{(k)}$ are vectors of size I_k with $1 \leq k \leq N$ and R is a positive integer.

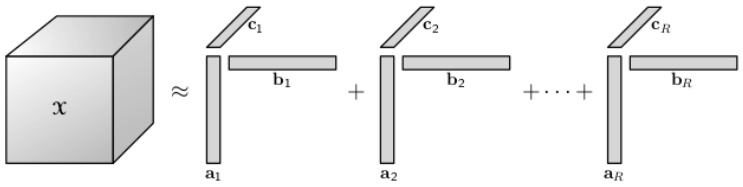


Figure 3.5: *CP decomposition of a third-order tensor.*

A CP decomposition of a tensor $\mathcal{A}$ is called an exact CP decomposition if $R = rank(\mathcal{A})$, with $rank(\mathcal{A})$ [56] represent the rank of the tensor $\mathcal{A}$ defined as the smallest number of rank-one tensors that generate $\mathcal{A}$ as their sum. Unlike matrices, who's the best rank-R approximation is given by the leading R factors of the SVD, the rank of a specific given tensor is hard to define [42]. In practice, the rank of a tensor is determined numerically by fitting various rank-R CP models. But an interesting property associated with CP decomposition for higher-order tensors is uniqueness under some conditions [41, 58]. If we define $A_n = \left[a_1^{(n)} a_2^{(n)} \cdots a_R^{(n)}\right]$ for $n \in \{1, \ldots, N\}$, the CP decomposition can be symbolically written as

$$\mathcal{A} = A_1 \circ A_2 \circ \cdots \circ A_N,$$

the matrices $A_n \in \mathbb{R}^{I_n \times R}$ are called factor matrices. Often, the vectors $a_r^{(n)}$ are chosen such that $\left\| a_r^{(n)} \right\| = 1$. In this case, the CP decomposition is written as

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)},$$

where λ_r is a scalar that compensates for the magnitudes of vectors $a_r^{(n)}$. Using the n-mode multiplication of a tensor by a matrix, we obtain the following representations:

$$\mathcal{A} = \Lambda \times_1 A_1 \times_2 \cdots \times_N A_N,$$

where $\Lambda \in \mathbb{R}^{R \times R \times \cdots \times R}$, with entries

$$\Lambda_{i_1, \cdots, i_N} = \begin{cases} \lambda_r & \text{for} \quad i_1 = i_2 = \cdots = i_N = r, \\ 0 & \text{otherwise.} \end{cases}$$

For a given integer R, there are many algorithms to compute CP decomposition. The most popular approach is to apply the alternating least squares method (ALS) [17, 41, 56].

3.3.4.2 Tucker decomposition

Definition 3.23. A tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is said to be in Tucker format if it can be represented as

$$\mathcal{A} = \mathcal{S} \times_1 \mathrm{U}^{(1)} \times_2 \mathrm{U}^{(2)} \times_3 \cdots \times_N \mathrm{U}^{(N)}$$

were $\mathcal{S} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is a tensor called the core tensor and $\mathrm{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ are matrices called the factor matrices, with $R_n = rank_n(\mathcal{A})$, for $n = 1, 2, \ldots, N$. Element-wise,

$$a_{i_1 i_2 \cdots i_N} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} s_{r_1 r_2 \cdots r_N} u_{i_1 r_1}^{(1)} u_{i_2 r_2}^{(2)} \cdots u_{i_N r_N}^{(N)}.$$

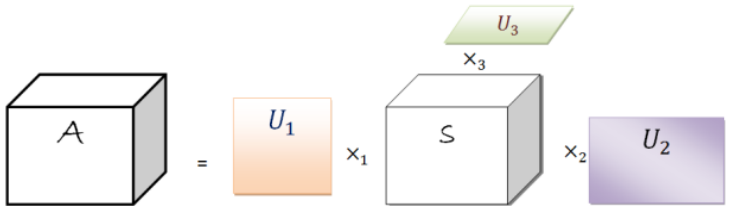


Figure 3.6: *Tucker representation of a third-order tensor.*

There are multiple approaches to compute the Tucker decomposition, the well-known approach is the method known as the higher-order singular value decomposition (HOSVD), that generalize the matrix SVD.

Theorem 3.24.

Any tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ *can be expressed as the product:*

$$\mathcal{A} = \mathcal{S} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times \times_N U^{(N)}$$

With the properties:

1. $U^{(n)} = \left(U_1^{(n)}, U_2^{(n)}, \ldots, U_{I_n}^{(n)}\right) \in \mathbb{R}^{I_n \times I_n}$ *are orthogonal matrices.*
2. *The sub-tensors* $\mathcal{S}_{i_n=\alpha}$ *of* $\mathcal{S} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ *have the following properties:*
   (a) *All-orthogonality: two sub-tensors* $< \mathcal{S}_{i_n=\alpha}, \mathcal{S}_{i_n=\beta} >= 0$ *for all possible values of* n, α *and* β *subject to* $\alpha \neq \beta$.
   (b) *Ordering:* $\|\mathcal{S}_{i_n=1}\| \geq \|\mathcal{S}_{i_n=2}\| \geq \cdots \geq \|\mathcal{S}_{i_n=I_n}\| \geq 0$ *for all possible values of* n.
   The matrix representation of HOSVD can be obtained by unfolding $\mathcal{A}$ *and* $\mathcal{S}$ *:*

$$\mathcal{A}_{(n)} = U^{(n)} \cdot \mathcal{S}_{(n)} \cdot \left(U^{(n+1)} \otimes U^{(n+2)} \otimes \ldots \otimes U^{(N)} \otimes U^{(1)} \otimes U^{(2)} \ldots \otimes U^{(n-1)}\right)$$

The Frobenius-norms $\|\mathcal{S}_{i_n=\alpha}\|$*, symbolized by* $\sigma_\alpha^{(n)}$*, are n-mode singular values of* $\mathcal{A}$ *and the vectors* $U_i^{(n)}$ *are* $i^{th} n-$ *mode singular vectors.*

The idea of the HOSVD method is to unfold the tensor onto each of its modes and perform SVD on each n-mode matrix of the tensor. The left singular vectors will be the factor matrices.

Algorithm 1 HOSVD for computing Tucker decomposition

1: **procedure** HOSVD($\mathcal{A}$)
2: **for** $n = 1, 2, 3, \cdots, N$ **do**
3: $U^{(n)} \longleftarrow$ left singular vectors of $\mathcal{A}_{(n)}$
4: **end for**
5: $\mathcal{S} = \mathcal{A} \times_1 U_1^T \times_2 U_2^T \times_3 \cdots \times_N U_N^T$.
6: return $\mathcal{S}$, $U^{(1)}, \cdots, \mathrm{U}^{(N)}$.
7: **end procedure**

We point out that a rank $(r_1, \cdots, r_N)$ approximation with $r_n \leq R_n$ for $n = 1, \ldots, N$ can be obtained simply by restricting the factor matrices U_n to the first r_n columns (truncated SVD) for $n = 1, \ldots, N$, and by restricting the core tensor $\mathcal{S}$.

Chapter 4

Color image and video restoration using tensor CP decomposition

In this chapter we propose a new approach to image and video restoration. This approach constructs a degradation model based on a tensor representation, where a color image is represented by a third-order tensor, and a video composed of color images is a fourth-order tensor. Applying tensor CP decomposition to our original problem leads to three subproblems. To solve those subproblems, we apply global LSQR algorithm, and a new algorithm based on Golub Kahan bidiagonalization. Some numerical tests are presented to show the effectiveness of the proposed methods.

4.1 Introduction

The field of image restoration is concerned with the problem of undoing the effects of imperfections in the image formation process. More specifically, the goal of image restoration is to remove blur and noise from a degraded image to recover an approximation of the original image. The well-known mathematical model associated with image restoration is formulated as follows

$$g = Kx. \tag{4.1}$$

If the images are assumed to have $m \times n$ pixels, then $g \in \mathbb{R}^{mn}$ and $x \in \mathbb{R}^{mn}$ are vectors that denote the corrupted and the true images respectively, and $K \in \mathbb{R}^{mn \times mn}$ is a matrix that denotes the blurring operator. This model is obtained from discretization of Fredholm integral equations

$$g(u,v) = \int_{\mathbb{R}^2} k(u,v;s,t)x(s,t)\mathrm{d}s\mathrm{d}t + \eta(u,v), \tag{4.2}$$

where k models the blurring operation, called the point spread function (PSF) and η is additive noise. The solution of (4.1) is very sensitive to perturbations, since the matrix K is severely ill-conditioned, with singular values decaying and clustering at 0. Due to the presence of the noise, we cannot solve the problem (4.1) directly. Instead, we use Tikhonov regularization [13, 27, 28] which replaces the minimization problem $\min_x \|Kx - g\|_2^2$ by a problem of the form:

$$\min_x \|Kx - g\|_2^2 + \lambda^2 \|Lx\|_2^2, \tag{4.3}$$

where $\lambda > 0$ is the regularization parameter, and L is the regularization operator. To recover multichannel images from their corrupted observations, we use the RGB representation, by treating each channel; see [38, 79], under the assumption that we have the same within-channel blurring(i.e., the same PSF) in all three channels. The problem of recovering multi-channels images was discussed in several works, such as in [7], where they present approaches to determine a low-rank approximation of the linear system of equations $AX = B$, based on Golub–Kahan bidiagonalization or block Golub–Kahan bidiagonalization.

In this chapter, we discuss the use of the tensor representation of multichannel images seen as third or fourth-order tensors, and the approximation of the blur matrix as a Kronecker product, to construct the degradation model expressed by the following tensor equation

$$\mathcal{X} \times_1 K_c \times_2 K_r = \mathcal{G}, \tag{4.4}$$

where $\mathcal{X}, \mathcal{G} \in \mathbb{R}^{m \times n \times p}$ denote the original image and the degraded image respectively,

and $\times_i$ for $i = 1, 2$ denote the i-mode matrix product [56]. $K_r \in \mathbb{R}^{n\times n}$ and $K_c \in \mathbb{R}^{m\times m}$ define within-channel blurring and they model the horizontal within blurring and the vertical within blurring matrices, respectively. We define an approximate solution $\bar{\mathcal{X}}$ of the tensor equation (4.4) as sum of s rank-one tensors (CP decomposition[41]) which can be written in case of third-order tensor as follows

$$\bar{\mathcal{X}} = \sum_{l=1}^{s} a_l \circ b_l \circ c_l \equiv [\![A, B, C]\!], \tag{4.5}$$

where the symbol $\circ$ denotes the outer product [20], s is a positive integer and $a_l \in \mathbb{R}^m$, $b_l \in \mathbb{R}^n$, $c_l \in \mathbb{R}^p$, for $l = 1, \ldots, s$, are respectively the constituent vectors of the corresponding factor matrices A, B, and C. The advantage of using the CP model is that it is unique under mild assumptions [58], and the number of terms in the right-hand side of the decomposition (4.5) can be decided freely without any restriction. Besides, the interpretation of the matrices A, B, and C is easier than other tensor decomposition such as HOSVD [56] or tensor train decomposition [64] since we deal with matrices instead of tensors. In practice, the best value of s is determined numerically by fitting various rank-s CP models. Based on this approximation, the degradation model becomes

$$[\![K_c A, K_r B, C]\!] = \mathcal{G}. \tag{4.6}$$

We use the alternating least squares (ALS [41]) method to solve (4.6), this leads us to three subproblems of the form $EXF^T = H$. We then use a method based on Golub Kahan bidiagonalization to solve them. Attractive results are obtained for numerical applications.

4.2 Degradation model

Tensors are multi-dimensional arrays, which are higher-order generalizations of matrices and vectors [20, 56]. Tensors provide a natural way to represent multidimensional data whose entries are indexed by several continuous or discrete variables, they can be used in several applications, such as images restoration. For instance, a color image is

represented as a third-order tensor and a video comprised of color images is seen as a fourth-order tensor. We assume the image to be represented by an array of $m \times n$ pixels in each one of the p channels (third-order tensor of size $m \times n \times p$). Let $g^{(i)} \in \mathbb{R}^{mn}$and $x^{(i)} \in \mathbb{R}^{mn}$ for $i = 1, 2, \ldots, p$ represent respectively the i-th channel of the corrupted and the original image. The degradation model is of the form

$$Kx^{(i)} = g^{(i)}, \quad i = 1, 2, \ldots, p, \tag{4.7}$$

where $K \in \mathbb{R}^{mn \times mn}$ represents within channel blurring, which is assumed to be the same in all channels. Using the Kronecker product approximation of the blurring matrix in [48, 62, 61, 76] or by assuming that the PSF is separable (the horizontal and vertical components of the blur are separated) [38], the matrix K given in (4.7) can be decomposed into a Kronecker product of two matrices $K_r = (k^r_{i,j})_{1 \leq i,j \leq n}$ and K_c.

$$K = K_r \otimes K_c = \begin{vmatrix} k^r_{1,1}K_c & \ldots & \ldots & k^r_{1,n}K_c \\ \vdots & & & \\ k^r_{n,1}K_c & \ldots & \ldots & k^r_{n,n}K_c \end{vmatrix}. \tag{4.8}$$

In the non-separable case, one can approximate the matrix K by solving the Kronecker product approximation (KPA) problem [76].

$$(\hat{K}_r, \hat{K}_c) = \underset{K_r, K_c}{\mathbf{arg\,min}} \|K - K_r \otimes K_c\|. \tag{4.9}$$

The degradation model using tensor representation is described in the following proposition

Proposition 4.1. *The blur model associated with a p-channel image is given by the tensor equation*

$$\mathcal{X} \times_1 K_c \times_2 K_r = \mathcal{G}, \tag{4.10}$$

where $\mathcal{X}, \mathcal{G} \in \mathbb{R}^{m \times n \times p}$ denote the original image and the degraded image respectively.

Proof. Using Kronecker product notation, (4.7) can be written as

$$(I_p \otimes K)x = g, \tag{4.11}$$

where x and g are defined by

$$x = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(p)} \end{bmatrix}, \quad g = \begin{bmatrix} g^{(1)} \\ g^{(2)} \\ \vdots \\ g^{(p)} \end{bmatrix},$$

Using (4.8), equation (4.11) can be expressed as

$$(I_p \otimes K_r \otimes K_c)x = g, \tag{4.12}$$

which is equivalent to

$$K_c[\mathcal{X}(:,:,1), \mathcal{X}(:,:,2), \ldots, \mathcal{X}(:,:,p)](I_p \otimes K_r^T) = [\mathcal{G}(:,:,1), \mathcal{G}(:,:,2), \ldots, \mathcal{G}(:,:,p)], \tag{4.13}$$

where $\mathcal{G} \in \mathbb{R}^{m \times n \times p}$ and $\mathcal{X} \in \mathbb{R}^{m \times n \times p}$ are third-order tensors defined by $G^{(i)}$, and $X^{(i)}$ for $i = 1, \ldots, p$, respectively as its frontal slices, with $g^{(i)} = vec(G^{(i)})$ and $x^{(i)} = vec(X^{(i)})$ for $i = 1, \ldots, p$.

From (4.13) and based on the transpose of each frontal slice, we obtain the following equation

$$K_r[\mathcal{X}(:,:,1)^T, \mathcal{X}(:,:,2)^T, \ldots, \mathcal{X}(:,:,p)^T](I_p \otimes K_c^T) = [\mathcal{G}(:,:,1)^T, \mathcal{G}(:,:,2)^T, \ldots, \mathcal{G}(:,:,p)^T]. \tag{4.14}$$

Using the relations

$$[\mathcal{X}(:,:,1)^T, \mathcal{X}(:,:,2)^T, \ldots, \mathcal{X}(:,:,p)^T] = \mathcal{X}_{(2)}, \quad \text{and} \quad [\mathcal{G}(:,:,1)^T, \mathcal{G}(:,:,2)^T, \ldots, \mathcal{G}(:,:,p)^T] = \mathcal{G}_{(2)},$$

equation (4.14) can be expressed as

$$K_r \mathcal{X}_{(2)} (I_p \otimes K_c^T) = \mathcal{G}_{(2)}. \tag{4.15}$$

Applying tensor properties (see [56] p.8), equation (4.15) can be written as

$$(\mathcal{X} \times_1 K_c \times_2 K_r \times_3 I_p)_{(2)} = \mathcal{G}_{(2)}, \tag{4.16}$$

which leads to the equation

$$\mathcal{X} \times_1 K_c \times_2 K_r = \mathcal{G}. \tag{4.17}$$

□

Let us define the linear operator

$$\begin{aligned} \mathcal{M} : \mathbb{R}^{m \times n \times p} &\rightarrow \mathbb{R}^{m \times n \times p} \\ \mathcal{X} &\rightarrow \mathcal{X} \times_1 K_c \times_2 K_r. \end{aligned}$$

Then, the tensor minimization problem becomes

$$\min_{\mathcal{X}} \|\mathcal{M}(\mathcal{X}) - \mathcal{G}\|. \tag{4.18}$$

The general least squares problem $\min_{\mathcal{X}} \left\| \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} - \mathcal{G} \right\|$ was treated in the next chapter, where we work on the coefficient matrices $A^{(i)} (i = 1, \ldots, N)$.

4.2.1 Degradation model using the CP decomposition

The CP decomposition [20, 41, 51, 56] factorizes a tensor into a sum of component rank-one tensors. For example, given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times P}$, we wish to write it as

$$\mathcal{X} = \sum_{l=1}^{s} a_l \circ b_l \circ c_l \equiv [\![A, B, C]\!], \tag{4.19}$$

where the symbol $\circ$ denotes the outer product [20], s is a positive integer and $a_l \in \mathbb{R}^I$, $b_l \in \mathbb{R}^J$, $c_l \in \mathbb{R}^P$, for $l = 1, \ldots, s$ are respectively the constituent vectors of the

corresponding factor matrices A, B, and C. The CP decomposition can be represented in matricized forms by applying unfolding representations of the tensor $\mathcal{X}$:

$$\begin{aligned}\mathcal{X}_{(1)} &\cong A(C \odot B)^T,\\ \mathcal{X}_{(2)} &\cong B(C \odot A)^T,\\ \mathcal{X}_{(3)} &\cong C(B \odot A)^T.\end{aligned}$$

It is often convenient to assume that all vectors have unit length so that we can use the modified Harshman's PARAFAC model given by

$$\mathcal{X} = \sum_{l=1}^{s} \lambda_l a_l \circ b_l \circ c_l \equiv [\![\lambda, A, B, C]\!].$$

Proposition 4.2. *Let $\mathcal{X} = a_1 \circ a_2 \circ \cdots \circ a_N$ be a rank-one tensor and V a set of N matrices $V_1, V_2, \ldots, V_N$. Then we have*

$$\mathcal{X} \times_1 V_1 \times_2 V_2 \cdots \times_N V_N = V_1 a_1 \circ V_2 a_2 \circ \cdots \circ V_N a_N. \tag{4.20}$$

Proof. The proof follows from properties of the n-mode matrix product. □

Using Proposition 4.2, the degradation model given in (4.10) is formulated by

$$[\![K_c A, K_r B, C]\!] = \mathcal{G}.$$

Then, the tensor minimization problem defined in (4.18) becomes

$$\min_{A,B,C} \|\mathcal{G} - [\![K_c A, K_r B, C]\!]\|. \tag{4.21}$$

In the presence of noise, the minimization problem defined in (4.21) is replaced by a regularized problem

$$\min_{A,B,C} \|\mathcal{G} - [\![K_c A, K_r B, C]\!]\| + \alpha_A \|A\| + \alpha_B \|B\| + \alpha_C \|C\|, \tag{4.22}$$

where α_A, α_B, and α_C are nonnegative regularization parameters.

4.3 Alternating Least Squares (ALS)

In order to estimate the factor matrices A, B and C defined in (4.21), we use ALS (Alternating Least Squares) algorithm [20, 22, 41]. Let $\mathcal{T}$ be a third-order tensor defined by the following canonical decomposition

$$\mathcal{T} = [\![K_c A, K_r B, C]\!],$$

we have

$$\begin{aligned} \mathcal{T}_{(1)} &\cong K_c A(C \odot K_r B)^T, \\ \mathcal{T}_{(2)} &\cong K_r B(C \odot K_c A)^T, \\ \mathcal{T}_{(3)} &\cong C(K_r B \odot K_c A)^T. \end{aligned}$$

Applying the unfolding representations of the tensors $\mathcal{T}$ and $\mathcal{G}$. The minimization problem (4.21) is equivalent to the following three expressions

$$\begin{aligned} &\min_{A,B,C} \left\| G_{(1)} - K_c A(C \odot K_r B)^T \right\|, \\ &\min_{A,B,C} \left\| G_{(2)} - K_r B(C \odot K_c A)^T \right\|, \\ &\min_{A,B,C} \left\| G_{(3)} - C(K_r B \odot K_c A)^T \right\|. \end{aligned} \tag{4.23}$$

As a result, instead of solving (4.21) for the three variables one time, we can use the three equations given in (4.23) by fixing all factor matrices but one each time. Thus, given three initial factor matrices A^0, B^0 and C^0, the ALS method solves the three least-squares subproblems in (4.23) to obtain the factor matrices A, B, and C. Starting from the initial guesses A^0, B^0, and C^0, the ALS approach fixes B and C to solve for A, then fixes A and C to solve for B, and then fixes A and B to solve for C. This process continues iteratively until some convergence criterion is satisfied. Therefore, this method translates the original generalistic minimization problem to a three subproblems where each one is just a least-squares problem. The approach is summarized in the following algorithm.

Algorithm 2 ALS-Algorithm

procedure ALS-ALGORITHM($\mathcal{G}$, K_r, K_C, A^0,B^0,C^0)
  for $k = 0, 2, 3, \cdots, M-1$ **do**
    $A^{k+1} = \underset{A}{argmin} \left\| G_{(1)} - K_c A (C^k \odot K_r B^k)^T \right\|$
    $B^{k+1} = \underset{B}{argmin} \left\| G_{(2)} - K_r B (C^k \odot K_c A^{k+1})^T \right\|$
    $C^{k+1} = \underset{C}{argmin} \left\| G_{(3)} - C (K_r B^{k+1} \odot K_c A^{k+1})^T \right\|$
  end for
  return A^M, B^M, C^M.
end procedure

4.4 Solving the problem $EXF^T = H$

The minimization problems given in (4.23) have the following form

$$\min_X \left\| H - EXF^T \right\|, \tag{4.24}$$

where $E \in \mathbb{R}^{l_1 \times l_2}$, $F \in \mathbb{R}^{l_3 \times l_4}$ and $H \in \mathbb{R}^{l_1 \times l_3}$. For example, in the first equation defined in (4.23), we have:

$$E = K_c, \quad F = (C^k \odot K_r B^k), \quad and \quad H = G_{(1)}.$$

The **Global LSQR** is an iterative regularized method that can be used to solve the problem (4.24), this method is a generalization of the classical LSQR [66], which is based on the bidiagonalization procedure of Golub and Kahan. After applying m steps of the Golub-Kahan procedure, the problem is reduced to solving the minimization problem

$$\min_{y_m} ||\beta_1 e_1 - T_m y_m||_2, \tag{4.25}$$

where β_1 is equal to $\|H\|$, $e_1 \in \mathbb{R}^m$ represents the first unit vector, and T_m is a lower bidiagonal matrix. The problem (4.25) is solved by computing the QR factorization of the matrix T_m (e.g using m-Givens rotation). More details concerning Global LSQR can

be found in [47, 67, 71].

In the presence of noise, we introduce Tikhonov regularization method by considering the following minimization problem:

$$\min_{X} \left\| H - EXF^T \right\| + \lambda \left\| LX \right\|, \tag{4.26}$$

where L is the regularization operator, and λ represents the regularization parameter. In case of LSQR method, the Tikhonov regularization is included at each iteration, by replacing the minimization problem given in (4.25) by the following minimization

$$\min_{y_m} ||\beta_1 e_1 - T_m y_m||_2 + \lambda \left\| L y_m \right\|_2. \tag{4.27}$$

This goes through a class of methods called hybrid projection methods that combine iterative projection methods and variational regularization methods (see [18]). The suitable parameter λ can be found using different techniques such as generalized cross validation (GCV) [30, 34, 35], and L-curve criterion [15, 37]. Note that both Chung and Gazzola have done a lot of work on choosing regularization parameters for hybrid methods, e.g.,[19, 31]. If the regularization operator L is equal to the identity matrix, the minimization problem given in (4.27) can be written as

$$\left\| \begin{bmatrix} T_m \\ \lambda I \end{bmatrix} y_m - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \right\|_2. \tag{4.28}$$

Then, the minimization problem (4.27) is equivalent to

$$||\bar{e_1} - \hat{T}_m y_m||_2, \tag{4.29}$$

where $\hat{T}_m = \begin{bmatrix} T_m \\ \lambda I \end{bmatrix}$, $\bar{e_1} = \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix}$.

The problem (4.29) is solved by applying the QR factorization of the matrix $\hat{T}_m$ ($2m$-Givens rotation). To compute a suitable regularization parameter, we will use the discrepancy principle, discussed in [28].

In the next section, we propose a new approach for solving the minimization problem (4.24) based on the Golub-Kahan bidiagonalization for the matrices E and F, instead of using the Golub Kahan bidiagonalization associated with the linear operator $\mathcal{L}(X) = EXF^T$.

4.4.1 Simpler case: $l = rank(H) = 1$

In this section, we assume that the matrix H in equation (4.24) is of rank-one, i.e., it can be written as follows

$$H = h_1 h_2^T.$$

Applying the Golub-Kahan bidiagonalization algorithm (described in Algorithm 3 below) of the matrix E, taking as initial vector h_1, leads to the following relations

$$U_{k+1}(\beta_1 e_1) = h_1, \qquad EZ_k = U_{k+1}B_k, \tag{4.30}$$

where U_{k+1}, $Z_k = [z_1\ z_2 \cdots z_k]$ are orthonormal matrices, $\beta_1 = ||h_1||$ and B_k is a lower bidiagonal matrix. The coefficients $\beta_2, \ldots, \beta_k$ and $\alpha_1, \ldots, \alpha_k$ determined by Algorithm 3 define the lower bidiagonal matrix

$$B_k = \begin{bmatrix} \alpha_1 & & & & \\ \beta_2 & \alpha_2 & & & \\ & \beta_3 & \alpha_3 & & \\ & & \ddots & \ddots & \\ & & & \beta_k & \alpha_k \end{bmatrix}.$$

In the same way, applying the Golub-Kahan bidiagonalization algorithm of the matrix F, taking as initial vector h_2 leads to the following relations

$$Q_{k+1}(\hat{\beta_1} e_1) = h_2, \qquad FV_k = Q_{k+1}C_k, \tag{4.31}$$

where Q_{k+1}, $V_k = [v_1\ v_2 \cdots v_k]$ are orthonormal matrices, $\hat{\beta_1} = ||h_2||$ and C_k is a lower bidiagonal matrix.

Algorithm 3 Golub-Kahan bidiagonalization of the matrix A.

input: matrix $A \in \mathbb{R}^{m \times n}$, initial vector g

$\beta_1 = ||g||_2$

$u_1 = g/\beta_1$

$\hat{v}_1 = A^T u_1$

$\alpha_1 = ||\hat{v}_1||_2$

$v_1 = \hat{v}_1/\alpha_1$

for $k = 2, 3, \cdots$ **do**

    $\hat{u}_k = Av_{k-1} - \alpha_{k-1}u_{k-1}$

    $\beta_k = ||\hat{u}||_2$

    $u_k = \hat{u}_k/\beta_k$

    $\hat{v}_k = A^T u_k - \beta_k v_{k-1}$

    $\alpha_k = ||\hat{v}_k||_2$

    $v_k = \hat{v}_k/\alpha_k$

end for

The method consists of searching for a low-rank approximation of the form

$$X_k = x_k{}^{(1)} x_k{}^{(2)^T} = Z_k Y_k V_k^T, \tag{4.32}$$

which may be written as

$$X_k = Z_k y_k{}^{(1)} y_k{}^{(2)^T} V_k{}^T = x_k{}^{(1)} x_k{}^{(2)^T}, \tag{4.33}$$

where $x_k^{(1)} = Z_k y_k^{(1)}$, and $x_k^{(2)} = V_k y_k^{(2)}$.

Theorem 4.3. *Let* $R_k = H - EX_kF^T$.

Minimizing the residual norm $||R_k||$ *is equivalent to minimizing* $||(\beta_1 e_1)(\hat{\beta_1} e_1^T) - B_k y_k{}^{(1)} y_k{}^{(2)^T} C_k{}^T||$.

Proof. Consider the approximate solution $X_k = Z_k {y_k}^{(1)} {y_k}^{(2)^T} V_k^T$. Then the corresponding residual is given by

$$\begin{aligned} R_k &= H - EX_kF^T, \\ &= h_1h_2^T - EZ_k {y_k}^{(1)} {y_k}^{(2)^T} {V_k}^T F^T, \\ &= U_{k+1}(\beta_1 e_1)(\hat{\beta}_1 e_1^T){Q_{k+1}}^T - U_{k+1}B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T {Q_{k+1}}^T, \\ &= U_{k+1}\left[(\beta_1 e_1)(\hat{\beta}_1 {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T\right] Q_{k+1}^T. \end{aligned}$$

Since U_{k+1} and Q_{k+1} are orthonormal, we get

$$||R_k|| = ||(\beta_1 e_1)(\hat{\beta}_1 {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T||.$$

□

The following minimization problem is accomplished by using the QR factorization of the matrices B_k and C_k.

$$||(\beta_1 e_1)(\hat{\beta}_1 {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T||. \tag{4.34}$$

Thus

$$Q_k^{(1)} B_k = \begin{bmatrix} R_k^{(1)} \\ 0 \end{bmatrix}, \quad Q_k^{(2)} C_k = \begin{bmatrix} R_k^{(2)} \\ 0 \end{bmatrix},$$

and

$$Q_k^{(1)}(\beta_1 e_1) = \begin{pmatrix} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} \end{pmatrix}, \qquad Q_k^{(2)}(\hat{\beta}_1 e_1) = \begin{pmatrix} f_k^{(2)} \\ \bar{\phi}_{k+1}^{(2)} \end{pmatrix},$$

where $Q_k^{(1)}$ and $Q_k^{(2)}$ are the product of k-Givens rotation to eliminate the subdiagonals elements of the matrices B_k and C_k respectively. The minimizer $y_k^{(1)}$ and $y_k^{(2)}$ of (4.34) can then be obtained from

$${f_k}^{(1)} {f_k}^{(2)^T} = {R_k}^{(1)} {y_k}^{(1)} {y_k}^{(2)^T} {R_k}^{(2)^T}, \tag{4.35}$$

which can be split into two subproblems

$$R_k^{(1)} y_k^{(1)} = f_k^{(1)}, \quad \textbf{and} \quad R_k^{(2)} y_k^{(2)} = f_k^{(2)}.$$

Therefore an approximate solution is formed as

$$x_k = {x_k}^{(1)} {x_k}^{(2)^T},$$

where $x_k^{(1)}$ and $x_k^{(2)}$ are obtained from the following relations

$$\begin{cases} x_k^{(1)} &= x_{k-1}^{(1)} + \phi_k^{(1)} d_k^{(1)} \\ x_0^{(1)} &= 0 \end{cases}, \qquad \begin{cases} x_k^{(2)} &= x_{k-1}^{(2)} + \phi_k^{(2)} d_k^{(2)} \\ x_0^{(2)} &= 0 \end{cases},$$

$d_k^{(1)}$ and $d_k^{(2)}$ can be updated by

$$\begin{cases} d_k^{(1)} &= \dfrac{1}{\rho_k^{(1)}} (z_k - d_{k-1}^{(1)} \theta_k^{(1)}) \\ d_0^{(1)} &= 0 \end{cases}, \qquad \begin{cases} d_k^{(2)} &= \dfrac{1}{\rho_k^{(2)}} (v_k - d_{k-1}^{(2)} \theta_k^{(2)}) \\ d_0^{(2)} &= 0 \end{cases}.$$

Proposition 4.4. *The residual norm $||R_k||^2$ can be computed as follows*

$$\|R_k\|^2 = \left\| \bar{\phi}_{k+1}^{(2)} f_k^{(1)} \right\|^2 + \left\| \bar{\phi}_{k+1}^{(1)} f_k^{(2)} \right\|^2 + (\bar{\phi}_{k+1}^{(1)} \bar{\phi}_{k+1}^{(2)})^2.$$

Proof. Using Theorem 4.3 and since $Q_k^{(1)}$ and $Q_k^{(2)}$ are orthonormal. We have

$$\begin{aligned} \|R_k\|^2 &= \left\| (\beta_1 e_1)(\hat{\beta}_1 {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T \right\|^2, \\ &= \left\| {Q_k}^{(1)} \left((\beta_1 e_1)(\hat{\beta}_1 {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T \right) {Q_k}^{(2)^T} \right\|^2, \\ &= \left\| \begin{bmatrix} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} \end{bmatrix} \begin{bmatrix} {f_k}^{(2)^T} & \bar{\phi}_{k+1}^{(2)} \end{bmatrix} - \begin{bmatrix} R_k^{(1)} \\ 0 \end{bmatrix} {y_k}^{(1)} {y_k}^{(2)^T} \begin{bmatrix} {R_k}^{(2)^T} & 0 \end{bmatrix} \right\|^2. \end{aligned}$$

Since $R_k^{(1)} y_k^{(1)} = f_k^{(1)}$, and $R_k^{(2)} y_k^{(2)} = f_k^{(2)}$. We have

$$\begin{aligned} \|R_k\|^2 &= \left\| \begin{bmatrix} 0 & \bar{\phi}_{k+1}^{(2)} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} {f_k^{(2)}}^T & \bar{\phi}_{k+1}^{(1)} \bar{\phi}_{k+1}^{(2)} \end{bmatrix} \right\|^2, \\ &= \left\| \bar{\phi}_{k+1}^{(2)} f_k^{(1)} \right\|^2 + \left\| \bar{\phi}_{k+1}^{(1)} f_k^{(2)} \right\|^2 + (\bar{\phi}_{k+1}^{(1)} \bar{\phi}_{k+1}^{(2)})^2. \end{aligned}$$

□

4.4.2 General case: $l = rank(H) > 1$

In this section, we assume that the matrix H in equation (4.24) is of rank l, i.e, it can be written as follows

$$H = \sum_{i=1}^{l} {h_1}^{(i)} {h_2}^{(i)^T}.$$

We set $H_1 = \left[h_1^{(1)}, h_1^{(2)}, \ldots, h_1^{(l)}\right]$, and $H_2 = \left[h_2^{(1)}, h_2^{(2)}, \ldots, h_2^{(l)}\right]$. In the numerical examples, we compute $H_1$and H_2 by applying the SVD to the matrix H.

Applying the Global Golub-Kahan bidiagonalization to the matrices E and F, starting with H_1 and H_2, leads to the following relations

$$\mathbf{U}_{k+1}((\beta_1 e_1) \otimes I_l) = H_1, \qquad E\mathbf{Z}_k = \mathbf{U}_{k+1}(B_k \otimes I_l), \tag{4.36}$$

$$\mathbf{Q}_{k+1}((\hat{\beta}_1 e_1) \otimes I_l) = H_2, \qquad F\mathbf{V}_k = \mathbf{Q}_{k+1}(C_k \otimes I_l), \tag{4.37}$$

where $\mathbf{U}_{k+1}$, $\mathbf{Z}_k$, $\mathbf{Q}_{k+1}$ and $\mathbf{V}_k$ are F-orthonormal matrices. An approximate solution is given by

$$X_k = \sum_{i=1}^{l} {x_1}^{(i)} {x_2}^{(i)^T} = (\mathbf{Z}_k({y_k}^{(1)} \otimes I_l))(\mathbf{V}_k({y_k}^{(2)} \otimes I_l))^T = {X_k}^{(1)} {X_k}^{(2)^T}, \tag{4.38}$$

where $X_k^{(1)} = \mathcal{Z}_k(y_k^{(1)} \otimes I_l)$, and $X_k^{(2)} = \mathcal{V}_k(y_k^{(2)} \otimes I_l)$.

Lemma 4.5. *Let $\mathbf{S}_k = \left[S^{(1)} S^{(2)} \cdots S^{(k)}\right]$ be an F-orthonormal matrix with $S^{(i)} \in \mathbb{R}^{m \times l}$ for i=1,. . .,k, and let $X \in \mathbb{R}^{l.k \times n}$ and $Y \in \mathbb{R}^{k \times k}$. We have the following relations:*

$$\|\mathbf{S}_k X\| \leq \|X\|, \tag{4.39}$$

$$\|\mathbf{S}_k(Y \otimes I_l)\| = \|Y\|. \tag{4.40}$$

Proof. We have

$$\begin{aligned}\|\mathbf{S}_k X\|^2 &= \sum_{i=1}^{m}\sum_{j=1}^{n}(\mathbf{S}_k X)_{ij}^2, \\ &= \sum_{j=1}^{n}\left\|(\mathbf{S}_k X)_{:j}\right\|^2,\end{aligned}$$

with $(\mathbf{S}_k X)_{ij} = \sum_{p=1}^{k.n}(\mathbf{S}_k)_{ip}X_{pj} = \sum_{r=1}^{k} S_{i:}^{(r)} x_{rj}$, and x_{rj} is defined using Matlab notation as follows $x_{rj} = X((r-1)l+1 : rl, j)$. As consequence

$$(\mathbf{S}_k X)_{:j} = \sum_{r=1}^{k} S^{(r)} x_{rj},$$

which leads to

$$\begin{aligned}\|\mathbf{S}_k X\|^2 &= \sum_{j=1}^{n}\left\|\sum_{r=1}^{k} S^{(r)} x_{rj}\right\|^2 \\ &\leq \sum_{j=1}^{n}\sum_{r=1}^{k}\left\|x_{rj}\right\|^2 \\ &\leq \|X\|^2.\end{aligned}$$

Therefore (4.39) is achieved. On the other hand, we have:

$$(Y \otimes I_l)_{:p} = Y_{:i} \otimes (I_l)_{:j} \quad \textit{with } p = j + (i-1)l \textit{ for } i = 1, \ldots, k, \textit{ and } j = 1, \ldots, l.$$

Then

$$\begin{aligned}\|\mathbf{S}_k(Y \otimes I_l)\|^2 &= \sum_{p=1}^{l.k}\left\|(\mathbf{S}_k(Y \otimes I_l))_{:p}\right\|^2 \\ &= \sum_{j=1}^{l}\sum_{i=1}^{k}\left\|\sum_{r=1}^{k} S^{(r)} Y_{ri} \otimes (I_l)_{:j}\right\|^2 \\ &= \sum_{j=1}^{l}\sum_{i=1}^{k}\left\|\sum_{r=1}^{k} Y_{ri} S^{(r)} \otimes (I_l)_{:j}\right\|^2.\end{aligned}$$

Since $\mathbf{S}_k$ is F-orthonormal matrix. we obtain

$$\|\mathbf{S}_k(Y \otimes I_l)\|^2 = \sum_{i=1}^{k}\sum_{r=1}^{k} Y_{ri}^2 = \|Y\|.$$

This proves (4.40).

Theorem 4.6. *Let $R_k = H - EX_kF^T$. Then minimizing the residual norm $||R_k||$ is equivalent to minimizing $\left\|(\beta_1 e_1)(\hat{\beta_1} {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T\right\|$.*

Proof. Consider the approximate solution $X_k = (\mathbf{Z}_k({y_k}^{(1)} \otimes I_l))(\mathbf{V}_k({y_k}^{(2)} \otimes I_l))^T$. Then the associated residual is given by

$$\begin{aligned} R_k &= H - EX_kF^T, \\ &= H_1 {H_2}^T - E(\mathbf{Z}_k({y_k}^{(1)} \otimes I_l))(\mathbf{V}_k({y_k}^{(2)} \otimes I_l))^T F^T, \\ &= \mathbf{U}_{k+1}\left[((\beta_1 e_1)(\hat{\beta_1} e_1^T) \otimes I_l) - (B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T \otimes I_l)\right] {\mathbf{Q}_{k+1}}^T. \end{aligned}$$

Since $\mathbf{U}_{k+1}$ and $\mathbf{Q}_{k+1}$ are F-orthonormal, by applying the relations in Lemma 4.5, we get

$$\|R_k\| \leq \left\|(\beta_1 e_1)(\hat{\beta_1} {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T\right\|.$$

□

The following minimization problem is accomplished by using the QR factorization of the matrices B_k and C_k (see section 4.4.1).

$$\left\|(\beta_1 e_1)(\hat{\beta_1} {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T\right\|. \tag{4.41}$$

□

The algorithm can be summarized as follows

Algorithm 4

input: matrix $E \in \mathbb{R}^{I_1 \times I_2}$, $F \in \mathbb{R}^{I_3 \times I_4}$, $H \in \mathbb{R}^{I_1 \times I_3}$, r

Decompose H as $H = H_1 H_2^T$ where $H_1 \in \mathbb{R}^{I_1 \times r}$, $H_2 \in \mathbb{R}^{I_3 \times r}$

Compute ${\beta_1}^{(1)} = ||H_1||, {U_1}^{(1)} = H_1/{\beta_1}^{(1)}, {\alpha_1}^{(1)} = ||E^T {U_1}^{(1)}||$ and ${V_1}^{(1)} = E^T {U_1}^{(1)}/{\alpha_1}^{(1)}$

Compute ${\beta_1}^{(2)} = ||H_2||, {U_1}^{(2)} = H_2/{\beta_1}^{(2)}, {\alpha_1}^{(2)} = ||F^T {U_1}^{(2)}||$ and ${V_1}^{(2)} = F^T {U_1}^{(2)}/{\alpha_1}^{(2)}$

Set ${W_1}^{(l)} = {V_1}^{(l)}$, $\bar{\phi}_1^{(l)} = {\beta_1}^{(l)}$, $\bar{\rho}_1^{(l)} = {\alpha_1}^{(l)}$, $n_0^{(l)} = 0$, for $l = 1, 2$

for $i = 1, 2, \ldots, itermax$ **do**

$\bar{W}_i^{(1)} = E V_i^{(1)} - \alpha_i^{(1)} U_i^{(1)}, \quad \bar{W}_i^{(2)} = F V_i^{(2)} - {\alpha_i}^{(2)} U_i^{(2)}$

for $l = 1, 2$ **do**

${\beta_{i+1}}^{(l)} = ||\bar{W}_i^{(l)}||$

${U_{i+1}}^{(l)} = \bar{W}_i^{(l)}/{\beta_{i+1}}^{(l)}$

$\bar{S}_i^{(l)} = E^T {U_{i+1}}^{(l)} - \beta_{i+1}^{(l)} V_i^{(l)}$

$\alpha_{i+1}^{(l)} = ||\bar{S}_i^{(l)}||$

${V_{i+1}}^{(l)} = \bar{S}_i^{(l)}/{\alpha_{i+1}}^{(l)}$

$\rho_i{}^{(l)} = \sqrt{\left(\bar{\rho}_i^{(l)}\right)^2 + \left({\beta_{i+1}}^{(l)}\right)^2}$

$c_i{}^{(l)} = \bar{\rho}_i^{(l)}/\rho_i{}^{(l)}$

$s_i{}^{(l)} = {\beta_i}^{(l)}/\rho_i{}^{(l)}$

$\theta_{i+1}{}^{(l)} = s_i{}^{(l)} {\alpha_{i+1}}^{(l)}$,

$\bar{\rho}_{i+1}^{(l)} = c_i{}^{(l)} {\alpha_{i+1}}^{(l)}$

$\phi_i{}^{(l)} = c_i{}^{(l)} \bar{\phi}_i^{(l)}$

$\bar{\phi}_{i+1}^{(l)} = s_i^{(l)} \bar{\phi}_i^{(l)}$

$X_i^{(l)} = X_{i-1}^{(l)} + \frac{\phi_i^{(l)}}{\rho_i^{(l)}} W_i^{(l)}$

$W_{i+1}^{(l)} = V_{i+1}^{(l)} - \frac{\phi_{i+1}^{(l)}}{\rho_i^{(l)}} W_i^{(l)}$

end for

$X_i = {X_i}^{(1)} {X_i}^{(2)^T}$

$n_i{}^{(l)} = {n_{i-1}}^{(l)} + {\phi_i}^{(l)^2}$, for $l = 1, 2$

If $\sqrt{\bar{\phi}_{i+1}^{(1)2} n_i{}^{(2)} + \bar{\phi}_{i+1}^{(2)2} n_i{}^{(1)} + \bar{\phi}_{i+1}^{(1)2} \bar{\phi}_{i+1}^{(2)2}}$ is small enough then stop

end for

Remark 4.4.1. In line 26 of Algorithm 4, we compute an upper bound of the residual norm $||R_k||$ using Proposition 4.4, where $n_i^{(1)}$ and $n_i^{(2)}$ denote $||f_i^{(1)}||^2$ and $||f_i^{(2)}||^2$ respectively.

In the presence of noise, the minimization problem defined in (4.34) is replaced by a regularized problem

$$\min_{y_k{}^{(1)}, y_k{}^{(2)}} \left\| (\beta_1 e_1)(\hat{\beta_1} {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} {C_k}^T \right\| + \lambda \left\| L_1 {y_k}^{(1)} {y_k}^{(2)^T} {L_2}^T \right\|, \tag{4.42}$$

where L_1 and L_2 are the regularization operators, and λ is the regularization parameter. Which is equivalent to the following minimization problem

$$\min_{y_k^{(1)}, y_k^{(2)}} ||(\beta_1 \hat{\beta_1}) e_1 \otimes e_1 - (C_k \otimes B_k)(y_k^{(2)} \otimes y_k^{(1)})||_2 + \lambda ||(L_2 \otimes L_1) y_k^{(2)} \otimes y_k^{(1)}||_2, \tag{4.43}$$

which can be written as

$$\min_{y_k^{(1)}, y_k^{(2)}} \left\| \begin{bmatrix} C_k \otimes B_k \\ \lambda L_2 \otimes L_1 \end{bmatrix} y_k^{(2)} \otimes y_k^{(1)} - \begin{bmatrix} (\beta_1 \hat{\beta_1}) e_1 \otimes e_1 \\ 0 \end{bmatrix} \right\|_2. \tag{4.44}$$

If we consider the particular case where the matrix L_2 reduces to the identity I_k, and $L_1 = B_k$. The minimization problem (4.44) can be written as

$$\min_{y_k^{(1)}, y_k^{(2)}} \left\| \left(\begin{bmatrix} C_k \\ \lambda I_k \end{bmatrix} \otimes B_k \right) (y_k^{(2)} \otimes y_k^{(1)}) - \begin{bmatrix} (\beta_1 \hat{\beta_1}) e_1 \otimes e_1 \\ 0 \end{bmatrix} \right\|_2, \tag{4.45}$$

which is equivalent in matrix form to

$$\min_{y_k{}^{(1)}, y_k{}^{(2)}} \left\| (\beta_1 e_1)(\hat{\beta_1} {e_1}^T) - B_k {y_k}^{(1)} {y_k}^{(2)^T} \hat{C_k}^T \right\|, \tag{4.46}$$

where $\hat{C_k} = \begin{bmatrix} C_k \\ \lambda I_k \end{bmatrix}$.

The minimization problem (4.46) is accomplished by using the QR factorization of the matrices B_k and $\hat{C}_k$,

$$Q_k^{(1)} B_k = \begin{bmatrix} R_k^{(1)} \\ 0 \end{bmatrix}, \quad Q_k^{(1)}(\beta_1 e_1) = \begin{pmatrix} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} \end{pmatrix}, \tag{4.47}$$

and

$$Q_k^{(2)} \hat{C}_k = Q_k^{(2)} \begin{bmatrix} C_k \\ \lambda I_k \end{bmatrix} = \begin{bmatrix} R_k^{(2)} \\ 0 \end{bmatrix} \qquad Q_k^{(2)}(\hat{\beta}_1 e_1) = \begin{pmatrix} f_k^{(2)} \\ \bar{\phi}_{k+1}^{(2)} \\ g_k \end{pmatrix}. \tag{4.48}$$

The QR factorization (4.48) is formed similarly to the case $\lambda = 0$, except that two rotations are needed instead of one [65].

Proposition 4.7. *An upper bound of the residual norm $||R_k||^2$ is given as follows*

$$\|R_k\|^2 \leq \|\bar{g}_k\|^2 \left\|f_k^{(1)}\right\|^2 + \left\|\bar{\phi}_{k+1}^{(1)} f_k^{(2)}\right\|^2 + \left\|\bar{\phi}_{k+1}^{(1)} \bar{g}_k\right\|^2,$$

where $\bar{g}_k = \begin{pmatrix} \bar{\phi}_{k+1}^{(2)} \\ g_k \end{pmatrix}$.

Proof. Similar to the proof of Proposition 4.4. □

4.4.3 A parameter selection method

For an appropriate selection of the regularization parameter λ appearing in (4.43), we suggest to use the GCV method [13, 34]. The GCV method is a well-known method for computing the regularization parameter λ, the regularization parameter λ_{GCV} is chosen to minimize the GCV function.

$$GCV(\lambda) = \frac{||Kf_\lambda - g||_2^2}{(trace(I - KK_\lambda^{-1}K^T))^2},$$

where $K_\lambda = K^T K + \lambda L^T L$, and f_λ is solution of $K_\lambda f = K^T g$.

Let $K = (C_k \otimes B_k)$ and $L = L_2 \otimes L_1$, the GCV function can be simplified for Tikhonov regularization method using the generalized singular value decompositions (GSVD) of

the pairs (C_k, L_2) and (B_k, L_1). Thus, there exist orthogonal matrices U_1, U_2, V_1, V_2 and invertible matrices W_1, W_2 such that

$$\begin{aligned} U_1^T C_k W_1 = C_1 &= \text{diag}(c_{1,1}, \cdots, c_{k,1}), \\ U_2^T B_k W_2 = C_2 &= \text{diag}(c_{1,2}, \cdots, c_{k,2}), \end{aligned}$$

and

$$\begin{aligned} V_1^T L_1 W_1 = S_1 &= \text{diag}(s_{1,1}, \cdots, s_{k,1}), \\ V_2^T L_2 W_2 = S_2 &= \text{diag}(s_{1,2}, \cdots, s_{k,2}). \end{aligned}$$

Then the GSVD of the pair (K, L) is given by

$$\begin{aligned} U^T K W_{=} C &= \text{diag}(c_1, \cdots, c_N), \\ V^T L W = S &= \text{diag}(s_1, \cdots, s_N), \end{aligned}$$

where $U = U_2 \otimes U_1$, $V = V_2 \otimes V_1$, $C = C_2 \otimes C_1$, $S = S_2 \otimes S_1$.

4.5 Numerical examples

In this section, we provide some numerical results that illustrate the performance of the two algorithms for solving the problem given in (4.24), with application to image and video restoration.

To determine the effectiveness of our method, we evaluate the relative error:

$$\text{Relative error} = \frac{||\mathcal{X}^{(k)} - \mathcal{X}_{true}||}{||\mathcal{X}_{true}||},$$

where $\mathcal{X}^{(k)}$ denotes the approximate restoration. In addition, we evaluate the Signal-to-Noise Ratio (SNR) defined by

$$\text{SNR} = 10\log_{10} \frac{||\mathcal{X}_{true} - E(\mathcal{X}_{true})||^2}{||\mathcal{X}^{(k)} - \mathcal{X}_{true}||^2},$$

where $E(\mathcal{X}_{true})$ denotes the mean gray-level of the uncontaminated image $\mathcal{X}_{true}$. All experiments are performed on an AMD Ryzen 7 4700U 2 GHz with 16 Ram using Matlab 2013*a*.

4.5.1 Example 1

In the first example, we present the numerical results recovered by Algorithm 4 and global LSQR when applied to the restoration of gray-scale and color images that have been contaminated by Defocus blur, whose entries are given by:

$$k(i,j) = \begin{cases} \frac{1}{\pi r^2}, \text{if} \sqrt{i^2+j^2} \leq r \\ 0, \quad \text{otherwise} \end{cases},$$

with $r = 3$. In the case of gray-scale images, the blurred image has been built by the product $K_c X K_r^T$. Table 4.1 compares the computing time (in seconds) and the relative error of the computed restorations for the max number of iterations (Iter) showed in the table.

Table 4.1: *Results for Example 1*

Size	Method	Iter	Relative error	CPU-time(sec)
128×128	Global LSQR	30	5.5×10^{-2}	0.1
	Algorithm 4	20	2.45×10^{-2}	0.09
400×318	Global LSQR	40	2.74×10^{-2}	0.78
	Algorithm 4	30	1.0×10^{-2}	0.71

The next table shows the results obtained from the restoration of color images with Algorithm 2, using Algorithm 4 to solve the three subproblems, or using global LSQR denoted respectively by Algorithm 2(Algo 3), and Algorithm 2(G-LSQR), with $s = 250$. We point out that in the next table, and in all the tables below, “Iter” stands for the number of iterations and has the following form Iter1(iter2)", where "Iter1" indicates the number of iterations of Algorithm 2, and "iter2" indicates the number of iterations of Algorithm 4 or global LSQR.

Table 4.2: *Results for Example 1*

Size	Method	Iter	Relative error	CPU-time(sec)
$227 \times 303 \times 3$	Algorithm 2(G-LSQR)	4(30)	4.32×10^{-2}	4.79
	Algorithm 2(Algo 3)	5(20)	3.1×10^{-2}	3.64
$384 \times 512 \times 3$	Algorithm 2(G-LSQR)	4(30)	6.07×10^{-2}	12.2
	Algorithm 2(Algo 3)	5(20)	5.71×10^{-2}	9.72

4.5.2 Example 2

This example illustrates the performance of Algorithm 2(Algo 3) applied to the restoration of a 3-channel RGB color image that has been contaminated by blur and noise (generated by Matlab's randn function) with different noise levels. This noise level is defined as follows $\nu = \frac{\|\mathcal{E}\|}{\|\hat{\mathcal{G}}\|}$ where $\mathcal{E}$ is a tensor that represents the noise in $\mathcal{G}$, i.e., $\mathcal{G} := \hat{\mathcal{G}} + \mathcal{E}$, and $\hat{\mathcal{G}}$ is the noise-free image associated with the original image $\mathcal{X}$. In this example, we set $s = 180$ fixed for noise levels $\nu = 10^{-3}$ and $\nu = 5 \times 10^{-3}$, and $s = 250$ for noise level $\nu = 10^{-2}$. The true image of size $384 \times 512 \times 3$ is blurred by Gaussian blur of size 3×3 with deviation $\sigma = 2$, or by motion blur of length 21 and angle equal to 0. For comparison with the existing approaches in the literature, we compare our approach with the methods proposed in [25]. We refer to these methods as T-global GMRES and T-global Golub Kahan. These algorithms represent the tensor version of GMRES and Golub Kahan bidiagonalization using the T-product, when they are applied for solving the problem arising from recovering multichannel images that have the following form

$$\mathcal{K}_c * \mathcal{X} * \mathcal{K}_r^T = \mathcal{G}, \tag{4.49}$$

where $\mathcal{X}, \mathcal{G} \in \mathbb{R}^{m \times n \times p}$ denote the original image and the degraded image respectively, $\mathcal{K}_c \in \mathbb{R}^{m \times m \times p}$, and $\mathcal{K}_r \in \mathbb{R}^{n \times n \times p}$ such that $\mathcal{K}_c(:,:,1) = K_c, \quad \mathcal{K}_c(:,:,i) = 0$, for $i = 2, \ldots, p$ and $\mathcal{K}_r(:,:,1) = K_r$, $\mathcal{K}_r(:,:,i) = 0$, for $i = 2, \ldots, p$.

(a) (b) (c)

Figure 4.1: *Example 2. (a) Exact image, (b) Degraded image(motion blur + noise $\nu = 10^{-3}$), Restored images by (c) Algorithm 2(Algo 3)*

The following table compares the computing CPU-time, the relative errors and the SNR of the computed restorations for a fixed number of iterations.

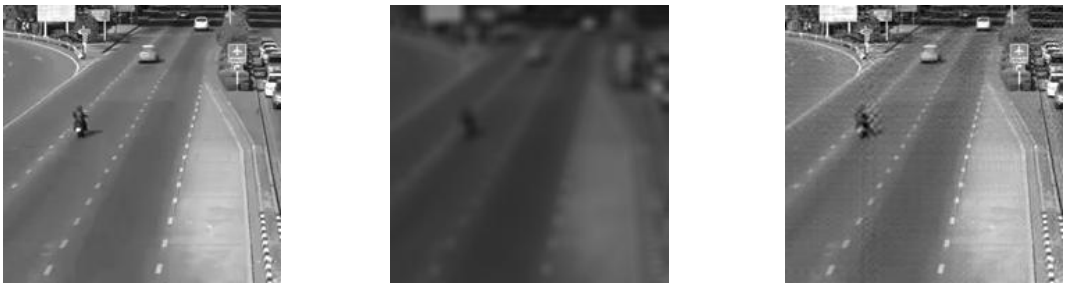
Table 4.3: *Comparison of image restoration on image 'peppers' blurred by Gaussian blur*

Noise level	Method	Iter	SNR	Relative error	CPU-time(sec)
0.001	T-global GMRES	10(m=10)	21.6	4.91×10^{-2}	20.89
	T-global Golub Kahan	15	20.44	5.55×10^{-2}	8.21
	Algorithm 2(Algo 3)	3(25)	21.86	4.81×10^{-2}	6.3
0.005	T-global GMRES	10(m=10)	19.86	6.20×10^{-2}	21.89
	T-global Golub Kahan	6	18.17	7.37×10^{-2}	8.05
	Algorithm 2(Algo 3)	4(20)	20.45	5.66×10^{-2}	7.66
0.01	T-global GMRES	10(m=10)	19.61	6.24×10^{-2}	21.8
	T-global Golub Kahan	8	18.2	7.34×10^{-2}	4
	Algorithm 2(Algo 3)	4(25)	20.47	5.65×10^{-2}	10.7

Based on the testes reported in Table 4.3 and many unstated tests, we remark that our proposed algorithm works effectively for image restoration problems both in terms of the SNR and the relative error. In terms of the CPU-time, we note that T-global Golub Kahan is faster than the other two algorithms stated in 4.3. We point out that when we increase the number of iterations for the T-global Golub Kahan algorithm, for noise levels $\nu = 10^{-2}$ and $\nu = 5 \times 10^{-3}$, the associated algorithm fails to converge.

4.5.3 Example 3

Video processing is not a continuous process but a discrete one. That means each time we deal with videos, we are dealing with the sequence of frames themselves, each frame is just an image, which might be represented as an $m \times n$ array of pixels in case of gray-scale videos, and $m \times n \times 3$ array in case of color videos. In this example, we evaluate the effectiveness of our algorithm applied to the restoration of a grayscale video of size $200 \times 200 \times 20$ that have been contaminated by both blur (Gaussian blur of size 9×9 with deviation $\sigma = 4$) and additive noise with noise level equal to 10^{-2}. The following figure shows the third frame associated with the true, the blurred, and the restored video.



(a) (b) (c)

Figure 4.2: *Example 3: Frame no. 3: (a) Original frame, (b) Blurred and noisy frame, (c) Restored frame by Algorithm2(Algo 3)*

The following table shows the resulting relative error of the previous example and time of execution

Table 4.4: *Results for Example 3*

s	Iter	Relative error	CPU-time(sec)
100	5(40)	6.78×10^{-2}	10.9
200	5(40)	5.99×10^{-2}	14.82
300	4(40)	5.77×10^{-2}	11.74

4.6 Conclusion

In this chapter, we have proposed a new approach to restore multichannel images, by using its tensor representation to construct the blurring model (4.10). To solve the problem (4.18), we define an approximate solution in CP decomposition format that leads to three subproblems (4.23). We use global LSQR algorithm and a new algorithm based on Golub Kahan's bidiagonalization to solve those subproblems. Numerical examples show that our approaches lead to satisfactory results when applied to image and video restoration.

Chapter 5

The LSQR method for solving tensor least squares problem

In this chapter, we are interested in finding an approximate solution $\hat{\mathcal{X}}$ of the tensor least squares minimization problem $\min_{\mathcal{X}} \left\| \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} - \mathcal{G} \right\|$ where $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ and $A^{(i)} \in \mathbb{R}^{J_i \times I_i}$ $(i = 1, \ldots, N)$ are known, and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the unknown tensor to be approximated. Our approach is based on two steps. Firstly, we apply the CP or HOSVD decomposition to the right-hand side tensor $\mathcal{G}$. Secondly, we perform the well-known Golub-Kahan bidiagonalization to each coefficient matrix $A^{(i)}(i = 1, \ldots, N)$ to obtain a reduced tensor least squares minimization problem. This type of equations may appear in color image and video restorations as we described below. Some numerical tests are performed to show the effectiveness of our proposed method.

5.1 Introduction

The LSQR algorithm of Paige and Sanders [66] is one of the most efficient algorithms for solving the following problem

$$Ax = b,$$

or the linear least squares minimization,

$$\min_x \|Ax - b\|_2,$$

where A is a matrix of size $m \times n$ and b is a vector of size m. The LSQR method is analytically equivalent to the conjugate gradient method applied to the associated normal equation. The LSQR algorithm is based on the bidiagonalization procedure of Golub and Kahan [33]. In the few past years, many researches have generated the LSQR algorithm for solving various equations. For instance, in [71] the authors proposed a global version of LSQR (GL-LSQR) to obtain an approximate solution of the matrix equation $AX = B$ with $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times s}$. Recently, a Golub-Kahan bidiagonalization based on tensor format is presented in [5, 49] to solve the following tensor equation

$$\mathcal{A}(\mathcal{X}) = \mathcal{B},$$

with $\mathcal{X}$ is a tensor of size $n_d \times n_{d-1} \times \cdots \times n_1$, and $\mathcal{A}$ is a linear operator defined by

$$\begin{aligned} \mathcal{A} : \mathbb{R}^{n_d\times n_{d-1}\times\cdots\times n_1} &\longrightarrow \mathbb{R}^{n_d\times n_{d-1}\times\cdots\times n_1} \\ \mathcal{X} &\longrightarrow \mathcal{A}(\mathcal{X}) = \sum_{i=1}^{I} \mathcal{X} \times_1 A_{i,d} \times_2 A_{i,d-1} \times_3 \cdots \times_d A_{i,1}. \end{aligned}$$

Here, $\times_i$ for $i = 1, \ldots, d$ denote the i-mode product defined below. In those references, the authors used Golub-Kahan bidiagonalization to the linear operator $\mathcal{A}$. For an extensive survey on the subject of higher-order tensors we refer to [20, 56].

This chapter is concerned with the numerical solution of a tensor least squares problem of the form:

$$\min_{\mathcal{X}} \left\| \mathcal{L}(\mathcal{X}) - \mathcal{S} \times_1 G^{(1)} \times_2 G^{(2)} \times_3 \cdots \times_N G^{(N)} \right\|, \tag{5.1}$$

where $\mathcal{L}$ is a linear tensor operator, defined by

$$\begin{aligned} \mathcal{L} : \mathbb{R}^{I_1\times I_2\times\cdots\times I_N} &\longrightarrow \mathbb{R}^{J_1\times J_2\times\cdots\times J_N} \\ \mathcal{X} &\longrightarrow \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)}, \end{aligned}$$

where $\mathcal{S} \in \mathbb{R}^{m_1\times m_2\times\cdots\times m_N}$, $G^{(i)} \in \mathbb{R}^{J_i\times m_i}$ and $A^{(i)} \in \mathbb{R}^{J_i\times I_i}$ $(i = 1, \ldots, N)$ are known, and

$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the unknown tensor to be approximated. We point out that our approach can be generally applied to solve the least square problem:

$$\min_{\mathcal{X}} \|\mathcal{L}(\mathcal{X}) - \mathcal{G}\|, \tag{5.2}$$

for an arbitrary right-hand side tensor $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, by decomposing the tensor $\mathcal{G}$ using CP [41] or Tucker decomposition [73, 74] better known as the higher-order SVD (HOSVD) [24]. It is easy to verify that if the right-hand side tensor $\mathcal{G}$ is written in CP or HOSVD format, then the solution $\mathcal{X}$ can also be written in CP or HOSVD format. The tensor least squares minimization problem (5.2) is a generalization of the equations arise in color image and video restoration which we discuss in section 5.5. It is not difficult to verify that (5.2) is equivalent to the following minimization problem:

$$\min_{\mathcal{X}} \|\mathbb{A}vec(\mathcal{X}) - vec(\mathcal{G})\|,$$

with $\mathbb{A} = A^{(N)} \otimes \cdots \otimes A^{(2)} \otimes A^{(1)}$, where $\otimes$ denotes the Kronecker product. The operator '*vec*' stacks the columns of a matrix or tensor to form a vector. If $I_i = J_i$, for $i = 1, \ldots, N$, the eigenvalues of the matrix $\mathbb{A}$ arise as a product of eigenvalues of the matrices $A^{(i)}(i = 1, \ldots, N)$. The set of the spectrum of $\mathbb{A}$ denoted by $\lambda(\mathbb{A})$ is given by (see [45]).

$$\lambda(\mathbb{A}) = \{\lambda_1 \lambda_2 \cdots \lambda_N \quad \text{such that} \quad \lambda_i \in \lambda(A^{(i)}),\ i = 1, \ldots, N\},$$

which leads to the following result

Lemma 5.1. *If $I_i = J_i$, for $i = 1, \ldots, N$, then the solution of the tensor problem (5.2) is unique if and only if*

$$\lambda_1 \lambda_2 \cdots \lambda_N \neq 0,$$

for all $\lambda_i \in \lambda(A^{(i)})$, $i = 1, \ldots, N$.

When $\mathcal{X}$ is an order 2 tensor, that is a matrix X, the tensor least squares minimization (5.2) becomes

$$\min_X \left\| A^{(1)} X A^{(2)^T} - G \right\|.$$

In this work, we are interested in finding an approximate solution of the tensor least squares problem (5.1). Our approach is based on performing the well-known Golub-Kahan bidiagonalization to the pair of matrices $\left(A^{(i)}, G^{(i)}\right)$ for $i = 1, \ldots, N$. More generally, we are interested on solving the tensor problem (5.2), by written the right-hand side tensor $\mathcal{G}$ in CP or HOSVD format. Using this approach we can solve the problem (5.2) for higher orders, since we are dealing with matrices instead of tensors. In fact, when the dimension increases the problem becomes harder (computation and memory requirement), since the data size of a tensor increases exponentially with the increase of the dimensionality of the tensor. As consequence tensor computations become much expensive. For example the n-mode product given in Definition 3.14 has a computational complexity of $O(J\prod_{\substack{i=1 \\ i\neq n}}^{N} I_i)$. In addition, by writing the approximate solution in CP or HOSVD decomposition format, we reduce the required memory. For instance, the CP decomposition transforms the storage complexity of an I^N tensors into $O(NRI)$, where R is the CP rank.

The remainder of the chapter is organized as follows. In the next section, we introduce notations adopted in this chapter and some basic definitions, properties related to tensors. In section 5.3, we construct an approximate solution of the minimization problem (5.2) based on Golub-Kahan bidiagonalization. We work on coefficient matrices $A^{(i)}(i = 1, \ldots, N)$ by taking the right-hand side tensor $\mathcal{G}$ in rank one format, then we generalized to the case where the right-hand side tensor is approximated using HOSVD decomposition in section 5.4. An example of application to image and video restoration is given in section 5.5. Finally, numerical examples are presented in section 5.6, to show the effectiveness of the proposed approach.

5.2 Notations and preliminary concepts

In this section, we recall some of the basic properties about tensors, that will be used in the chapter.

Definition 5.2 ([20, 56]). The n-mode matrix of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathcal{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ and arranges the mode-n fibers into columns of a matrix. More specifically, we have

$$\mathcal{A}_{(n)}(i_n, j) = \mathcal{A}(i_1, i_2, ..., i_N),$$

where $j = 1 + \sum_{k=1, k\neq n}^{N} (i_k - 1) J_k$, and $J_k = \prod_{m=1, m\neq n}^{k-1} I_m$.

Remark 5.2.1. Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, we can express the n-mode matrix using slices:

$$\begin{aligned}
\mathcal{A}_{(1)} &= [\mathcal{A}(:,1,:), \mathcal{A}(:,2,:), \ldots, \mathcal{A}(:,I_2,:)] \\
\mathcal{A}_{(2)} &= \left[\mathcal{A}(:,:,1)^T, \mathcal{A}(:,:,2)^T, \ldots, \mathcal{A}(:,:,I_3)^T\right] \\
\mathcal{A}_{(3)} &= [\mathcal{A}(1,:,:), \mathcal{A}(2,:,:), \ldots, \mathcal{A}(I_1,:,:)] .
\end{aligned}$$

Proposition 5.3 ([20, 56]). *Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a N-th order tensor and $\{U_i\}_{1 \leq i \leq N}$ a set of matrices with $U_i \in \mathbb{R}^{J_i \times I_i}$, $i = 1, \ldots, N$.*

1. $(\mathcal{X} \times_{i=1}^{N} U_i)_{(n)} = U_n \mathcal{X}_{(n)} (U_N \otimes \ldots \otimes U_{n+1} \otimes U_{n-1} \otimes \ldots \otimes U_1)^T$.
2. $vec(\mathcal{X} \times_{i=1}^{N} U_i) = (U_N \otimes U_{N-1} \otimes \cdots \otimes U_1) vec(\mathcal{X})$.

Proposition 5.4. *Let $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{X} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ be two N order tensors, we have the following result*

$$\|\mathcal{Y} \otimes \mathcal{X}\| = \|\mathcal{Y}\| \|\mathcal{X}\| .$$

Proof. Easy to verify using Definition 3.18. □

Proposition 5.5 ([20]). *Let $(a_i)_{1 \leq i \leq N}$ be a family of N vectors of sizes I_i with $i = 1, \ldots, N$. We have the following relation*

$$vec(a_1 \circ a_2 \circ \cdots \circ a_N) = a_N \otimes a_{N-1} \otimes \cdots \otimes a_1.$$

Proposition 5.6 ([9]). *Let $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{X} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ be two tensors, $U_n \in \mathbb{R}^{K \times I_n}$ and $V_n \in \mathbb{R}^{L \times J_n}$. We have*

$$(\mathcal{Y} \otimes \mathcal{X}) \times_n (U_n \otimes V_n) = (\mathcal{Y} \times_n U_n) \otimes (\mathcal{X} \times_n V_n).$$

5.3 Rank one approximation

In the following section, we assume that the right-hand side tensor $\mathcal{G}$ in (5.2) is of rank one, which mean, it can be written in this form

$$\mathcal{G} = g^{(1)} \circ g^{(2)} \circ \cdots \circ g^{(N)},$$

where $g^{(i)} \in \mathbb{R}^{J_i}$ for $i = 1, \ldots, N$. Applying Golub-Kahan bidiagonalization algorithm to the pairs $(A^{(i)}, g^{(i)})$ for $i = 1, \ldots, N$, leads to the following relations for $i = 1, \ldots, N$,

$$\begin{aligned} U_{k+1}^{(i)}(\beta_1^{(i)} e_1) &= g^{(i)}, \\ A^{(i)} V_k^{(i)} &= U_{k+1}^{(i)} B_k^{(i)}, \end{aligned} \tag{5.3}$$

where $U_{k+1}^{(i)} = [u_1^{(i)}, \ldots, u_{k+1}^{(i)}]$ and $V_k^{(i)} = [v_1^{(i)}, \ldots, v_k^{(i)}]$ are orthonormal basis, e_1 is the first unit vector of $\mathbb{R}^k$ and $B_k^{(i)}$ is a bidiagonal matrix defined as follow

$$B_k^{(i)} = \begin{bmatrix} \alpha_1^{(i)} & & & & \\ \beta_2^{(i)} & \alpha_2^{(i)} & & & \\ & \beta_3^{(i)} & \alpha_3^{(i)} & & \\ & & \ddots & \ddots & \\ & & & \beta_k^{(i)} & \alpha_k^{(i)} \\ & & & & \beta_{k+1}^{(i)} \end{bmatrix}.$$

Proposition 5.7. *Let $\mathcal{X}^{(k)} = \mathcal{Y}_k \times_1 V_k^{(1)} \times_2 V_k^{(2)} \times_3 \cdots \times_N V_k^{(N)}$ be an approximate solution of (5.2) with $\mathcal{Y}_k \in \mathbb{R}^{k \times k \times \cdots \times k}$. The corresponding residual $\mathcal{R}_k$ can be expressed as*

$$\mathcal{R}_k = (\beta_1^{(1)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1 - \mathcal{Y}_k \times_1 B_k^{(1)} \times_2 \cdots \times_N B_k^{(N)}) \times_1 U_{k+1}^{(1)} \times_2 \cdots \times_N U_{k+1}^{(N)}. \tag{5.4}$$

And the residual norm is given by

$$\|\mathcal{R}_k\| = \left\| \beta_1^{(1)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1 - \mathcal{Y}_k \times_1 B_k^{(1)} \times_2 \cdots \times_N B_k^{(N)} \right\|. \tag{5.5}$$

Proof. Using the relations (5.3), we get

$$\begin{aligned}
\mathcal{R}_k &= \mathcal{G} - \mathcal{X}_k \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} \\
&= \mathcal{G} - \mathcal{Y}_k \times_1 V_k^{(1)} \times_2 V_k^{(2)} \times_3 \cdots \times_N V_k^{(N)} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} \\
&= \mathcal{G} - \mathcal{Y}_k \times_1 A^{(1)} V_k^{(1)} \times_2 A^{(2)} V_k^{(2)} \times_3 \cdots \times_N A^{(N)} V_k^{(N)} \\
&= \mathcal{G} - \mathcal{Y}_k \times_1 U_{k+1}^{(1)} B_k^{(1)} \times_2 U_{k+1}^{(2)} B_k^{(2)} \times_3 \cdots \times_N U_{k+1}^{(N)} B_k^{(N)} \\
&= \mathcal{G} - \mathcal{Y}_k \times_1 B_k^{(1)} \times_2 B_k^{(2)} \times_3 \cdots \times_N B_k^{(N)} \times_1 U_{k+1}^{(1)} \times_2 U_{k+1}^{(2)} \times_3 \cdots \times_N U_{k+1}^{(N)}.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\mathcal{G} &= g^{(1)} \circ g^{(2)} \circ \cdots \circ g^{(N)} \\
&= U_{k+1}^{(1)}(\beta_1^{(1)} e_1) \circ U_{k+1}^{(2)}(\beta_1^{(2)} e_1) \circ \cdots \circ U_{k+1}^{(N)}(\beta_1^{(N)} e_1) \\
&= (\beta_1^{(1)} e_1 \circ \beta_1^{(2)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1) \times_1 U_{k+1}^{(1)} \times_2 U_{k+1}^{(2)} \times_3 \cdots \times_N U_{k+1}^{(N)},
\end{aligned}$$

which shows (5.4). On the other hand $U_{k+1}^{(i)}$ for $i = 1, \ldots, N$ are orthonormal matrices, which proves (5.5). □

The method determines the tensor $\mathcal{Y}_k$ which minimize $\|\mathcal{R}_k\|$,

$$\mathcal{Y}_k = \underset{\mathcal{Y}}{argmin} \left\| \beta_1^{(1)} e_1 \circ \beta_1^{(2)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1 - \mathcal{Y} \times_1 B_k^{(1)} \times_2 B_k^{(2)} \times_3 \cdots \times_N B_k^{(N)} \right\|. \tag{5.6}$$

This minimization problem is accomplished by using the QR decomposition to each matrix $B_k^{(i)}$ for $i = 1, \ldots, N$.

$$Q_k^{(i)} B_k^{(i)} = \begin{pmatrix} R_k^{(i)} \\ 0 \end{pmatrix}, \quad Q_k^{(i)}(\beta_1^{(i)} e_1) = \begin{pmatrix} f_k^{(i)} \\ \bar{\phi}_{k+1}^{(i)} \end{pmatrix},$$

where the matrix $Q_k^{(i)}$ is a product of k Givens rotation chosen to eliminate the subdiagonal elements $\beta_2^{(i)}, \beta_3^{(i)}, \ldots, \beta_{k+1}^{(i)}$ and

$$R_k^{(i)} = \begin{bmatrix} \rho_1^{(i)} & \theta_2^{(i)} & & & \\ & \rho_2^{(i)} & \theta_3^{(i)} & & \\ & & \ddots & \ddots & \\ & & & \rho_{k-1}^{(i)} & \theta_k^{(i)} \\ & & & & \rho_k^{(i)} \end{bmatrix}, \quad \textbf{and} \quad f_k^{(i)} = \begin{bmatrix} \phi_1^{(i)} \\ \phi_2^{(i)} \\ \vdots \\ \phi_{k-1}^{(i)} \\ \phi_k^{(i)} \end{bmatrix}.$$

Then, the minimizer $\mathcal{Y}_k$ of the minimization problem (5.6) can be obtained from the following equation

$$\mathcal{Y}_k \times_1 R_k^{(1)} \times_2 R_k^{(2)} \times_3 \cdots \times_N R_k^{(N)} = f_k^{(1)} \circ f_k^{(2)} \circ \cdots \circ f_k^{(N)}.$$

Therefore an approximate solution is given by

$$\begin{aligned} \mathcal{X}^{(k)} &= \mathcal{Y}_k \times_1 V_k^{(1)} \times_2 V_k^{(2)} \times_3 \cdots \times_N V_k^{(N)} \\ &= V_k{}^{(1)} R_k{}^{(1)^{-1}} f_k{}^{(1)} \circ V_k^{(2)} R_k^{(2)^{-1}} f_k{}^{(2)} \circ \cdots \circ V_k^{(N)} R_k^{(N)^{-1}} f_k^{(N)} \\ &= x_k^{(1)} \circ x_k^{(2)} \circ \cdots \circ x_k^{(N)}, \end{aligned}$$

where $x_k{}^{(i)} = V_k^{(i)} R_k^{(i)^{-1}} f_k{}^{(i)}$ for $i = 1, \ldots, N$. Which can be formulated in this form (see [66]),

$$\begin{cases} x_k^{(i)} &= x_{k-1}^{(i)} + \phi_k^{(i)} d_k^{(i)}, \\ x_0^{(i)} &= 0, \end{cases}$$

where $d_k^{(i)}$ can be updated using the following expression:

$$\begin{cases} d_k^{(i)} &= \dfrac{1}{\rho_k^{(i)}} (v_k^{(i)} - \theta_k^{(i)} d_{k-1}^{(i)}), \\ d_0^{(i)} &= 0. \end{cases}$$

The following lemma is used to prove Theorem 5.9.

Lemma 5.8. *Let $\mathcal{F}$ and $\hat{\mathcal{F}}$ be two N order tensors of size $k+1 \times \cdots \times k+1$ defined by:*

$$\begin{aligned}\mathcal{F} &= \begin{bmatrix} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} \end{bmatrix} \circ \begin{bmatrix} f_k^{(2)} \\ \bar{\phi}_{k+1}^{(2)} \end{bmatrix} \circ \cdots \circ \begin{bmatrix} f_k^{(N)} \\ \bar{\phi}_{k+1}^{(N)} \end{bmatrix}, \\ \hat{\mathcal{F}} &= \begin{bmatrix} f_k^{(1)} \\ 0 \end{bmatrix} \circ \begin{bmatrix} f_k^{(2)} \\ 0 \end{bmatrix} \circ \cdots \circ \begin{bmatrix} f_k^{(N)} \\ 0 \end{bmatrix}.\end{aligned}$$

Then $\left\langle \mathcal{F}, \hat{\mathcal{F}} \right\rangle = \left\| \hat{\mathcal{F}} \right\|^2$.

Proof.

$$\left\langle \mathcal{F}, \hat{\mathcal{F}} \right\rangle = \sum_{i_1=1}^{k+1} \sum_{i_2=1}^{k+1} \cdots \sum_{i_N=1}^{k+1} \mathcal{F}_{i_1 \ldots i_N} \hat{\mathcal{F}}_{i_1 \ldots i_N}.$$

From Definition 3.17, it is easy to verify that

$$\hat{\mathcal{F}}_{i_1 \ldots i_N} = \begin{cases} \mathcal{F}_{i_1 \ldots i_N} & \text{for} \quad 1 \leq i_1, i_2, \ldots, i_N \leq k \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned}\left\langle \mathcal{F}, \hat{\mathcal{F}} \right\rangle &= \sum_{i_1}^{k} \sum_{i_2}^{k} \cdots \sum_{i_N}^{k} \mathcal{F}_{i_1 \ldots i_N} \mathcal{F}_{i_1 \ldots i_N} \\ &= \sum_{i_1}^{k} \sum_{i_2}^{k} \cdots \sum_{i_N}^{k} \mathcal{F}_{i_1 \ldots i_N}^2 \\ &= \left\| \hat{\mathcal{F}} \right\|^2.\end{aligned}$$

□

In the following proposition, we give an upper bound of the residual norm.

Theorem 5.9. *The residual norm $||\mathcal{R}_k||^2$ satisfy the following inequality*

$$\|\mathcal{R}_k\|^2 \leq P_k \sum_{i=1}^{N} \frac{\bar{\phi}_{k+1}^{(i)2}}{\left\| \hat{f}_k^{(i)} \right\|^2},$$

where $\hat{f}_k^{(i)} = \begin{bmatrix} {f_k}^{(i)} \\ \bar{\phi}_{k+1}^{(i)} \end{bmatrix}$ and $P_k = \prod_{i=1}^{N} \left\| \hat{f}_k^{(i)} \right\|^2$.

Proof. We have

$$\begin{aligned}
\|\mathcal{R}_k\|^2 &= \left\|\mathcal{Y}_k \times_1 B_k^{(1)} \times_2 \cdots \times_N B_k{}^{(N)} - \beta_1{}^{(1)} e_1 \circ \cdots \circ \beta_1{}^{(N)} e_1\right\|^2 \\
&= \left\|\left(\mathcal{Y}_k \times_1 B_k^{(1)} \times_2 \cdots \times_N B_k{}^{(N)} - \beta_1^{(1)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1\right) \times_1 Q_k{}^{(1)} \times_2 \cdots \times_N Q_k{}^{(N)}\right\|^2 \\
&= \left\|\mathcal{Y}_k \times_1 \begin{bmatrix} R_k^{(1)} \\ 0 \end{bmatrix} \times_2 \cdots \times_N \begin{bmatrix} R_k^{(N)} \\ 0 \end{bmatrix} - \begin{bmatrix} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} \end{bmatrix} \circ \cdots \circ \begin{bmatrix} f_k^{(N)} \\ \bar{\phi}_{k+1}^{(N)} \end{bmatrix}\right\|^2.
\end{aligned}$$

Let $\mathcal{Y}_k$ be the solution to the following problem:

$$\mathcal{Y}_k \times_1 R_k^{(1)} \times_2 R_k^{(2)} \times_3 \cdots \times_N R_k^{(N)} = f_k^{(1)} \circ f_k^{(2)} \circ \cdots \circ f_k^{(N)},$$

then

$$\|\mathcal{R}_k\|^2 = \left\|\begin{bmatrix} f_k^{(1)} \\ \bar{\phi}_{k+1}^{(1)} \end{bmatrix} \circ \begin{bmatrix} f_k^{(2)} \\ \bar{\phi}_{k+1}^{(2)} \end{bmatrix} \circ \cdots \circ \begin{bmatrix} f_k^{(N)} \\ \bar{\phi}_{k+1}^{(N)} \end{bmatrix} - \begin{bmatrix} f_k^{(1)} \\ 0 \end{bmatrix} \circ \begin{bmatrix} f_k^{(2)} \\ 0 \end{bmatrix} \circ \cdots \circ \begin{bmatrix} f_k^{(N)} \\ 0 \end{bmatrix}\right\|^2.$$

From Lemma 5.8, the residual norm can be expressed as follow:

$$\begin{aligned}
\|\mathcal{R}_k\|^2 &= \left\|\mathcal{F} - \hat{\mathcal{F}}\right\|^2 \\
&= \|\mathcal{F}\|^2 - 2\left\langle \mathcal{F}, \hat{\mathcal{F}}\right\rangle + \left\|\hat{\mathcal{F}}\right\|^2 \\
&= \|\mathcal{F}\|^2 - \left\|\hat{\mathcal{F}}\right\|^2 \\
&= \sum_{i_1=1}^{k+1}\sum_{i_2=1}^{k+1}\cdots\sum_{i_N=1}^{k+1} \mathcal{F}_{i_1 i_2 \ldots i_N}^2 - \sum_{i_1=1}^{k}\sum_{i_2=1}^{k}\cdots\sum_{i_N=1}^{k} \mathcal{F}_{i_1 i_2 \ldots i_N}^2 \\
&\leq \sum_{i_2=1}^{k+1}\cdots\sum_{i_N=1}^{k+1} \mathcal{F}_{k+1 i_2 \ldots i_N}^2 + \sum_{i_1=1}^{k+1}\cdots\sum_{i_N=1}^{k+1} \mathcal{F}_{i_1 k+1 \ldots i_N}^2 + \cdots + \sum_{i_1=1}^{k+1}\sum_{i_2=1}^{k+1}\cdots\sum_{i_{N-1}=1}^{k+1} \mathcal{F}_{i_1 i_2 \ldots k+1}^2.
\end{aligned}$$

Using Matlab notation, we have

$$\left\|\mathcal{R}_k\right\|^2 \leq \|\mathcal{F}(k+1,:,\ldots,:)\|^2 + \|\mathcal{F}(:,k+1,\ldots,:)\|^2 + \cdots + \|\mathcal{F}(:,:,\ldots,k+1)\|^2.$$

With $\underbrace{\mathcal{F}(:,\ldots,k+1,\ldots,:)}_{\text{i-th index}}$ for $i = 1,\ldots,N$, are expressed by

$$\mathcal{F}(:,\ldots,k+1,\ldots,:) = \bar{\phi}_{k+1}^{(i)} \hat{f}_k^{(1)} \circ \cdots \circ \hat{f}_k^{(i-1)} \circ \hat{f}_k^{(i+1)} \circ \cdots \circ \hat{f}_k^{(N)}.$$

Using Proposition 5.5, we have

$$vec(\mathcal{F}(:,\ldots,k+1,\ldots,:)) = \bar{\phi}_{k+1}^{(i)} \hat{f}_k^{(N)} \otimes \cdots \otimes \hat{f}_k^{(i+1)} \otimes \hat{f}_k^{(i-1)} \cdots \otimes \hat{f}_k^{(1)}.$$

Then

$$||\mathcal{R}_k||^2 \leq \sum_{i=1}^{N} \bar{\phi}_{k+1}^{(i)2} \left\|\hat{f}_k^{(1)}\right\|^2 \cdots \left\|\hat{f}_k^{(i-1)}\right\|^2 \left\|\hat{f}_k^{(i+1)}\right\|^2 \cdots \left\|\hat{f}_k^{(N)}\right\|^2.$$

□

5.4 Approximation in HOSVD format

In this section, we assume that the right-hand side tensor $\mathcal{G}$ of (5.2) is written in HOSVD format:

$$\mathcal{G} = \mathcal{S} \times_1 G^{(1)} \times_2 \cdots \times_N G^{(N)},$$

where $\mathcal{S} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_N}$ and $G^{(i)} \in \mathbb{R}^{J_i \times m_i}$ for $i = 1,\ldots,N$. The CP decomposition is a particular case of HOSVD, when $m_1 = m_2 = \cdots = m_N = R$ and $\mathcal{S} = \mathcal{I}_R$, where $\mathcal{I}_R$ is the tensor identity. Applying global Golub-Kahan bidiagonalization [71] to the pairs $(A^{(i)}, G^{(i)})$ for $i = 1,\ldots,N$, leads to the following relations:

$$\begin{aligned} \mathbb{U}_{k+1}{}^{(i)}({\beta_1}^{(i)} e_1 \otimes I_{m_i}) &= G^{(i)}, \\ A^{(i)} \mathbb{V}_k^{(i)} &= \mathbb{U}_{k+1}^{(i)} (B_k^{(i)} \otimes I_{m_i}). \end{aligned} \tag{5.7}$$

with $\mathbb{U}_{k+1}^{(i)} = \left[U_1^{(i)}\ U_2^{(i)} \cdots U_{k+1}^{(i)}\right]$ and $\mathbb{V}_k^{(i)} = \left[V_1^{(i)}\ V_2^{(i)} \cdots V_k^{(i)}\right]$ are F-orthonormal.
The method consists in searching an approximate solution of the form

$$\mathcal{X}^{(k)} = \mathcal{Y}_k \times_1 \mathbb{V}_k^{(1)} \times_2 \mathbb{V}_k^{(2)} \times_3 \cdots \times_N \mathbb{V}_k^{(N)},$$

where $\mathcal{Y}_k$ is solution to the following minimization problem

$$\min_{\mathcal{Y}} \left\| \mathcal{S} \times_1 (\beta_1^{(i)} e_1 \otimes I_{m_1}) \times_2 \cdots \times_N (\beta_1^{(N)} e_1 \otimes I_{m_N}) - \mathcal{Y} \times_1 (B_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (B_k^{(N)} \otimes I_{m_N}) \right\|. \tag{5.8}$$

In particular, when $\mathcal{S}$ is reduce to $\mathcal{I}_R$ and $m_1 = \ldots = m_N = R$, we have the following proposition

Proposition 5.10. *Let $\mathcal{X}^{(k)} = (\mathcal{Y}_k \otimes \mathcal{I}_R) \times_1 \mathbb{V}_k^{(1)} \times_2 \mathbb{V}_k^{(2)} \cdots \times_N \mathbb{V}_k^{(N)}$ with $\mathcal{Y}_k \in \mathbb{R}^{k\times k\times\cdots\times k}$, be an approximate solution of (5.2), with the right-hand side tensor $\mathcal{G}$ is written in CP decomposition format. Then the corresponding residual $\mathcal{R}_k$ can be expressed as*

$$\mathcal{R}_k = (\beta_1^{(1)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1 - \mathcal{Y}_k \times_1 B_k^{(1)} \times_2 \cdots \times_N B_k^{(N)}) \otimes \mathcal{I}_R \times_1 \mathbb{U}_{k+1}^{(1)} \times_2 \cdots \times_N \mathbb{U}_{k+1}^{(N)}. \tag{5.9}$$

In this case, The method chooses the tensor $\mathcal{Y}_k$ which satisfy the minimization problem

$$\mathcal{Y}_k = \underset{\mathcal{Y}}{argmin} \left\| \beta_1^{(1)} e_1 \circ \beta_1^{(2)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1 - \mathcal{Y} \times_1 B_k^{(1)} \times_2 B_k^{(2)} \times_3 \cdots \times_N B_k^{(N)} \right\|. \tag{5.10}$$

Proof. Using the relations (5.7), we get

$$\begin{aligned}
\mathcal{R}_k &= \mathcal{G} - \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} \\
&= \mathcal{G} - (\mathcal{Y}_k \otimes \mathcal{I}_R) \times_1 \mathbb{V}_k^{(1)} \times_2 \mathbb{V}_k^{(2)} \times_3 \cdots \times_N \mathbb{V}_k^{(N)} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} \\
&= \mathcal{G} - (\mathcal{Y}_k \otimes \mathcal{I}_R) \times_1 \mathbb{U}_{k+1}^{(1)}(B_k^{(1)} \otimes I_R) \times_2 \mathbb{U}_{k+1}^{(2)}(B_k^{(2)} \otimes I_R) \times_3 \cdots \times_N \mathbb{U}_{k+1}^{(N)}(B_k^{(N)} \otimes I_R) \\
&= \mathcal{G} - \mathcal{Y}_k \times_1 B_k^{(1)} \times_2 B_k^{(2)} \times_3 \cdots \times_N B_k^{(N)} \otimes \mathcal{I}_{\mathcal{R}} \times_1 \mathbb{U}_{k+1}^{(1)} \times_2 \mathbb{U}_{k+1}^{(2)} \times_3 \cdots \times_N \mathbb{U}_{k+1}^{(N)}.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\mathcal{G} &= \mathcal{I}_R \times_1 G^{(1)} \times_2 \cdots \times_N G^{(N)} \\
&= \mathcal{I}_R \times_1 \mathbb{U}_{k+1}^{(1)}(\beta_1^{(1)} e_1 \otimes I_R) \times_2 \cdots \times_N \mathbb{U}_{k+1}^{(N)}(\beta_1^{(N)} e_1 \otimes I_R) \\
&= (\beta_1^{(1)} e_1 \circ \cdots \circ \beta_1^{(N)} e_1 \otimes \mathcal{I}_{\mathcal{R}}) \times_1 \mathbb{U}_{k+1}^{(1)} \times_2 \cdots \times_N \mathbb{U}_{k+1}^{(N)},
\end{aligned}$$

which prove (5.9). □

The minimization problem (5.8) is accomplished by using QR decomposition associated to the matrices $B_k^{(i)}$ for $i = 1, \ldots, N$. Then, the minimizer $\mathcal{Y}_k$ of the problem (5.8), can be obtained from the following equation:

$$\mathcal{Y}_k \times_1 (R_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (R_k^{(N)} \otimes I_{m_N}) = \mathcal{S} \times_1 (f_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (f_k^{(N)} \otimes I_{m_N}).$$

Therefore an approximate solution is formed as

$$\begin{aligned}\mathcal{X}^{(k)} &= \mathcal{S} \times_1 \mathbb{V}_k^{(1)} \times_2 \cdots \times_N \mathbb{V}_k^{(N)} \\ &= \mathcal{S} \times_1 \mathbb{V}_k{}^{(1)}(R_k{}^{(1)-1} f_k{}^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N \mathbb{V}_k^{(N)}(R_k{}^{(N)-1} f_k{}^{(N)} \otimes I_{m_N}) \\ &= \mathcal{S} \times_1 X_k^{(1)} \times_2 \cdots \times_N X_k^{(N)},\end{aligned} \tag{5.11}$$

where $X_k{}^{(i)} = \mathbb{V}_k{}^{(i)}(R_k^{(i)^{-1}} f_k{}^{(i)} \otimes I_{m_i})$ for $i = 1, \ldots, N$. Which can be formulated in this form,

$$\begin{cases} X_k^{(i)} &= X_{k-1}^{(i)} + \phi_k^{(i)} D_k^{(i)} \\ X_0^{(i)} &= 0, \end{cases} \tag{5.12}$$

where $D_k^{(i)}$ can be updated using the following expression:

$$\begin{cases} D_k^{(i)} &= \dfrac{1}{\rho_k^{(i)}}(V_k^{(i)} - \theta_k^{(i)} D_{k-1}^{(i)}) \\ D_0^{(i)} &= 0. \end{cases}$$

Lemma 5.11. *Let $\mathbb{V}_k^{(i)} = \left[V_1^{(i)}\ V_2^{(i)} \cdots V_k^{(i)}\right]$ be an F-orthonormal basis with $V_l^{(i)} \in \mathbb{R}^{I_i \times m_i}$ for $l = 1, \ldots, k$, and let $\mathcal{X}$ be an N order tensors of size $km_1 \times \cdots \times km_N$, and $\mathcal{Z} \in \mathbb{R}^{k \times \cdots \times k}$. Then*

$$\left\| \mathcal{X} \times_i \mathbb{V}_k^{(i)} \right\| \leqslant \|\mathcal{X}\|, \tag{5.13}$$

$$\left\| (\mathcal{Z} \otimes \mathcal{I}_{m_i}) \times_i \mathbb{V}_k^{(i)} \right\| = \|\mathcal{Z}\|. \tag{5.14}$$

Proof. See [9]. □

Theorem 5.12. *The residual norm $||\mathcal{R}_k||^2$ satisfy the following inequality,*

$$\|\mathcal{R}_k\|^2 \leq \|\mathcal{S}\|^2 P_k \sum_{i=1}^{N} \frac{\bar{\phi}_{k+1}^{(i)2}}{\left\| \hat{f}_k^{(i)} \right\|^2},$$

where $\hat{f}_k^{(i)} = \begin{bmatrix} f_k{}^{(i)} \\ \bar{\phi}_{k+1}^{(i)} \end{bmatrix}$ *and* $P_k = \prod_{i=1}^{N} ||\hat{f}_k^{(i)}||^2$.

Proof. Using the first relation in Lemma 5.11, we have

$$\|\mathcal{R}_k\|^2 \leqslant \left\|\mathcal{Y}_k \times_1 (B_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (B_k^{(N)} \otimes I_{m_N}) - \mathcal{S} \times_1 (\beta_1^{(i)} e_1 \otimes I_{m_1}) \times_2 \cdots \times_N (\beta_1^{(N)} e_1 \otimes I_{m_N})\right\|^2.$$

Let $Z = \left\|\mathcal{Y}_k \times_1 (B_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (B_k^{(N)} \otimes I_{m_N}) - \mathcal{S} \times_1 (\beta_1^{(i)} e_1 \otimes I_{m_1}) \times_2 \cdots \times_N (\beta_1^{(N)} e_1 \otimes I_{m_N})\right\|^2$.
We have

$$Z = \left\|\mathcal{Y}_k \times_{i=1}^{N} \left(\begin{bmatrix} R_k^{(i)} \\ 0 \end{bmatrix} \otimes I_{m_i} \right) - \mathcal{S} \times_{i=1}^{N} \left(\begin{bmatrix} f_k^{(i)} \\ \bar{\phi}_{k+1}^{(i)} \end{bmatrix} \otimes I_{m_i} \right) \right\|^2.$$

Let $\mathcal{Y}_k$ be the solution to the following equation:

$$\mathcal{Y} \times_1 (R_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (R_k^{(N)} \otimes I_{m_N}) = \mathcal{S}_k \times_1 (f_k^{(1)} \otimes I_{m_1}) \times_2 \cdots \times_N (f_k^{(N)} \otimes I_{m_N}).$$

Then

$$Z = \left\|\mathcal{S} \times_{i=1}^{N} \left(\begin{bmatrix} f_k^{(i)} \\ \bar{\phi}_{k+1}^{(i)} \end{bmatrix} \otimes I_{m_i} \right) - \mathcal{S} \times_{i=1}^{N} \left(\begin{bmatrix} f_k^{(i)} \\ 0 \end{bmatrix} \otimes I_{m_i} \right) \right\|^2.$$

Let

$$\hat{f}^{(i)} = \begin{bmatrix} f_k^{(i)} \\ \bar{\phi}_{k+1}^{(i)} \end{bmatrix}, \quad \text{and} \quad f^{(i)} = \begin{bmatrix} f_k^{(i)} \\ 0 \end{bmatrix} \; i = 1, \ldots, N.$$

Using Proposition 5.3, we have

$$
\begin{aligned}
Z &\leq \|\mathcal{S}\|^2 \left\| \left(\hat{f}^{(N)} \otimes I_{m_N}\right) \otimes \cdots \otimes \left(\hat{f}^{(1)} \otimes I_{m_1}\right) - \left(f^{(N)} \otimes I_{m_N}\right) \otimes \cdots \otimes \left(f^{(1)} \otimes I_{m_1}\right) \right\|_2^2 \\
&= \|\mathcal{S}\|^2 \left\| \left(\hat{f}^{(N)} \otimes \cdots \otimes \hat{f}^{(1)}\right) \otimes \left(I_{m_N} \otimes \cdots \otimes I_{m_1}\right) - \left(f^{(N)} \otimes \cdots \otimes f^{(1)}\right) \otimes \left(I_{m_N} \otimes \cdots \otimes I_{m_1}\right) \right\|_2^2 \\
&= \|\mathcal{S}\|^2 \left\| \left(\hat{f}^{(N)} \otimes \cdots \otimes \hat{f}^{(1)} - f^{(N)} \otimes \cdots \otimes f^{(1)}\right) \otimes \underbrace{\left(I_{m_N} \otimes \cdots \otimes I_{m_1}\right)}_{I_M} \right\|_2^2 \\
&= \|\mathcal{S}\|^2 \left\| \left(\hat{f}^{(N)} \otimes \cdots \otimes \hat{f}^{(1)} - f^{(N)} \otimes \cdots \otimes f^{(1)}\right) \otimes I_M \right\|_2^2 \\
&= \|\mathcal{S}\|^2 \left\| \hat{f}^{(N)} \otimes \cdots \otimes \hat{f}^{(1)} - f^{(N)} \otimes \cdots \otimes f^{(1)} \right\|_2^2 \\
&= \|\mathcal{S}\|^2 \left\| vec\left(\hat{f}^{(1)} \circ \cdots \circ \hat{f}^{(N)}\right) - vec\left(f^{(1)} \circ \cdots \circ f^{(N)}\right) \right\|_2^2 \\
&= \|\mathcal{S}\|^2 \left\| \hat{f}^{(1)} \circ \cdots \circ \hat{f}^{(N)} - f^{(1)} \circ \cdots \circ f^{(N)} \right\|^2 .
\end{aligned}
$$

The result achieved using Theorem 5.9. □

The discussed approach can be summarized in Algorithm 5 given below.

Algorithm 5

1: **input:** Coefficient matrices $A^{(i)}$ for $i = 1, \ldots, N$. The right-hand side tensor $\mathcal{G}$.

2: **output:** An approximate solution $\mathcal{X}_k$ to (5.2).

3: Decompose $\mathcal{G}$ as $\mathcal{S} \times_1 G^{(1)} \times_2 G^{(2)} \times_3 \cdots \times_N G^{(N)}$

4: Set ${\beta_1}^{(i)} = ||G^{(i)}||, {U_1}^{(i)} = G^{(i)}/\beta_1, {\alpha_1}^{(i)} = ||{A^{(i)}}^T {U_1}^{(i)}||$ and ${V_1}^{(i)} = {A^{(i)}}^T U_1^{(i)}/\alpha_1^{(i)}$

5: Set $W_1^{(i)} = V_1^{(i)}, \bar{\phi}_1^{(i)} = \beta_1^{(i)}, \bar{\rho}_1^{(i)} = \alpha_1^{(i)}$

6: **for do** $j = 1, 2, 3 \ldots, k$

7: **for do** $i = 1, 2, 3 \ldots, N$

8: $\bar{W}_j = A^{(i)} V_j^{(i)} - \alpha_j^{(i)} U_j^{(i)},\ \beta_{j+1}^{(i)} = ||\bar{W}_j^{(i)}||,\ U_{j+1}^{(i)} = \bar{W}_j^{(i)}/\beta_{j+1}^{(i)}$

9: $\bar{S}_j^{(i)} = {A^{(i)}}^T {U_{j+1}}^{(i)} - {\beta_{j+1}}^{(i)} V_j^{(i)},\ \alpha_{j+1}^{(i)} = ||\bar{S}_j^{(i)}||,\ {V_{j+1}}^{(i)} = \bar{S}_j^{(i)}/{\alpha_{j+1}}^{(i)}$

10: ${\rho_j}^{(i)} = \sqrt{\bar{\rho}_j^{(i)2} + {\beta_{j+1}}^{(i)2}}$

11: ${c_j}^{(i)} = \bar{\rho}_j^{(i)}/{\rho_j}^{(i)}$

12: $s_j^{(i)} = \beta_j^{(i)}/\rho_j^{(i)}$

13: $\theta_{j+1}^{(i)} = s_j^{(i)} \alpha_{j+1}^{(i)}$

14: $\bar{\rho}_{j+1}^{(i)} = c_j^{(i)} \alpha_{j+1}^{(i)}$

15: $\phi_j^{(i)} = c_j^{(i)} \bar{\phi}_j^{(i)}$

16: ${n_j}^{(i)} = {n_{j-1}}^{(i)} {\phi_j}^{(i)2}$

17: $\bar{\phi}_{j+1}^{(i)} = s_j^{(i)} \bar{\phi}_j^{(i)}$

18: $X_j^{(i)} = X_{j-1}^{(i)} + \frac{\phi_j^{(i)}}{\rho_j^{(i)}} W_j^{(i)}$

19: ${W_{j+1}}^{(i)} = {V_{j+1}}^{(i)} - \frac{{\phi_{i+1}}^{(i)}}{{\rho_i}^{(i)}} {W_j}^{(i)}$

20: $\hat{n}_j^{(i)} = {n_j}^{(i)} + \bar{\phi}_{j+1}^{(i)2}$

21: **end for**

22: $\mathcal{X}_j = \mathcal{S} \times_1 {X_j}^{(1)} \times_2 \cdots \times_N {X_j}^{(N)}$

23: If $\|\mathcal{S}\| \sqrt{\prod_{i=1}^{N} \hat{n}_j^{(i)} \sum_{i=1}^{N} \frac{\bar{\phi}_{j+1}^{(i)2}}{\hat{n}_j^{(i)}}}$ is small enough then stop

24: **end for**

Remark 5.4.1. In the line 23 of Algorithm 5, we compute the upper bound of the residual norm $||\mathcal{R}_k||$ given in Theorem 5.12 where $\hat{n}_j^{(i)}$ denote $\left\| \hat{f}_j^{(i)} \right\|^2$.

5.5 Example of application to image and video restoration

Many applications require the solution of the problem of the form (5.2). The problem (5.2) may appear in color image and video restoration. We recall that the blur model associated to p-channel images (color images and gray-scale videos) is given by

$$\mathcal{X} \times_1 K_c \times_2 K_r = \mathcal{G}, \tag{5.15}$$

where $\mathcal{X}, \mathcal{G} \in \mathbb{R}^{m\times n\times p}$ denote the original image and the degraded image respectively. For more details about the degradation model associated to p-channel images, see chapter 4 section 4.2. This model is constructed under the assumption that the blur is the same in all channels.

In case of color videos, video restoration is the problem of restoring a sequence of k color images (frames). Each frame is represented by a third-order tensor of size $m \times n \times 3$ which mean, a color video can be represented as a fourth-order tensor of size $m \times n \times 3 \times k$. Under the same assumptions used to construct the blurring model associated to p-channel images, the degradation model associated color videos is given by

Proposition 5.13. *The blur model associated with color videos is given by the tensor equation*

$$\mathcal{X} \times_1 K_c \times_2 K_r = \mathcal{G}, \tag{5.16}$$

where $\mathcal{X}, \mathcal{G} \in \mathbb{R}^{m\times n\times 3\times k}$ denote the original and the degraded color video.

Proof. Let $\mathcal{G}(:,:,i,j)$ for $i = 1,2,3$ and $j = 1,2,\cdots,k$ denotes the gray scale image that constitute the channel of each frame of the blurred video. Assuming that the blur is spatially invariant. The blurring model is modeled by

$$K\mathbf{vec}(\mathcal{X}(:,:,i,j)) = \mathbf{vec}(\mathcal{G}(:,:,i,j)) \quad \text{for} \quad i = 1,2,3 \quad \text{and} \quad j = 1,2,\cdots,k \tag{5.17}$$

Using the approximation of the blurring matrix as Kronecker product, we have

$$K_c\mathcal{X}(:,:,i,j)K_r^T = \mathcal{G}(:,:,i,j) \quad \text{for} \quad i = 1,2,3 \quad \text{and} \quad j = 1,2,\cdots,k \tag{5.18}$$

which lead to

$$K_c[\mathcal{X}(:,:,1,1),\ldots,\mathcal{X}(:,:,3,k)](I_3 \otimes K_r^T) = [\mathcal{G}(:,:,1,1),\ldots,\mathcal{G}(:,:,3,k)] \tag{5.19}$$

which is equivalent to

$$K_c[\mathcal{X}(:,:,1,1),\cdots,\mathcal{X}(:,:,3,k)](I_3 \otimes I_k \otimes K_r^T) = [\mathcal{G}(:,:,1,1),\cdots,\mathcal{X}(:,:,3,k)] \tag{5.20}$$

The transpose of (5.20) is given by

$$K_r[\mathcal{X}(:,:,1,1)^T,\ldots,\mathcal{X}(:,:,3,k)^T](I_3 \otimes I_k \otimes K_c^T) = \left[\mathcal{G}(:,:,1,1)^T,\ldots,\mathcal{X}(:,:,3,k)^T\right] \tag{5.21}$$

As consequence, equation (5.21) can be expressed as

$$K_r\mathcal{X}_{(2)}(I_3 \otimes I_k \otimes K_c^T) = \mathcal{G}_{(2)} \tag{5.22}$$

As conclusion the blur model associated to color videos is given by the tensorial equation

$$\mathcal{X} \times_1 K_c \times_2 K_r = \mathcal{G} \tag{5.23}$$

□

Remark 5.5.1. In case where each channel of the p channels are not affected by the same blur. The blurring model associated is of the form

$$K_c^{(i)T}\mathcal{X}(:,:,i)K_r^{(i)} = \mathcal{G}(:,:,i), \quad \text{for} \quad i = 1,2,\cdots,p.$$

By imposing periodic boundary conditions, we assume that the matrices $K_c^{(i)}, K_r^{(i)}$ for $i = 1,2,\cdots,p$ are circulant matrices. Using the following well known propriety associated to the diagonalization of circulant matrices.

Lemma 5.14. *Let $C \in \mathbb{R}^{n\times n}$ be a circulant matrix, Then C is diagonalized matrix. More precisely,*

$$C = F^*\Lambda F,$$

where F is the Fourier matrix, $F^\star$ is the complex conjugate transpose of $F\Lambda = \operatorname{diag}(\lambda_1, \lambda_1, \lambda_2, \ldots, \lambda_n)$. where λ_k are the eigenvalues of C.

The blurring model associated to the restoration of multi-channel images is described in the following proposition:

Proposition 5.15. *Imposing periodic boundary conditions, the blur model associated to p-channel images is given by the tensorial equation*

$$\mathcal{X} \times_1 F_1 \times_2 F_2^* = \mathcal{S},$$

where $F_1 \in \mathbb{R}^{m\times m}, F_2 \in \mathbb{R}^{n\times n}$ are Fourier matrix and $\mathcal{S}$ is defined by

$$\mathcal{S}(:,:,i) = \Lambda_c^{(i)-1} F_1 \mathcal{G}(:,:,i) F_2^* \Lambda_r^{(i)-1}.$$

with $K_c^{(i)} = F_1 \Lambda_c^{(i)} F_1^*$ and $K_r^{(i)} = F_2 \Lambda_r^{(i)} F_2^*$ for $i = 1, \cdots, p$.

5.6 Numerical examples

In this section, we perform some numerical tests to show the effectiveness of the approach described in this chapter. The first part is devoted to solving the problem (5.2) for given matrices $A^{(i)}$, $i = 1, \ldots, N$. In the second part, we present some results of application to image and video restoration. In order to solve the problem (5.2), we decompose the right-hand side tensor $\mathcal{G} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$ using HOSVD or CP decomposition. In the next tables, we use Algorithm 5-CP to denote Algorithm 5 by decomposing the right-hand side tensor $\mathcal{G} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$ using CP decomposition as follows,

$$\mathcal{G} = G^{(1)} \circ G^{(2)} \circ \cdots \circ G^{(N)},$$

where $G^{(i)} \in \mathbb{R}^{J_i\times R}$ for $i = 1, \ldots, N$. In addition, we use Algorithm 5-HOSVD to denote Algorithm 5 by decomposing $\mathcal{G}$ using HOSVD decomposition as follows,

$$\mathcal{G} = \mathcal{S} \times_1 G^{(1)} \times_2 \cdots \times_N G^{(N)},$$

where $\mathcal{S} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_N}$, and $G^{(i)} \in \mathbb{R}^{J_i \times m_i}$. All experiments are performed on a 2.7 GHz Intel(R) Core i5 and 8 Ram with Matlab 2016a. In all the tables below, "Iter" stands for the number of iterations.

5.6.1 Part 1:

In this part, we present two numerical examples in order to show the effectiveness of our approach for solving the problem (5.2), for given matrices $A^{(i)}$, $i = 1, \ldots, N$.

Example 1

In the first example, the coefficient matrices $A^{(i)}$, $i = 1, \ldots, N$ are generated using the following Matlab command:

$$A^{(i)} = gallery('cycol', [n\ p], l),$$

with $l = 20$. In this case, $A^{(i)}$ are n-by-p matrix with cyclically repeating columns, with rank cannot exceed l. We construct the right-hand side tensor so that all the entries of the exact solution $\mathcal{X}^\star$ are equal to one. The following table display the results obtained. The used stopping criterion is

$$\|\mathcal{R}_k\| \leq \epsilon$$

where ϵ is a given tolerance equal to 10^{-6}, with number max of iterations equal to 30. In this example, we decompose the right-hand side tensor using CP decomposition.

Table 5.1: *Numerical results for Example 1.*

N	n	p	R	**Iter**	$\|\mathcal{R}_k\|$	$\|\mathcal{X}_k - \mathcal{X}^\star\|$	**CPU-time(sec)**
3	200	100	5	21	3.14×10^{-8}	7.66×10^{-13}	0.18
	400	300	10	30	1.95×10^{-6}	2.46×10^{-12}	0.78
	500	400	10	30	8.40×10^{-6}	4.37×10^{-12}	5.16
4	50	50	10	23	2.53×10^{-7}	8.16×10^{-12}	0.11
	100	50	20	30	1.92×10^{-7}	3.31×10^{-12}	0.17

We point out that the CPU time includes the required time for computing CP decomposition and the construction of the solution $\mathcal{X}^{(k)}$ (5.11).

Example 2

In this example, we keep the same data as the previous example, except for the coefficient matrices $A^{(i)}$, $i = 1, \ldots, N$, they are taken from [72] and they have the same size n:

$$A^{(i)} = eye(n) + \frac{0.5}{sqrt(n)} rand(n),$$

where *eye* and *rand* are Matlab functions corresponding to the identity and random matrix, respectively. We compared our approach with GLS-BTF described in [71]. The numerical results are listed in Table 5.2. and they are obtained by applying Algorithm 5, For this example, the stopping criterion is considered as

$$\frac{\|\mathcal{R}_k\|}{\|\mathcal{G}\|} < 10^{-10},$$

with number max of iterations equal to 160. We point out the CPU-time cover the required time for computing HOSVD decomposition of the right-hand side tensor and the time of construction of the solution $\mathcal{X}^{(k)}$. In the next table 'error' denotes the upper bound of the residual norm described in Theorem 5.12. In this example , we set $m_i = J_i = n$ for $i = 1, \ldots, N$.

Table 5.2: *Comparison of algorithm 5 and GLS-BTF.*

Method	N	n	**Iter**	$\|\mathcal{R}_k\|$	error	$\|\mathcal{X}_k - \mathcal{X}^\star\|$	**CPU-time(sec)**
Algorithm5-HOSVD	3	100	27	3.42×10^{-10}	1.34×10^{-9}	3×10^{-11}	0.21
		400	30	4.72×10^{-8}	7.39×10^{-8}	1.39×10^{-9}	17.94
	4	50	27	6.12×10^{-10}	2.81×10^{-9}	4.60×10^{-11}	2.01
		100	27	6.34×10^{-8}	1.33×10^{-7}	6.55×10^{-10}	35
GLS-BTF [71]	3	100	131	4.06×10^{-6}		5.41×10^{-6}	11.71
		400	–	–		–	–
	4	50	160	8.5×10^{-5}		1.18×10^{-4}	106.18
		100	–	–		–	–

The previous table demonstrates the efficiency of our approach especially in term of time of execution. Note that for GLS-BTF method we did not give the exact error and the residual norm, when $N = 3$, $n = 400$, and when $N = 4$, $n = 100$, due to large CPU-time needed to execute its associated algorithm. In order to show the quality of bound given in Theorem 5.12, we compare between the residual norm and 'error' which represent the upper bound of the residual norm.

We point out that when solving the problem (5.2) for higher dimensions, the approximate solution $\mathcal{X}^{(k)}$(5.11) is not explicitly computed, only the coefficient matrices $X_k^{(i)}$, $i = 1, \ldots, N$ are constructed. The numerical results are shown in Table 5.3. We keep the same matrices $A^{(i)}$, $i = 1, \ldots, N$ defined in this example, under the assumption that the right-hand side tensor is written in CP decomposition format:

$$\mathcal{G} = G^{(1)} \circ G^{(2)} \circ \cdots \circ G^{(N)},$$

with $G^{(i)} = rand(J_i, R)$, $i = 1 \ldots, N$. The used stopping criterion is

$$\|\mathcal{R}_k\| \leq \epsilon$$

where ϵ is a given tolerance equal to 10^{-10}. with number max of iterations equal to 25 when $N = 3$, and equal to 30 when $N = 4$.

Table 5.3: *Numerical results for Example 2 with $n = 1000,\ 10.000$.*

N	n	R	**Iter**	error	**CPU-time(sec)**
3	1000	10	25	1.97×10^{-9}	0.22
	10,000	10	25	3.45×10^{-7}	10.27
4	1000	10	30	2.07×10^{-10}	0.34
	10,000	5	30	3.03×10^{-8}	16.13

5.6.2 Part 2: application to image and video restoration

In this part, we provide some numerical results that illustrate the performance of the approach described in this work applied to the problem of image restoration. To determine the effectiveness of our methods, we evaluate the Relative error.

$$\text{Relative error} = \frac{||\mathcal{X}^{(k)} - \mathcal{X}_{true}||}{||\mathcal{X}_{true}||},$$

where $\mathcal{X}^{(k)}$ denotes the computed restoration. In addition we evaluate the Signal-to-Noise Ratio (SNR) defined by

$$\text{SNR} = 10\log_{10}\frac{||\mathcal{X}_{true} - E(\mathcal{X}_{true})||^2}{||\mathcal{X}^{(k)} - \mathcal{X}_{true}||^2},$$

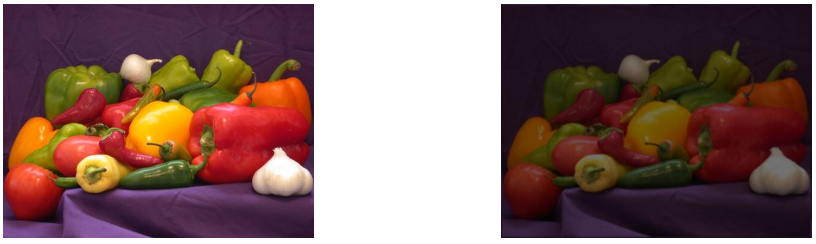
where $E(\mathcal{X}_{true})$ denotes the mean gray-level of the uncontaminated image $\mathcal{X}_{true}$. In the following examples, we point out that the CPU time covers both the time of decomposition of the right-hand side tensor $\mathcal{G}$ and the construction of the solution. In the next example, we set $m = m_1 = m_2$ and $m_i = J_i$ for $i = 3, \ldots, N$, with $N = 3$ in case of color images and grayscal videos, and $N = 4$ in case of color videos.

5.6.2.1 Example 1

This example illustrates the performance of Algorithm 5 applied to the restoration of a 3-channel RGB color image that has been contaminated by Gaussian blur whose function is given by

$$k(s,t) = \frac{1}{2\pi\alpha^2} exp\left\{-\frac{1}{2\alpha^2}\left(s^2+t^2\right)\right\},$$

and noise (generated by Matlab's randn function) with noise level $\nu = 10^{-3}$. This noise level is defined as follows $\nu = \frac{\|\mathcal{E}\|}{\|\hat{\mathcal{G}}\|}$ where $\mathcal{E}$ is a tensor that represents the noise in $\mathcal{G}$, i.e., $\mathcal{G} := \hat{\mathcal{G}} + \mathcal{E}$, and $\hat{\mathcal{G}}$ is the noise-free image. The true and blurred noisy image of size $388 \times 516 \times 3$ are shown in Figure 5.1. Table 5.4 compares the computing CPU-time, the relative errors and the SNR of the computed restorations for a fixed number of iterations.



(a) (b)

Figure 5.1: *Example 1. (a) Exact image, (b) blurred image.*

Table 5.4: *Results for Example 1.*

$R = m$	**Method**	**Iter**	**SNR**	**Relative error**	**CPU-time(sec)**
100	Algorithm 5-CP	40	19.51	6.32×10^{-2}	1.96
	Algorithm 5-HOSVD	40	22.44	4.51×10^{-2}	0.63
150	Algorithm 5-CP	20	21.77	4.87×10^{-2}	2.75
	Algorithm 5-HOSVD	20	24.23	3.67×10^{-2}	0.54

(a) (b)

Figure 5.2: *Example 1. restored images for R=m=150 (a) approach based on CP decomposition, (b) approach based on HOSVD decomposition.*

5.6.2.2 Example 2

In this example, we illustrate the effectiveness of our approach applied to the restoration of gray-scale and color videos, seen as third order and fourth order tensors, respectively. The following table shows the results obtained after 20 iterations of Algorithm 5. For completeness, the Figure 5.3 shows the results obtained from the restoration of a video of size $360 \times 640 \times 30$ using Algorithm 5-HOSVD.

Table 5.5: *Comparison of the performance of the two approaches for videos restoration of different sizes.*

Size	$R = m$	**Method**	**Relative error**	**CPU-time(sec)**
$200 \times 200 \times 30$	150	Algorithm 5-CP	2.79×10^{-2}	1.15
	150	Algorithm 5-HOSVD	3.12×10^{-2}	0.25
$360 \times 640 \times 30$	200	Algorithm 5-CP	7.79×10^{-2}	4.56
	200	Algorithm 5-HOSVD	3.13×10^{-2}	1.19
$360 \times 640 \times 100$	200	Algorithm 5-HOSVD	5.75×10^{-2}	3.49
$200 \times 200 \times 3 \times 20$	150	Algorithm 5-HOSVD	3.13×10^{-2}	0.68
$360 \times 640 \times 3 \times 30$	200	Algorithm 5-HOSVD	3.17×10^{-2}	6.82

Note that in the last three experiments displayed in Table 5.5, we present only the results associated to the restoration obtained with algorithm 5-HOSVD, due to the time needed to built the approximate solution tensor.

Figure 5.3: *Example 2. First row: original frames. Second row: Blurred frames. Third row: Restored frames.*

5.7 Conclusion

In this chapter, we proposed a new approach to solve the tensor least squares minimization problem (5.2). We worked under the assumption that the right-hand side tensor is written (or approximated) using CP or higher order singular value decomposition (HOSVD) format. Our goal is to solve the problem (5.2) for higher dimensions, by applying Golub-Kahan bidiagonalization process to each coefficient matrix $A^{(i)}$ for $i = 1, \ldots, N$, and using an LSQR-like method to construct the approximate solution.The presented numerical examples show the effectiveness of the proposed approach.

Chapter 6

Tensor products with application to face recognition

In this chapter, we explore the use of the t-product, the cosine product, and the outer product, applied to face recognition. The proposed approaches are based on using tensor decompositions of an arrangement of images in a database, when we add a factor such as illumination, view angle, or expression. Our algorithms can be applied to a database of images represented by a third or fourth-order tensor. In the numerical results, we compare our approaches with some of the existing methods based on tensor format.

6.1 Introduction

Face recognition represents one of the most successful applications of image analysis. It has recently gained increasing interest [14, 26, 46]. It is due to several applications such as security and video-surveillance. In the past few years, there have been multiple methods being developed to achieve high performance. The PCA (Principal Component Analysis), often goes by the name "eigenfaces" is one of those techniques which is based on matrix decomposition. The eigenfaces form a basis set of all images used to built the covariance matrix. This generates dimension reduction by letting the smaller set of basis images to represent the original training images. Classification can

be obtained by comparing how faces are represented by the basis set. This approach was generalized in [40] using the t-product where the images are considered as two-dimensional rather than vectorized objects. However, this method works better when all pictures are taken under similar conditions, and it does not perform well when several factors are varied such as illumination, view angel, and expression. Recently, the face recognition problem was considered using a tensor model. By letting the modes of the tensor represent a different viewing condition, e.g., illumination or facial expression, the precision of the recognition algorithms is improved compared to the PCA method. This tensor representation was recently considered in several works. For instance, in [26, 77] the High-Order SVD (HOSVD, see [24, 74]) is used to classify the image of an unknown person, and in [14] the authors explore the use of the Tensor-Train decomposition (for TT-decomposition, see [64]) for multi-feature recognition strategies. In this chapter, we are interested in using the tensor decompositions: QR and SVD defined via the t-product or the cosine product, and the CP decomposition defined via the outer product. The approach based on the t-product and the cosine product differs from the one given in [40], in terms of the data representation. In fact, by adding a factor such as illumination, view angel or expression to construct a dataset of images, we use third-order tensors to represent the dataset of images, when the images are in vectorized format, or as fourth-order tensors, when the images are considered as a two-dimensional array. This approach can also be extended to the case of fifth-order tensors when the images are considered as third-order arrays. Note that in [40], the representation is limited to the case of third-order tensors.

The remainder of the chapter is organized as follows. In sections 6.2 and 6.3, we recall some properties associated to the t-product and the cosine product between third-order tensors, at the same time, we establish some properties for general higher-order tensors. In section 6.4, we give a brief introduction to the CP decomposition. In section 6.5, we present new algorithms for tensor face recognition problem, using t-product, cosine product, and CP decomposition. Finally, numerical examples are presented in Section 6.6 that show the effectiveness of the proposed approaches.

6.2 t-product, definition and properties

In this part, we recall some definitions and properties related to the t-product in case of third-order tensors [54, 52]. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, the operators **bcirc**, **unfold** and **fold** are described by:

$$\textbf{bcirc}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & A^{(n_3)} & A^{(n_3-1)} & \dots & A^{(2)} \\ A^{(2)} & A^{(1)} & A^{(n_3)} & \dots & A^{(3)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A^{(n_3)} & A^{(n_3-1)} & \dots & A^{(2)} & A^{(1)} \end{bmatrix}, \ \textbf{unfold}(\mathcal{A}) = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n_3)} \end{bmatrix},$$

bcirc is a block circulant matrix of size $n_1 n_3 \times n_2 n_3$, and the command **unfold**$(\mathcal{A})$ takes the tensor $\mathcal{A}$ into a block $n_1 n_3 \times n_2$ matrix. The operation **fold** takes **unfold**$(\mathcal{A})$ back to the tensor form:

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

Before stating some properties associated to the block circulant matrices, it is helpful to look at the simplest form of a block-circulant matrix, that is a circulant matrix [36, 57].

Definition 6.1. A square matrix C of size $n \times n$ is a circulant matrix if it has the following form:

$$C = \begin{bmatrix} r_1 & r_n & r_{n-1} & \cdots & r_2 \\ r_2 & r_1 & r_n & \cdots & r_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ r_n & r_{n-1} & \cdots & r_2 & r_1 \end{bmatrix},$$

where each row is a cyclic shift of the row above to the right.

The matrix C is clearly determined by its first column $r = (r_1, r_2, \ldots, r_n)$. Therefore, the above circulant matrix is also denoted by **circ**(r). Let Z be an $n \times n$ defined by:

$$Z = \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix}.$$

Then it is seen easily that

$$C = \mathbf{circ}(r) = \sum_{i=1}^{n} r_i Z^{i-1}.$$

The most important property of circulant matrices [32] is that they are diagonalizable by the Fourier matrix F_n:

$$C = F_n \Lambda F_n^*,$$

where $\Lambda = diag(F_n r)$ is a diagonal matrix with the eigenvalues of C, F_n^* is the conjugate transpose of F_n. Just as in the case of circulant matrices, a block circulant matrix can be diagonalized by the Discrete Fourier Transform (DFT) [53]. In this case a block circulant matrix can be a block diagonalized.

Lemma 6.2. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, we have:*

$$\boldsymbol{bcirc}(\mathcal{A}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \boldsymbol{circ}(\mathcal{A}(i, j, :)) \otimes e_i e_j^T. \tag{6.1}$$

Proof. Let $e_k \in \mathbb{R}^{n_2 n_3}$, then e_k can be written as $e_l \otimes e_m$, with $e_m \in \mathbb{R}^{n_2}$, $e_l \in \mathbb{R}^{n_3}$, and $k = l + (m-1)n_3$. In analogue to the case of circulant matrices, we have

$$\mathbf{bcirc}(\mathcal{A}) = I \otimes A^{(1)} + Z \otimes A^{(2)} + \cdots + Z^{n_3 - 1} \otimes A^{(n_3)} = \sum_{i=1}^{n_3} Z^{i-1} \otimes A^{(i)}.$$

Follows from a simple calculations, that $Z^i e_l = e_{i+l}$, with all indices are interpreted *mod* n_3, which mean if $i+l > n_3$, then $i+l$ becomes $\hat{l}$ with $i+l = \hat{l} + n_3$.

$$\begin{aligned}
\mathbf{bcirc}(\mathcal{A})e_k &= \mathbf{bcirc}(\mathcal{A})e_l \otimes e_m \\
&= e_l \otimes A^{(1)}e_m + e_{l+1} \otimes A^{(2)}e_m + \cdots + e_{l-1} \otimes A^{(n_3)}e_m \\
&= e_l \otimes \sum_{i=1}^{n_1} A^{(1)}(i,m)e_i + e_{l+1} \otimes \sum_{i=1}^{n_1} A^{(2)}(i,m)e_i + \cdots + e_{l-1} \otimes \sum_{i=1}^{n_1} A^{(n_3)}(i,m)e_i \\
&= \sum_{i=1}^{n_1} \left(A^{(1)}(i,m)e_l + A^{(2)}(i,m)e_{l+1} + \cdots + A^{(n_3)}(i,m)e_{l-1} \right) \otimes e_i.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\left(\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{circ}(\mathcal{A}(i,j,:)) \otimes e_i e_j^T \right) e_k &= \left(\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{circ}(\mathcal{A}(i,j,:)) \otimes e_i e_j^T \right) e_l \otimes e_m \\
&= \sum_{i=1}^{n_1} \mathbf{circ}(\mathcal{A}(i,m,:))e_l \otimes e_i \\
&= \sum_{i=1}^{n_1} \left(A^{(1)}(i,m)I + A^{(2)}(i,m)Z + \cdots + A^{(n_3)}(i,m)Z^{n_3-1} \right) e_l \otimes e_i \\
&= \sum_{i=1}^{n_1} \left(A^{(1)}(i,m)e_l + A^{(2)}(i,m)e_{l+1} + \cdots + A^{(n_3)}(i,m)e_{l-1} \right) \otimes e_i.
\end{aligned}$$

Therefore, the k-th column of $\mathbf{bcirc}(\mathcal{A})$ and $\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{circ}(\mathcal{A}(i,j,:)) \otimes e_i e_j^T$ are equal. This proves (6.1). □

Theorem 6.3. *The matrix* $\boldsymbol{bcirc}(\mathcal{A})$ *is block diagonalizable. We have:*

$$(F_{n_3} \otimes I_{n_1})\boldsymbol{bcirc}(\mathcal{A})(F_{n_3}^\star \otimes I_{n_2}) = \begin{bmatrix} D^{(1)} & & & \\ & D^{(2)} & & \\ & & \ddots & \\ & & & D^{(n_3)} \end{bmatrix}.$$

Proof. It follows from:

$$\mathbf{bcirc}(\mathcal{A}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{circ}(\mathcal{A}(i,j,:)) \otimes e_i e_j^T.$$

□

Definition 6.4 ([52, 54])**.** The t-product $*_t$ between two tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$ is a tensor given by:

$$\mathcal{C} = \mathcal{A} *_t \mathcal{B} = \textbf{fold}(\textbf{bcirc}(\mathcal{A})\textbf{unfold}(\mathcal{B})), \tag{6.2}$$

$\mathcal{C}$ is of size $n_1 \times m_2 \times n_3$.

Since $\textbf{bcirc}(\mathcal{A})$ is a block diagonalized using the discrete Fourier transform. The t-product can be computed as

$$\begin{aligned} \textbf{bcirc}(\mathcal{A})\textbf{unfold}(\mathcal{B}) &= (F_{n_3}^{\star} \otimes I_{n_1})(F_{n_3} \otimes I_{n_1})\textbf{bcirc}(\mathcal{A})(F_{n_3}^{\star} \otimes I_{n_2})(F_{n_3} \otimes I_{n_2})\textbf{unfold}(\mathcal{B}) \\ &= (F_{n_3}^{\star} \otimes I_{n_1}) \begin{bmatrix} D^{(1)} & & & \\ & D^{(2)} & & \\ & & \ddots & \\ & & & D^{(n_3)} \end{bmatrix} \textbf{unfold}(\hat{\mathcal{B}}), \end{aligned}$$

with $\textbf{unfold}(\hat{\mathcal{B}})) = (F_{n_3} \otimes I_{n_2})\textbf{unfold}(\mathcal{B})$. This can be computed by applying the Fast Fourier Transform (FFT) along the tubes of $\mathcal{B}$. Using the FFT along the third dimension, the t-product can be computed using the following algorithm

Algorithm 6 t-Product Computing Using FFT [52]

1: **input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$

2: **output :** $\mathcal{C} = \mathcal{A} *_t \mathcal{B} \in \mathbb{R}^{n_1 \times m_2 \times n_3}$

3: $\hat{\mathcal{A}} \longleftarrow fft(\mathcal{A}, [], 3)$, $\hat{\mathcal{B}} \longleftarrow fft(\mathcal{B}, [], 3)$

4: **for** $i = 1, 2, \ldots, n_3$ **do**

5: $\quad \hat{\mathcal{C}}(:, :, i) = \hat{\mathcal{A}}(:, :, i)\hat{\mathcal{B}}(:, :, i)$

6: **end for**

7: $\mathcal{C} \longleftarrow ifft(\hat{\mathcal{C}}, [], 3)$

The command $fft(\mathcal{A}, [], 3)$ is a Matlab command that computes the fast Fourier transform (FFT) on the third dimension of a multiway array. In the next part of this section, we introduce some properties and definitions associated to the t-product, that we are

going to use in this work.

Proposition 6.5. *The tensor $\mathcal{I}_{n_1 n_1 n_3}$, whose first frontal slice is the $n_1 \times n_1$ identity matrix, and whose other frontal slices are all zeros, is the identity element of the t-product:*

$$\mathcal{I}_{n_1 n_1 n_3} *_t \mathcal{A} = \mathcal{A} *_t \mathcal{I}_{n_1 n_1 n_3} = \mathcal{A}.$$

Proof. Easy to verify from Definition 6.4. □

Definition 6.6. A third-order tensor is f-diagonal if each frontal slice is a diagonal matrix. Likewise, a tensor is f-upper triangular or f-lower triangular if each frontal slice is upper or lower triangular, respectively.

Definition 6.7. The transpose of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a tensor $\mathcal{A}^T \in \mathbb{R}^{n_2 \times n_1 \times n_3}$ obtained by transposing each of the frontal slices and then reversing the order of transposed frontal slices 2 through n_3.

Definition 6.8. $\mathcal{Q} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ is said to be an orthogonal tensor if

$$\mathcal{Q} *_t \mathcal{Q}^T = \mathcal{Q}^T *_t \mathcal{Q} = \mathcal{I}_{n_1 n_1 n_3}.$$

Properties 6.2.1. [52]

1. The t-product is associative: Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be third-order tensors of appropriate size, then
$$\mathcal{A} *_t (\mathcal{B} *_t \mathcal{C}) = (\mathcal{A} *_t \mathcal{B}) *_t \mathcal{C}.$$

2. Let $\mathcal{A}$, $\mathcal{B}$ be third-order tensors such that $\mathcal{A} * \mathcal{B}$ and $\mathcal{B}^T *_t \mathcal{A}^T$ are defined, then
$$(\mathcal{A} *_t \mathcal{B})^T = \mathcal{B}^T *_t \mathcal{A}^T.$$

Proposition 6.9. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$ be third-order tensors, for $i = 1, \ldots, n_1$, $j = 1, \ldots, m_2$, we have the following equalities,*

$$(\mathcal{A} *_t \mathcal{B})(i,:,:) = \mathcal{A}(i,:,:) *_t \mathcal{B},\ (\mathcal{A} *_t \mathcal{B})(:,j,:) = \mathcal{A} *_t \mathcal{B}(:,j,:). \tag{6.3}$$

$$(\mathcal{A} *_t \mathcal{B})(i,j,:) = \mathcal{A}(i,:,:) *_t \mathcal{B}(:,j,:). \tag{6.4}$$

Proof. From the definition of the t- product, we have

$$\mathcal{A}(i,:,:) = \mathcal{I}_{n_1 n_1 n_3}(i,:,:) *_t \mathcal{A}, \quad \textbf{and} \quad \mathcal{B}(:,j,:) = \mathcal{B} *_t \mathcal{I}_{m_2 m_2 n_3}(:,j,:). \tag{6.5}$$

As consequence

$$(\mathcal{A} *_t \mathcal{B})(i,:,:) = \mathcal{I}_{n_1 n_1 n_3}(i,:,:) *_t (\mathcal{A} *_t \mathcal{B}) = (\mathcal{I}_{n_1 n_1 n_3}(i,:,:) *_t \mathcal{A}) *_t \mathcal{B} = \mathcal{A}(i,:,:) *_t \mathcal{B},$$

and

$$(\mathcal{A} *_t \mathcal{B})(:,j,:) = (\mathcal{A} *_t \mathcal{B}) *_t (:,j,:) = \mathcal{A} *_t (\mathcal{B} *_t \mathcal{I}_{m_2 m_2 n_3}(:,j,:)) = \mathcal{A} *_t \mathcal{B}(:,j,:).$$

Using the relations (6.5), we obtain

$$\begin{aligned}(\mathcal{A} *_t \mathcal{B})(i,j,:) &= \mathcal{I}_{n_1 n_1 n_3}(i,:,:) *_t \mathcal{A} *_t \mathcal{B} *_t \mathcal{I}_{m_1 m_1 n_3}(:,j,:) \\ &= (\mathcal{I}_{n_1 n_1 n_3}(i,:,:) *_t \mathcal{A}) *_t (\mathcal{B} *_t \mathcal{I}_{m_1 m_1 n_3}(:,j,:)) \\ &= \mathcal{A}(i,:,:) *_t \mathcal{B}(:,j,:).\end{aligned}$$

□

Theorem 6.10 ([52, 54])**.** *(t-SVD) Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ real-valued tensor. Then $\mathcal{A}$ can be factored as*

$$\mathcal{A} = \mathcal{U} *_t \mathcal{S} *_t \mathcal{V}^T, \tag{6.6}$$

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

The t-SVD can be computed using the fast Fourier transform, as described in the following algorithm

Algorithm 7 t-SVD Computing Using FFT

1: **input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$,

2: **output:** $\mathcal{U}, \mathcal{S}, \mathcal{V}$

3: $\hat{\mathcal{A}} \longleftarrow fft(\mathcal{A}, [\,], 3)$

4: **for** $i = 1, 2, \ldots, n_3$ **do**

5: $\left[\hat{\mathcal{U}}(:,:,i), \hat{\mathcal{S}}(:,:,i), \hat{\mathcal{V}}(:,:,i)\right] = svd(\hat{\mathcal{A}}(:,:,i))$

6: **end for**

7: $\mathcal{U} \longleftarrow ifft(\hat{\mathcal{U}}\,[\,], 3), \mathcal{S} \longleftarrow ifft(\hat{\mathcal{S}}\,[\,], 3), \mathcal{V} \longleftarrow ifft(\hat{\mathcal{V}}\,[\,], 3)$

Theorem 6.11 ([54]). *(t-QR) Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ real-valued tensor. Then $\mathcal{A}$ can be factored as*

$$\mathcal{A} = \mathcal{Q} *_t \mathcal{R}, \tag{6.7}$$

where $\mathcal{Q}$ is an orthogonal tensor of size $n_1 \times n_1 \times n_3$ and $\mathcal{R}$ is an f-upper triangular tensor of size $n_1 \times n_2 \times n_3$.

The t-QR can be computed using the fast Fourier transform as follows

Algorithm 8 t-QR Computing Using FFT

1: **input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$,

2: **output:** $\mathcal{U}, \mathcal{S}, \mathcal{V}$

3: $\hat{\mathcal{A}} \longleftarrow fft(\mathcal{A}, [\,], 3)$

4: **for** $i = 1, 2, \ldots, n_3$ **do**

5: $\left[\hat{\mathcal{Q}}(:,:,i), \hat{\mathcal{R}}(:,:,i)\right] = qr(\hat{\mathcal{A}}(:,:,i))$

6: **end for**

7: $\mathcal{Q} \longleftarrow ifft(\hat{\mathcal{Q}}\,[\,], 3), \mathcal{R} \longleftarrow ifft(\hat{\mathcal{R}}\,[\,], 3)$

6.2.1 Generalization of the t-product to higher-order tensors

The t-product can be extended to higher-order tensors, in a recursive manner(see [59]). For instance, in case of fourth-order tensors, the t-product is defined as follows

Definition 6.12. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3 \times n_4}$ be two fourth-order tensors. The t-product $\mathcal{A} *_t \mathcal{B}$ is a fourth-order tensor of size $n_1 \times m_2 \times n_3 \times n_4$ given by:

$$\mathcal{C} = \mathcal{A} *_t \mathcal{B} = \textbf{fold}(\textbf{bcirc}(\mathcal{A}) *_t \textbf{unfold}(\mathcal{B})), \tag{6.8}$$

where

$$\textbf{bcirc}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} & \mathcal{A}^{(n_4)} & \mathcal{A}^{(n_4-1)} & \dots & \mathcal{A}^{(2)} \\ \mathcal{A}^{(2)} & \mathcal{A}^{(1)} & \mathcal{A}^{(n_4)} & \dots & \mathcal{A}^{(3)} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \mathcal{A}^{(n_4)} & \mathcal{A}^{(n_4-1)} & \dots & \mathcal{A}^{(2)} & \mathcal{A}^{(1)} \end{bmatrix} \quad \text{and} \quad \textbf{unfold}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} \\ \mathcal{A}^{(2)} \\ \vdots \\ \mathcal{A}^{(n_4)} \end{bmatrix},$$

with $\mathcal{A}^{(i)} = \mathcal{A}(:,:,:,i)$, for $i = 1, \dots, n_4$.

Lemma 6.13. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$ be a fourth-order tensor, we have:*

$$\boldsymbol{bcirc}(\mathcal{A}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \boldsymbol{circ}(\mathcal{A}(i,j,k,:)) \otimes e_i \circ e_j \circ e_k. \tag{6.9}$$

Similarly, for higher-order tensors we get:

$$\boldsymbol{bcirc}(\mathcal{A}) = \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_{N-1}=1}^{n_{N-1}} \boldsymbol{circ}(\mathcal{A}(i_1, i_2, \cdots, i_{N-1}, :)) \otimes e_{i_1} \circ e_{i_2} \circ \cdots \circ e_{i_{N-1}}, \tag{6.10}$$

with $\mathcal{A}$ is a N-order tensor of size $n_1 \times n_2 \times \cdots \times n_N$.

Proof. For definiteness, we focus on proving (6.9), (6.10) is proved in recursive way. In analogue to the case of circulant matrices, we can write $\textbf{bcirc}(\mathcal{A})$ as follows,

$$\textbf{bcirc}(\mathcal{A}) = I \otimes \mathcal{A}^{(1)} + Z \otimes \mathcal{A}^{(2)} + \cdots + Z^{n_3-1} \otimes \mathcal{A}^{(n_3)} = \sum_{i=1}^{n_3} Z^{i-1} \otimes \mathcal{A}^{(i)}$$

Let $e_m \in \mathbb{R}^{n_3}$. Then

$$\begin{aligned} \textbf{bcirc}(\mathcal{A}) \times_3 e_m^T &= \sum_{i=1}^{n_3} Z^{i-1} \otimes \mathcal{A}^{(i)} \times_3 e_m^T \\ &= \sum_{i=1}^{n_3} Z^{i-1} \otimes \mathcal{A}^{(i)}(:,:,m). \end{aligned}$$

Based on Lemma 6.2, we obtain

$$\mathbf{bcirc}(\mathcal{A}) \times_3 e_m^T = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{circ}(\mathcal{A}(i,j,m,:)) \otimes e_i \circ e_j.$$

On the other hand, we have

$$\left(\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \mathbf{circ}(\mathcal{A}(i,j,k,:)) \otimes e_i \circ e_j \circ e_k \right) \times_3 e_m^T = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{circ}(\mathcal{A}(i,j,m,:)) \otimes e_i \circ e_j,$$

which proves that the m-th frontal slice of $\mathbf{bcirc}(\mathcal{A})$ and $\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \mathbf{circ}(\mathcal{A}(i,j,k,:)) \otimes e_i \circ e_j \circ e_k$ are equal. This proves (6.9). □

Theorem 6.14. *The tensor* $\boldsymbol{bcirc}(\mathcal{A})$ *is block diagonalizable.*

Proof. It follows from:

$$\mathbf{bcirc}(\mathcal{A}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \mathbf{circ}(\mathcal{A}(i,j,k,:)) \otimes e_i \circ e_j \circ e_k.$$

□

To compute the t-product, we apply the FFT along the fourth dimension. The associated algorithm is expressed as follows

Algorithm 9 Fourth-order t-Product using FFT

1: **input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3 \times n_4}$
2: **output:** $\mathcal{C} = \mathcal{A} *_t \mathcal{B} \in \mathbb{R}^{n_1 \times m_2 \times n_3 \times n_4}$
3: $\hat{\mathcal{A}} \longleftarrow fft(\mathcal{A}, [\,], 4)$, $\hat{\mathcal{B}} \longleftarrow fft(\mathcal{B}, [\,], 4)$
4: **for** $i = 1, 2, \ldots, n_4$ **do**
5: $\quad \hat{\mathcal{C}}(:,:,:,i) = \hat{\mathcal{A}}(:,:,:,i) *_t \hat{\mathcal{B}}(:,:,:,i)$
6: **end for**
7: $\mathcal{C} \longleftarrow ifft(\hat{\mathcal{C}}\,[\,], 4)$

Remark 6.2.1. In the same way, notations, definitions and decompositions associated to the t-product can be extended to higher-order tensors. Proposition 6.9 can be generalized also to higher-order tensors.

6.3 Cosine transform product, definition and properties

Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, we define

$$\mathbf{mat}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & A^{(2)} & \dots & A^{(n_3)} \\ A^{(2)} & A^{(1)} & \dots & A^{(n_3-1)} \\ \vdots & \ddots & \ddots & \vdots \\ A^{(n_3)} & A^{(n_3-1)} & \dots & A^{(1)} \end{bmatrix} + \begin{bmatrix} A^{(2)} & \dots & A^{(n_3)} & 0 \\ \vdots & \iddots & \iddots & A^{(n_3)} \\ A^{(n_3)} & 0 & \iddots & \vdots \\ 0 & A^{(n_3)} & \dots & A^{(2)} \end{bmatrix},$$

is block Toepltiz-plus-Hankel matrix. Here 0 denotes the zero matrix of size $n_1 \times n_2$. The command **ten**(.) is defined as the inverse of the **mat** operation:

$$\mathbf{ten}(\mathbf{mat}(\mathcal{A})) = \mathcal{A}.$$

Definition 6.15 ([50])**.** The cosine product $*_c$ between two tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$ is a tensor of size $n_1 \times m_2 \times n_3$ given by

$$\mathcal{C} = \mathcal{A} *_c \mathcal{B} = \mathbf{ten}(\mathbf{mat}(\mathcal{A})\mathbf{mat}(\mathcal{B})). \tag{6.11}$$

The cosine product can be computed without forming explicitly the block matrices in the previous definition. Indeed, let $y \in \mathbb{R}^n$ then **mat**(y) is a $n \times n$ Toeplitz-plus Hankel matrix. Let C_n denote the $n \times n$ orthogonal DCT matrix, we have

$$C_n \mathbf{mat}(y) C_n^T = diag(d),$$

where d is a vector of eigenvalues can be computed as

$$d = W^{-1}(C_n \mathbf{mat}(y) e_1),$$

where $W = diag(C_n(:, 1))$. Since $\mathbf{mat}(y)e_1 = (I + Z)vec(y)$, with Z is the $n \times n$ circulant up-shift matrix (In Matlab, we have: $Z = diag(ones(n-1, 1), 1)$). Therefore

$$d = W^{-1}(C_n(I + Z)vec(y)).$$

Let L be an operator defined as

$$\begin{aligned} L : \mathbb{R}^n &\longrightarrow \mathbb{R}^n \\ y &\longrightarrow L(y) = \underbrace{W^{-1}C_n(I+Z)}_{M} vec(y). \end{aligned}$$

Definition 6.16. Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3$ tensor. Then $\mathcal{L}(\mathcal{A}) = \hat{\mathcal{A}}$ is $n_1 \times n_2 \times n_3$ tensor whose tube fibers $\hat{a}_{i,j}$ are computed as

$$\hat{a}_{i,j} = \hat{\mathcal{A}}(i,j,:) = L(\mathcal{A}(i,j,:)) \quad i = 1,\ldots,n_1, \quad j = 1,\ldots,n_2.$$

Based on the definition of the n-mode product, we can notice that

$$\mathcal{L}(\mathcal{A}) = \mathcal{A} \times_3 M.$$

Since $\mathbf{mat}(\mathcal{A})$ and $\mathbf{mat}(\mathcal{B})$ are block diagonalized using the discrete cosine transform.

Theorem 6.17. *The matrix $\mathbf{mat}(\mathcal{A})$ is block diagonalizable by the matrix pair $C_{n_3} \otimes I_{n_1}$ and $C_{n_3}^T \otimes I_{n_2}$, with diagonal blocks given by the frontal slices of $\mathcal{L}(\mathcal{A})$.*

Proof. See [50] □

Using the previous theorem, the cosine product can be computed as

$$\begin{aligned} \mathbf{mat}(\mathcal{A})\mathbf{mat}(\mathcal{B}) &= (C_{n_3}^\star \otimes I_{n_1})(C_{n_3} \otimes I_{n_1})\mathbf{mat}(\mathcal{A})(C_{n_3}^\star \otimes I_{n_1})(C_{n_3} \otimes I_{n_1})\mathbf{mat}(\mathcal{B})(C_{n_3}^\star \otimes I_{n_1})(C_{n_3} \otimes I_{n_1}) \\ &= (C_{n_3}^\star \otimes I_{n_1}) \begin{bmatrix} \hat{\mathcal{A}}^{(1)} & & & \\ & \hat{\mathcal{A}}^{(2)} & & \\ & & \ddots & \\ & & & \hat{\mathcal{A}}^{(n_3)} \end{bmatrix} \begin{bmatrix} \hat{\mathcal{B}}^{(1)} & & & \\ & \hat{\mathcal{B}}^{(2)} & & \\ & & \ddots & \\ & & & \hat{\mathcal{B}}^{(n_3)} \end{bmatrix} (C_{n_3} \otimes I_{n_1}). \end{aligned}$$

The cosine product can be computed using the following algorithm:

Algorithm 10 Cosine Product: third-order tensor [50]

1: **input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$

2: **output:** $\mathcal{C} = \mathcal{A} *_c \mathcal{B} \in \mathbb{R}^{n_1 \times m_2 \times n_3}$

3: $\hat{\mathcal{A}} \longleftarrow \mathcal{L}(\mathcal{A})$, $\hat{\mathcal{B}} \longleftarrow \mathcal{L}(\mathcal{B})$

4: **for** $i = 1, 2, \ldots, n_3$ **do**

5: $\quad \hat{\mathcal{C}}(:,:,i) = \hat{\mathcal{A}}(:,:,i)\hat{\mathcal{B}}(:,:,i)$

6: **end for**

7: $\mathcal{C} \longleftarrow \mathcal{L}^{-1}(\hat{\mathcal{C}})$

Properties 6.3.1. [50]

1. The cosine product is associative: Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be third-order tensors of appropriate size, then
$$\mathcal{A} *_c (\mathcal{B} *_c \mathcal{C}) = (\mathcal{A} *_c \mathcal{B}) *_c \mathcal{C}.$$

2. Let $\mathcal{A}$, $\mathcal{B}$ be third-order tensors such that $\mathcal{A} *_c \mathcal{B}$ and $\mathcal{B}^T *_c \mathcal{A}^T$ are defined, then
$$(\mathcal{A} *_c \mathcal{B})^T = \mathcal{B}^T *_c \mathcal{A}^T.$$

Proposition 6.18. *[50] Let $\hat{\mathcal{J}} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ be so that $\hat{\mathcal{J}}(:,:,i) = I_{n_1 \times n_1}$. Then $\mathcal{L}^{-1}(\hat{\mathcal{J}}) = \mathcal{J}$ is the identity element under $\mathcal{L}$*

Proposition 6.19. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}, \mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$. Then*
$$(\mathcal{A} *_c \mathcal{B})(i,j,;) = \mathcal{A}(i,:,;) *_c \mathcal{B}(:,j,;) \quad i = 1, \ldots, n_1, \quad j = 1, \ldots, m_2.$$

Proof. We have

$$\mathcal{J}(i,:,:) *_c \mathcal{A} = \mathcal{A}(i,:,:) \quad for\, i = 1, \ldots, n_1 \quad and \quad \mathcal{B} *_c \mathcal{J}(:,j,:) = \mathcal{B}(:,j,:) \quad for\, j = 1, \ldots, m_2.$$

Then

$$\begin{aligned}(\mathcal{A} *_c \mathcal{B})(i,j,;) &= \mathcal{J}(i,:,:) *_c (\mathcal{A} *_c \mathcal{B}) *_c \mathcal{J}(:,j,:) \\ &= (\mathcal{J}(i,:,:) *_c \mathcal{A}) *_c (\mathcal{B} *_c \mathcal{J}(:,j,:)) \\ &= \mathcal{A}(i,:,;) *_c \mathcal{B}(:,j,;).\end{aligned}$$

□

Lemma 6.20. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3}$ and $\mathcal{C} \in \mathbb{R}^{m_2 \times p_2 \times n_3}$, we have*

$$(\mathcal{A} *_c \mathcal{B} *_c \mathcal{C})(i,j,:) = \mathcal{A}(i,:,:) *_c \mathcal{B} *_c \mathcal{C}(:,j,:).$$

6.3.1 Generalization of the cosine product to higher-order tensors

In the same way as the t-product, the cosine product can be extended to higher-order tensors. In fact, in case of fourth-order tensors, we have:

Definition 6.21. Let $\mathcal{A}$ be an $n_1 \times n_2 \times n_3 \times n_4$ tensor. Then $\mathcal{L}(\mathcal{A}) = \hat{\mathcal{A}}$ is $n_1 \times n_2 \times n_3 \times n_4$ is a tensor whose tube fibers $\hat{a}_{i,j,k}$ are computed as

$$\hat{a}_{i,j,k} = \hat{\mathcal{A}}(i,j,k,:) = L(\mathcal{A}(i,j,k,:)) \quad i = 1,\ldots,n_1, \quad j = 1,\ldots,n_2 \quad k = 1,\ldots,n_3.$$

Another way to compute $\hat{\mathcal{A}}$ is to notice that

$$\mathcal{L}(\mathcal{A}) = \mathcal{A} \times_4 M.$$

The cosine product can be computed using the following algorithm

Algorithm 11 Cosine Product: fourth-order tensor

1: **input:** $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$, $\mathcal{B} \in \mathbb{R}^{n_2 \times m_2 \times n_3 \times n_4}$
2: **output:** $\mathcal{C} = \mathcal{A} *_c \mathcal{B} \in \mathbb{R}^{n_1 \times m_2 \times n_3 \times n_4}$
3: $\hat{\mathcal{A}} \longleftarrow \mathcal{L}(\mathcal{A})$, $\hat{\mathcal{B}} \longleftarrow \mathcal{L}(\mathcal{B})$
4: **for** $i = 1, 2, \ldots, n_4$ **do**
5: $\quad \hat{\mathcal{C}}(:,:,:,i) = \hat{\mathcal{A}}(:,:,:,i) *_c \hat{\mathcal{B}}(:,:,:,i)$
6: **end for**
7: $\mathcal{C} \longleftarrow \mathcal{L}^{-1}(\hat{\mathcal{C}})$

Remark 6.3.1. In the same way as the t-product, the matrix SVD and QR can be generalized using the cosine product for higher-order tensors.

6.4 CP decomposition

Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an N^{th}-order tensor. The CP decomposition [20, 41, 51, 56] of $\mathcal{A}$ is given by

$$\mathcal{A} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)},$$

where $a_r^{(k)}$ are vectors of size I_k with $1 \leq k \leq N$ and R is a positive integer. A CP decomposition of a tensor $\mathcal{A}$ is called an exact CP decomposition if $R = rank(\mathcal{A})$, with $rank(\mathcal{A})$ [56] represent the rank of the tensor $\mathcal{A}$ defined as the smallest number of rank-one tensors that generate $\mathcal{A}$ as their sum. Unlike matrices, who's the best rank-R approximation is given by the leading R factors of the SVD, the rank of a specific given tensor is hard to define [42]. In practice, the rank of a tensor is determined numerically by fitting various rank-R CP models. But an interesting property associated with CP decomposition for higher-order tensors is uniqueness under some conditions [41, 58]. If we define $A_n = \left[a_1^{(n)} a_2^{(n)} \cdots a_R^{(n)}\right]$ for $n \in \{1, \ldots, N\}$, the CP decomposition can be symbolically written as

$$\mathcal{A} = A_1 \circ A_2 \circ \cdots \circ A_N,$$

the matrices $A_n \in \mathbb{R}^{I_n \times R}$ are called factor matrices. Often, the vectors $a_r^{(n)}$ are chosen such that $\left\| a_r^{(n)} \right\| = 1$. In this case, the CP decomposition is written as

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)},$$

where λ_r is a scalar that compensates for the magnitudes of vectors $a_r^{(n)}$. Using the n-mode multiplication of a tensor by a matrix, we obtain the following representation:

$$\mathcal{A} = \mathcal{S} \times_1 A_1 \times_2 \cdots \times_N A_N,$$

where $\mathcal{S} \in \mathbb{R}^{R\times R\times\cdots\times R}$, with entries

$$\mathcal{S}_{i_1,\cdots,i_N} = \begin{cases} \lambda_r & \text{for} \quad i_1 = i_2 = \cdots = i_N = r, \\ 0 & \text{otherwise.} \end{cases}$$

An another representation for the CP decomposition is using slices:

$$\mathcal{A}(:,:,i_3,\cdots,i_N) = A_1 \Lambda D(A(:,i_3)) \ldots D(A(:,i_N)) A_2^T,$$

where $D(A(:,i)) = diag(A(:,i))$, $\Lambda = diag(\lambda)$, and $\lambda = [\lambda_1, \ldots, \lambda_r]$.

6.5 Application to face recognition

In this section, we are going to present an application to face recognition of the tensor decompositions discussed in this chapter. We first use the CP decomposition and then the QR and SVD decompositions defined via the t-product or the cosine product. Assuming that we have a collection of images of n_p persons, where each image is an $m_{i_1} \times m_{i_2}$ array with $m_{i_1} m_{i_2} = n_i$. Further, we assume that each person has been photographed with n_e different facial expressions. Given an image z of an unidentified person in an unknown expression, represented by a vector in $\mathbb{R}^{n_i}$ or by a matrix of size $m_{i_1} \times m_{i_2}$, the aim is to determine which of the n_p persons the new image is closest to.

6.5.1 CP decomposition classification

6.5.1.1 Third-order classification

Consider $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$. The CP decomposition associated to $\mathcal{A}$ is given by

$$\mathcal{A} = \mathcal{S} \times_i F \times_e G \times_p H,$$

where $\times_i, \times_e, \times_p$ are the 1-mode, 2-mode, and 3-mode multiplications, respectively. If we fix the expression e and the person p, the image of a person p in expression e is given

by

$$\mathcal{A}(:, e, p) = F\Lambda D\left(g_e\right) h_p^T, \tag{6.12}$$

where $g_e = G(e, :)$, $h_p = H(p, :)$, and $\Lambda = D(\lambda)$. Then (6.12) can be expressed as follows

$$\mathcal{A}(:, e, p) = C^e h_p^T,$$

where $C^e = F\Lambda D\left(g_e\right)$. The columns of C^e are basis vectors for the expression e and the row p of H, i.e., h_p, holds the coordinates of the image of person p in this basis. Furthermore, the same h_p holds the coordinates of the images of person p in all expression bases. Then, if the image z that we want classify is an image of person p in expression e, then the coordinates of z in that basis are equal to h_p. Thus, we can classify z by computing its coordinates in all the expression bases and testing, for each expression, whether the coordinates of z coincide or nearly coincide with the elements of any row of H. The coordinates of z in the expression basis e can be obtained by solving the least squares problem

$$\min_{\alpha_e} \|C^e \alpha_e - z\| \quad \text{for} \quad e = 1, \ldots, n_e.$$

Then, for each $e = 1, \ldots, n_e$ and for each $p = 1, \ldots, n_p$, we compute

$$d(e, p) = \left\| \alpha_e - h_p^T \right\|.$$

We classify z as person $\hat{p}$ in expression $\hat{e}$, where $(\hat{e}, \hat{p}) = \underset{e,p}{\operatorname{argmin}}\, d(e, p)$.

Algorithm 12

1: **input:** Collection of images $\mathcal{A} \in \mathbb{R}^{n_i \times n_e \times n_p}$, $z \in \mathbb{R}^{n_i}$ image to classify
2: Decompose $\mathcal{A}$ as $\mathcal{A} = \mathcal{S} \times_i F \times_e G \times_p H$
3: **for** $e = 1; 2, \ldots, n_e$ **do**
4: Solve $\min_{\alpha_e} \|C^e \alpha_e - z\|$ with $C^e = F\Lambda D\left(g_e\right)$
5: **for** $p = 1; 2, \ldots, n_p$ **do**
6: Compute $d(e,p) = \left\|\alpha_e - h_p^T\right\|$
7: **end for**
8: **end for**
9: Classify z as person $\hat{p}$ in expression $\hat{e}$, where $(\hat{e}, \hat{p}) = \underset{e,p}{\operatorname{argmin}}\, d(e,p)$

6.5.1.2 Fourth-order classification

Now, we consider images as matrices of size $m_1 \times m_2$, which mean, the collection of n_p persons in n_e expressions is stored as fourth-order tensor $\mathcal{A} \in \mathbb{R}^{m_1 \times m_2 \times n_e \times n_p}$. The CP decomposition of the tensor $\mathcal{A}$ is expressed as follows

$$\mathcal{A} = \mathcal{S} \times_{i_1} F_1 \times_{i_2} F_2 \times_e G \times_p H,$$

where $\mathcal{S} \in \mathbb{R}^{R \times R \times R \times R}$, $\times_{i_1}, \times_{i_2}, \times_e, \times_p$ are the 1-mode, 2-mode, 3-mode, and 4-mode multiplications, respectively. Thus, the image of a person p in expression e in the database is given by

$$\mathcal{A}(:,:,e,p) = F_1 \Lambda D\left(g_e\right) D\left(h_p\right) F_2^T, \tag{6.13}$$

where $g_e = G(e,:), h_p = H(p,:)$. If we set $\mathcal{C} = \mathcal{L} \times_{i_1} F_1 \times_{i_2} F_2 \times_e G$, (6.13) becomes

$$\mathcal{A}(:,:,e,p) = \mathcal{C}^e \times_p h_p,$$

where $\mathcal{C}^e = \mathcal{C}(:,:e,:)$. So to classify the person $z \in \mathbb{R}^{m_1 \times m_2}$, we compute for each $e = 1, \ldots, n_e$ and for each $p = 1, \ldots, n_p$

$$d(e,p) = \left\|\alpha_e - h_p^T\right\|,$$

where α_e is solution of the minimization problem

$$\min_{\alpha_e} \left\| \mathcal{C}^e \times_p \alpha_e - z \right\| \quad \text{for} \quad e = 1, \ldots, n_e.$$

Equivalently, we can set α_e as solution of the following minimization problem

$$\min_{\alpha_e} \left\| F_1 D\left(g_e\right) X_{\alpha_e} F_2^T - z \right\| \quad \text{where} \quad X_{\alpha_e} = D\left(\alpha_e\right). \tag{6.14}$$

If we set $F_1^e = F_1 D\left(g_e\right)$, the minimization problem (6.14) becomes

$$\min_{\alpha_e} \left\| F_1^e X_{\alpha_e} F_2^T - z \right\| \quad \text{where} \quad X_{\alpha_e} = D\left(\alpha_e\right). \tag{6.15}$$

Using the vec operator, the minimization problem (6.15) is equivalent to the following minimization

$$\min_{\alpha_e} \left\| M^e \alpha_e - \text{vec}(z) \right\| \quad \text{where} \quad M^e = F_2 \odot F_1^e.$$

The symbol $\odot$ denotes the Khatri-Rao product [56]. Then, we classify $z \in \mathbb{R}^{m_1 \times m_2}$ as person $\hat{p}$ in expression $\hat{e}$, where $(\hat{e}, \hat{p}) = \underset{e,p}{\text{argmin}}\, d(e, p)$

Algorithm 13

1: **input:** Collection of images $\mathcal{A} \in \mathbb{R}^{m_1 \times m_2 \times n_e \times n_p}$, $z \in \mathbb{R}^{m_1 \times m_2}$ image to classify
2: Decompose $\mathcal{A}$ as $\mathcal{A} = \mathcal{S} \times_{i_1} F_1 \times_{i_2} F_2 \times_e G \times_p H$
3: **for** $e = 1, 2, \ldots, n_e$ **do**
4: Solve $\min_{\alpha_e} \left\| M^e \alpha_e - \text{vec}(z) \right\|$ where $M^e = F_2 \odot F_1^e$, and $F_1^e = F_1 D\left(g_e\right)$
5: **for** $p = 1, 2, \ldots, n_p$ **do**
6: Compute $d(e, p) = \left\| \alpha_e - h_p^T \right\|$
7: **end for**
8: **end for**
9: Classify z as person $\hat{p}$ in expression $\hat{e}$, where $(\hat{e}, \hat{p}) = \underset{e,p}{\text{argmin}}\, d(e, p)$

6.5.2 Cosine product and t-product decompositions classification

In the next sections, $*$ denote the t-product or the cosine product. The collection of images is stored as a third-order tensor:

$$\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times n_i},$$

or as a fourth-order tensor:

$$\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times m_1 \times m_2}.$$

6.5.2.1 Third-order classification

Using the tensor QR factorization, the tensor $\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times n_i}$ can be written as follows

$$\mathcal{A} = \mathcal{Q} * \mathcal{R}. \tag{6.16}$$

For a fixed expression e, and using Theorem 6.9, $\mathcal{A}(:, e, :)$ can be written as

$$\mathcal{A}(:, e, :) = \mathcal{Q} * \mathcal{R}(:, e, :), \quad for\ e = 1, \ldots, n_e. \tag{6.17}$$

Thus, the image of a person p in expression e is expressed by

$$\mathcal{A}(p, e, :) = \mathcal{Q}(p, :, :) * \mathcal{R}(:, e, :), \quad for \quad e = 1, \ldots, n_e,\ p = 1, \ldots, n_p. \tag{6.18}$$

Given an image $z \in \mathbb{R}^{n_i}$ of an unknown person in an unknown expression, that we want to classify. The coordinates of z in the expression basis 'e' can be found by solving the following least squares problem

$$\min_{\alpha_e} \left\| \mathcal{Z} - \alpha_e * \mathcal{R}(:, e, :) \right\|, \tag{6.19}$$

with $\mathcal{Z} \in \mathbb{R}^{1 \times 1 \times n_i}$, such that $\mathcal{Z}(1, 1, :) = z$. Then, for each $e = 1, \ldots, n_e$ and for each $p = 1, \ldots, n_p$, we compute:

$$\left\| \alpha_e - \mathcal{Q}(p, :, :). \right\|. \tag{6.20}$$

We can then recognize the person in the image by attributing the label corresponding to the closest match.

Algorithm 14

input: Collection of images $\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times n_i}$, z image to classify
Decompose $\mathcal{A}$ as $\mathcal{A} = \mathcal{Q} * \mathcal{R}$
$\mathcal{Z}(1,1,:) = z(:)$
for $e = 1, 2, \ldots, n_e$ **do**
    Solve $\min_{\alpha_e} \|\mathcal{Z} - \alpha_e * \mathcal{R}(:,e,:)\|$
    for $p = 1, 2, \ldots, n_p$ **do**
        if then $\|\alpha_e - \mathcal{Q}(p,:,:)\| < tol$, then classify z as the person p
        end if
    end for
end for

In the same way. Applying the tensor SVD decomposition to the third-order tensor $\mathcal{A}$, that contain the images data leads to the following relation

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T. \tag{6.21}$$

Fixing a particular value of the expression parameter, i.e $j = e$ corresponds to using $\mathcal{V}(:,e,:)$. In other word, we have

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T(:,e,:). \tag{6.22}$$

Thus, the image of a person p in the expression e is defined by

$$\mathcal{A}(:,e,p) = \mathcal{U}(p,:,:) * \mathcal{S} * \mathcal{V}^T(:,e,:). \tag{6.23}$$

The classification strategy is then analogous to the case of QR factorization, given an new image to be classified $z \in \mathbb{R}^{n_i}$. The distance of z from the person p is given by

$$\|\alpha_e - \mathcal{U}(p,:,:)\|, \tag{6.24}$$

where α_e is solution of the problem

$$\min_{\alpha_e} \left\| \mathcal{Z} - \alpha_e * \mathcal{S} * \mathcal{V}^T(:, e, :) \right\|, \tag{6.25}$$

with $\mathcal{Z} \in \mathbb{R}^{1\times 1\times n_i}$, such that $\mathcal{Z}(1, 1, :) = z$.

6.5.2.2 Fourth-order classification

In this section, we extend the classification method defined previously to the case of fourth-order tensors. We consider images as matrices of size $m_1 \times m_2$, instead of vectors. Using the t-QR or the cosine QR factorization, the tensor $\mathcal{A} \in \mathbb{R}^{n_p\times n_e\times m_1\times m_2}$ can be written as follows

$$\mathcal{A} = \mathcal{Q} * \mathcal{R}. \tag{6.26}$$

For a fixed expression e, $\mathcal{A}(:, e, :, :)$ can be written as

$$\mathcal{A}(:, e, :, :) = \mathcal{Q} * \mathcal{R}(:, e, :, :), \quad for\ e = 1, \ldots, n_e. \tag{6.27}$$

Thus, the image of a person p in the expression e is expressed by

$$\mathcal{A}(p, e, :, :) = \mathcal{Q}(p, :, :, :) * \mathcal{R}(:, e, :, :), \quad for \quad e = 1, \ldots, n_e,\ p = 1, \ldots, n_p. \tag{6.28}$$

Given an image $z \in \mathbb{R}^{m_1\times m_2}$ of an unknown person in an unknown expression, that we want to classify. The coordinates of z in the expression basis can be found by solving the following least squares problem

$$\min_{\alpha_e} \left\| \mathcal{Z} - \alpha_e * \mathcal{R}(:, e, :, :) \right\|, \tag{6.29}$$

with $\mathcal{Z} \in \mathbb{R}^{1\times 1\times m_1\times m_2}$, such that $\mathcal{Z}(1, 1, :, :) = z$. Then, for each $e = 1, \ldots, n_e$ and for each $p = 1, \ldots, n_p$, we compute:

$$\left\| \alpha_e - \mathcal{Q}(p, :, :, :) \right\|. \tag{6.30}$$

We can then recognize the person in the image by assigning the label corresponding to the closest match.

Algorithm 15

input: Collection of images $\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times m_1 \times m_2}$, z image to classify
Decompose $\mathcal{A}$ as $\mathcal{A} = \mathcal{Q} * \mathcal{R}$
$\mathcal{Z}(1,1,:,:) = z$
for $e = 1, 2, \ldots, n_e$ **do**
    Solve $\min_{\alpha_e} \|\mathcal{Z} - \alpha_e * \mathcal{R}(:,e,:,:)\|$
    for $p = 1, 2, \ldots, n_p$ **do**
        if then $\|\alpha_e - \mathcal{Q}(p,:,:,:)\| < tol$, then classify z as the person p
        end if
    end for
end for

6.6 Numerical results

In this section, we apply the algorithms mentioned above on a database of faces of n_p persons in n_e different expressions. Each image can be either considered as an $m_1 \times m_2$ matrix, or as a $m_1 m_2$ vector, so that a database can be represented either by a fourth-order tensor or by a third-order tensor. In this work, we consider the datasets described in the following table

Table 6.1: *Size, number of expressions and persons of each databases.*

Dataset	**Pixel(matrix format)**	**Pixel(vector format)**	n_p	n_e
Face96	196×196	38416	119	19
Orl	92×112	10304	40	10

Figure 6.1: *Sample face images of three individuals from the Face96 dataset*

Figure 6.2: *Sample face images of three individuals from the Orl dataset*

In all our experiments, we use percentage of the success rate as performance measure given by

$$\frac{\textbf{number of correctly matched images}}{\textbf{number of test images}}$$

We split each database in a training set and a test set. The split is made by taking the $s\%$ of the n_e expressions for each person as training set and the remaining ones as test set. The expressions used as training set are chosen randomly. In the following table we present the success rate of all considered algorithms applied to 'Orl' dataset, for $s\% = 50\%$. All percentages correspond to 3 consecutive run, in each run we test 30 images chosen randomly from the test set. We compare the approaches described in this chapter with the approach based on HOSVD described in [26], and the Principal Component Analysis (PCA-eingfaces) generalized to the tensor case (see [40]), where the data is represented by a third-order tensor $\mathcal{A} \in \mathbb{R}^{m_1 \times M \times m_3}$, with M represent the number of images. We point out, that in the next tables, CP 3D and CP 4D refer to three and four dimensional CP decomposition classification respectively. (t-SVD 3D, t-QR 3D) and (t-SVD 4D, t-QR 4D) refer to three and four dimensional classification based on t-SVD and t-QR decompositions respectively. In the same way, (dct-SVD 3D, dct-QR 3D) and (dct-SVD 4D, dct-QR 4D) refer to three and four dimensional classification based on cosine SVD and cosine QR factorizations respectively.

Table 6.2: *Comparison results -Orl dataset*

Method	Success rate	Time of decomposition	Time of recognition
CP 3D (R=40)	87,78	**0.085**	0.084
CP 4D (R=35)	88,89	1.53	0.11
t-QR 3D	**94.46**	0.52	1.88
t-QR 4D	93.33	1.28	3.18
dct-QR 3D	**94.46**	0.608	2,32
dct-QR 4D	91.13	1.68	4,32
HOSVD 3D	86.66	0.24	0.096
PCA-eignfaces	85.56	0.35	**0.03**

As we can remark, the t-QR 3D and the dct-QR 3D algorithms give a better classification performance than the other tensor based approaches. In term of CPU time, we can observe that the CP 3D algorithm require less CPU time than the other approaches.

6.6.1 Numerical results with compression via truncated HOSVD

In this section, we present the numerical results associated with 3D classifications, with compression via truncated HOSVD.

Definition 6.22. Let $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an N^{th}-order tensor. The Tucker decomposition (often referred to as the higher-order SVD (HOSVD)[24, 74]) of $\mathcal{A}$ is defined as:

$$\mathcal{A} = \mathcal{V} \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_N U_N,$$

where $\mathcal{V} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is the core tensor, $U_n \in \mathbb{R}^{I_n \times R_n}$ are factor matrices, and $(R_1, \ldots, R_N)$ is the multi-linear rank of the tensor $\mathcal{A}$, where $R_n = rank(\mathcal{A}_{(n)})$. In case of HOSVD decomposition, the factor matrices U_n, for $n = 1, \ldots, N$ are orthonormal.

For computing the orthonormal factors U_n for $n = 1, \ldots, N$, and the tensor $\mathcal{V}$. We compute the SVD associated with each n-mode matrix of the tensor $\mathcal{A}$:

$$\mathcal{A}_{(n)} = U_n \Sigma_n V_n^T,$$

and put

$$\mathcal{V} = \mathcal{A} \times_1 U_1^T \times_2 U_2^T \times_3 \cdots \times_N U_N^T.$$

We point out that a rank $(r_1, \ldots, r_N)$ approximation with $r_n \leq R_n$ for $n = 1, \ldots, N$ can be obtained simply by restricting the factor matrices U_n to the first r_n columns (truncated SVD) for $n = 1, \ldots, N$, and by restricting the core tensor $\mathcal{V}$. We write the HOSVD of the tensor $\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times n_i}$ in the following form:

$$\mathcal{A} = \mathcal{V} \times_3 U,$$

with $\mathcal{V} \in \mathbb{R}^{n_p \times n_e \times m_i}$, $U \in \mathbb{R}^{n_i \times m_i}$ and $m_i < n_i$. In application, we set $m_i = n_e n_p$. Since in our illustrating examples, the numbers of persons and different expressions are small to number of pixels of images in vector format. Applying the t-QR or the cosine QR to the core tensor $\mathcal{V}$, we obtain

$$\mathcal{A} = (\mathcal{Q} * \mathcal{R}) \times_3 U.$$

Following from Definition 3.14 and from Propositions 6.9 and 6.19, if we fix an expression 'e' and a person 'p', we have

$$\mathcal{A}(p, e, :) = (\mathcal{Q}(p, :, :) * \mathcal{R}(:, e, :)) \times_3 U.$$

So, for a given image $z \in \mathbb{R}^{n_i}$, we have to solve the least squares problems

$$\min_{\alpha_e} \|\mathcal{Z} - (\alpha_e * \mathcal{R}(:, e, :)) \times_3 U\|. \tag{6.31}$$

Since U is an orthonormal matrix, the problem (6.31) is equivalent to following minimization

$$\min_{\alpha_e} \left\|\hat{\mathcal{Z}} - (\alpha_e * \mathcal{R}(:, e, :))\right\|. \tag{6.32}$$

with $\hat{\mathcal{Z}} \in \mathbb{R}^{1 \times 1 \times m_i}$, such that $\hat{\mathcal{Z}}(1, 1, :) = U^T z$. Then, Algorithm 16 can be rewritten in this form:

Algorithm 16

input: Collection of images $\mathcal{A} \in \mathbb{R}^{n_p \times n_e \times n_i}$, z image to classify
Decompose $\mathcal{A}$ as $\mathcal{A} = \mathcal{V} \times_3 U$
Decompose the core tensor $\mathcal{V}$ as $\mathcal{V} = \mathcal{Q} * \mathcal{R}$
$\mathcal{Z}(1,1,:) = U^T z(:)$
for do $e = 1, 2, \ldots, n_e$
    Solve $\|\mathcal{Z} - \alpha_e * \mathcal{R}(:, e, :)\|$
    for do $p = 1, 2, \ldots, n_p$
        if then $\|\alpha_e - \mathcal{Q}(p, :, :)\| < tol$, then classify z as the person p
        end if
    end for
end for

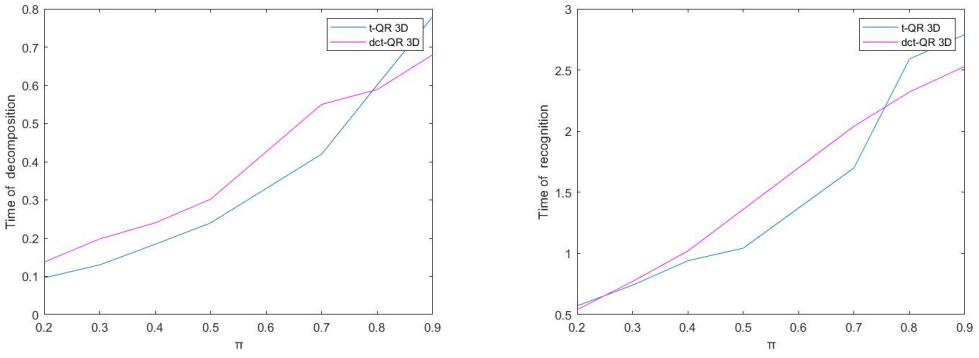


Figure 6.3: *Time of decomposition, and time of recognition of t-QR 3D, dct-QR 3D for different values of the truncation parameter π tested on Orl dataset, with s% = 50%.*

Table 6.3: *Success rate associated to 3D classification algorithms tested on Face96 dataset, with s% = 25%.*

Method	Success rate	Time of decomposition	Time of recognition
CP 3D	92.22	0.29	0.35
HOSVD 3D	92.22	1.762	1.11
t-QR 3D	92.22	**0.13**	0.36
t-SVD 3D	93.33	0.18	0.37
dct-QR 3D	92.22	0.14	0,38
dct-SVD 3D	**95.56**	0.15	0,38
PCA-eignfaces	90.56	2.82	**0.15**

6.7 Conclusion

In this chapter, we have proposed approaches for face recognition. Based on applying the tensor CP decomposition or QR/SVD decompositions defined using the t-product or the cosine product to a dataset of images. Our approaches are applied to a database represented by third or fourth-order tensors. In the numerical examples, we compare our approaches with PCA method based on tensor format and HOSVD based approach. In 3D classifications, we obtain comparable or better recognition results with the compressed versions of the algorithms.

Chapter 7

Conclusions and perspectives

In this chapter, we present a brief review of the results established in the previous chapters

Summary of results

In this thesis, we proposed algorithms for solving higher-order problems with applications to image restoration and face recognition.

In chapter 4, we constructed a degradation model associated with color images and videos. This model was constructed based on CP decomposition. Then we used the Alternating Least Squares (ALS) method to split the degradation model into three sub-problems. To solve those sub-problems, we used an algorithm based on Golub Kahan bidiagonalization.

In chapter 5, we solved the tensor least squares minimization problem $\min_{\mathcal{X}} \left\| \mathcal{X} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} - \mathcal{G} \right\|$, by decomposing the right-hand tensor using HOSVD or CP decomposition, and by working with the coefficient matrices $A^{(i)}$, for i= $1, \ldots, N$. In order to overcome the problem of the curse of dimensionality.

In chapter 6, we introduced approaches to face recognition defined via the t-product, the cosine product, and the outer product. Our approaches are applied to a database represented by third or fourth-order tensors.

Perspectives

During this work, multiple questions were raised, which represent interesting axes for future work:

- Applying the face recognition algorithms to a database contaminated by blur or noise.
- Using tensor decompositions for large data completion.

Bibliography

[1] Harry C Andrews and Boby Ray Hunt. Digital image restoration. 1977.

[2] ES Angel and Anil K Jain. Restoration of images degraded by spatially varying pointspread functions by a conjugate gradient method. *Applied Optics*, 17(14): 2186–2190, 1978.

[3] Mark R Banham and Aggelos K Katsaggelos. Digital image restoration. *IEEE signal processing magazine*, 14(2):24–41, 1997.

[4] Christian F Beckmann and Stephen M Smith. Tensorial extensions of independent component analysis for multisubject fmri analysis. *Neuroimage*, 25(1):294–311, 2005.

[5] Fatemeh PA Beik, Khalide Jbilou, Mehdi Najafi-Kalyani, and Lothar Reichel. Golub–kahan bidiagonalization for ill-conditioned tensor equations with applications. *Numer. Algorithms*, 2020.

[6] Peter N. Belhumeur, Joao P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.

[7] Abdeslem H Bentbib, M El Guide, K Jbilou, E Onunwor, and L Reichel. Solution methods for linear discrete ill-posed problems for color image restoration. *BIT Numerical Mathematics*, 58:555–576, (2018).

[8] Abdeslem Hafid Bentbib, Mohamed El Guide, Khalide Jbilou, and Lothar Reichel. Global golub–kahan bidiagonalization applied to large discrete ill-posed problems. *Journal of Computational and Applied Mathematics*, 322:46–56, 2017.

[9] AH Bentbib, S El-Halouy, and El M Sadek. Krylov subspace projection method for sylvester tensor equation with low rank right-hand side. *Numerical Algorithms*, pages 1–20, 2020.

[10] Richard Berry and James Burnell. The handbook of astronomical image processing. *The handbook of astronomical image processing*, 2000.

[11] Gregory Beylkin and Martin J Mohlenkamp. Numerical operator calculus in higher dimensions. *Proceedings of the National Academy of Sciences*, 99(16):10246–10251, 2002.

[12] Gregory Beylkin and Martin J Mohlenkamp. Algorithms for numerical analysis in high dimensions. *SIAM Journal on Scientific Computing*, 26(6):2133–2159, 2005.

[13] A Bouhamidi and K Jbilou. Sylvester Tikhonov-regularization methods in image restoration. *Journal of Computational and Applied Mathematics*, 206:86–98, 2007.

[14] D Brandoni and Valeria Simoncini. Tensor-train decomposition for image recognition. *Calcolo*, 57(1):9, 2020.

[15] Daniela Calvetti, Gene Howard Golub, and Lothar Reichel. Estimation of the L-curve via Lanczos bidiagonalization. *BIT Numerical Mathematics*, 39:603–619, 1999.

[16] Daniela Calvetti, Per Christian Hansen, and Lothar Reichel. L-curve curvature bounds via lanczos bidiagonalization. *ETNA. Electronic Transactions on Numerical Analysis [electronic only]*, 14:20–35, 2002.

[17] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[18] Julianne Chung and Silvia Gazzola. Computational methods for large-scale inverse problems: a survey on hybrid projection methods. *arXiv preprint arXiv:2105.07221*, (2021).

[19] Julianne Chung, James G Nagy, Dianne P O'leary, et al. A weighted GCV method for Lanczos hybrid regularization. *Electronic Transactions on Numerical Analysis*, 28: 149–167, 2008.

[20] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[21] Pierre Comon. Tensor decompositions. *Mathematics in signal processing V*, pages 1–24, 2002.

[22] Pierre Comon, Xavier Luciani, and André LF De Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 23:393–405, 2009.

[23] Lieven De Lathauwer and Bart De Moor. From matrix to tensor: Multilinear algebra and signal processing. In *Institute of mathematics and its applications conference series*, volume 67, pages 1–16. Citeseer, 1998.

[24] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[25] Mohamed El Guide, Alaa Ichi, Khalide Jbilou, and Rachid Sadaka. On tensor GMRES and Golub-Kahan methods via the T-product for color image processing. *The Electronic Journal of Linear Algebra*, 37:524–543, 2021.

[26] Lars Eldén. *Matrix methods in data mining and pattern recognition*. SIAM, 2007.

[27] Heinz W Engl. *Regularization Methods for The Stable Solution of Inverse Problems*. Univ., Institut für Mathematik, 1992.

[28] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of Inverse Problems*, volume 375. Springer Science & Business Media, 1996.

[29] Caterina Fenu, Lothar Reichel, and Giuseppe Rodriguez. Gcv for tikhonov regularization via global golub–kahan decomposition. *Numerical Linear Algebra with Applications*, 23(3):467–484, 2016.

[30] Caterina Fenu, Lothar Reichel, Giuseppe Rodriguez, and Hassane Sadok. GCV for Tikhonov regularization by partial SVD. *BIT Numerical Mathematics*, 57:1019–1039, 2017.

[31] Silvia Gazzola, Paolo Novati, and Maria Rosaria Russo. Embedded techniques for choosing the parameter in Tikhonov regularization. *Numerical Linear Algebra with Applications*, 21:796–812, 2014.

[32] Gene H Gloub and Charles F Van Loan. Matrix computations. *Johns Hopkins Universtiy Press, 3rd edtion*, 1996.

[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(2):205–224, 1965.

[34] Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

[35] Gene Howard Golub and Urs Von Matt. *Tikhonov Regularization for Large Scale Problems*. Citeseer, 1997.

[36] Robert M Gray. Toeplitz and circulant matrices: A review. 2006.

[37] Martin Hanke and Per Christian Hansen. Regularization methods for large-scale problems. *Surv. Math. Ind*, 3:253–315, 1993.

[38] Per Christian Hansen, James G Nagy, and Dianne P O'leary. *Deblurring Images: Matrices, Spectra, and Filtering*. SIAM, 2006.

[39] Per Christian Hansen, James G Nagy, Dianne P Leary, and Rodney Miller. Deblurring images: Matrices, spectra and filtering. *Journal of Electronic Imaging*, 17(1): 019901–019901, 2008.

[40] Ning Hao, Misha E Kilmer, Karen Braman, and Randy C Hoover. Facial recognition using tensor-tensor decompositions. *SIAM Journal on Imaging Sciences*, 6(1): 437–463, 2013.

[41] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

[42] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[43] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[44] Frank L Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematics and Physics*, 7(1-4):39–79, 1928.

[45] Roger A Horn, Roger A Horn, and Charles R Johnson. *Topics in matrix analysis*. Cambridge university press, 1994.

[46] Anil K Jain and Stan Z Li. *Handbook of face recognition*, volume 1. Springer, 2011.

[47] Khalide Jbilou, Abderrahim Messaoudi, and Hassane Sadok. Global FOM and GMRES algorithms for matrix equations. *Applied Numerical Mathematics*, 31:49–63, 1999.

[48] Julie Kamm and James G Nagy. Kronecker product and SVD approximations in image restoration. *Linear Algebra and its Applications*, 284:177–192, 1998.

[49] Saeed Karimi and Maryam Dehghan. Global least squares method based on tensor form to solve linear systems in kronecker format. *Transactions of the Institute of Measurement and Control*, 40(7):2378–2386, 2018.

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

[51] Henk AL Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 14(3):105–122, 2000.

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

[53] Misha E Kilmer, Carla D Martin, and Lisa Perrone. A third-order generalization of the matrix svd as a product of third-order tensors. *Tufts University, Department of Computer Science, Tech. Rep. TR-2008-4*, 2008.

[54] 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(1): 148–172, 2013.

[55] Stefan Kindermann. Convergence analysis of minimization-based noise level-free parameter choice rules for linear ill-posed problems. *Electron. Trans. Numer. Anal*, 38:233–257, 2011.

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

[57] Irwin Kra and Santiago R Simanca. On circulant matrices. *Notices of the AMS*, 59 (3):368–377, 2012.

[58] Joseph B Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, pages 7–18, 1989.

[59] Carla D Martin, Richard Shafer, and Betsy LaRue. An order-p tensor factorization with applications in imaging. *SIAM Journal on Scientific Computing*, 35(1):A474–A490, 2013.

[60] Eduardo Martınez-Montes, Pedro A Valdés-Sosa, Fumikazu Miwakeichi, Robin I Goldman, and Mark S Cohen. Concurrent eeg/fmri analysis by multiway partial least squares. *NeuroImage*, 22(3):1023–1034, 2004.

[61] James G Nagy and Misha Elena Kilmer. Kronecker product approximation for preconditioning in three-dimensional imaging applications. *IEEE Transactions on Image Processing*, 15:604–613, 2006.

[62] James G Nagy, Michael K Ng, and Lisa Perrone. Kronecker product approximations for image restoration with reflexive boundary conditions. *SIAM Journal on Matrix Analysis and Applications*, 25:829–841, 2003.

[63] Nhat Nguyen, Peyman Milanfar, and Gene Golub. Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement. *IEEE Transactions on image processing*, 10(9):1299–1308, 2001.

[64] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33:2295–2317, 2011.

[65] Christopher C Paige and Michael A Saunders. Algorithm 583: LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software (TOMS)*, 8:195–209, 1982.

[66] Christopher C Paige and Michael A Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[67] Zhen-Yun Peng. A matrix LSQR iterative method to solve matrix equation AXB= C. *International Journal of Computer Mathematics*, 87:1820–1830, 2010.

[68] Timothy M Persons, Paul F Hemler, and Robert J Plemmons. 3d iterative restoration of tomosynthetic images. In *Signal Recovery and Synthesis*, page JW4. Optical Society of America, 2001.

[69] Martin Schweiger, Adam Gibson, and Simon R Arridge. Computational aspects of diffuse optical tomography. *Computing in Science & Engineering*, 5(6):33–41, 2003.

[70] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

[71] Faezeh Toutounian and Saeed Karimi. Global least squares method (gl-lsqr) for solving general linear systems with several right-hand sides. *Applied Mathematics and Computation*, 178(2):452–460, 2006.

[72] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[73] Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15:122–137, 1963.

[74] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[75] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[76] Charles F Van Loan and Nikos Pitsianis. Approximation with kronecker products. In *Linear algebra for large scale and real-time applications*, pages 293–314. Springer, 1993.

[77] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear image analysis for facial recognition. In *Object recognition supported by user interaction for service robots*, volume 2, pages 511–514. IEEE, 2002.

[78] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear subspace analysis of image ensembles. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–93. IEEE, 2003.

[79] You-Wei Wen, Michael K Ng, and Yu-Mei Huang. Efficient total variation minimization methods for color image restoration. *IEEE Transactions on Image Processing*, 17:2081–2088, 2008.

Résumé

Les tenseurs sont des tableaux multidimensionnels. Ils sont une généralisation des matrices et des vecteurs. Ils fournissent un moyen naturel de représenter les données dans différents domaines, ce qui fait le domaine tensoriel un cadre idéal pour formuler et résoudre de nombreux problèmes dans différents domaines. L'une des applications les plus importantes des tenseurs se trouve dans le domaine du traitement des images, comme la restauration d'images et la reconnaissance des visages, où les images couleur (images RGB) sont présentées comme des tenseurs d'ordre 3 et une vidéo composée d'images couleur est un tenseur d'ordre 4. Mais lorsque nous travaillons dans des espaces de dimension supérieure pour résoudre des problèmes d'ordre supérieur, un ensemble de défis se pose, comme un problème connu sous le nom de "the curse of dimensionality". En effet, lorsque la dimension augmente, les problèmes d'ordre supérieur deviennent plus difficiles (besoin en calcul et en mémoire) car la taille des données d'un tenseur augmente exponentiellement avec l'augmentation de la dimensionnalité du tenseur. Par conséquent, les calculs tensoriels deviennent très coûteux. Dans cette thèse, nous nous sommes concentrés sur la résolution de certains problèmes tensoriels. Les algorithmes proposés sont obtenus en combinant des méthodes itératives telles que la méthode LSQR et des décompositions d'ordre supérieur pour surmonter le problème de la dimensionnalité. Les approches que nous proposons sont appliquées à la restauration d'images et de vidéos.

De plus, cette thèse étudie les méthodes de reconnaissance des visages basées sur le format tensoriel, où des outils d'algèbre multi-linéaire tels que la $HOSVD$ (Higher- Order Singular Value Decomposition) ont été utilisés. Nous proposons un nouvel algorithme qui peut être appliqué à une base de données d'images représentées par un tenseur

d'ordre 3 ou 4.

Mots clés: HOSVD, decomposition CP, LSQR, T-product, produit cosine , traitement des images et des vidéos, reconnaissance faciale.

Abstract

Tensors are multi-dimensional arrays. They are a generalization of matrices and vectors. They provide a natural way to represent the data in different fields, which make tensor field a great framework for formulating and solving many problems in different areas. One of the most important applications of tensors is in the field of image processing such as image restoration and face recognition, where color images (RGB images) are presented as third-order tensors and a video composed of color images is a fourth-order tensor. But when we work in higher dimension spaces to solve higher-order problems, a set of challenges arise such as a problem known as "the curse of dimensionality ". In fact, when the dimension increase the higher-order problems become harder (computation and memory requirement) since the data size of a tensor increases exponentially with the increase of the dimensionality of the tensor. As consequence tensor computations become much expensive. In this thesis, we have focused on solving some tensor problems. The suggested algorithms are obtained by combining iterative methods such as LSQR method and higher-order decompositions to overcome the problem of dimensionality. Our suggested approaches are applied to the restoration of images and videos.

In addition, this thesis studies the face recognition methods based on the tensor format, where multi-linear algebra tools such as the $HOSVD$ (Higher- Order Singular Value Decomposition) have been used. We suggest a new algorithm that can be applied to a database of images represented by a third or fourth-order tensor.

Keywords: HOSVD, CP decomposition, color image restoration, video restoration, LSQR, face recognition, t-product, cosine product.