# A Multilinear (Tensor) Algebraic Framework for Computer Graphics, Computer Vision, and Machine Learning

by

M. Alex O. Vasilescu

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

# A Multilinear (Tensor) Algebraic Framework for Computer Graphics, Computer Vision, and Machine Learning

M. Alex O. Vasilescu
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2009

## Abstract

This thesis introduces a multilinear algebraic framework for computer graphics, computer vision, and machine learning, particularly for the fundamental purposes of image synthesis, analysis, and recognition. Natural images result from the multifactor interaction between the imaging process, the scene illumination, and the scene geometry. We assert that a principled mathematical approach to disentangling and explicitly representing these causal factors, which are essential to image formation, is through numerical multilinear algebra, the algebra of higher-order tensors.

Our new image modeling framework is based on (i) a multilinear generalization of principal components analysis (PCA), (ii) a novel multilinear generalization of independent components analysis (ICA), and (iii) a multilinear projection for use in recognition that maps images to the multiple causal factor spaces associated with their formation. Multilinear PCA employs a tensor extension of the conventional matrix singular value decomposition (SVD), known as the $M$-mode SVD, while our multilinear ICA method involves an analogous $M$-mode ICA algorithm.

As applications of our tensor framework, we tackle important problems in computer graphics, computer vision, and pattern recognition; in particular, (i) image-based rendering, specifically introducing the multilinear synthesis of images of textured surfaces under varying view and illumination conditions, a new technique that we call "TensorTextures", as well as (ii) the multilinear analysis and recognition of facial images under variable face shape, view, and illumination conditions, a new technique that we call "TensorFaces". In developing these applications, we introduce a multilinear image-based rendering algorithm and a multilinear appearance-based recognition algorithm. As a final, non-image-based application of our framework, we consider the analysis, synthesis and recognition of human motion data using multilinear methods, introducing a new technique that we call "Human Motion Signatures".

# Acknowledgements

First and foremost, I am grateful to my advisor, Professor Demetri Terzopoulos. Although initially resisting the direction of my research, he eventually became its biggest supporter. This thesis would not have materialized in its present form without his valuable input and guidance. I am indeed fortunate to have had him as my advisor.

I thank the other members of my PhD Thesis Committee, Professors Geoffrey Hinton, Alan Jepson, and David Fleet, for their constructive critique, which improved the quality of this dissertation. I also appreciate their flexibility in accommodating my departmental and senate defenses into their busy schedules.

I am grateful to Professor Amnon Shashua of The Hebrew University of Jerusalem for agreeing to serve as the external examiner of my thesis on very short notice and for reviewing my dissertation in record time.

My thanks also go to Greg Ward, who provided encouragement and comments on a draft of my SIGGRAPH paper on "TensorTextures".

Many other people at the University of Toronto (UofT), New York University (NYU), the Massachusetts Institute of Technology (MIT), and elsewhere deserve special mention and my appreciation:

At the UofT, I thank Professor John Tsotsos for his interest in my work as well as for his kindness and encouragement. I had countless enjoyable discussions about computer vision and related topics with Chakra Chennubhotla. I also thank Geneviève Arboit, Andrew Brown, Kiam Choo, Florin Cutzu, Petros Faloutsos, Steven Myers, Michael Neff, Victor Ng, Alberto Paccanaro, Sageev Oore, Faisal Qureshi, Brian Sallans, Corina Wang, and Howard Zhang for their comradery. Geneviève, Howard, Kiam, and Steven generously volunteered their time and bodies for motion capture data acquisition. My work on human motion signatures started with an overambitious term project for a graphics course taught by Professors Eugene Fiume and James Stewart. Julie Weedmark kindly assisted in dealing with the administrative and financial obstacles that were unjustly imposed by the UofT School of Graduate Studies.

At NYU, I thank the MRL/CAT faculty, Professors Davi Geiger, Ken Perlin, Chris Bregler, Denis Zorin, Yann LeCun, Jack Schwartz, and last but not least Mike Uretsky for encouragement and interesting discussions. Professor Michael Overton deserves a special thanks for pointing me to the work on tensors by Dr. Tamara Kolda at Sandia. Finally, I have fond memories of interactions with Aaron Hertzmann, Evgueni Parilov, Wei Shao, Elif Tosun, Lexing Ying, and many other members of the MRL and CAT. I particularly thank Jared Silver for his valuable assistance with 3D modeling, animation, and

A/V production, and Svetlana Stenchikova for exploring some programming issues.

At MIT, I am grateful to Professor Rosalind Picard for bringing me into the Media Lab and for her generous support during my two years there. I thank Professors John Maeda, Alex Pentland, and Marvin Minsky for helpful discussions. I also had memorable conversations with Ian Eslick, my office mate at the Media Lab. Several years earlier, while I was at the UofT, I was inspired by Professor Josh Tenenbaum's bilinear models, and he generously provided me some of his code.

I would like to gratefully acknowledge my amicable interactions with members of the "tensor decomposition" research community. In particular, I thank Tamara Kolda, Lieven de Lathauwer, Pietr Kroonenberg, Rasmus Bro, Pierre Comon, Jos ten Berge, Henk Kiers, Lek-Heng Lim, and Charlie van Loan. I am also grateful to (the late) Richard Harshman and Gene Golub for their interest in my work and their encouragement.

Motion data were collected at the Gait Laboratory of the Bloorview MacMillan Medical Centre in Toronto. The data acquisition work was done with the permission of Professor Stephen Naumann, Director of the Rehabilitation Engineering Department and with the helpful assistance of Mr. Alan Morris. The 3D Morphable faces database was obtained courtesy of Prof. Sudeep Sarkar of the University of South Florida as part of the USF HumanID 3D database.

The portions of this research that were carried out at the UofT were funded in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada through grants to Demetri Terzopoulos, and at NYU in part by the Technical Support Working Group (TSWG) of the US Department of Defense through a grant to him and me.

Finally, I am grateful for the essential financial support that my parents, George and Ioanna Vasilescu, provided to me while I was a student and I greatly appreciate their sacrifices.

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# Love and Tensor Algebra

Come, let us hasten to a higher plane
Where dyads tread the fairy fields of Venn,
Their indices bedecked from one to $n$
Commingled in an endless Markov chain!

Come, every frustum longs to be a cone
And every vector dreams of matrices.
Hark to the gentle gradient of the breeze:
It whispers of a more ergodic zone.

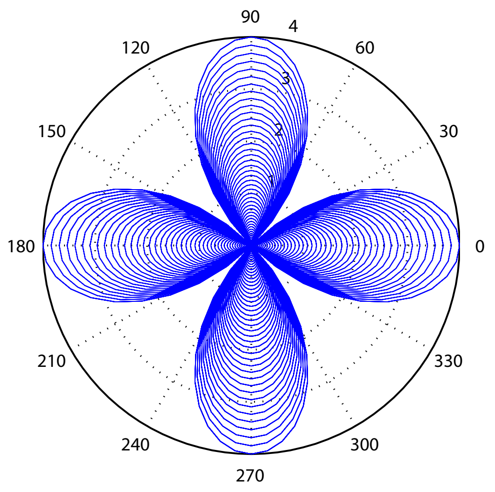
In Riemann, Hilbert or in Banach space
Let superscripts and subscripts go their ways.
Our asymptotes no longer out of phase,
We shall encounter, counting, face to face.
I'll grant thee random access to my heart,
Thou'lt tell me all the constants of thy love;
And so we two shall all love's lemmas prove,
And in our bound partition never part.

For what did Cauchy know, or Christoffel,
Or Fourier, or any Boole or Euler,
Wielding their compasses, their pens and rulers,
Of thy supernal sinusoidal spell?

Cancel me not – for what then shall remain?
Abscissas some mantissas, modules, modes,
A root or two, a torus and a node:
The inverse of my verse, a null domain.

Ellipse of bliss, converge, O lips divine!
the product of four scalars it defines!
Cyberiad draws nigh, and the skew mind
Cuts capers like a happy haversine.

I see the eigenvalue in thine eye,
I hear the tender tensor in thy sigh.
Bernoulli would have been content to die,
Had he but known such $a^2 \cos 2\phi$!



– Stanislaw Lem
"The Cyberiad"
(tr. Michael Kandel)

# 1 Introduction

**Contents**

*Image computing* concerns the synthesis, analysis, and recognition of images. Image synthesis, mathematically a forward problem, is primarily of interest in the field of *computer graphics*. Image analysis, an inverse problem, is primarily of interest in the field of *computer vision*. Image recognition, is primarily of interest in the field of *machine learning* or *pattern recognition*. Image computing cuts across all of these fields.

Linear algebra—i.e., the algebra of matrices—has traditionally been of tremendous value in the context of image synthesis, analysis, and recognition. The Fourier transform, the Karhunen-Loeve transform, and other linear techniques have been veritable workhorses. For example, principal components analysis (PCA) has been a popular technique in facial image recognition, as has its extension, independent components analysis (ICA) [Chellappa et al. 1995]. By their very nature, these products of linear algebra model single-factor, linear variation in image formation or the linear combination of sources.

Natural images, however, result from the interaction of *multiple* causal factors related to scene structure, illumination, and imaging. For example, facial images are the result of facial geometry (person, facial expression, etc.), the pose of the head relative to the camera, the lighting conditions, and the type of camera employed. To deal with this complexity, we propose a more sophisticated mathematical approach to the synthesis, analysis, and recognition of images that can explicitly represent each of the multiple causal factors underlying image formation.

This thesis introduces and develops a tensor algebraic framework for image computing. Our novel approach is that of multilinear algebra—the algebra of higher-order tensors.[1] The natural generalization of matrices (i.e., linear operators defined over a vector space), tensors define multilinear operators over a *set* of vector spaces. Hence, multilinear algebra and higher-order tensors, which subsume linear algebra and matrices/vectors/scalars as special cases, serves as a unifying mathematical framework suitable for addressing a variety of challenging problems in image computing.

Beyond image computing, we will demonstrate that our tensor algebraic framework is also applicable to non-image data; e.g., motion computing. In particular, we will investigate the analysis, synthesis, and recognition of human motion data.

## 1.1 Data Analysis and Multilinear Representations

The goal of many statistical data analysis problems, among them those arising in the domains of computer graphics, computer vision, and machine learning, is to find a suitable representation of multivariate data that facilitates the analysis, visualization, compression, approximation, and/or interpretation of the data. This is often done by applying a suitable transformation to the space in which the observational data reside. Several methods have been developed to compute transformations, resulting in representations with different assumptions and with different optimality properties defined in terms of dimensionality reduction, the statistical interestingness of the resulting components, the simplicity of the transformation, and various applications-related criteria.

Representations that are derived through *linear transformations* of the original observed data have traditionally been preferred due to their conceptual and computational simplicity. Principal components analysis (PCA), factor analysis, projection pursuit, and independent components analysis (ICA) are methods that employ linear transformations. A linear transformation $\mathbf{T}$ is a function or mapping from a domain vector space $\mathbb{R}^m$ to a range vector space $\mathbb{R}^n$:

$$\mathbf{T} : \mathbb{R}^m \mapsto \mathbb{R}^n. \tag{1.1}$$

The transformation is linear if for $c \in \mathbb{R}$ and for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$,

$$\mathbf{T}(c\mathbf{x} + \mathbf{y}) = c\mathbf{T}(\mathbf{x}) + \mathbf{T}(\mathbf{y}). \tag{1.2}$$

Methods for finding a suitable linear transformation can be categorized as second-order statistical methods and higher-order statistical methods.

Second-order statistical methods assume that the data are Gaussian distributed, hence completely

[1]In our work, the term "tensor" denotes a multilinear map from a set of domain vector spaces to a range vector space. The term "data tensor" denotes a multi-dimensional (or $m$-way) array. This is in contrast to differential geometry, engineering, or physics where the word "tensor" often refers to a *tensor field* or tensor-valued functions on manifolds (e.g., metric tensor, curvature tensor, strain-tensor, stress tensor, or gravitational field tensor).

determined by their mean vector and covariance matrix, which encode the first-order and second-order statistics of the data, respectively. Second-order methods provide a data representation in the sense of minimal reconstruction (mean-squared) error using only the information contained in the mean vector and covariance matrix. PCA and factor analysis are second-order methods whose goal is to reduce the dimensionality of the data using the covariance matrix. The representation provided by PCA is an optimal linear dimensionality reduction approach in the least-squares sense.

Second-order methods, however, neglect potentially important aspects of non-Gaussian data, such as clustering and independence. Unlike second-order methods, which seek an *accurate* representation, higher-order methods seek a *meaningful* representation. Methods such as projection pursuit and ICA employ additional information about the data distribution not contained in the covariance matrix; they exploit higher-order statistics. The basic goal of ICA is to find a transformation in which the components are statistically independent. ICA can be applied to feature extraction or blind source separation where the observed data are decomposed into a linear combination of source signals.

Whether derived through second-order or higher-order statistical considerations, linear transformations are limited in their ability to facilitate data analysis. *Multilinear transformations* are a natural generalization of linear transformations, which provide significantly more power. Whereas linear transformations are the topic of linear algebra—the algebra of scalars, vectors, and matrices—multilinear transformations require a more sophisticated multilinear algebra—the algebra of higher-order tensors.

A multilinear transformation is a *nonlinear* function or mapping from not just one, but a *set* of $M$ domain vector spaces $\mathbb{R}^{m_i}$, $1 \leq i \leq M$, to a range vector space $\mathbb{R}^n$:

$$\boldsymbol{\mathcal{T}} : \{\mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \ldots \times \mathbb{R}^{m_M}\} \mapsto \mathbb{R}^n. \tag{1.3}$$

The function is linear with respect to each of its arguments; i.e., for all $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^{m_i}$,

$$\begin{aligned}
\boldsymbol{\mathcal{T}}(c\mathbf{x}_1 + \mathbf{y}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) &= c\boldsymbol{\mathcal{T}}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) + \boldsymbol{\mathcal{T}}(\mathbf{y}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) \\
\boldsymbol{\mathcal{T}}(\mathbf{x}_1, c\mathbf{x}_2 + \mathbf{y}_2, \ldots, \mathbf{x}_M) &= c\boldsymbol{\mathcal{T}}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) + \boldsymbol{\mathcal{T}}(\mathbf{x}_1, \mathbf{y}_2, \ldots, \mathbf{x}_M) \\
&\vdots \\
\boldsymbol{\mathcal{T}}(\mathbf{x}_1, \mathbf{x}_2, \ldots, c\mathbf{x}_M + \mathbf{y}_M) &= c\boldsymbol{\mathcal{T}}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M) + \boldsymbol{\mathcal{T}}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{y}_M).
\end{aligned} \tag{1.4}$$

Multilinear transformations lead to generative models that explicitly capture how the observed data are influenced by multiple underlying *causal factors*. These causal factors can be fundamental physical, behavioral, or biological processes that cause patterns of variation in the observational data, which comprise a set of measurements or *response variables* that are affected by the causal factors (see Appendix B). Analogously to the aforementioned methods for finding linear transformations, there are second-order and higher-order statistical methods for finding suitable multilinear transformations that compute accurate or meaningful representations of the causal factors.

The term "multi" in our multilinear generative models connotes "multiple" as well as "multiplicative". Indeed, our models assume that the multiple causal factors combine with one another in a multi-

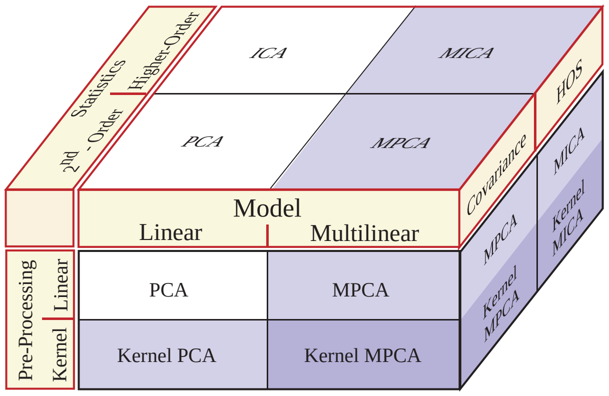


*Figure 1.1: Taxonomy of models.*

plicative manner. Thus, although our multilinear models are nonlinear, the nonlinearity is not arbitrary. The tacit assumption (cf. (1.4)) is that the variability in the measurement data due to changes in any single factor can be adequately modeled in a linear manner *when all other factors remain fixed*. This assumption can be further relaxed through particular types of data pre-processing (via nonlinear kernel functions).

Figure 1.1 presents a taxonomy of the models within the scope of this thesis. We will develop and gainfully apply multilinear generalizations of the linear PCA and ICA methods. Multilinear generalizations of factor analysis, projection pursuit, and other related methods can be developed, but we will not consider them in this thesis. Whereas conventional PCA and ICA are suitable for representing and analyzing unifactor data through the use of flat, linear subspaces, multilinear PCA and multilinear ICA are applicable to the representation and analysis of multifactor data through the use of curved, multilinear manifolds. Unlike arbitrarily nonlinear manifolds, however, each of the multiple sets of basis vectors comprising the multilinear model define linear subspaces that combine multiplicatively to form the multilinear manifold. In the context of manifold learning methods, a simple way of visualizing (in low-dimensional space) the essential difference between linear and multilinear methods, which are fundamentally nonlinear, is shown in Figure 1.2.

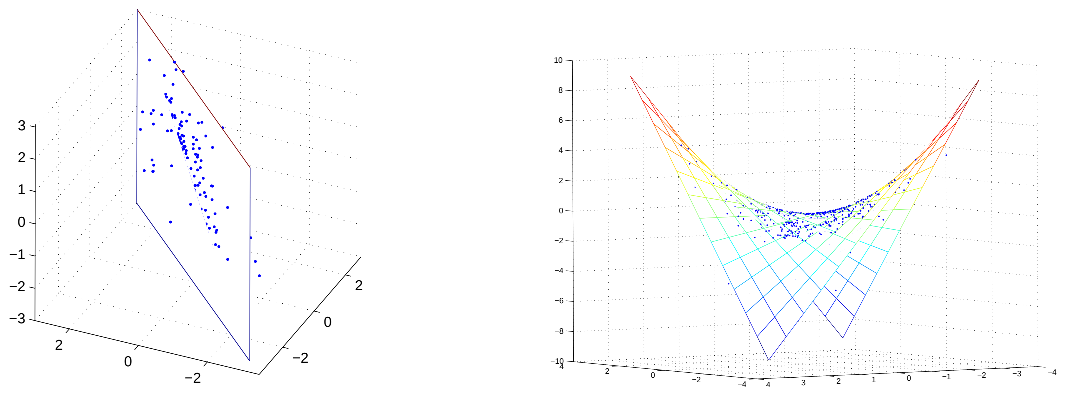
## 1.2 Thesis Contributions

The overarching contribution of this thesis is

- a multilinear (tensor) algebraic approach to computer graphics, computer vision, and machine learning.

Our approach shows promise as a unifying framework spanning these fields. Our primary technical contributions are as follows:

- The introduction of multilinear PCA (MPCA) to the domains of image-based rendering, appearance-based face recognition, and human motion computing;



(a) (b)

*Figure 1.2: Linear (a) and multilinear (b) manifolds populated by observational data (the blue dots) in three-dimensional space.*

- The introduction of multilinear ICA (MICA) and its application to appearance-based face recognition;
- The introduction of multilinear projection, with associated novel definitions of the identity and pseudo-inverse tensors, for the purposes of recognition in our tensor framework.

More specifically, our technical contributions are the following:

1. **Multilinear (tensor) analysis of image ensembles:** We introduce a multilinear analysis framework for appearance-based image representation. We demonstrate that multilinear algebra offers a potent mathematical approach to analyzing the multifactor structure of image ensembles and for addressing the fundamental yet difficult problem of disentangling the causal factors.

2. **Multilinear PCA/ICA (MPCA/MICA):** We introduce nonlinear, multifactor generalizations of the principal components analysis (PCA) and independent components analysis (ICA) methods. Our novel, multilinear ICA model of image ensembles learns the statistically independent components of multiple factors. Whereas the conventional PCA and ICA are based on linear algebra, our efficient multilinear PCA (MPCA) and multilinear ICA (MICA) algorithms exploit the more general and powerful multilinear algebra. We develop MPCA and MICA dimensionality reduction algorithms that enable subspace analysis within our multilinear framework. These algorithms are based on a tensor decomposition known as the $M$-mode SVD, a natural extension to tensors of the conventional matrix singular value decomposition (SVD).

3. **TensorTextures (Figure 1.3):** In the context of computer graphics, we introduce a tensor framework for image synthesis. In particular, we develop a new image-based rendering algorithm called TensorTextures that learns a parsimonious model of the bidirectional texture function (BTF) from observational data. Given an ensemble of images of a textured surface, our nonlinear, generative model explicitly represents the multifactor interaction implicit in the detailed appearance

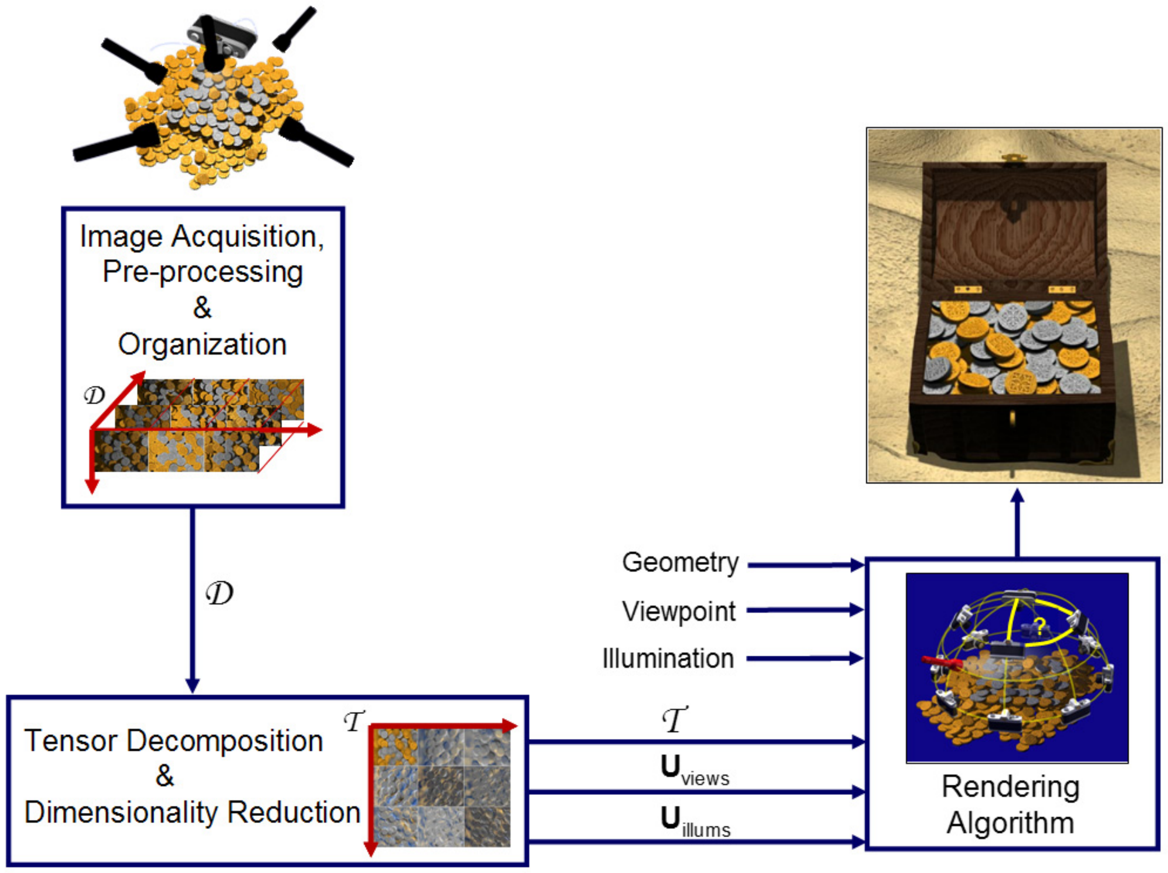


Image Acquisition, Pre-processing & Organization
$\mathcal{D}$
$\mathcal{D}$
Tensor Decomposition & Dimensionality Reduction
$\mathcal{T}$
Geometry
Viewpoint
Illumination
$\mathcal{T}$
$\mathbf{U}_{\text{views}}$
$\mathbf{U}_{\text{illums}}$
Rendering Algorithm

*Figure 1.3: TensorTextures analysis/synthesis system diagram. The left half of the figure illustrates the offline, analysis component, while the right half illustrates the online, synthesis component. Counterclockwise from the upper left, (1) training images acquired under different view and illumination conditions are organized into a data tensor $\mathcal{D}$, (2) the tensor is decomposed and the dimensionality of the view and illumination bases are reduced in order to compute the TensorTextures basis $\mathcal{T}$, (3) the base geometry plus view and illumination parameters are input to the learned generative model in order to synthesize an image (4) using a multilinear rendering algorithm.*

of the surface under varying photometric angles, including local (per-texel) reflectance, complex mesostructural self-occlusion, interreflection and self-shadowing, and other BTF-relevant phenomena. Applying TensorTextures, we demonstrate the image-based rendering of natural and synthetic textured surfaces under continuously varying view and illumination conditions.

4. **TensorFaces (Figure 1.4):** In the context of computer vision, we consider the multilinear analysis of ensembles of facial images that combine several factors, including different facial geometries (people), facial expressions, head poses, and lighting conditions. We introduce the TensorFaces representation, which has important advantages, demonstrating that it yields superior facial recognition rates relative to standard, linear (PCA/eigenfaces) approaches. We demonstrate the power of multilinear subspace analysis in the context of facial image ensembles, where the relevant causal factors include different faces, expressions, views, and illuminations. We demonstrate

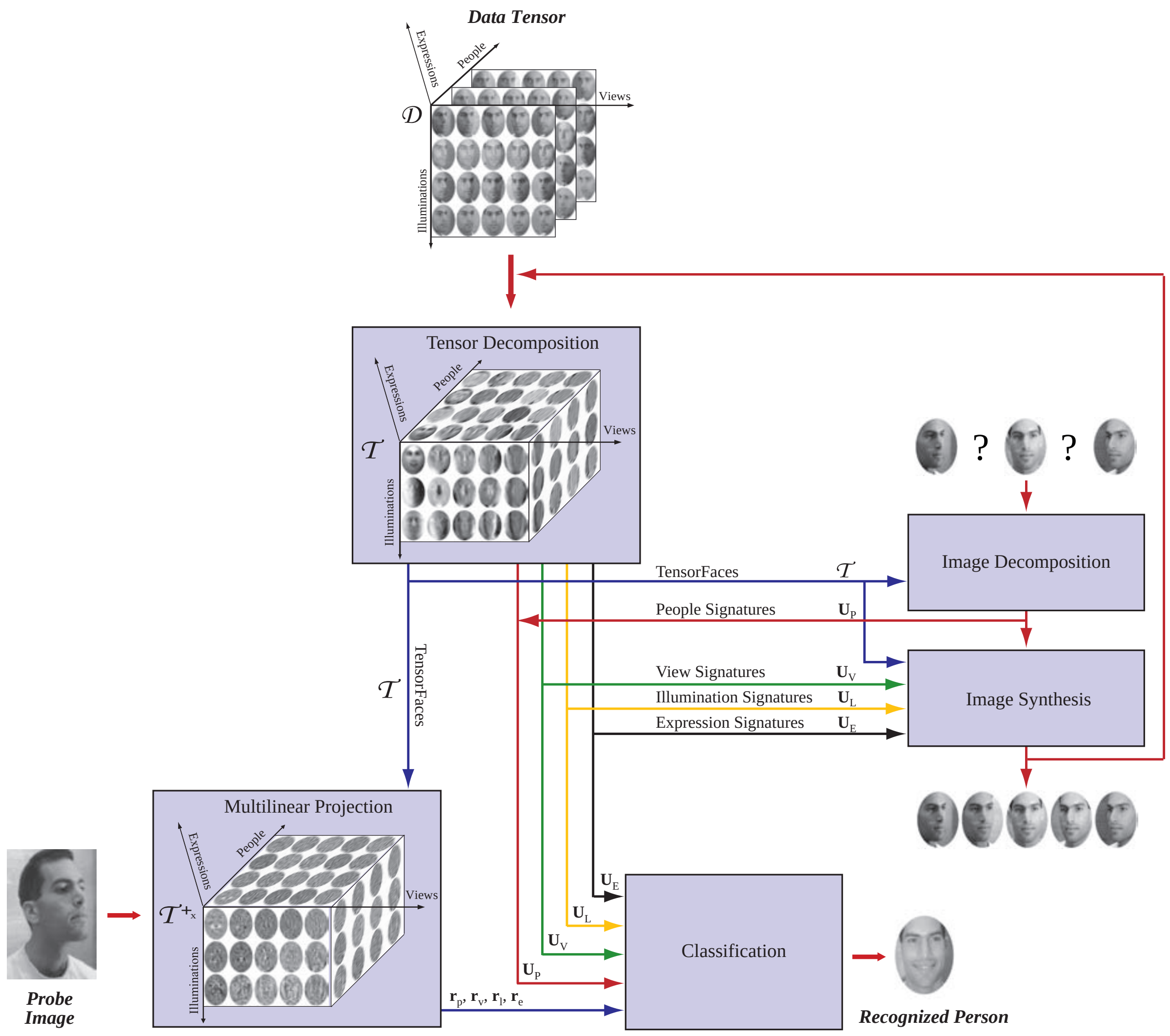


*Figure 1.4: Architecture of a multilinear facial image recognition system. A facial training image ensemble including different people, expressions, views, illuminations, and expressions is organized as a data tensor. The tensor is decomposed in the (offline) learning phase to train a multilinear model. In the (online) recognition phase, the model recognizes a previously unseen probe image as one of the known people in the database. In principle, the trained generative model can also synthesize novel images of known or unknown persons from one or more of their facial images.*

factor-specific dimensionality reduction of facial image ensembles. For example, we suppress illumination effects (shadows, highlights) while preserving detailed facial features.

5. **Multilinear projection for recognition (Figure 1.4):** TensorFaces leads to a multilinear recognition algorithm that projects an unlabeled test image into the multiple causal factor spaces to simultaneously infer its factor labels. In the context of facial image ensembles, where the factors are person, view, illumination, expression, etc., we demonstrate that the statistical regularities learned by MPCA and MICA capture information that, in conjunction with our multilinear projection algorithm, improves automatic face recognition rates. The multilinear projection algorithm employs mode-$m$ identity and mode-$m$ pseudoinverse tensors, concepts that we generalize from matrix algebra.

6. **TensorMotions and human motion signatures:** We also investigate the application of our tensor framework to human motion data, introducing an algorithm that extracts human motion signatures, which capture the distinctive patterns of movement of particular individuals. The algorithm analyzes motion data spanning multiple subjects performing different actions. The analysis yields a generative motion model that can synthesize new motions in the distinctive styles of these individuals. Our algorithms can also recognize people and actions from new motions by comparing motion signatures and action parameters.

The above contributions have been reported in the following publications: [Vasilescu 2001b; Vasilescu 2001a; Vasilescu and Terzopoulos 2002b; Vasilescu 2002; Vasilescu and Terzopoulos 2002a; Vasilescu and Terzopoulos 2003b; Vasilescu and Terzopoulos 2003c; Vasilescu and Terzopoulos 2003a; Vasilescu and Terzopoulos 2004b; Vasilescu and Terzopoulos 2004a; Vasilescu and Terzopoulos 2005a; Vasilescu and Terzopoulos 2005b; Vasilescu 2006a; Vasilescu 2006b; Vasilescu and Terzopoulos 2007a; Vasilescu and Terzopoulos 2007b; Vasilescu and Terzopoulos 2007c].

## 1.3 Thesis Overview

The remainder of this thesis is organized as follows:

Chapter 2 reviews related work. We begin, in Section 2.1, with a historical perspective of multilinear algebra in factor analysis and applied mathematics. We then review, in Section 2.2, the genesis of multilinear analysis in vision, graphics, and learning. Then we consider, in turn, the key background literature on our application topics of image-based rendering in computer graphics (Section 2.3), appearance-based facial recognition in computer vision (Section 2.4), and human motion analysis, synthesis, and recognition in graphics and vision (Section 2.5).

Chapter 3 introduces the mathematical background of our work. First, we review the relevant linear algebraic concepts in Section 3.1. Then, we introduce the terminology and relevant fundamentals of multilinear algebra in Section 3.2, among them the important concept of tensor decompositions.

Chapter 4 develops our tensor algebraic framework. We first review PCA and ICA in Section 4.1. In Section 4.2 we generalize PCA to multilinear PCA (MPCA) and develop an efficient MPCA algorithm.

Section 4.3 develops a multilinear ICA (MICA) technique and associated MICA algorithm. Finally, Section 4.4 proposes arbitrarily nonlinear, kernel-based multifactor generalizations of PCA/ICA and recent manifold mapping methods.

In Chapter 5, we apply our tensor algebraic framework in a machine learning approach to image-based rendering in computer graphics. Focusing on the rendering of textured surfaces, we develop a novel, multilinear texture mapping method that we call TensorTextures. Section 5.1 discusses the multilinear analysis of ensembles of texture images. Section 5.2 discusses the computation of TensorTextures. Section 5.3 investigates their parsimonious representation through dimensionality reduction. Section 5.4 develops the multilinear TensorTextures synthesis algorithm for planar surfaces and Section 5.5 generalizes it to handle curved surfaces. In Section 5.6, we present additional results, demonstrating applications to synthetic and natural textures.

In Chapter 6, we apply our framework to facial image ensemble analysis in computer vision. Section 6.1 develops the TensorFaces representation using a database of natural facial images. Section 6.2 investigates strategic dimensionality reduction in our multilinear model. Section 6.3 applies TensorFaces to a different facial image dataset based on 3D face scans. Section 6.4 introduces an independent TensorFaces model, which is based on MICA. Section 6.5 concludes the chapter with a closer look at the structure of facial image manifolds.

In Chapter 7, we expand our mathematical framework and apply it to the challenging problem of facial image recognition under unconstrained view, illumination, and expression conditions. First, in Section 7.1, we develop a set of linear projection operators for recognition patterned after the conventional linear one used in PCA-based recognition methods. Then, in Section 7.2, we propose a more natural and powerful multilinear projection operator and associated algorithms, which yields improved recognition results. We present facial recognition experiments and results in Section 7.3. Section 7.4 discusses the limitations of our face recognition approach.

In Chapter 8, we apply our framework to motion analysis, synthesis, and recognition. Section 8.1 discusses the motion data acquisition process. We develop our multilinear motion analysis, synthesis, and recognition algorithms in Sections 8.2, 8.3, and 8.4 in turn, and Section 8.5 presents our results. Finally, section 8.6 discusses the limitations of our multilinear motion analysis/synthesis/recognition technique.

Chapter 9 concludes the thesis, wherein we summarize our main contributions in Section 9.1 and propose promising avenues for future work in Section 9.2.

Appendix A discusses the treatment of data as vectors, matrices, and tensors. Appendix B discusses proper data acquisition for the purposes of multilinear analysis, synthesis, and recognition. Appendix C presents details regarding motion capture data processing.

# 2

# Related Work

**Contents**

In this chapter, we review prior work that is relevant to the various topics of this thesis. First we review the history of multilinear algebra for factor analysis and applied mathematics. We then examine the introduction of bilinear and multilinear analysis in vision, graphics, and learning, including recent follow-on efforts by other researchers inspired by our work. Finally, we consider, in turn, the key background literature on our application topics of image-based rendering in computer graphics, appearance-based facial recognition in computer vision, and human motion analysis, synthesis, and recognition in graphics and vision.

## 2.1 Linear and Multilinear Algebra for Factor Analysis

The analysis and decomposition of multi-way arrays, or higher-order data arrays, also known as data tensors, which generalize two-way arrays, or second-order arrays, also known as data matrices, first emerged in factor analysis.[1] Factor analysis using higher-order tensors has been studied in the field of psychometrics for the last forty years. References [Law et al. 1984; Coppi and Bolasco 1989] are collections of important early papers.

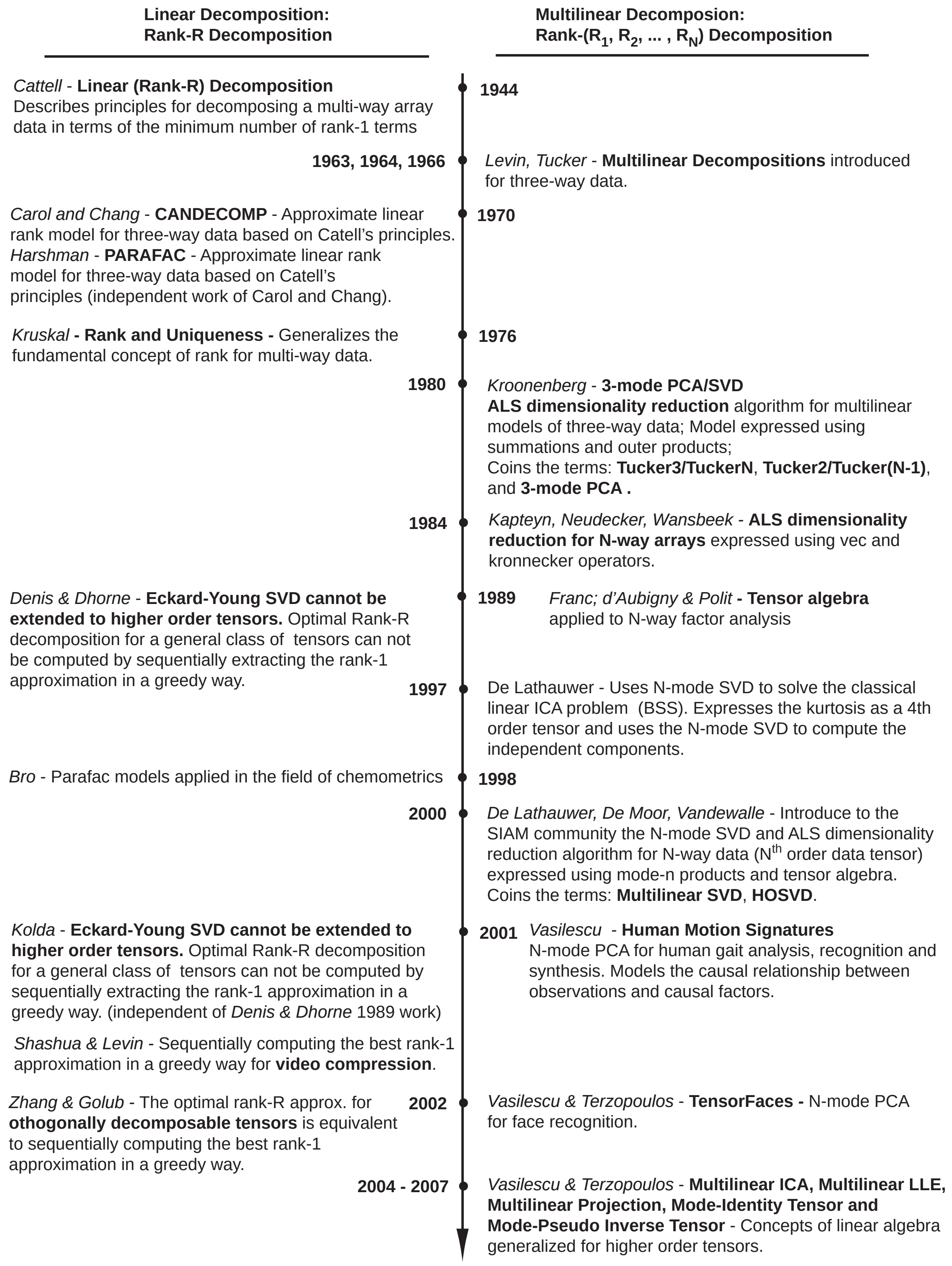
The analysis and decomposition of data matrices, or second-order tensors, which have also been used in factor analysis, have a much longer history. Stewart [1993] discusses the early history of the singular value decomposition. The SVD was discovered independently in 1873, by Beltrami [1873] and in 1874 by Jordan [1874] for square, real-valued matrices. Beltrami's proof used the relationship of the SVD to the eigenvalue decomposition of the matrices $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, while Jordan used an inductive argument that constructs the SVD from the largest singular value and its associated singular vectors. A 1936 article by Eckart and Young [1936] published in the journal *Psychometrika* extends the theorem underlying the singular value decomposition (SVD) to rectangular matrices, where they show that the optimal rank-$R$ approximation to a matrix can be reduced to $R$ successive rank-1 approximation problems to a diminishing residual. The SVD was not used as a computational tool until the 1960s, when efficient and stable algorithms were developed as a result of Golub's pioneering efforts [Golub and Kahn 1965; Golub and Reinsch 1970], and its usefulness in a variety of applications was established.

A major breakthrough in the factor analysis of higher order tensors came thirty years later after the Eckart-Young paper, when Tucker published a 1966 *Psychometrika* article that described the decomposition of third-order data tensors [Tucker 1966]. Tucker's decomposition computed three orthonormal vector spaces associated with the three modes of the tensor. In a 1980 article, Kroonenberg explicitly made the connection between Tucker's decomposition and the SVD, and coined the terms "3-mode PCA", "Tucker3", and "Tucker2", and he also presented a dimensionality reduction algorithm which employed an alternating least squares algorithm [Kroonenberg and de Leeuw 1980]. These decompositions and associated dimensionality reduction methods were generalized to $M$-mode factor analysis by various authors in the 1980's, first by Kapteyn et al. [1986] who expressed the decomposition in terms of *vec* operators and Kronecker products, and later by Franc [1989; 1992] and d'Aubigny and Polit [1989] who applied tensor algebra to $M$-way factor analysis. Figure 2.1 shows a timeline of key publications.

Tucker's decomposition and associated alternating least squares dimensionality-reduced subspace decomposition have recently been restated and introduced to the applied math community by De Lathauwer as the multilinear SVD and the multilinear (i.e., rank-$(R_1, R_2, \dots, R_M)$) decomposition, respectively [de Lathauwer 1997; de Lathauwer et al. 2000a; de Lathauwer et al. 2000b].

Unfortunately, unlike the situation in linear algebra, there does not exist a unique tensor decomposition that has all the nice properties of the matrix SVD. Although the $M$-mode SVD or multilinear SVD computes $M$ orthonormal subspaces, one for each mode, it does not compute the rank-$R$ tensor

[1]Factor analysis is related to principal components analysis. Factor analysis postulates that the observed data are the sum of their true values plus additive identically and independently distributed (IID) Gaussian noise. The representation is computed by applying PCA or a maximum likelihood estimation method to a modified covariance matrix—the covariance of the observed data minus the covariance of the noise, which is either known or must be estimated. Once the principal factors are computed, it is common to search for a rotation matrix that makes the results easy to interpret; e.g., a sparse representation can elucidate the relationship between the factors and the observed variables. The factors in factor analysis are intended to correspond to real-world phenomena, whereas the components of principal components analysis are geometrical abstractions that may not correspond to real-world phenomena.



**Linear Decomposition:**
**Rank-R Decomposition**

**Multilinear Decomposion:**
**Rank-($R_1$, $R_2$, ... , $R_N$) Decomposition**

**1944** — *Cattell* - **Linear (Rank-R) Decomposition**
Describes principles for decomposing a multi-way array data in terms of the minimum number of rank-1 terms

**1963, 1964, 1966** — *Levin, Tucker* - **Multilinear Decompositions** introduced for three-way data.

**1970** — *Carol and Chang* - **CANDECOMP** - Approximate linear rank model for three-way data based on Catell's principles.
*Harshman* - **PARAFAC** - Approximate linear rank model for three-way data based on Catell's principles (independent work of Carol and Chang).

**1976** — *Kruskal* **- Rank and Uniqueness -** Generalizes the fundamental concept of rank for multi-way data.

**1980** — *Kroonenberg* - **3-mode PCA/SVD**
**ALS dimensionality reduction** algorithm for multilinear models of three-way data; Model expressed using summations and outer products;
Coins the terms: **Tucker3/TuckerN**, **Tucker2/Tucker(N-1)**, and **3-mode PCA .**

**1984** — *Kapteyn, Neudecker, Wansbeek* - **ALS dimensionality reduction for N-way arrays** expressed using vec and kronnecker operators.

**1989** — *Denis & Dhorne* - **Eckard-Young SVD cannot be extended to higher order tensors.** Optimal Rank-R decomposition for a general class of tensors can not be computed by sequentially extracting the rank-1 approximation in a greedy way.

*Franc; d'Aubigny & Polit* **- Tensor algebra** applied to N-way factor analysis

**1997** — De Lathauwer - Uses N-mode SVD to solve the classical linear ICA problem (BSS). Expresses the kurtosis as a 4th order tensor and uses the N-mode SVD to compute the independent components.

**1998** — *Bro* - Parafac models applied in the field of chemometrics

**2000** — *De Lathauwer, De Moor, Vandewalle* - Introduce to the SIAM community the N-mode SVD and ALS dimensionality reduction algorithm for N-way data (Nth order data tensor) expressed using mode-n products and tensor algebra.
Coins the terms: **Multilinear SVD**, **HOSVD**.

**2001** — *Kolda* - **Eckard-Young SVD cannot be extended to higher order tensors.** Optimal Rank-R decomposition for a general class of tensors can not be computed by sequentially extracting the rank-1 approximation in a greedy way. (independent of *Denis & Dhorne* 1989 work)

*Shashua & Levin* - Sequentially computing the best rank-1 approximation in a greedy way for **video compression**.

*Vasilescu* - **Human Motion Signatures**
N-mode PCA for human gait analysis, recognition and synthesis. Models the causal relationship between observations and causal factors.

**2002** — *Zhang & Golub* - The optimal rank-R approx. for **othogonally decomposable tensors** is equivalent to sequentially computing the best rank-1 approximation in a greedy way.

*Vasilescu & Terzopoulos* - **TensorFaces -** N-mode PCA for face recognition.

**2004 - 2007** — *Vasilescu & Terzopoulos* - **Multilinear ICA, Multilinear LLE, Multilinear Projection, Mode-Identity Tensor and Mode-Pseudo Inverse Tensor** - Concepts of linear algebra generalized for higher order tensors.

*Figure 2.1: The development of multilinear algebra.*

decomposition.

A second major development in tensor decompositions came in 1970. Carroll and Chang [1970] introduced the CANDECOMP (Canonical Decomposition), while the PARAFAC (Parallel Factor Analysis) was introduced independently by Harshman [1970].

The CANDECOMP/PARAFAC (CP) decompositions are based on the principles put forth in 1944 by Catell [1944]. CP decomposes a tensor as a sum of $K$ optimal rank-1 tensors.[2] Note that this decomposition does not provide $M$ orthonormal subspaces. Descriptions of some of the development of these two higher-order SVDs can be found in Figure 2.1.

Denis et al. [1989] and independently Kolda [2001] show that the Eckard-Young theorem does not extend to higher order tensors, thus the rank-$R$ decomposition for a general class of tensors cannot be computed sequentially by extracting the rank-1 approximation in a greedy way. Zhang and Golub [2002] show that for a special class of tensors—orthogonally decomposable tensors—optimal rank-$R$ decomposition can be computed by sequentially computing the rank-1 approximation in a greedy way.

Vasilescu and Terzopoulos [2005a] introduce a Multilinear-ICA (as opposed to the tensorized computation of the conventional, linear ICA that is described in the work of De Lathauwer [1997]). In 2007, Vasilescu and Terzopoulos also generalize the concepts of identity, pseudo-inverse, and projection from linear algebra to multilinear algebra.

## 2.2 Bilinear and Multilinear Analysis in Vision, Graphics, and Learning

Some of the factor analysis methods reviewed above have recently been applied to vision, graphics and machine learning problems.

Bilinear models were the first to be explored. The *2-mode analysis* technique for analyzing (statistical) data matrices of scalar entries is described by Magnus and Neudecker [1988]. 2-mode analysis was extended to vector entries by Marimont and Wandel [1992] in the context of characterizing color surface and illuminant spectra. Freeman and Tenenbaum [1997; 2000] applied this extension in three different perceptual domains, including face recognition.

In computer vision, Shashua and Levin [2001] organize a collection of image "matrices" as a three-dimensional array, or 3rd-order data tensor, rather than as a matrix of vectorized images. They develop an approximate greedy rank-$R$ algorithm for higher-order tensors analogous to Eckard-Young algorithm and apply it to compress collections of images, such as video images.[3] Their algorithm takes advantage of temporal redundancies as well as horizontal/vertical spatial redundancies, but it fails to take advantage of diagonal spatial redundancies. Nevertheless, the authors report higher compression rates compared to applying conventional PCA on vectorized image data matrices. This result is interesting, but when dealing with higher order tensors, even a true rank-$R$ decomposition can result in the use of more storage

[2]This decomposition should not be confused with a rank-$R$ decomposition which expresses a tensor as the summation of the minimum number of rank-1 terms. However, one can discover the rank-$R$ tensor decomposition using CP through trial and error.

[3]This extension of the Eckart and Young algorithm has been shown to be suboptimal for higher-order tensors [Denis and Dhorne 1989; Kolda 2001].

than required to store the original data tensor.[4]

In computer graphics, Furukawa *et al.* [2002] proposed a compression method like Sashua's that expresses sampled bidirectional texture function (BTF) data as a linear combination of lower-rank tensors, but this is inadequate as a possible generalization of PCA. Although the authors report improved compression rates over PCA, besides the storage issue discussed above, their method does not permit separate dimensionality reduction (compression) to be guided independently in viewing, illumination, and spatial variation.

Note that both of the preceding algorithms are approximations of the rank-$R$ tensor decomposition problem. Other examples in the recent literature are [Wang and Ahuja 2004; Shashua and Hazan 2005]. These are linear models that approximate a tensor as a linear combination of rank-1 tensors. They are greedy algorithms which do not guarantee to compute the best rank-$R$ decomposition [Denis and Dhorne 1989; Kolda 2001]. They are not guaranteed to find the smallest number of rank-1 elements, nor are they guaranteed to find the $K$ optimal rank-1 tensors as can be achieved by the CANDECOMP/PARAFAC algorithms.

Vasilescu and Terzopoulos were the first to introduce true multilinear tensor models to computer vision [2001a; 2002b; 2003b; 2005a; 2007a], computer graphics [2001b; 2003c; 2004b], and machine learning [2002; 2002a; 2003a; 2004a; 2006a]. Multilinear models and related tensor methods are currently attracting increasing interest in these fields.

In computer graphics, interesting follow-up work includes that by Vlasic et al. [2005], which uses a multilinear model for mapping video-recorded performances of one individual to facial animations of another. Hsu et al. [2005] show results in synthesizing human motion by implementing the multilinear algorithm introduced by Vasilescu [2001b; 2001a; 2002]. Wang et al. [2005] proposes a variation on the TensorTextures algorithm due to Vasilescu and Terzopoulos [2003c; 2004b] by treating a texture measurement as a matrix.

In computer vision, tensor decompositions have recently been applied to face, expression and human motion recognition. Like [Vasilescu 2002], Davis et al. [2003] and Elgammal and Lee [2004] use multilinear models for motion recognition. Davis et al. decompose human motion into pose and effort, while Elgammal decomposes it in terms of style and content, but precedes the decomposition by performing LLE dimensionality reduction. While interesting, performing dimensionality reduction prior to multilinear analysis can remove stylistic differences which are important in recognition. Wang et al. [2003] applied multilinear decomposition for expression recognition. Xu et al. [2005] claim novelty in their organization of facial images into separate tensors for each person and their computation of the different image formation factors in a concurrent way across the different tensors. This approach, however,

[4]An important task for data analysis is to develop a compressed representation for the data matrix that might be easier to interpret. Such a compressed representation can be accomplished by a low rank matrix approximation—a rank-$R$ approximation. The rank of a matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ is the maximum number of linearly independent columns (rows). In other words, the rank is *at most* the minimum of the matrix dimensions, $\mathrm{rank}(\mathbf{A}) \leq \min(I_1, I_2)$. By comparison, Kruskal [1989] showed that the rank of a third order tensor, $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is *at least* the maximum of the tensor dimensions. The rank can be bounded by the following inequality: $\max(I_1, I_2, I_3) \leq \mathrm{rank}(\boldsymbol{\mathcal{T}}) \leq \min(I_1 I_2, I_1 I_3, I_2 I_3)$. Thus a rank-$R$ decomposition can result in more storage than the original data tensor. For example, Kruskal reported that for random third order tensors of dimensionality $2 \times 2 \times 2$, $79\%$ have rank 2, and $21\%$ have rank 3.

is equivalent to assembling all the images into one tensor and simply not extracting the person representation, which is known as person-specific or view-specific TensorFaces [Vasilescu and Terzopoulos 2002b; Vasilescu and Terzopoulos 2002a].

A primary advantage of multilinear algebra for recognition as executed in this thesis is that through a relatively more computationally expensive offline algorithm, one computes a unique representation per person that is invariant of the other factors inherent in image formation. This yields a linearly separable representation for each person without the need for support vector machines (SVM) or Discriminant Local Linear Embedding (DLLE) algorithms, hence making the online recognition computation cheap. Recently, however, a set of papers have appeared in the machine learning [Ye 2005; Dai and Yeung 2006; He et al. 2005; Cai et al. 2006] and computer vision [Xu et al. 2005; Wang and Ahuja 2005; Xia et al. 2006] literature that, due to their treatment of images as matrices, prompts them to use multilinear algebra, but with none of the benefits of our work. These approaches yield multiple representations per person that are not linearly separable despite the use of multilinear algebra. The authors attempt to deal with the mathematical problems that arize from their treatment of images as matrices by generalizing SVMs, or by applying SVMs or DLLE to columns/rows of images.[5]

It is interesting to note that there appears to be some confusion in the literature regarding the fact that multilinear models are fundamentally nonlinear (refer to Figure 1.2); for example, the recent survey article by Shakhnarovich and Moghaddam [2004] incorrectly categorizes them under linear models. This may be because the name "multilinear" inappropriately suggests "multiple linear" models to some.[6]

For completeness, the remaining sections in this chapter cover relevant, pre-multilinear background work in the three application areas of our framework—image-based rendering, appearance-based facial recognition, and human motion analysis, synthesis, and recognition.

## 2.3 Background on Image-Based Rendering

In computer graphics, "rendering" refers to the synthesis of images by computer from mathematical, traditionally geometric, models of 3D scenes. Image-based rendering (IBR) is a recent rendering technique [Shum et al. 2007]. In its purest form, instead of using geometric primitives, a collection of sample images of a 3D scene are used to synthesize novel views. There are also instances of IBR with implicit geometry and even IBR with some amount of explicit geometry. IBR was introduced by Chen and Williams [1993; 1995] and further developed in numerous papers (see, e.g., [McMillan and

[5]The conventional way of organizing multivariate observations for statistical analysis is as a vector. In recent years, however, several papers have advocated organizing the data elements associated with a single observation, say an ordinary image, as the sensor provides them; e.g., representing the array of image pixels as a matrix. What these authors overlook, however, is that when an image is treated as a matrix, from a statistical perspective it becomes an ensemble of row/column measurement variables. Most arguments in favor of the latter approach that are found in the literature are provably false. Appendix A discusses this issue in detail.

[6]Interestingly, Kernel PCA (KPCA) is often given as an example of a "true" nonlinear model. KPCA first applies a nonlinear transformation to the data and then it performs a linear decomposition. Thus, KPCA derives its nonlinearity from its preprocessing step. By contrast, a multilinear, or rank-$(R_1, R_2, \ldots, R_M)$, decomposition derives its nonlinearity from the decomposition step. Clearly, one can have doubly nonlinear models where one performs a nonlinear preprocessing step, followed by a multilinear decomposition (see Section 4.4).

Bishop 1995; Gortler et al. 1996; Levoy and Hanrahan 1996; Debevec et al. 1996; Seitz and Dyer 1996; Debevec et al. 2000; Matusik et al. 2003]).

In this thesis, we will be concerned with the image-based rendering of textured surfaces. The appearance of physical surfaces is determined by a complex interaction of multiple factors related to scene geometry, illumination, and imaging. The well-known bi-directional reflectance distribution function (BRDF) [Nicodemus et al. 1977] accounts for surface microstructure at a point [Larson 1992]. Its generalization, the *bidirectional texture function* (BTF) [Dana et al. 1999] captures the appearance of extended, textured surfaces. The BTF, essentially an array of BRDFs, accommodates spatially varying reflectance, surface mesostructure (i.e., 3D texture caused by local height variation over rough surfaces) [Koenderink and van Doorn 1996], subsurface scattering, and other phenomena over a finite region of the surface. It is a function of six variables $(x, y, \theta_v, \phi_v, \theta_i, \phi_i)$, where $(x, y)$ are surface parametric (texel) coordinates, and where $(\theta_v, \phi_v)$ is the view direction and $(\theta_i, \phi_i)$ is the illumination direction (a.k.a. the photometric angles).

Several BTF acquisition devices have been described in the literature (see, e.g., [Dana et al. 1999; Malzbender et al. 2001; Sattler et al. 2003; Han and Perlin 2003]). In essence, these devices sample the BTF by acquiring images of a surface of interest from several different views under several different illuminations.

Given only sparsely sampled BTF data, IBR is applicable to the challenging problem of rendering the appearance of a textured surface viewed from an arbitrary vantage point under arbitrary illumination [Debevec et al. 1996]. This problem has recently attracted considerable attention [Liu et al. 2001; Malzbender et al. 2001; Tong et al. 2002; Furukawa et al. 2002; Meseth et al. 2003; Suykens et al. 2003; Koudelka et al. 2003].

## 2.4 Background on Appearance-Based Facial Recognition

Object recognition is one of the most fundamental problems in computer vision, and the recognition of faces has received enormous attention in the literature [Chellappa et al. 1995; Zhao et al. ; Shakhnarovich and Moghaddam 2004].

Appearance-based recognition attempts to recognize objects, such as human faces, directly from their appearance in ordinary grey-level images. Sirovich and Kirby were the first to propose the use of PCA for appearance-based facial image analysis and representation [Sirovich and Kirby 1987]. The idea was applied by Turk and Pentland [1991b; 1991a] in their famous “Eigenfaces” face recognition technique (and subsequently by Murase and Nayar [1995] to the appearance-based recognition of arbitrary objects). Their subspace projection technique for measuring the similarity between facial images was improved by Moghaddam and Pentland [1997], who proposed a probabilistic similarity measure and explicitly represented the principal face subspace and its orthogonal complement.

Arguing that Eigenfaces, although well suited to appearance-based facial representation, are less than ideal for face recognition, Belhumeur et al. [1997] report better recognition results from their linear discriminant analysis (LDA) approach, which applies the Fisher linear discriminant to compute the linear subspace of greatest separability by maximizing the ratio of between-class scatter to within-class scatter. A generalization of this idea was presented by Moghaddam et al. [2000] in their Bayesian approach, which distinguishes between intrapersonal and extrapersonal variations in facial appearance, reporting better results than LDA.

The above methods apply PCA to dimensionality reduction of the training image ensemble. PCA represents the second-order statistics of the image ensemble. Independent components analysis (ICA) represents higher-order statistics as well, but it too yields a linear subspace albeit with different properties, primarily a non-orthogonal basis set. Bartlett et al. [2001; 2002] applied ICA to face recognition in two different ways, yielding independent basis images and a factorial representation, which represent local and global properties of faces, respectively, and they too report better recognition rates than Eigenfaces.

As we stated previously, our multilinear approach to recognition is fundamentally nonlinear. This is in contrast to the above linear methods. Other nonlinear methods include nonlinear PCA (NLPCA) [Kramer 1991], as well as kernel PCA (KPCA) [Schölkoph et al. 1998] and kernel LDA (KLDA) [Yang 2002] methods in which kernel functions that satisfy Mercer's theorem correspond to inner products in infinite-dimensional space. Shakhnarovich and Moghaddam [2004] present an empirical comparative evaluation of some of the above linear and nonlinear techniques.

## 2.5 Background on Human Motion Synthesis, Analysis, and Recognition

Johansson [1974] and Cutting et al. [1977; 1978] [Kozlowski and Cutting 1977] showed in the 1970s that observers can recognize actions, classify gender, and identify individuals familiar to them by just watching video sequences of lights affixed to human joints. These experiments suggest that joint angles are sufficient for the recognition of people by their gait.

Human gait is determined by a person's weight, limb length, and typical posture, which makes it a useful identifying biometric due to its non-intrusive and non-concealable nature, particularly when face, iris, or fingerprint information is not available. There are two approaches to the study of gait and action recognition. The first emulates moving light display perception in humans, by tracking a set of feature points on the body whose motions are used in recognizing the individual or the activity performed [Tanawongsuwan and Bobick 2001]. However, because the recovery of joint angles from a video of a walking person is a difficult problem [Bregler and Malik 1998; Cham and Rehg 1999; Sidenbladh et al. 2000], most research focuses on the statistics of patterns generated by a silhouette of the walking person in an image, such as eigengait space recognition [BenAbdelkader et al. 2002; Murase and Nayar 1995; Huang et al. 1999]. An important approach that falls in the latter category was developed by Niyogi and Adelson [1994], which characterizes motion via the entire 3D spatio-temporal (XYT) data volume spanned by the moving person in the image.

Motion synthesis is the goal of computer animation, which has a vast literature. Human motion synthesis through the analysis of motion capture data is currently attracting a great deal of attention within the community as a means of animating graphical characters. Several authors have introduced generative motion models for this purpose. Recent papers report the use of neural network learning models [Grzeszczuk et al. 1998], hidden Markov models [Brand and Hertzmann 2000], and Gaussian process models [Grochow et al. 2004].

# 3 Linear and Multilinear Algebra

**Contents**

**Algorithms**

In this chapter, we introduce the relevant mathematical background of our work. We first review linear algebraic methods, particularly the concepts of rank and the singular value decomposition (SVD) of matrices. We then present tensor terminology and the fundamentals of multilinear algebra, leading up to tensor decompositions.

We will use standard textbook notation, denoting scalars by lowercase italic letters $(a, b, \ldots)$, vectors by bold lowercase letters $(\mathbf{a}, \mathbf{b}, \ldots)$, matrices by bold uppercase letters $(\mathbf{A}, \mathbf{B}, \ldots)$, and higher-order tensors by bold uppercase calligraphic letters $(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \ldots)$.

## 3.1 Relevant Linear (Matrix/Vector) Algebra

The column span of a matrix $\mathbf{A}$ is the *range* of $\mathbf{A}$ and its dimension is the *rank* of $\mathbf{A}$:

**Definition 3.1.1 (Matrix Rank)** *The rank of a matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ denoted by $\mathrm{rank}(\mathbf{A})$ is the maximum number of linearly independent columns (rows).* ■

The rank can be bounded by the minimum of the matrix dimensions, $\mathrm{rank}(\mathbf{A}) \leq \min(I_1, I_2)$.

**Definition 3.1.2 (Rank-1 Matrix)** *$\mathbf{A}$ is said to be a rank-1 matrix, i.e., $\mathrm{rank}(\mathbf{A}) = 1$, if it is decomposable in terms of an outer product, denoted by $\circ$, of two vectors $\mathbf{u} = [u_1, u_2, \ldots, u_{I_1}]^{\mathrm{T}}$ and $\mathbf{v} = [v_1, v_2, \ldots, v_{I_2}]^{\mathrm{T}}$:*

$$\mathbf{A} = \mathbf{u} \circ \mathbf{v} = \begin{bmatrix} u_1v_1 & u_1v_2 & \ldots & u_1v_{I_2} \\ u_2v_1 & u_2v_2 & \ldots & u_2v_{I_2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{I_1}v_1 & u_{I_1}v_2 & \ldots & u_{I_1}v_{I_2} \end{bmatrix}. \tag{3.1}$$

■

**Definition 3.1.3 (Rank-$R$ Decomposition)** *The rank-$R$ decomposition of a matrix $\mathbf{A}$ is the minimal number of rank-1 matrices whose linear combination yields $\mathbf{A}$:*

$$\mathbf{A} = \sum_{r=1}^{R} \sigma_r \, \mathbf{u}^{(r)} \circ \mathbf{v}^{(r)}. \tag{3.2}$$

■

**Theorem 3.1.1 (SVD)** *If $\mathbf{A}$ is an $I_1 \times I_2$ matrix with $\mathrm{rank}(\mathbf{A}) = R$, its singular value decomposition can be written as*

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} = \mathbf{U}_1\boldsymbol{\Sigma}\mathbf{U}_2^{\mathrm{T}} \tag{3.3}$$

*where $\mathbf{U}_1$ and $\mathbf{U}_2$ are orthonormal matrices of dimensionality $I_1 \times I_1$ and $I_2 \times I_2$, respectively, and $\boldsymbol{\Sigma}$ is a $I_1 \times I_2$ diagonal matrix that contains the singular values $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_R > 0$.* ■

The best low rank approximation with respect to the Frobenius norm of a matrix is accomplished by truncating its SVD. This is stated in the Schmidt/Eckart-Young Theorem:[1]

**Theorem 3.1.2 (Eckart-Young Theorem)** *Let $\mathbf{A}$ be a $I_1 \times I_2$ matrix with $\mathrm{rank}(\mathbf{A}) = R$ and let*

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{3.4}$$

$$= \sum_{r=1}^{R} \sigma_r \, \mathbf{u}^{(r)} \circ \mathbf{v}^{(r)} \tag{3.5}$$

[1]This is the fundamental theorem of the singular value decomposition. It was proved in 1907 by Schmidt [1907] and rediscovered in 1936 by Eckart and Young [1936].

*be the singular value decomposition of* $\mathbf{A}$*, where the singular values* $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_R > 0$.[2] *If* $\operatorname{rank}(\tilde{\mathbf{A}}) = \tilde{R} < R$*, then the approximation error*

$$e = \|\mathbf{A} - \tilde{\mathbf{A}}\|^2, \tag{3.6}$$

*where* $\|\mathbf{A}\| = \langle \mathbf{A}, \mathbf{A} \rangle^{1/2} = \sqrt{\sum_{ij} \mathbf{A}_{ij}^2}$ *is the Frobenius norm, is minimized by*

$$\tilde{\mathbf{A}} = \sum_{r=1}^{\tilde{R}} \sigma_r \, \mathbf{u}^{(r)} \circ \mathbf{v}^{(r)}. \tag{3.7}$$

■

## 3.2 Relevant Multilinear (Tensor) Algebra

Next, we review related definitions for higher-order tensors. A *tensor*, or $m$-way array, is a generalization of a vector (first-order tensor) and a matrix (second-order tensor).

**Definition 3.2.1 (Tensor)** *Tensors are multilinear mappings over a set of vector spaces. The order of tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ *is* $M$*. An element of* $\boldsymbol{\mathcal{A}}$ *is denoted as* $\boldsymbol{\mathcal{A}}_{i_1 \ldots i_m \ldots i_M}$ *or* $a_{i_1 \ldots i_m \ldots i_M}$*, where* $1 \leq i_m \leq I_m$. ■

**Definition 3.2.2 (Rank-1 Tensor)** *An* $M^{th}$*-order tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ *is a rank-1 tensor when it is expressible as the outer product of* $M$ *vectors:* $\boldsymbol{\mathcal{A}} = \mathbf{u}_1 \circ \mathbf{u}_2 \circ \ldots \circ \mathbf{u}_M$. ■

Next, we generalize the definition of column and row rank of matrices. In tensor terminology, column vectors are referred to as mode-1 vectors and row vectors as mode-2 vectors.

**Definition 3.2.3 (Mode-$m$ Vectors)** *The mode-*$m$ *vectors of an* $M^{th}$*-order tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ *are the* $I_m$*-dimensional vectors obtained from* $\boldsymbol{\mathcal{A}}$ *by varying index* $i_m$ *while keeping the other indices fixed.* ■

The mode-$m$ vectors of a tensor are also known as *fibers*. The mode-$m$ vectors are the column vectors of matrix $\mathbf{A}_{[m]}$ that results from *matrixizing* (a.k.a. *flattening*) the tensor $\boldsymbol{\mathcal{A}}$ (Figure 3.1).

**Definition 3.2.4 (Mode-$m$ Matrixizing)** *The mode-*$m$ *matrixizing of tensor* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_M}$ *is defined as the matrix* $\mathbf{A}_{[m]} \in \mathbb{R}^{I_m \times (I_1 \ldots I_{m-1} I_{m+1} \ldots I_M)}$*. As the parenthetical ordering indicates, the mode-*$m$ *column vectors are arranged by sweeping all the other mode indices through their ranges, with smaller mode indexes varying more rapidly than larger ones; thus,*

$$[\mathbf{A}_{[m]}]_{jk} = a_{i_1 \ldots i_m \ldots i_M}, \quad \text{where} \quad j = i_m \quad \text{and} \quad k = 1 + \sum_{\substack{n=1 \\ n \neq m}}^{M} (i_n - 1) \prod_{\substack{l=1 \\ l \neq m}}^{n-1} I_l. \tag{3.8}$$

[2] This equation can be rewritten as $\mathbf{A} = \sum_{r=1}^{R} \sigma_r \mathbf{u}^{(r)} \mathbf{v}^{(r)\mathrm{T}}$.

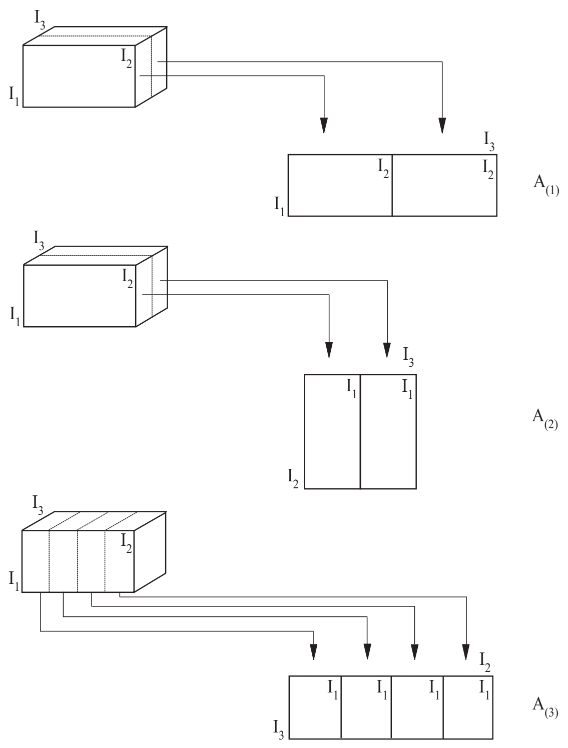


*Figure 3.1: Matrixizing a (3rd-order) tensor. The tensor can be matrixized in 3 ways to obtain matrices comprising its 1-mode, 2-mode, and 3-mode vectors.*

■

**Definition 3.2.5 (Mode-$m$ Rank)** *The mode-$m$ rank of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$, denoted $R_m$, is defined as the rank of the vector space generated by the mode-$m$ vectors:*

$$R_m = \text{rank}_m(\boldsymbol{\mathcal{A}}) = \text{rank}(\mathbf{A}_{[m]}). \tag{3.9}$$

■

A generalization of the product of two matrices is the product of a tensor and a matrix [Carroll et al. 1980; de Lathauwer et al. 2000a].

**Definition 3.2.6 (Mode-$m$ Product, $\times_m$)** *The mode-$m$ product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_m \times \ldots \times I_M}$ and a matrix $\mathbf{B} \in \mathbb{R}^{J_m \times I_m}$, denoted by $\boldsymbol{\mathcal{A}} \times_m \mathbf{B}$, is a tensor of dimensionality $\mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times J_m \times I_{m+1} \times \ldots \times I_M}$ whose entries are computed by*

$$[\boldsymbol{\mathcal{A}} \times_m \mathbf{B}]_{i_1 \ldots i_{m-1} j_m i_{m+1} \ldots i_M} = \sum_{i_m} a_{i_1 \ldots i_{m-1} i_m i_{m+1} \ldots i_M} b_{j_m i_m}. \tag{3.10}$$

■

The mode-$m$ product can be expressed in tensor notation as

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_m \mathbf{B}, \tag{3.11}$$

or in terms of matrixized tensors as

$$\mathbf{C}_{[m]} = \mathbf{B}\mathbf{A}_{[m]}. \tag{3.12}$$

The mode-$m$ product of a tensor and a matrix is a special case of the inner product in multilinear algebra and tensor analysis.[3] In the literature it is often specified using the Einstein summation convention, but the mode-$m$ product symbol is more intuitive since it highlights its multiplicative nature and more clearly expresses the analogy between matrix and tensor SVD.[4]

The mode-$m$ product has the following properties:

1. Given a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots I_n \times \ldots I_m \cdots}$ and two matrices, $\mathbf{U} \in \mathbb{R}^{J_m \times I_m}$ and $\mathbf{V} \in \mathbb{R}^{J_n \times I_n}$,

$$\begin{aligned} \boldsymbol{\mathcal{A}} \times_m \mathbf{U} \times_n \mathbf{V} &= (\boldsymbol{\mathcal{A}} \times_m \mathbf{U}) \times_n \mathbf{V} && (3.13)\\ &= (\boldsymbol{\mathcal{A}} \times_n \mathbf{V}) \times_m \mathbf{U} && (3.14)\\ &= \boldsymbol{\mathcal{A}} \times_n \mathbf{V} \times_m \mathbf{U}. && (3.15) \end{aligned}$$

2. Given a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots \times I_m \times \ldots \times I_M}$ and two matrices, $\mathbf{U} \in \mathbb{R}^{J_m \times I_m}$ and $\mathbf{V} \in \mathbb{R}^{K_m \times J_m}$,

$$(\boldsymbol{\mathcal{A}} \times_m \mathbf{U}) \times_m \mathbf{V} = \boldsymbol{\mathcal{A}} \times_m (\mathbf{V}\mathbf{U}). \tag{3.16}$$

3. The mode-$m$ product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots \times I_m \times \ldots \times I_M}$ with a set of matrices, $\mathbf{U}_m \in \mathbb{R}^{J_m \times I_m}$ where $1 \le m \le M$, can be expressed in tensor, matrix, or vector notation as follows:

$$\begin{aligned} \boldsymbol{\mathcal{C}} &= \boldsymbol{\mathcal{A}} \times_1 \mathbf{B}_1 \ldots \times_m \mathbf{B}_m \ldots \times_M \mathbf{B}_M && (3.17)\\ \mathbf{C}_{[m]} &= \mathbf{B}_m \mathbf{A}_{[m]} (\mathbf{B}_M \otimes \ldots \otimes \mathbf{B}_{m+1} \otimes \mathbf{B}_{m-1} \otimes \ldots \otimes \mathbf{B}_1)^{\mathrm{T}} && (3.18)\\ \mathrm{vec}(\boldsymbol{\mathcal{C}}) &= (\mathbf{B}_M \otimes \ldots \otimes \mathbf{B}_1)\mathrm{vec}(\boldsymbol{\mathcal{A}}) && (3.19) \end{aligned}$$

where $\otimes$ is the Kronecker product (see below) and $\mathrm{vec}(\cdot)$ denotes the vectorizing operator.[5] The order of the Kronecker products in (3.18) and (3.19) depends on how tensor $\boldsymbol{\mathcal{A}}$ is matrixized or on how it is vectorized. The index that changes fastest across the columns of $\mathbf{A}_{[m]}$ is that associated with mode 1 (see Figure 3.1); thus, $\mathbf{B}_1$ is the last matrix in the series of Kronecker products.

**Definition 3.2.7 (Kronecker Product, $\otimes$)** *The Kronecker product of* $\mathbf{U} \in \mathbb{R}^{I \times J}$ *and* $\mathbf{V} \in \mathbb{R}^{K \times L}$ *is the*

[3]Definition 3.2.6 will be generalized in Definition 7.2.1, which defines mode-$m$ products between two tensors.

[4]Carroll et al. [1980] denoted the mode-$m$ product between a tensor $\boldsymbol{\mathcal{A}}$ and a matrix $\mathbf{B}$ using the symbol $\odot_m$ as $\boldsymbol{\mathcal{C}} = \mathbf{B} \odot_m \boldsymbol{\mathcal{A}}$, which is equivalent to the $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} \times_m \mathbf{B}$ notation of de Lathauwer et al. [2000a].

[5]The vectorizing operator applied to a matrix $\mathbf{A}$ yields a vector $\mathrm{vec}(\mathbf{A}) = [\mathbf{a}_1^{\mathrm{T}} \mathbf{a}_2^{\mathrm{T}} \ldots \mathbf{a}_{I_2}^{\mathrm{T}}]^{\mathrm{T}}$ in which each of the $I_1$-dimensional column vectors $\mathbf{a}_i$ of $\mathbf{A}$ are concatenated. The vectorizing operator applied to a tensor $\boldsymbol{\mathcal{T}}$ is $\mathrm{vec}(\boldsymbol{\mathcal{T}}) = \mathrm{vec}(\mathbf{T}_{[1]})$.

*$IK \times JL$ matrix defined as $[\mathbf{U} \otimes \mathbf{V}]_{ik,jl} = u_{ij}v_{kl}$. Thus,*

$$\mathbf{U} \otimes \mathbf{V} = \begin{bmatrix} u_{11}\mathbf{V} & \dots & u_{1J}\mathbf{V} \\ \vdots & \ddots & \vdots \\ u_{I1}\mathbf{V} & \dots & u_{IJ}\mathbf{V} \end{bmatrix}. \tag{3.20}$$

■

The following are useful *vec-Kronecker properties*:

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

   — in particular, $\mathrm{vec}(\mathbf{a} \circ \mathbf{c}) = \mathrm{vec}(\mathbf{ac}^{\mathrm{T}}) = (\mathbf{c} \otimes \mathbf{a})$;

2. $(\mathbf{A} \otimes \mathbf{B})^{\mathrm{T}} = (\mathbf{A}^{\mathrm{T}} \otimes \mathbf{B}^{\mathrm{T}})$;

3. $(\mathbf{A} \otimes \mathbf{B})^{+} = (\mathbf{A}^{+} \otimes \mathbf{B}^{+})$.

**Definition 3.2.8 (Khatri-Rao Product, $\odot$ )** *The Khatri-Rao product of $\mathbf{U} \in \mathbb{R}^{I \times L}$ and $\mathbf{V} \in \mathbb{R}^{K \times L}$ is denoted as $\mathbf{U} \odot \mathbf{V}$ and its entries are computed by $[\mathbf{U} \odot \mathbf{V}]_{ik,l} = u_{il}v_{kl}$. It is a columnwise Kronecker product; therefore, it can be expressed as $\mathbf{U} \odot \mathbf{V} = [(\mathbf{u}^{(1)} \otimes \mathbf{v}^{(1)}) \dots (\mathbf{u}^{(l)} \otimes \mathbf{v}^{(l)}) \dots (\mathbf{u}^{(L)} \otimes \mathbf{v}^{(L)})]$ [Rao and Mitra 1971].* ■

The Kahtri-Rao product of a set of matrices $\mathbf{U}_m \in \mathbb{R}^{I_m \times L}$ for $1 \leq m \leq M$ is

$$\mathbf{U}_1 \odot \dots \odot \mathbf{U}_M = \left[ \begin{array}{ccc} (\mathbf{u}_1^{(1)} \otimes \dots \otimes \mathbf{u}_M^{(1)}) & \dots & (\mathbf{u}_1^{(L)} \otimes \dots \otimes \mathbf{u}_M^{(L)}) \end{array} \right], \tag{3.21}$$

where $\mathbf{u}_m^{(l)}$ is the $l^{\text{th}}$ column of $\mathbf{U}_m$ for $1 \leq l \leq L$. An element of the resulting matrix is defined as $[\mathbf{U}_1 \odot \dots \odot \mathbf{U}_M]_{i_1 i_2 \dots i_M, l} = [\mathbf{U}_1]_{i_1 l}[\mathbf{U}_2]_{i_2 l} \dots [\mathbf{U}_M]_{i_M l}$.

**Definition 3.2.9 (Hadamard Product, $\circledast$)** *Hadamard product of $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{I \times J}$ is an element-wise product defined as $[\mathbf{U} \circledast \mathbf{V}]_{ij} = u_{ij}v_{ij}$.* ■

The Hadamard product is useful in the computation of the rank-$K$ decomposition of a tensor. It is related to the Khatri-Rao product as follows:

$$(\mathbf{U} \odot \mathbf{V})^{\mathrm{T}}(\mathbf{U} \odot \mathbf{V}) = \begin{bmatrix} \mathbf{u}_1^{\mathrm{T}} \otimes \mathbf{v}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{u}_L^{\mathrm{T}} \otimes \mathbf{v}_L^{\mathrm{T}} \end{bmatrix} [\mathbf{u}_1 \otimes \mathbf{v}_1 \ \dots \ \mathbf{u}_L \otimes \mathbf{v}_L] \tag{3.22}$$

$$= \begin{bmatrix} \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 \otimes \mathbf{v}_1^{\mathrm{T}}\mathbf{v}_1 & \dots & \mathbf{u}_1^{\mathrm{T}}\mathbf{u}_L \otimes \mathbf{v}_1^{\mathrm{T}}\mathbf{v}_L \\ \vdots & \ddots & \vdots \\ \mathbf{u}_L^{\mathrm{T}}\mathbf{u}_1 \otimes \mathbf{v}_L^{\mathrm{T}}\mathbf{v}_1 & \dots & \mathbf{u}_L^{\mathrm{T}}\mathbf{u}_L \otimes \mathbf{v}_L^{\mathrm{T}}\mathbf{v}_L \end{bmatrix} \tag{3.23}$$

$$= (\mathbf{U}^{\mathrm{T}}\mathbf{U}) \circledast (\mathbf{V}^{\mathrm{T}}\mathbf{V}) \tag{3.24}$$

When dealing with a set of matrices $\mathbf{U}_m \in \mathbb{R}^{I\times J}$, for $1 \leq m \leq M$, the Hadamard product is related to the Khatri-Rao product as follows:

$$(\mathbf{U}_1 \odot \ldots \odot \mathbf{U}_M)^T(\mathbf{U}_1 \odot \ldots \odot \mathbf{U}_M) = (\mathbf{U}_1^T\mathbf{U}_1) \circledast (\mathbf{U}_2^T\mathbf{U}_2) \circledast \ldots \circledast (\mathbf{U}_M^T\mathbf{U}_M). \tag{3.25}$$

The generalization of scalar products, orthogonality, and the Frobenius norm to tensors is straightforward.

**Definition 3.2.10 (Scalar Product)** *The scalar product of two tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1\times\ldots\times I_M}$ is defined as*

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}\rangle = \sum_{i_1=1}^{I_1}\ldots\sum_{i_M=1}^{I_M} a_{i_1 i_2\ldots i_M} b_{i_1 i_2 \ldots i_M}. \tag{3.26}$$

■

If $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}\rangle = 0$, tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ are said to be *orthogonal* tensors.

**Definition 3.2.11 (Frobenius Norm)** *The Frobenius norm of a tensor $\boldsymbol{\mathcal{A}}$ is*

$$\|\boldsymbol{\mathcal{A}}\| = \langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}}\rangle^{1/2}. \tag{3.27}$$

■

### 3.2.1 Tensor Decompositions and Dimensionality Reduction

Unfortunately, there does not exist a higher-order SVD for tensors that inherits all the properties of the matrix SVD [Denis and Dhorne 1989; Kolda 2001],[6] but there are two higher-order tensor decompositions that each retain some of the nice properties—the linear or rank-$R$ decomposition and the multilinear or rank-$(R_1, R_2, \ldots, R_M)$ decomposition. For matrices, the linear rank-$R$ decomposition and the multilinear (bilinear) rank-$(R_1, R_2)$ decomposition (with $R_1 = R_2 = R$ necessarily) are equivalent, since $\mathbf{\Sigma}$ is a diagonal matrix, and the multilinear rank decomposition reduces to a linear rank decomposition; hence, one need not distinguish between these decompositions in the case of matrices. They are both computed by the matrix SVD (along with the orthonormal subspaces associated with the column space $\mathbf{U}_1$ and row space $\mathbf{U}_2$). This is demonstrated by the following simple $R = 2$ example:

$$\mathbf{A} = \mathbf{U}_1\mathbf{\Sigma}\mathbf{U}_2^{\mathrm{T}} \tag{3.28}$$

$$= \begin{bmatrix} \mathbf{u}_1^{(1)} & \mathbf{u}_1^{(2)} \end{bmatrix} \begin{bmatrix} \sigma_{11} = \sigma_1 & \sigma_{12} = 0 \\ \sigma_{21} = 0 & \sigma_{22} = \sigma_2 \end{bmatrix} \begin{bmatrix} \mathbf{u}_2^{(1)} & \mathbf{u}_2^{(2)} \end{bmatrix}^{\mathrm{T}} \tag{3.29}$$

$$= \sum_{r=1}^{R} \sigma_r\, \mathbf{u}_1^{(r)} \circ \mathbf{u}_2^{(r)} \qquad \text{Rank-}R \text{ linear decomposition} \tag{3.30}$$

[6]An SVD is a *combinatorial orthogonal rank decomposition*, but the converse is not necessarily true. In general, rank decomposition is not necessarily singular value decomposition. For further discussion on the differences between matrix SVD, rank decomposition, and orthogonal rank decomposition for higher order tensors, see [Kolda 2001].

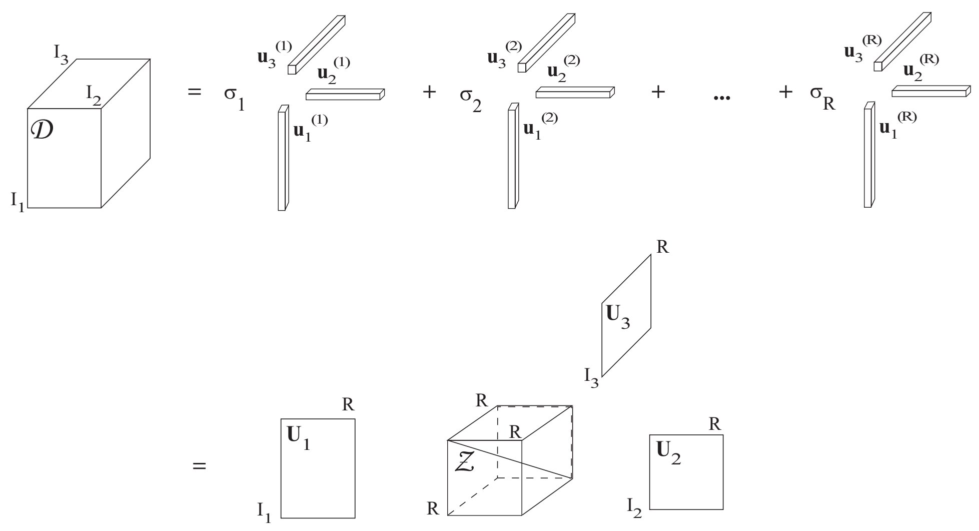


*Figure 3.2: The rank-$R$ decomposition of the data tensor $\boldsymbol{\mathcal{D}}$ yields a sum of rank-$1$ tensors which can be expressed as a product of a diagonal core tensor $\boldsymbol{\mathcal{Z}}$ and $M$ (not necessarily orthonormal) mode matrices $\mathbf{U}_1 \dots \mathbf{U}_M$. The decomposition $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$ of a third-order $(M = 3)$ tensor $\boldsymbol{\mathcal{D}}$ is illustrated.*

$$= \sum_{i=1}^{R} \sum_{j=1}^{R} \sigma_{ij}\, \mathbf{u}_1^{(i)} \circ \mathbf{u}_2^{(j)} \qquad \text{Rank-}(R,R) \text{ multilinear decomposition} \tag{3.31}$$

When dealing with higher-order tensors, however, the difference between the linear rank-$R$ and multilinear rank-$(R_1, R_2, \dots, R_M)$ decompositions becomes apparent and important. Furthermore, for higher-order tensors, there is no trivial counterpart to dimensionality reduction in the matrix SVD case through truncation.

### Linear, Rank-$R$ Decomposition

The tensor rank-$R$ decomposition is a natural generalization of the matrix rank-$R$ decomposition, but unfortunately the $\mathbf{U}_m$ are no longer orthonormal matrices.

**Definition 3.2.12 (Tensor Rank and Rank-$R$ Decomposition)** *The rank of an $M^{th}$-order tensor $\boldsymbol{\mathcal{A}}$, denoted $R = \operatorname{rank}(\boldsymbol{\mathcal{A}})$, is the minimal number of rank-1 tensors that yield $\boldsymbol{\mathcal{A}}$ in a linear combination:*

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \sigma_r\, \mathbf{u}_1^{(r)} \circ \mathbf{u}_2^{(r)} \circ \dots \circ \mathbf{u}_M^{(r)} \tag{3.32}$$

$$= \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \dots \times_M \mathbf{U}_M, \tag{3.33}$$

*which is known as the rank-$R$ decomposition of $\boldsymbol{\mathcal{A}}$, where for $1 \le m \le M$ the $\mathbf{u}_m^{(r)}$ are unit vectors, $\mathbf{U}_m \in \mathbb{R}^{I_m \times R} = [\mathbf{u}_m^{(1)} \dots \mathbf{u}_m^{(r)} \dots \mathbf{u}_m^{(R)}]$, and $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{R \times R \times \dots \times R} = \operatorname{diag}(\sigma_1 \dots \sigma_r \dots \sigma_R)$ is a diagonal tensor of order $M$.* ■

**Algorithm 3.2.1** CANDECOMP/PARAFAC (CP) algorithm.

```
Input $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$ and a desired $K$.

1. Initialization (Random):
   For $m := 1, \ldots, M$, set $\mathbf{U}_m$ to be a random $I_m \times K$ matrix.

2. Local optimization:
   Iterate for $n := 1, \ldots, N$
      For $m := 1, \ldots, M$,
         Set $\mathbf{U}_m := \mathbf{D}_{[m]} (\mathbf{U}_M \odot \ldots \odot \mathbf{U}_{m+1} \odot \mathbf{U}_{m-1} \odot \ldots \odot \mathbf{U}_1)$
             $\left(\mathbf{U}_1^{\mathrm{T}}\mathbf{U}_1 \circledast \ldots \circledast \mathbf{U}_{m-1}^{\mathrm{T}}\mathbf{U}_{m-1} \circledast \mathbf{U}_{m+1}^{\mathrm{T}}\mathbf{U}_{m+1} \circledast \ldots \circledast \mathbf{U}_M^{\mathrm{T}}\mathbf{U}_M\right)^{-1}$.
   until convergence.[a]

Output the converged mode matrices $\mathbf{U}_1, \ldots, \mathbf{U}_M$.
```

[a]Note that $N$ is a pre-specified maximum number of iterations. A possible convergence criterion is to compute at each iteration the approximation error $e_n := \|\boldsymbol{\mathcal{D}} - \tilde{\boldsymbol{\mathcal{D}}}\|^2$ and test if $e_{n-1} - e_n \leq \epsilon$ for a sufficiently small tolerance $\epsilon$.

This is illustrated in Figure 3.2 for the case $M = 3$. Denis and Dhorne [1989] and, independently, Kolda [2001] show that the Eckard-Young theorem does not extend to higher-order tensors. Thus, the rank-$R$ decomposition for a general class of tensors cannot be computed sequentially by extracting the rank-1 approximation in a greedy manner as in the matrix case.

The *CANDECOMP/PARAFAC (CP) algorithm* (Algorithm 3.2.1) [Carroll and Chang 1970; Harshman 1970] iteratively computes the best low-rank approximation of a tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_1 \times \ldots \times I_m \ldots \times I_M}$ as a sum of a given number $K$ of rank-1 terms.[7] The approximation is expressed as

$$\tilde{\boldsymbol{\mathcal{D}}} = \sum_{k=1}^{K} \mathbf{u}_1^{(k)} \circ \mathbf{u}_2^{(k)} \circ \ldots \circ \mathbf{u}_M^{(k)} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_M \mathbf{U}_M, \tag{3.34}$$

where $\boldsymbol{\mathcal{Z}}$ is a diagonal tensor of ones along its main diagonal, and $\mathbf{U}_m \in \mathbb{R}^{I_m \times K}$. In view of the form of $\boldsymbol{\mathcal{Z}}$, the approximation may be expressed in matrix form as

$$\begin{aligned}\tilde{\mathbf{D}}_{[m]} &= \mathbf{U}_m \mathbf{Z}_{[m]} (\mathbf{U}_M \otimes \ldots \otimes \mathbf{U}_{m+1} \otimes \mathbf{U}_{m-1} \otimes \ldots \otimes \mathbf{U}_1)^{\mathrm{T}} && (3.35)\\ &= \mathbf{U}_m (\mathbf{U}_M \odot \ldots \odot \mathbf{U}_{m+1} \odot \mathbf{U}_{m-1} \odot \ldots \odot \mathbf{U}_1)^{\mathrm{T}} && (3.36)\\ &= \mathbf{U}_m \mathbf{W}_m^{\mathrm{T}}. && (3.37)\end{aligned}$$

The CP algorithm computes it by finding a local minimum of $e = \|\boldsymbol{\mathcal{D}} - \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \ldots \times_M \mathbf{U}_M\|^2$ by cycling through the modes, solving for $\mathbf{U}_m$ in the equation $\partial e / \partial \mathbf{U}_m = 0$ while holding all the other

[7] Note that $K$ must be specified to the CP algorithm; it cannot solve the classical "rank-$R$" problem for tensors [Kruskal 1989], which is to find the minimal number $R$ of rank-1 tensors that yield a rank $R$ tensor $\boldsymbol{\mathcal{A}}$ in linear combination.

mode matrices constant, and repeating until convergence. Note that

$$\frac{\partial e}{\partial \mathbf{U}_m} = \frac{\partial}{\partial \mathbf{U}_m}\|\tilde{\mathbf{D}}_{[m]} - \mathbf{U}_m \mathbf{W}_m^{\mathrm{T}}\|^2 = -\tilde{\mathbf{D}}_{[m]}\mathbf{W}_m + \mathbf{U}_m \mathbf{W}_m^{\mathrm{T}}\mathbf{W}_m. \tag{3.38}$$

Thus, $\partial e/\partial \mathbf{U}_m = 0$ implies that

$$\begin{aligned}\mathbf{U}_m &= \tilde{\mathbf{D}}_{[m]}\mathbf{W}_m(\mathbf{W}_m^{\mathrm{T}}\mathbf{W}_m)^{-1}\\ &= \tilde{\mathbf{D}}_{[m]}(\mathbf{U}_M \odot \ldots \odot \mathbf{U}_{m+1} \odot \mathbf{U}_{m-1} \odot \ldots \odot \mathbf{U}_1)\\ &\quad(\mathbf{U}_1^{\mathrm{T}}\mathbf{U}_1 \circledast \ldots \circledast \mathbf{U}_{m-1}^{\mathrm{T}}\mathbf{U}_{m-1} \circledast \mathbf{U}_{m+1}^{\mathrm{T}}\mathbf{U}_{m+1} \circledast \ldots \mathbf{U}_M^{\mathrm{T}}\mathbf{U}_M)^{-1},\end{aligned} \tag{3.39}$$

where $\mathbf{W}_m = (\mathbf{U}_M \odot \ldots \odot \mathbf{U}_{m+1} \odot \mathbf{U}_{m-1} \odot \ldots \odot \mathbf{U}_1)$, and $\mathbf{W}_m^{\mathrm{T}}\mathbf{W}_m$ may be computed more efficiently by taking advantage of the Hadamard/Khatri-Rao property (3.25). This optimization is the basis of Algorithm 3.2.1.

### Multilinear, Rank-$(R_1, R_2, \ldots, R_M)$ Decomposition

The rank-$(R_1, R_2, \ldots, R_M)$ decomposition (a.k.a. the Tucker decomposition) does not reveal the rank of the tensor, but it naturally generalizes the orthonormal subspaces corresponding to the left/right matrices computed by the matrix SVD [Tucker 1966; Kroonenberg and de Leeuw 1980; Kapteyn et al. 1986; de Lathauwer et al. 2000b].

**Theorem 3.2.1 (Rank-$(R_1, R_2, \ldots, R_M)$ (or Multilinear) Decomposition / $M$-mode SVD)** *Let $\boldsymbol{\mathcal{A}}$ be a $I_1 \times I_2 \ldots \times I_m \ldots \times I_M$ tensor for $1 \le m \le M$. Every such tensor can be decomposed as follows:*

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_m \mathbf{U}_m \ldots \times_M \mathbf{U}_M \tag{3.40}$$

$$= \sum_{i_1=1}^{R_1}\sum_{i_2=1}^{R_2}\ldots\sum_{i_m=1}^{R_m}\ldots\sum_{i_M=1}^{R_M} \sigma_{i_1 i_2 \ldots i_M}\, \mathbf{u}_1^{(i_1)} \circ \mathbf{u}_2^{(i_2)} \ldots \circ \mathbf{u}_m^{(i_m)} \ldots \circ \mathbf{u}_M^{(i_M)}, \tag{3.41}$$

*where $\mathbf{U}_m \in \mathbb{R}^{I_m \times R_m} = [\mathbf{u}_m^{(1)}\mathbf{u}_m^{(2)} \ldots \mathbf{u}_m^{(i_m)} \ldots \mathbf{u}_m^{(R_m)}]$ are orthonormal mode-$m$ matrices, where $1 \le R_m \le I_m$ for $1 \le m \le M$, and $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{R_1 \times R_2 \ldots \times R_m \ldots \times R_M}$.[8] The subtensors $\boldsymbol{\mathcal{Z}}_{i_m=a}$ and $\boldsymbol{\mathcal{Z}}_{i_m=b}$ obtained by fixing the $m^{th}$ index to $a$ and $b$ are orthogonal for all values of $m$, $a$, and $b$ when $a \neq b$. The $\|\boldsymbol{\mathcal{Z}}_{i_m=a}\| = \sigma_a^{(m)}$ is the $a^{th}$ mode-$m$ singular value of $\boldsymbol{\mathcal{A}}$ and the $a^{th}$ column vector of $\mathbf{U}_m$, such that $\|\boldsymbol{\mathcal{Z}}_{i_m=1}\| \ge \|\boldsymbol{\mathcal{Z}}_{i_m=2}\| \ge \ldots \|\boldsymbol{\mathcal{Z}}_{i_m=R_m}\| \ge 0$.* ■

This is illustrated in Figure 3.3 for the case $M = 3$. Tensor $\boldsymbol{\mathcal{Z}}$, known as the *core tensor*, is analogous to the diagonal singular value matrix in conventional matrix SVD (although it does not have a simple, diagonal structure). The core tensor contains normalizing parameters associated with the different *mode matrices* $\mathbf{U}_1, \ldots, \mathbf{U}_M$. Mode matrix $\mathbf{U}_m$ contains the orthonormal vectors spanning the column space of matrix $\mathbf{A}_{[m]}$ resulting from the *mode-$m$ matrixizing* of $\boldsymbol{\mathcal{A}}$ (Figure 3.1).

[8]From (3.19), we can also write in vector form, $\mathrm{vec}(\boldsymbol{\mathcal{A}}) = (\mathbf{U}_M \otimes \mathbf{U}_{M-1} \ldots \otimes \mathbf{U}_m \ldots \otimes \mathbf{U}_1)\mathrm{vec}(\boldsymbol{\mathcal{Z}})$.

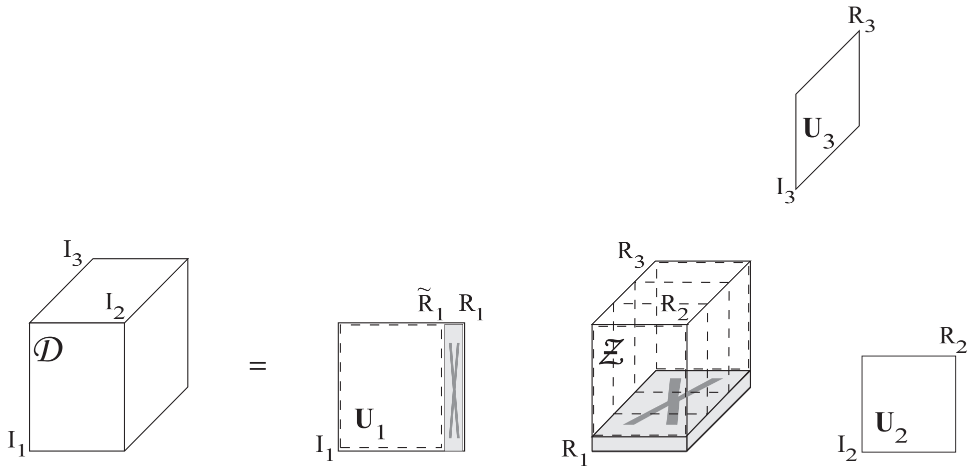


*Figure 3.3: The multilinear, or rank-$(R_1, R_2, \ldots, R_M)$, decomposition decomposes the data tensor $\boldsymbol{\mathcal{D}}$ into the product of a core tensor $\boldsymbol{\mathcal{Z}}$ and $M$ orthogonal mode matrices $\mathbf{U}_1 \ldots \mathbf{U}_M$. For the $M = 3$ case illustrated here, $\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$. Deletion of the last mode-1 eigenvector of $\mathbf{U}_1$ incurs an error in the approximation equal to $\sigma^2_{I_1}$, which equals the Frobenius norm of the (grey) subtensor of $\boldsymbol{\mathcal{Z}}$ whose row vectors would normally multiply the eigenvector in the mode-1 product $\boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1$.*

**Algorithm 3.2.2** $M$-mode SVD algorithm.

```
Input the data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$.

  1. For $m := 1, \ldots, M$,
     Let $\mathbf{U}_m$ be the left orthonormal matrix of the SVD of $\mathbf{D}_{[m]}$, the mode-$m$ matrixized $\boldsymbol{\mathcal{D}}$.
  2. Set $\boldsymbol{\mathcal{Z}} := \boldsymbol{\mathcal{D}} \times_1 \mathbf{U}_1^{\mathrm{T}} \times_2 \mathbf{U}_2^{\mathrm{T}} \ldots \times_m \mathbf{U}_m^{\mathrm{T}} \ldots \times_M \mathbf{U}_M^{\mathrm{T}}$.

Output the mode matrices $\mathbf{U}_1, \ldots, \mathbf{U}_M$ and the core tensor $\boldsymbol{\mathcal{Z}}$.
```

Algorithm 3.2.2 is the $M$*-mode SVD algorithm* for decomposing an $M$th-order tensor $\boldsymbol{\mathcal{D}}$ according to equation (3.40). It employs $M$ matrix SVDs to compute the $M$ orthonormal mode matrices and then computes the core tensor.

As we stated earlier, there is no trivial counterpart for higher-order tensors to dimensionality reduction in the matrix SVD case. According to [Kroonenberg and de Leeuw 1980; Kapteyn et al. 1986; de Lathauwer et al. 2000b], a useful generalization in the tensor case involves an optimal rank-$(\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_M)$ approximation which iteratively optimizes each of the modes of the given tensor, where each optimization step involves a best reduced-rank approximation of a positive semi-definite symmetric matrix.

A truncation of the mode matrices of the data tensor $\boldsymbol{\mathcal{D}}$ results in an approximation $\tilde{\boldsymbol{\mathcal{D}}}$ with reduced ranks $\tilde{R}_m \leq R_m$, where $R_m = \mathrm{rank}_m(\boldsymbol{\mathcal{D}}) = \mathrm{rank}(\mathbf{D}_{[m]}) = \mathrm{rank}(\mathbf{U}_m)$ is the $m$-rank of $\boldsymbol{\mathcal{D}}$ for

**Algorithm 3.2.3** $M$-mode SVD dimensionality reduction algorithm.

```
Input the data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$ and desired ranks $\tilde{R}_1, \ldots, \tilde{R}_M$.

  1. Initialization (Truncated $M$-Mode SVD):
     Apply Step 1 of the $M$-mode SVD algorithm (Algorithm 3.2.2) to $\boldsymbol{\mathcal{D}}$.
     For $m := 1, 2, \ldots, M$, truncate to $\tilde{R}_m$ columns the mode matrix $\mathbf{U}_m \in \mathbb{R}^{I_m \times \tilde{R}_m}$.[a]

  2. Local optimization via alternating least squares:
     Iterate for $n := 1, \ldots, N$

         For $m := 1, \ldots, M$,
             Set $\boldsymbol{\mathcal{X}} := \boldsymbol{\mathcal{D}} \times_1 \mathbf{U}_1^{\mathrm{T}} \ldots \times_{m-1} \mathbf{U}_{m-1}^{\mathrm{T}} \times_{m+1} \mathbf{U}_{m+1}^{\mathrm{T}} \ldots \times_M \mathbf{U}_M^{\mathrm{T}}$.
             Set $\mathbf{U}_m$ to the $\tilde{R}_m$ leading left-singular vectors of the SVD of $\mathbf{X}_{[m]}$.
         Set $\boldsymbol{\mathcal{Z}} := \boldsymbol{\mathcal{X}} \times_M \mathbf{U}_M^{\mathrm{T}}$.

     until convergence.[b]

Output the converged core tensor $\boldsymbol{\mathcal{Z}}$ and mode matrices $\mathbf{U}_1, \ldots, \mathbf{U}_M$.
```

[a] The complexity of computing the SVD of an $m \times n$ matrix $\mathbf{A}$ is $O(mn \min(m, n))$, which is costly when both $m$ and $n$ are large. However, we can efficiently compute the $\tilde{R}$ leading left-singular vectors of $\mathbf{A}$ by first computing the rank-$\tilde{R}$ modified Gram-Schmidt (MGS) orthogonal decomposition $\mathbf{A} \simeq \mathbf{QR}$, where $\mathbf{Q}$ is $m \times \tilde{R}$ and $\mathbf{R}$ is $\tilde{R} \times n$, and then computing the SVD of $\mathbf{R}$ and multiplying it as follows: $\mathbf{A} \simeq \mathbf{Q}(\tilde{\mathbf{U}}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}) = (\mathbf{Q}\tilde{\mathbf{U}})\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$.

[b] See Footnote *a* in Algorithm 3.2.1.

$1 \le m \le M$. The error of this approximation is

$$\|\boldsymbol{\mathcal{D}} - \tilde{\boldsymbol{\mathcal{D}}}\|^2 = \sum_{i_1=\tilde{R}_1+1}^{R_1} \sum_{i_2=\tilde{R}_2+1}^{R_2} \cdots \sum_{i_M=\tilde{R}_M+1}^{R_M} \boldsymbol{\mathcal{Z}}^2_{i_1 i_2 \ldots i_M} \tag{3.42}$$

The error is bounded by the sum of squared singular values associated with the discarded singular vectors:

$$\|\boldsymbol{\mathcal{D}} - \tilde{\boldsymbol{\mathcal{D}}}\|^2 \le \sum_{i_1=\tilde{R}_1+1}^{R_1} \sigma^2_{i_1} + \sum_{i_2=\tilde{R}_2+1}^{R_2} \sigma^2_{i_2} + \cdots + \sum_{i_M=\tilde{R}_M+1}^{R_M} \sigma^2_{i_M}. \tag{3.43}$$

Note that the singular value associated with the $n^{\text{th}}$ singular vector in mode matrix $\mathbf{U}_m$ is equal to $\|\boldsymbol{\mathcal{Z}}_{i_m=n}\|$; i.e., the Frobenius norm of subtensor $\boldsymbol{\mathcal{Z}}_{i_m=n}$ of the core tensor $\boldsymbol{\mathcal{Z}}$ (Figure 3.3).

Truncation of the mode matrices resulting from the $M$-mode SVD algorithm may yield a good reduced-dimensionality approximation $\tilde{\boldsymbol{\mathcal{D}}}$, but it is generally not optimal. A locally optimal dimensionality reduction scheme for tensors is to compute a *best rank-($\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_M$) approximation* $\tilde{\boldsymbol{\mathcal{D}}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_M \mathbf{U}_M$ with orthonormal $I_m \times \tilde{R}_m$ mode matrices $\mathbf{U}_m$, for $m = 1, \ldots, M$, which minimizes the error function

$$e = \|\boldsymbol{\mathcal{D}} - \tilde{\boldsymbol{\mathcal{D}}}\| + \sum_{m=1}^{M} \boldsymbol{\Lambda}_m \|\mathbf{U}_m^{\mathrm{T}}\mathbf{U}_m - \mathbf{I}\|, \tag{3.44}$$

where the $\boldsymbol{\Lambda}_m$ are Lagrange multiplier matrices.[9]

To this end, the *$M$-mode dimensionality reduction algorithm* (Algorithm 3.2.3) iteratively computes $\mathbf{U}_m$, for $m = 1, \ldots, M$, and $\boldsymbol{\mathcal{Z}}$ by performing in Step 2 a higher-order extension of the orthogonal iteration for matrices. Note that the $M$-mode orthogonal iteration finds only a local minimum of this generally non-convex error function.

[9]This best rank-$(\tilde{R}_1, \tilde{R}_2, \ldots, \tilde{R}_M)$ problem should also not be confused with the classical "rank-$R$" problem for tensors mentioned in Footnote 7.

4

# The Tensor Algebraic Framework

**Contents**

**Algorithms**

In this chapter, we develop a multilinear framework for the analysis of multimodal observational data. An observation comprises a set of measurements whose values are influenced by multiple underlying causal factors. The measurements are also known as response variables since their values change in response to changes in the causal factors. The causal factors are not directly measurable, and the variables extracted by data analysis in order to represent them are known as explanatory variables. For example, an image is an observation whose measurements are pixels, the values of which vary with changes in the causal factors—scene structure, illumination, view, etc. In the previous chapter, we discussed the decomposition of an $M^{\text{th}}$-order data tensor in terms of $M$ mode matrices. These matrices are associated with the measurement mode as well as with the $M-1$ causal factor modes. When performing data analysis, we are interested in modeling the causal relationship, therefore we will be decomposing the data tensor only in terms of the $M-1$ causal factors.

First we review Principle Components Analysis (PCA) and Independent Components Analysis (ICA). Next, we develop Multilinear PCA (MPCA) and Multilinear ICA (MICA), the key methods in our framework, which overcome the limitations of their linear counterparts (in Chapter 7, we will expand our tensor framework by introducing multilinear projection for recognition). We close the chapter by discussing generalizations of our multilinear algorithms and the extension of our multifactor approach to include other manifold learning methods.

## 4.1 PCA and ICA

Principal Components Analysis computes an optimal orthonormal linear data representation by modeling the variance in the data, but without distinguishing between different sources of variation. Thus, it is best suited at capturing single causal factor linear variation in the observed data.

Let a multivariate sensory observation be represented by a column vector. An ensemble of observations form a *data matrix*, or two-way array, $\mathbf{D}$ where each column is a measurement vector. The underlying assumption is that the observations reside in a low-dimensional subspace; i.e., that the column vectors of $\mathbf{D}$ span a low dimensional subspace.

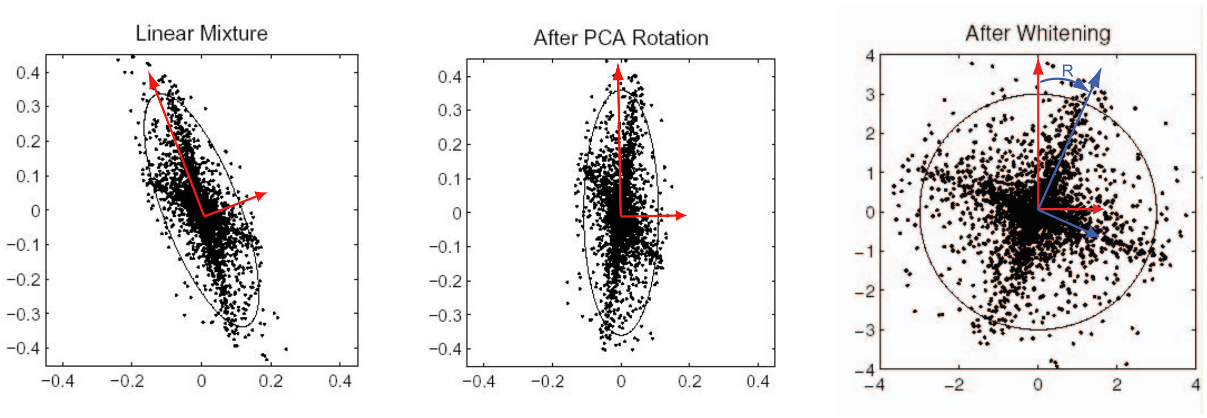
Principal components analysis of an ensemble of $I_2$ observations is computed by performing an SVD on a $I_1 \times I_2$ data matrix $\mathbf{D}$ whose columns are the "vectorized", "centered" observations. Each vectorized, centered observation is obtained by subtracting the mean observation from each input observation of the ensemble and identically arranging the resulting measurements into a column vector. It is important to realize that this regards entire observations (e.g., images) as vectors or points in high ($I_1$) dimensional space. This enables PCA to model the full second-order statistics, the covariance between all possible pairs of measurements (pixel) in the observation (image) as we discuss in detail in Appendix A.

The data matrix $\mathbf{D} \in \mathbb{R}^{I_1 \times I_2}$ is a two-mode mathematical object that has two associated vector spaces, a row space and a column space. In a factor analysis of $\mathbf{D}$, the SVD orthogonalizes these two spaces and decomposes the matrix as

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} = \underbrace{\mathbf{U}}_{\substack{\text{PCA Basis}\\ \text{Matrix}}} \quad \underbrace{\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}}_{\substack{\text{PCA Coefficient}\\ \text{Matrix}}}, \tag{4.1}$$

the product of an orthonormal column-space represented by the left matrix $\mathbf{U} \in \mathbb{R}^{I_1 \times R}$, a diagonal singular value matrix $\mathbf{\Sigma} \in \mathbb{R}^{R \times R}$ with diagonal entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_R \geq 0$ called the singular values of $\mathbf{D}$, and an orthonormal row space represented by the right matrix $\mathbf{V} \in \mathbb{R}^{I_2 \times R}$. The left-singular vectors in $\mathbf{U}$ are also called the *principal component* (or Karhunen-Loeve) directions of $\mathbf{D}$.[1]

[1]This is because the covariance matrix is given by $\mathbf{S} = \mathbf{D}\mathbf{D}^{\mathrm{T}}/I_2$, and from (4.1) we have the eigen-decomposition of $\mathbf{D}\mathbf{D}^{\mathrm{T}} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^{\mathrm{T}}$, which is the eigen-decomposition of $\mathbf{S}$ up to a factor $I_2$.



(a) (b) (c)

*Figure 4.1: Geometric view of the relationship between PCA and ICA. (a) Compute the PCA basis vectors associated with the data. (b) Represent the data relative to the PCA basis vectors. PCA applies a rotation to the data. (c) Whiten the data and compute the rotation* $\mathbf{R}$ *that finds the arms in the data, thus yielding the ICA basis vectors.*

Optimal dimensionality reduction in matrix principal components analysis is obtained by truncation of the singular value decomposition (i.e., deleting eigenvectors associated with the smallest singular values).

While PCA seeks an *accurate* representation in least-square sense, ICA seeks a *meaningful* representation by employing additional information not contained in the covariance matrix. The independent components analysis of multivariate data seeks a sequence of projections such that the projected data are statistically independent. One way of achieving statistical independence is by seeking a set of projections which results in projected data that looks as far from Gaussian as possible. Figure 4.1 provides a geometric interpretation of the computation of ICA. ICA and PCA may yield different subspaces, as shown in Figure 4.2, which will yield different recognition results. The subspaces for PCA and ICA are identical when the directions of greatest variation and the directions of the independent components span the same subspace, in which case the recognition results will be identical.

ICA can be applied in two ways [Bartlett et al. 2002]:

1. to $\mathbf{D}^{\mathrm{T}}$, each of whose rows is a different image, which finds a spatially independent basis set that reflects the local properties of imaged objects, or

2. to $\mathbf{D}$, which finds a set of coefficients that are statistically independent while the basis reflects the global properties of imaged objects.

In **Approach 1**, ICA starts essentially from the factor analysis or PCA solution (4.1), and applies an invertible transformation to the principal components such that they become *independent components* [Hastie et al. 2001] as follows:

$$\mathbf{D}^{\mathrm{T}} \simeq \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}} \tag{4.2}$$

$$= \left(\mathbf{V}\boldsymbol{\Sigma}\mathbf{W}^{-1}\right)\left(\mathbf{W}\mathbf{U}^{\mathrm{T}}\right) \tag{4.3}$$

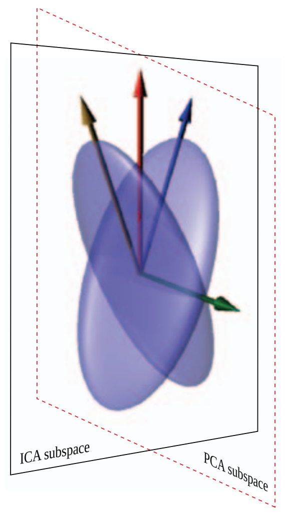


*Figure 4.2: PCA and ICA may yield different subspaces for analysis. The ellipsoids represent a bi-lobed dataset. The red and green axes capture the greatest data variation and indicate the top two principal components, while the blue and yellow axes indicate the independent components. Thus, in this example, the directions of greatest variations and the directions of independent components define two subspaces that are mutually othogonal.*

$$= \mathbf{K}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}, \tag{4.4}$$

where every column of $\mathbf{D}$ is a different image, $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$ are rank-reduced, $\mathbf{W}$ is an invertible transformation matrix that is computed by the ICA algorithm, $\mathbf{C} = \mathbf{U}\mathbf{W}^{\mathrm{T}}$ are the independent components, and $\mathbf{K} = \mathbf{W}^{-T}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ are the coefficients. Various objective functions, such as those based on mutual information, negentropy, higher-order cumulants, etc., are presented in the literature for computing the independent components along with different optimization methods for extremizing these objective functions [Hyvärinen et al. 2001]. Dimensionality reduction with ICA is usually performed in the PCA preprocessing stage.

Alternatively, in **Approach 2**, ICA is applied to $\mathbf{D}$ and it rotates the principal components directions such that the coefficients are statistically independent, as follows:

$$\mathbf{D} \simeq \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{4.5}$$

$$= \left(\mathbf{U}\mathbf{W}^{-1}\right)\left(\mathbf{W}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\right) \tag{4.6}$$

$$= \mathbf{C}\mathbf{K}, \tag{4.7}$$

where $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{\Sigma}$ are rank-reduced, $\mathbf{C} = \mathbf{U}\mathbf{W}^{-1}$ is the basis matrix and $\mathbf{K} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ are the statistically independent coefficients. Note that the $\mathbf{W}$ matrices in (4.6) differ from the analogous $\mathbf{W}$ matrices in (4.3); hence, the $\mathbf{K}$ and $\mathbf{C}$ matrices also differ.

PCA and ICA are linear analysis methods, hence they are not well suited to the representation of multifactor image ensembles. To address this shortcoming, the next two sections develop their multilinear generalizations in turn.

## 4.2 Multilinear PCA (MPCA)

The analysis of an ensemble of observations (e.g., images) resulting from the confluence of multiple causal factors (e.g., scene structure, illumination, view, etc.) is a problem in multilinear algebra. Within this mathematical framework, the ensemble of observations is organized as a data tensor. This data tensor $\boldsymbol{\mathcal{D}}$ must be decomposed in order to separate and parsimoniously represent the causal factors. To this end, we prescribe the $M$-mode SVD (3.40):

$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_m \mathbf{U}_m \ldots \times_M \mathbf{U}_M. \tag{4.8}$$

As was stated in Section 3.2, multilinear analysis decomposes the tensor as the mode-$m$ product of $M$-orthogonal spaces.

Now, let us assume that $m = 1$ is the measurement (pixel) mode and that the remaining modes $m = 2, \ldots, M$ are the causal factor modes; i.e., those modes associated with the relevant causal factors that underlie the generation of observations (images). Rather than computing the core tensor $\boldsymbol{\mathcal{Z}}$, we prefer to define the basis tensor

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1, \tag{4.9}$$

which we call the *extended core tensor*, that makes explicit the interaction between the causal factors:

$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{T}} \times_2 \mathbf{U}_2 \ldots \times_m \mathbf{U}_m \ldots \times_M \mathbf{U}_M. \tag{4.10}$$

Given this multilinear model, an observation (image) may be synthetized according to the expression

$$\mathbf{d} = \boldsymbol{\mathcal{T}} \times_2 \mathbf{r}_2^{\mathrm{T}} \ldots \times_m \mathbf{r}_m^{\mathrm{T}} \ldots \times_M \mathbf{r}_M^{\mathrm{T}}, \tag{4.11}$$

where, for $m = 2, \ldots, M$, vector $\mathbf{r}_m$ is the representation vector associated with causal factor $m$. Note, in particular, that if $\mathbf{r}_m$ is chosen to be $\mathbf{c}_i$, $i = 1, \ldots, I_M$, a row of $\mathbf{U}_m$, for $m = 2, \ldots, M$, then $\mathbf{d}$ will reconstruct any one of the images in the ensemble $\boldsymbol{\mathcal{D}}$.

### Efficient MPCA Algorithm

Recall the $M$-mode SVD algorithm (Algorithm 3.2.2) for decomposing $\boldsymbol{\mathcal{D}}$ according to (4.8). Regarding Step 1 of the algorithm, note that when $\mathbf{D}_{[m]}$ is a nonsquare matrix, the computation of $\mathbf{U}_m$ in the SVD $\mathbf{D}_{[m]} = \mathbf{U}_m \boldsymbol{\Sigma} \mathbf{V}_m^{\mathrm{T}}$ can be performed efficiently, depending on which dimension of $\mathbf{D}_{[m]}$ is smaller, by decomposing either $\mathbf{D}_{[m]} \mathbf{D}_{[m]}^{\mathrm{T}} = \mathbf{U}_m \boldsymbol{\Sigma}^2 \mathbf{U}_m^{\mathrm{T}}$ (note that $\mathbf{V}_m^{\mathrm{T}} = \boldsymbol{\Sigma}^{+} \mathbf{U}_m^{\mathrm{T}} \mathbf{D}_{[m]}$) or by decomposing $\mathbf{D}_{[m]}^{\mathrm{T}} \mathbf{D}_{[m]} = \mathbf{V}_m \boldsymbol{\Sigma}^2 \mathbf{V}_m^{\mathrm{T}}$ and then computing $\mathbf{U}_m = \mathbf{D}_{[m]} \mathbf{V}_m \boldsymbol{\Sigma}^{+}$.

Despite the above improvement, the $M$-mode SVD algorithm will still be inefficient in practice when one or more of the dimensions $I_m$ are large. We can improve it further for practical use in applications. To this end, the mode-$m$ covariance matrix $\mathbf{D}_{[m]} \mathbf{D}_{[m]}^{\mathrm{T}}$ for the causal factor modes $m = 2, \ldots, M$ can be computed directly, element by element, by summing the appropriate inner products between pairs of data points (e.g., images) in the data tensor $\boldsymbol{\mathcal{D}}$ that are associated with mode $m$. In

**Algorithm 4.2.1** Multilinear PCA (MPCA) algorithm.

```
Input the data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$, where mode $m = 1$ is the measurement mode, and the desired ranks $\tilde{R}_2, \ldots, \tilde{R}_M$.

1. For $m := 2, \ldots, M$,
      Compute the elements of the mode-$m$ covariance matrix, for $j, k := 1, \ldots, I_m$, as follows:
      $$\left[\mathbf{D}_{[m]}\mathbf{D}^{\mathrm{T}}_{[m]}\right]_{jk} := \sum_{i_2=1}^{I_2} \cdots \sum_{i_{m-1}=1}^{I_{m-1}} \sum_{i_{m+1}=1}^{I_{m+1}} \cdots \sum_{i_M=1}^{I_M} \mathbf{d}^{\mathrm{T}}_{i_2 \ldots i_{m-1}\, j\, i_{m+1} \ldots i_M} \mathbf{d}_{i_2 \ldots i_{m-1}\, k\, i_{m+1} \ldots i_M}. \quad (4.13)$$
      Set mode matrix $\mathbf{U}_m$ to the left matrix of the SVD of $\mathbf{D}_{[m]}\mathbf{D}^{\mathrm{T}}_{[m]} = \mathbf{U}_m \mathbf{\Sigma}^2 \mathbf{U}^{\mathrm{T}}_m$.
      Truncate to $\tilde{R}_m$ columns $\mathbf{U}_m \in \mathbb{R}^{I_m \times \tilde{R}_m}$. [a]
2. Set $\boldsymbol{\mathcal{T}} := \boldsymbol{\mathcal{D}} \times_2 \mathbf{U}^{\mathrm{T}}_2 \ldots \times_m \mathbf{U}^{\mathrm{T}}_m \ldots \times_M \mathbf{U}^{\mathrm{T}}_M$.
3. Local optimization via alternating least squares:
   Iterate for $n := 1, \ldots, N$
      For $m := 2, \ldots, M$,
         Set $\boldsymbol{\mathcal{X}} := \boldsymbol{\mathcal{D}} \times_2 \mathbf{U}^{\mathrm{T}}_2 \ldots \times_{m-1} \mathbf{U}^{\mathrm{T}}_{m-1} \times_{m+1} \mathbf{U}^{\mathrm{T}}_{m+1} \ldots \times_M \mathbf{U}^{\mathrm{T}}_M$.
         Set $\mathbf{U}_m$ to the $\tilde{R}_m$ leading left-singular vectors of the SVD of $\mathbf{X}_{[m]}$.
      Set $\boldsymbol{\mathcal{T}} := \boldsymbol{\mathcal{X}} \times_M \mathbf{U}^{\mathrm{T}}_M$.
   until convergence.[b]

Output the converged extended core tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \tilde{R}_2 \times \ldots \times \tilde{R}_M}$ and causal factor mode matrices $\mathbf{U}_2, \ldots, \mathbf{U}_M$.
```

[a]See Footnote *a* of Algorithm 3.2.3.
[b]See Footnote *a* in Algorithm 3.2.1.

particular, element $j, k$ of the mode-$m$ covariance matrix is given by

$$\left[\mathbf{D}_{[m]}\mathbf{D}^{\mathrm{T}}_{[m]}\right]_{jk} = \sum_{i_2=1}^{I_2} \cdots \sum_{i_{m-1}=1}^{I_{m-1}} \sum_{i_{m+1}=1}^{I_{m+1}} \cdots \sum_{i_M=1}^{I_M} \mathbf{d}^{\mathrm{T}}_{i_2 \ldots i_{m-1}\, j\, i_{m+1} \ldots i_M} \mathbf{d}_{i_2 \ldots i_{m-1}\, k\, i_{m+1} \ldots i_M}, \qquad (4.12)$$

where $\mathbf{d}_{i_2,\ldots,i_M}$ is a vectorized image that is packed into the data tensor $\boldsymbol{\mathcal{D}}$. This computation will also be useful in generalizations of our multilinear algorithms, as we will discuss in Section 4.4.

Given the above considerations, we obtain a *Multilinear PCA (MPCA) algorithm* (Algorithm 4.2.1) which includes dimensionality reduction. Step 1 of this algorithm performs an efficient $(M-1)$-Mode SVD of input data tensor $\boldsymbol{\mathcal{D}}$, where the measurement mode is not explicitly computed, and truncates each of the causal factor mode matrices $\mathbf{U}_m$, for $m = 2, \ldots, M$, to the specified reduced ranks $\tilde{R}_m$. Then, Step 3 of the algorithm iteratively optimizes the reduced mode matrices. Its theoretical underpinnings are the same as for the $M$-mode dimensionality reduction algorithm (Algorithm 3.2.3). Like the $M$-mode orthogonal iteration in Algorithm 3.2.3, the $(M-1)$-mode orthogonal iteration in Algorithm 4.2.1 finds only a local minimum of the generally non-convex error function (3.44). Note that $\boldsymbol{\mathcal{T}}$ is computed as specified since $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{D}} \times_2 \mathbf{U}_2^{\mathrm{T}} \ldots \times_{m-1} \mathbf{U}_{m-1}^{\mathrm{T}} \times_{m+1} \mathbf{U}_{m+1}^{\mathrm{T}} \ldots \times_M \mathbf{U}_M^{\mathrm{T}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_m \mathbf{U}_m$. Hence, upon completion of the previous "For $m$" step, $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 \times_M \mathbf{U}_M$. Therefore, after convergence, $\boldsymbol{\mathcal{X}} \times_M \mathbf{U}_M^{\mathrm{T}} = \boldsymbol{\mathcal{Z}} \times_1 \mathbf{U}_1 = \boldsymbol{\mathcal{T}}$.

After applying Algorithm 4.2.1 to perform an MPCA with dimensionality reduction on an input data tensor $\boldsymbol{\mathcal{D}}$ of order $M$, where mode $m = 1$ is the measurement mode, the resulting approximation of the input data tensor is given by

$$\boldsymbol{\mathcal{D}} \simeq \tilde{\boldsymbol{\mathcal{D}}} = \boldsymbol{\mathcal{T}} \times_2 \mathbf{U}_2 \ldots \times_M \mathbf{U}_M. \tag{4.14}$$

### Relationships to PCA

To show mathematically that PCA is a special case of our MPCA, we write the latter in terms of matrix notation. A matrix representation of the $M$-mode SVD can be obtained by matrixizing with respect to the measurement (i.e., pixel) mode, and using (3.18), as follows:

$$\underbrace{\mathbf{D}_{[1]}}_{\text{Data}} = \underbrace{\mathbf{U}_1}_{\substack{\text{PCA Basis}\\\text{Matrix}}} \underbrace{\mathbf{Z}_{[1]}(\mathbf{U}_M \otimes \ldots \otimes \mathbf{U}_m \otimes \ldots \otimes \mathbf{U}_2)^{\mathrm{T}}}_{\substack{\text{PCA Coefficient}\\\text{Matrix}}}. \tag{4.15}$$

The above matrix product can be interpreted as a standard linear decomposition of the data ensemble (4.1). Note that $\mathbf{D}_{[1]}$ is a matrix whose columns are a set of observations; hence, it is the PCA data matrix. The measurement mode matrix $\mathbf{U}_1$ is computed by performing an SVD on $\mathbf{D}_{[1]}$; hence, it is the PCA basis matrix. Thus, the product of the matrixized core tensor $\mathbf{Z}_{[1]}$ and the Kronecker product of the mode matrices $\mathbf{U}_M, \ldots, \mathbf{U}_2$ associated with the causal factors is the PCA coefficient matrix. As a consequence, our multilinear analysis subsumes linear, PCA analysis. MPCA goes beyond PCA by further decomposing the PCA coefficient matrix into a set of coefficient matrices associated with the different causal factors.

Although the computation of the measurement mode matrix $\mathbf{U}_1$ in our MPCA is the PCA basis matrix (4.15), more interesting is the relationship between standard PCA and the MPCA causal factor mode matrices. Computing the causal factor mode matrices in our multilinear model amounts to the computation of a set of mutually constrained, cluster-based PCAs. When dealing with data that can be separated into clusters, the standard machine learning approach is to compute a separate PCA. When data from different clusters are generated by the same underlying process (e.g., facial images of the same people under different viewing conditions), the underlying data can be concatenated in the measurement mode and the common causal factor can be modeled by one PCA.[2] Thus, we have defined a *constrained, cluster-based PCA* in the sense that the resulting causal factor representation is constrained

[2]The active appearance model [Cootes et al. 2001] concatenated two different measurements, facial feature locations and texture, to compute a person representation for a particular viewpoint invariant of measurement. More generally, one can concatenate the measurements of a person from different viewpoint clusters to compute a person representation that is invariant of the measurement and the viewpoint.

to be identical in every cluster.

MPCA performs $M$ constrained, cluster-based PCAs, since the computation of the mode-$m$ matrix $\mathbf{U}_m$, which involves a mode-$m$ data tensor flattening and subsequent SVD, is equivalent to performing a constrained, cluster-based PCA; i.e., data cluster concatenation followed by an SVD. In the context of our multifactor data analysis, we define a cluster as a set of observations for which all factors are fixed except one, the $m^{\text{th}}$ factor. Note that there are $N_m = I_2 I_3 \dots I_{m-1} I_{m+1} \dots I_M$ possible clusters and the data in each cluster varies with the same causal mode.[3] Thus, the data across different clusters share one of the underlying causal factors. The constrained, cluster-based PCA concatenates the clusters in the measurement mode and analyzes the data with a linear model, such as PCA.

To see this, let $\boldsymbol{\mathcal{D}}_{i_2 \dots i_{m-1} i_{m+1} \dots i_M} \in \mathbb{R}^{I_1 \times 1 \times \dots \times 1 \times I_m \times 1 \times \dots \times 1}$ denote a subtensor of $\boldsymbol{\mathcal{D}}$ that is obtained by fixing all modes except causal factor mode $m$ and mode 1 (the measurement mode). Matrixizing this subtensor in mode 1, we obtain $\mathbf{D}_{i_2 \dots i_{m-1} i_{m+1} \dots i_M [1]} \in \mathbb{R}^{I_1 \times I_m}$. This data matrix comprises a cluster of data obtained by varying the $m^{\text{th}}$ causal factor, to which one can traditionally apply PCA. Since there are $N_m = I_2 I_3 \dots I_{m-1} I_{m+1} \dots I_M$ possible clusters that share the same underlying space associated with the $m^{\text{th}}$ factor, the data can be concatenated and PCA performed in order to extract the same representation for the $m^{\text{th}}$ factor regardless of the cluster. Now, consider the multilinear PCA computation of mode matrix $\mathbf{U}_m$, which can be written in terms of matrixized subtensors as

$$\mathbf{D}_{[m]} = \begin{bmatrix} \mathbf{D}_{1 \dots 11 \dots 1}{}_{[m]}^{\mathrm{T}} \\ \vdots \\ \mathbf{D}_{I_2 \dots 11 \dots 1}{}_{[m]}^{\mathrm{T}} \\ \vdots \\ \mathbf{D}_{I_2 \dots I_{m-1} I_{m+1} \dots I_M}{}_{[m]}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} = \mathbf{U}_m \mathbf{\Sigma}_m \mathbf{V}_m^{\mathrm{T}}. \tag{4.16}$$

Clearly, this is equivalent to computing a set of $N_m = I_2 I_3 \dots I_{m-1} I_{m+1} \dots I_M$ cluster-based PCAs concurrently by combining them into a single statistical model and representing the underlying causal factor $m$ common to the clusters. Thus, rather than computing a separate linear PCA model for each cluster, MPCA concatenates the clusters into a single statistical model and computes a representation (coefficient vector) for mode $m$ that is invariant relative to the other causal factor modes $2, \dots, (m-1), (m+1), \dots, M$. Therefore, MPCA is a multilinear, constrained, cluster-based PCA.

To clarify the relationship, let us number each of the matrices $\mathbf{D}_{i_2 \dots i_{m-1} i_{m+1} \dots i_M [m]} = \mathbf{D}_m^{(n)}$ with a parenthetical superscript $1 \le n = 1 + \sum_{k=2, k \ne m}^{M} (i_n - 1) \prod_{l=2, l \ne m}^{k-1} I_l \le N_m$. Let each of the linear SVDs be $\mathbf{D}_m^{(n)} = \mathbf{U}_m^{(n)} \mathbf{\Sigma}_m^{(n)} \mathbf{V}_m^{(n)\mathrm{T}}$. Then $\mathbf{D}_{[m]} = [\mathbf{U}_m^{(1)} \mathbf{\Sigma}_m^{(1)} \dots \mathbf{U}_m^{(N_m)} \mathbf{\Sigma}_m^{(N_m)}] \,\mathrm{diag}([\mathbf{V}_m^{(1)} \dots \mathbf{V}_m^{(N_m)}])^{\mathrm{T}}$, where $\mathrm{diag}(\cdot)$ denotes a diagonal matrix whose elements are each of the elements of its vector argument. The measurement mode matrix $\mathbf{V}_m^{(n)}$ contains the eigenvectors that span the observed data in cluster $n$. The SVD of the first matrix is $[\mathbf{U}_m^{(1)} \mathbf{\Sigma}_m^{(1)} \dots \mathbf{U}_m^{(N_m)} \mathbf{\Sigma}_m^{(N_m)}] = \mathbf{U}_m \mathbf{\Sigma}_m \mathbf{W}_m^{\mathrm{T}}$. Then we can write (4.16)

[3] Observations in the cluster may not be in Euclidean proximity in the measurement space. Consequently, a cluster may not be easily identified through a standard EM algorithm, but may be computable using a "coupled-EM" that would generalize the EM algorithm to extend a set of clusters. Such an algorithm has not been applied to define the clusters in these experiments.

as $\mathbf{D}_{[m]} = \mathbf{U}_m \mathbf{\Sigma}_m \mathbf{V}_m^{\mathrm{T}}$, where $\mathbf{V}_m^{\mathrm{T}} = \mathbf{W}_m^{\mathrm{T}} \operatorname{diag}([\mathbf{V}_m^{(1)} \dots \mathbf{V}_m^{(N_m)}])^{\mathrm{T}}$. Matrix $\mathbf{W}_m$ rotates each of the measurement mode matrices $\mathbf{V}_m^{(n)}$ for each cluster computed by linear PCA such that the representation $\mathbf{U}_m$ is the same across the clusters.

## 4.3 Multilinear ICA (MICA)

We have discussed MPCA, which computes *accurate* representations of observational data by optimizing the reduced-dimensional data approximation error in a least-squares sense. This is done in a multifactor manner, by accurately representing the underlying causal factor subspaces. However, MPCA ignores important aspects of non-Gaussian data, such as clustering and independence. Therefore, we will next develop Multilinear ICA, which employs higher-order statistics to derive *meaningful* representations for each of the underlying causal factors. To compute such representations we will propose a novel, *Multilinear ICA (MICA) algorithm*. Our nonlinear, tensor algebraic algorithm should not be confused with existing tensor algorithms for computing the conventional, linear ICA [de Lathauwer 1997].

Analogously to (4.8) and (4.14), multilinear ICA is obtained by approximating the data tensor $\boldsymbol{\mathcal{D}}$ as the mode-$m$ product of $M$ mode matrices $\mathbf{C}_m$ and a core tensor $\boldsymbol{\mathcal{S}}$, as follows:

$$\tilde{\boldsymbol{\mathcal{D}}} = \boldsymbol{\mathcal{S}} \times_1 \mathbf{C}_1 \times_2 \mathbf{C}_2 \dots \times_m \mathbf{C}_m \dots \times_M \mathbf{C}_M. \tag{4.17}$$

Analogously to (4.9), (4.10), and (4.11), we have the extended core

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{S}} \times_1 \mathbf{C}_1, \tag{4.18}$$

the multilinear approximation

$$\tilde{\boldsymbol{\mathcal{D}}} = \boldsymbol{\mathcal{M}} \times_2 \mathbf{C}_2 \dots \times_m \mathbf{C}_m \dots \times_M \mathbf{C}_M, \tag{4.19}$$

and the multilinear expression for an observation

$$\tilde{\mathbf{d}} = \boldsymbol{\mathcal{M}} \times_2 \mathbf{r}_2^{\mathrm{T}} \dots \times_m \mathbf{r}_m^{\mathrm{T}} \dots \times_M \mathbf{r}_M^{\mathrm{T}}. \tag{4.20}$$

As in ICA, there are two approaches for multilinear independent components analysis (MICA). The first results in a factorial code, where each set of coefficients is statistically independent, while the second finds a set of independent bases.

In **Approach 1**, transposing the matrixized data tensor $\boldsymbol{\mathcal{D}}$ in mode $m$ and computing the ICA as in (4.2)–(4.4), we obtain

$$\begin{aligned} \mathbf{D}_{[m]}^{\mathrm{T}} &\simeq \mathbf{V}_m \mathbf{\Sigma}_m \mathbf{U}_m^{\mathrm{T}} && (4.21)\\ &= \left(\mathbf{V}_m \mathbf{\Sigma}_m \mathbf{W}_m^{-1}\right) \left(\mathbf{W}_m \mathbf{U}_m^{\mathrm{T}}\right) && (4.22)\\ &= \mathbf{K}_m^{\mathrm{T}} \mathbf{C}_m^{\mathrm{T}}, && (4.23) \end{aligned}$$

where $\mathbf{U}_m$, $\mathbf{V}_m$, and $\mathbf{\Sigma}_m$ are rank-reduced, and the mode matrices are given by

$$\mathbf{C}_m = \mathbf{U}_m \mathbf{W}_m^{\mathrm{T}}. \tag{4.24}$$

The columns associated with each of the mode matrices, $\mathbf{C}_m$ are statistically independent. We can derive the relationship between MICA and MPCA (4.14) in the context of this approach as follows:

$$\begin{aligned}
\tilde{\mathcal{D}} &= \mathcal{Z} \times_1 \mathbf{U}_1 \ldots \times_M \mathbf{U}_M \\
&= \mathcal{Z} \times_1 \mathbf{U}_1 \mathbf{W}_1^{\mathrm{T}} \mathbf{W}_1^{-T} \ldots \times_M \mathbf{U}_M \mathbf{W}_M^{\mathrm{T}} \mathbf{W}_M^{-T} \\
&= \mathcal{Z} \times_1 \mathbf{C}_1 \mathbf{W}_1^{-T} \ldots \times_M \mathbf{C}_M \mathbf{W}_M^{-T} \\
&= (\mathcal{Z} \times_1 \mathbf{W}_1^{-T} \ldots \times_M \mathbf{W}_M^{-T}) \times_1 \mathbf{C}_1 \ldots \times_M \mathbf{C}_M \\
&= \mathcal{S} \times_1 \mathbf{C}_1 \ldots \times_M \mathbf{C}_M,
\end{aligned}$$

where the core tensor $\mathcal{S} = \mathcal{Z} \times_1 \mathbf{W}_1^{-T} \ldots \times_M \mathbf{W}_M^{-T}$.

Alternatively, in **Approach 2**, matrixizing the data tensor $\mathcal{D}$ in the $m^{\text{th}}$ mode and computing the ICA as in (4.5)–(4.7), we obtain:

$$\begin{aligned}
\mathbf{D}_{[m]} &\simeq \mathbf{U}_m \mathbf{\Sigma}_m \mathbf{V}_m^{\mathrm{T}} && (4.25)\\
&= \left(\mathbf{U}_m \mathbf{W}_m^{-1}\right) \left(\mathbf{W}_m \mathbf{\Sigma}_m \mathbf{V}_m^{\mathrm{T}}\right) && (4.26)\\
&= \mathbf{C}_m \mathbf{K}_m, && (4.27)
\end{aligned}$$

where $\mathbf{U}_m$, $\mathbf{V}_m$, and $\mathbf{\Sigma}_m$ are rank-reduced, and the mode matrices are given by

$$\mathbf{C}_m = \mathbf{U}_m \mathbf{W}_m^{-1}. \tag{4.28}$$

Note that the $\mathbf{W}$ matrices in (4.26) and (4.28) differ from the analogous $\mathbf{W}$ matrices in (4.22) and (4.24); hence, the $\mathbf{K}$ and $\mathbf{C}$ matrices also differ between the two methods. We can derive the relationship between MICA and MPCA (4.14) in the context of the second approach as follows:

$$\begin{aligned}
\tilde{\mathcal{D}} &= \mathcal{Z} \times_1 \mathbf{U}_1 \ldots \times_M \mathbf{U}_M \\
&= \mathcal{Z} \times_1 \mathbf{U}_1 \mathbf{W}_1^{-1} \mathbf{W}_1 \ldots \times_M \mathbf{U}_M \mathbf{W}_M^{-1} \mathbf{W}_M \\
&= \mathcal{Z} \times_1 \mathbf{C}_1 \mathbf{W}_1 \ldots \times_M \mathbf{C}_M \mathbf{W}_M \\
&= (\mathcal{Z} \times_1 \mathbf{W}_1 \ldots \times_M \mathbf{W}_M) \times_1 \mathbf{C}_1 \ldots \times_M \mathbf{C}_M \\
&= \mathcal{S} \times_1 \mathbf{C}_1 \ldots \times_M \mathbf{C}_M,
\end{aligned}$$

where the core tensor $\mathcal{S} = \mathcal{Z} \times_1 \mathbf{W}_1 \ldots \times_M \mathbf{W}_M$.

Algorithm 4.3.1 is an efficient *Multilinear ICA (MICA) algorithm*, which is the analog of our efficient MPCA algorithm (Algorithm 4.2.1). Step 1 of Algorithm 4.3.1 has a structure that is similar to the $M$-mode SVD algorithm (Algorithm 3.2.2), but applies one of the above to ICA approaches. Anal-

**Algorithm 4.3.1** Multilinear ICA (MICA) algorithm.

```
Input the data tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$, where mode $m = 1$ is the measurement mode, and the desired ranks $\tilde{R}_2, \ldots, \tilde{R}_M$.

  1. For $m := 2, \ldots, M$,
         Compute the causal factor mode matrix $\mathbf{C}_m \in \mathbb{R}^{I_m \times \tilde{R}_m}$ in (4.17) using Approach 1 or
         Approach 2, truncating the initial SVD to dimensionality $\tilde{R}_m$.
  2. Set $\mathcal{M} := \mathcal{D} \times_2 \mathbf{C}_2^+ \ldots \times_m \mathbf{C}_m^+ \ldots \times_M \mathbf{C}_M^+$.
  3. Local optimization via alternating least squares:
     Iterate for $n := 1, \ldots, N$
         For $m := 2, \ldots, M$,
             Set $\mathcal{X} := \mathcal{D} \times_2 \mathbf{C}_2^+ \ldots \times_{m-1} \mathbf{C}_{m-1}^+ \times_{m+1} \mathbf{C}_{m+1}^+ \ldots \times_M \mathbf{C}_M^+$.
             Compute $\mathbf{C}_m$ from $\mathbf{X}_{[m]}$ either by Approach 1 (4.24) or Approach 2 (4.28).
         Set $\mathcal{M} := \mathcal{X} \times_M \mathbf{C}_M^+$.
     until convergence.[a]

Output the converged extended core tensor $\mathcal{M} \in \mathbb{R}^{I_1 \times \tilde{R}_2 \times \ldots \times \tilde{R}_M}$ and causal factor mode matrices $\mathbf{C}_2, \ldots, \mathbf{C}_M$.
```

[a]See Footnote *a* in Algorithm 3.2.1.

ogously to the case of MPCA, optimal dimensionality reduction in MICA is achieved in Step 3 of the algorithm by optimizing mode per mode using a straightforward variant of the $(M-1)$-mode orthogonal iteration, with an additional step that computes the transformation matrix $\mathbf{W}$. Thus, we compute the independent components for each mode in an iterative manner using alternating least squares, by holding fixed all mode components except one and solving for the remaining mode. Note that we can compute the approximation of $\mathcal{D}$ from the converged reduced extended core tensor $\mathcal{M}$ and reduced causal factor mode matrices $\mathbf{C}_m$ as

$$\mathcal{D} \simeq \tilde{\mathcal{D}} = \mathcal{M} \times_2 \mathbf{C}_2 \ldots \times_M \mathbf{C}_M. \tag{4.29}$$

As in the MPCA case, this algorithm finds only a local minimum of the generally non-convex error function.

To see the difference between the subspaces spanned by ICA and MICA, consider the representation computed for the causal factors of the process used to generate the data shown in Figure 4.3. MICA computes an overcomplete set of basis vectors that extract all the arms in the data while ICA extracts a combination of two of the arms. To see the difference between MPCA and MICA, consider the data shown in Figure 4.4. MPCA yields a subspace for a causal factor that is spanned by the directions of greatest variation, while MICA yields a subspace that is spanned by the directions that define indepen-

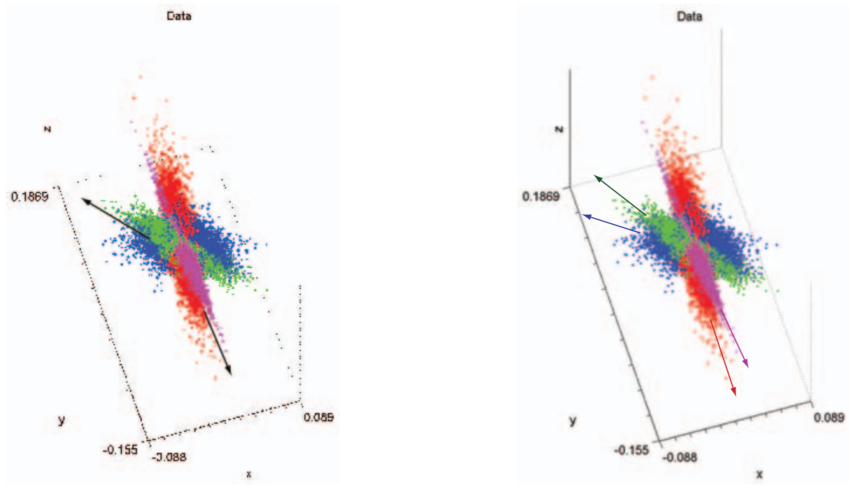


*Figure 4.3: This data set contains four different arms, but ICA is able to extract only a combination of two of the arms while MICA extracts all four.*

dent components. These two sets of directions may yield different subspaces.

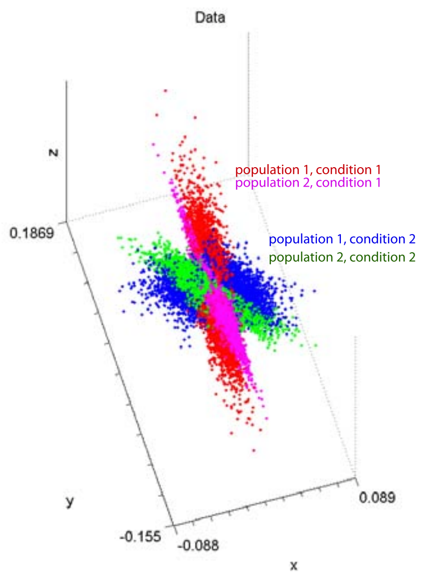
## 4.4 Kernel MPCA/MICA, Multifactor LLE/Isomap/LE, and Other Generalizations

The effectiveness of conventional PCA and ICA is limited by their global linearity and their susceptibility to non-Gaussian noise. Our approach to mitigating the linearity limitation has been to generalize these methods in a natural way, resulting in globally nonlinear, multifactor models and methods that have a multilinear manifold structure. We will now discuss two alternative classes of nonlinear models, and then briefly discuss the issue of noise and outliers.

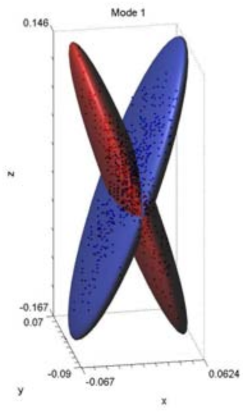
One approach is to handle data complexity via data clustering and the combination (or mixing) of linear PCA or ICA models fitted to each local data cluster [Tipping and Bishop 1999; Lee et al. 2000]. An important distinction is that mixture-of-PCA/ICA models are additive combinations of locally Gaussian models, whereas our multilinear approach prescribes a multiplicative combination of globally Gaussian models. Moreover, a mixture-of-PCA/ICA models yields a different set of basis vectors for each local cluster. In our applications, we seek a single parsimonious set of basis vectors that globally represent the entire dataset. That said, there seems to be no reason why one could not define mixture-of-MPCA/MICA models and methods that would generalize the existing mixture-of-PCA/ICA models. However, this is beyond the scope of the thesis.

An alternative approach is to apply linear models to nonlinear problems through the "kernel trick", specifically the kernel PCA [Schölkoph et al. 1998] and kernel ICA [Yang et al. 2005] techniques.[4]

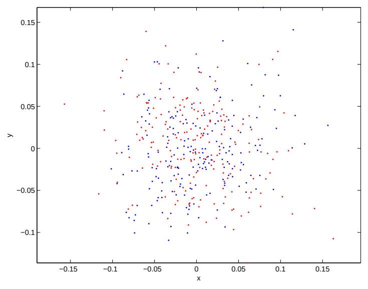
[4]The so-called "kernel trick" maps the original non-linear measurements into a higher-dimensional space, where a linear classifier is subsequently used; this makes a linear classification in the new space equivalent to non-linear classification in the original space. This is done using Mercer's theorem, which states that any continuous, symmetric, positive semi-definite kernel $K(\mathbf{u}, \mathbf{v})$ can be expressed as an inner product in a high-dimensional space. Wherever an inner product between two vectors is



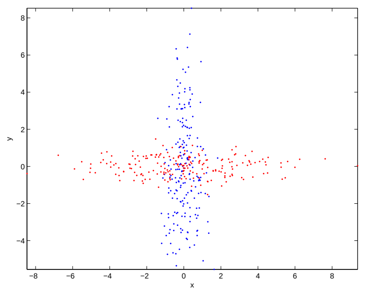
(a) Observed Data



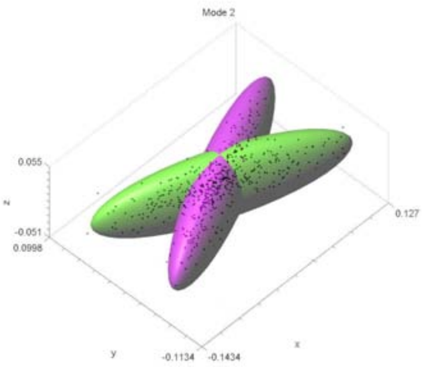
(b) Factor 1



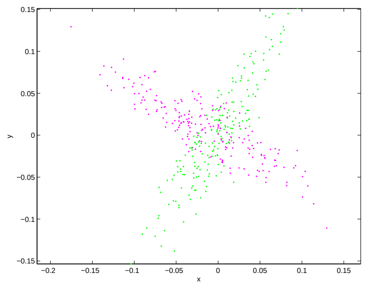
(c) MPCA - Factor 1 Subspace



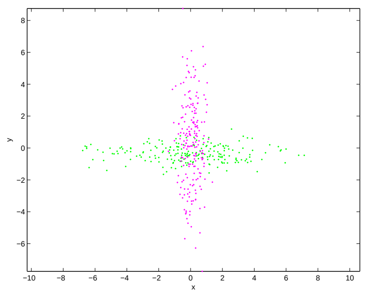
(d) MICA - Factor 1 Subspace



(e) Factor 2



(f) MPCA - Factor 2 Subspace



(g) MICA - Factor 2 Subspace

*Figure 4.4: A Comparison of the causal factor extraction/representation by MPCA and MICA. The observed data (a) were generated by the interaction of two unmeasurable causal factors. The distribution of Factor 1 (b) comprises two populations; that of Factor 2 (e) comprises two conditions. Without dimensionality reduction, both methods extract the factors in (b), (e). However,* analysis *implies dimensionality reduction, and MPCA and MICA compute the subspaces in (c),(f) and (d),(g). The subspaces for Factor 1 computed by MPCA (c) and MICA (d) are mutually orthogonal, but the subspaces for Factor 2 computed by MPCA (f) and MICA (g) are identical as the directions of greatest variation and the independent component directions span the same subspace in this case.*

| | |
|---|---|
| Linear kernel: | $K(\mathbf{u},\mathbf{v}) = \mathbf{u}^{\mathrm{T}}\mathbf{v} = \mathbf{u}\cdot\mathbf{v}$ |
| Polynomial kernel of degree $d$: | $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}^{\mathrm{T}}\mathbf{v})^d$ |
| Polynomial kernel up to degree $d$: | $K(\mathbf{u},\mathbf{v}) = (\mathbf{u}^{\mathrm{T}}\mathbf{v}+1)^d$ |
| Sigmoidal kernel: | $K(\mathbf{u},\mathbf{v}) = \tanh(\alpha\,\mathbf{u}^{\mathrm{T}}\mathbf{v}+\beta)$ |
| Gaussian (radial basis function (RBF)) kernel: | $K(\mathbf{u},\mathbf{v}) = \exp\left(-\frac{\lVert\mathbf{u}-\mathbf{v}\rVert^2}{2\sigma^2}\right)$ |

*Table 4.1: Common kernel functions. Kernel functions are symmetric, positive semi-definite functions (corresponding to symmetric, positive semi-definite Gram matrices). The linear kernel does not modify or warp the feature space.*

Kernel PCA/ICA are nonlinear versions of their conventional linear counterparts, but of course they are not multifactor models.

The kernel trick can also be applied to our multilinear, multifactor PCA/ICA models to further nonlinearize them, thus enabling them to deal with arbitrarily nonlinear data. To accomplish this, recall that the computation (4.13) of the mode-$m$ covariance matrix $\mathbf{D}_{[m]}\mathbf{D}_{[m]}^{\mathrm{T}}$ involves inner products $\mathbf{d}^{\mathrm{T}}_{i_2\ldots i_{m-1}\,j\,i_{m+1}\ldots i_M}\mathbf{d}_{i_2\ldots i_{m-1}\,k\,i_{m+1}\ldots i_M}$ between pairs of images in the image data tensor $\boldsymbol{\mathcal{D}}$ associated with causal factor mode $m$, for $m = 2,\ldots,M$. We replace the inner products with a generalized distance measure between images, $K(\mathbf{d}_{i_2\ldots i_{m-1}\,j\,i_{m+1}\ldots i_M}, \mathbf{d}_{i_2\ldots i_{m-1}\,k\,i_{m+1}\ldots i_M})$, where $K(\cdot,\cdot)$ is a suitable kernel function (Table 4.1), which corresponds to an inner product in some expanded feature space. This generalization naturally leads us to a *Kernel Multilinear PCA (K-MPCA) Algorithm*, where (4.13) in Algorithm 4.2.1 is replaced by

$$\left[\mathbf{D}_{[m]}\mathbf{D}_{[m]}^{\mathrm{T}}\right]_{jk} := \sum_{i_2=1}^{I_2}\cdots\sum_{i_{m-1}=1}^{I_{m-1}}\sum_{i_{m+1}=1}^{I_{m+1}}\cdots\sum_{i_M=1}^{I_M} K(\mathbf{d}_{i_2\ldots i_{m-1}\,j\,i_{m+1}\ldots i_M}, \mathbf{d}_{i_2\ldots i_{m-1}\,k\,i_{m+1}\ldots i_M}). \tag{4.30}$$

Similarly, a *Kernel Multilinear ICA (K-MICA) Algorithm* results from making the same generalization in the MICA algorithm (Algorithm 4.3.1). Algorithm 4.4.1 specifies both K-MPCA and K-MICA.

The relationship between the aforementioned models is shown in Figure 1.1. Furthermore, the kernel $M$-mode SVD algorithm straightforwardly leads to multifactor generalizations of recent manifold learning methods, including Locally Linear Embedding (LLE) [Roweis and Saul 2000], Isomap (IM) [Tenenbaum et al. 2000], and Laplacian Eigenmaps (LE) [Belkin and Niyogi 2003]. This is because these methods are based on non-Euclidean distance measures between pairs of training data samples (e.g., images) with respect to the manifold defined by the data, which can be regarded as particular

used, it is replaced with a kernel of the vectors. Thus, a linear algorithm is easily transformed into a nonlinear algorithm. This trick has been applied to numerous algorithms in machine learning and statistics.

**Algorithm 4.4.1** Kernel Multilinear PCA/ICA (K-MPCA/MICA) algorithm.

```
Input the data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$, where mode $m = 1$ is the measurement mode, and the desired ranks $\tilde{R}_2, \ldots, \tilde{R}_M$.

1. For $m := 2, \ldots, M$,

      Compute the elements of the mode-$m$ covariance matrix, for $j, k := 1, \ldots, I_m$, as follows:

      $\left[\mathbf{D}_{[m]}\mathbf{D}^{\mathrm{T}}_{[m]}\right]_{jk} := \sum_{i_2=1}^{I_2} \cdots \sum_{i_{m-1}=1}^{I_{m-1}} \sum_{i_{m+1}=1}^{I_{m+1}} \cdots \sum_{i_M=1}^{I_M} K(\mathbf{d}_{i_2 \ldots i_{m-1}\, j\, i_{m+1} \ldots i_M}, \mathbf{d}_{i_2 \ldots i_{m-1}\, k\, i_{m+1} \ldots i_M}).$

      For K-MPCA:  Set $\mathbf{Y}_m$ to the left matrix of the SVD of $\mathbf{D}_{[m]}\mathbf{D}^{\mathrm{T}}_{[m]} = \mathbf{Y}_m \boldsymbol{\Sigma}^2 \mathbf{Y}^{\mathrm{T}}_m$
                   Truncate to $\tilde{R}_m$ columns $\mathbf{Y}_m \in \mathbb{R}^{I_m \times \tilde{R}_m}$.
      For K-MICA:  Compute $\mathbf{Y}_m := \mathbf{C}_m \in \mathbb{R}^{I_m \times \tilde{R}_m}$ using Approach 1 (4.24) or
                   Approach 2 (4.28), and truncating the initial SVD to dimensionality $\tilde{R}_m$.

2. Set $\boldsymbol{\mathcal{T}} := \boldsymbol{\mathcal{D}} \times_2 \mathbf{Y}^+_2 \ldots \times_m \mathbf{Y}^+_m \ldots \times_M \mathbf{Y}^+_M$.

3. Local optimization via alternating least squares:
   Iterate for $n := 1, \ldots, N$

      For $m := 2, \ldots, M$,
         Set $\boldsymbol{\mathcal{X}} := \boldsymbol{\mathcal{D}} \times_2 \mathbf{Y}^+_2 \ldots \times_{m-1} \mathbf{Y}^+_{m-1} \times_{m+1} \mathbf{Y}^+_{m+1} \ldots \times_M \mathbf{Y}^+_M$.
         Set $\mathbf{Y}_m$ to the $\tilde{R}_m$ leading left-singular vectors of the SVD of $\mathbf{X}_{[m]}$.
      Set $\boldsymbol{\mathcal{T}} := \boldsymbol{\mathcal{X}} \times_M \mathbf{Y}^+_M$.

   until convergence.

Output the converged extended core tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \tilde{R}_2 \times \ldots \times \tilde{R}_M}$ and causal factor mode matrices $\mathbf{Y}_2, \ldots, \mathbf{Y}_M$.
```

types of kernels $K(\cdot,\cdot)$ in (4.30) [Ham et al. 2004]; i.e.,

$$K(\mathbf{u},\mathbf{v}) = \begin{cases} K_{\text{LLE}}(\mathbf{u},\mathbf{v}) & \text{Locally Linear Embedding,} \\ K_{\text{IM}}(\mathbf{u},\mathbf{v}) & \text{Isomap,} \\ K_{\text{LE}}(\mathbf{u},\mathbf{v}) & \text{Laplacian Eigenmap.} \end{cases}$$

Unlike conventional kernels, these kernels are not explicit functions; they must be computed algorithmically. The first step involves constructing a graph, where each training image is connected by edges to its near neighbors in the pixel space, thus describing the local neighborhood relationships between the training images, and inter-image distances are associated with the edges of the graph. The kernel $K_{\text{LLE}}$ is then based on a specially constructed graph operator. Kernel $K_{\text{IM}}$ is based on the geodesic distances on the manifold between pairs of images, computed using Dijkstra's shortest path algorithm. Kernel $K_{\text{LE}}$ is related to the commute time of diffusion on the graph, which is related to the graph Laplacian. Ham et al. [2004] provide the details for each of these cases. Note that these kernels matrixize the curved input manifold situated in the Euclidean space to a hyperplanar manifold in a "feature space". Their values depend not only on the particular pair of training images that are input as arguments, but also on all the other images in the training set.

The further development and application of our proposed multifactor generalizations of the aforementioned manifold mapping methods is beyond the scope of this thesis. However, we can expect them to be effective so long as measurement data are available that densely sample the curved manifolds of interest. Given arbitrarily dense training data, these local methods can, in principle, map out more or less arbitrarily curved manifolds. Unfortunately, the applications that we will pursue in the remainder of this thesis are not blessed by observational datasets that densely sample the manifolds of interest. As they are global models, our multilinear models can be fitted to relatively sparse measurement data, which is another one of their strengths.

Our nonlinear PCA and ICA models are susceptible to the same problems with noise and outliers as are the conventional linear PCA and ICA models. Both classes of models are robust against Gaussian noise, but not against other noise distributions. Fortunately, since our multilinear models are natural generalizations of their linear counterparts, noisy data can be addressed through the judical application of the same techniques for dealing with noise and outliers that have been developed for conventional PCA and ICA. In particular, the techniques known as bootstrap, jackknife, cross validation, etc., as well as techniques for robust regression and outlier detection are applicable when employing our multilinear models. However, we will not apply these techniques in this thesis.

## Preview and Notation:

In the remainder of the thesis, we will proceed to apply our tensor algebraic framework and the multilinear generalizations of PCA and ICA to computer graphics (TensorTextures), computer vision (TensorFaces), and machine learning or pattern recognition (face recognition and human motion signatures).

In our ensuing applications, rather than generically numbering tensor modes, as we have done in this and the previous chapter, we will name them. By convention, the first mode, mode 1, will represent the measurement mode—depending on the application, texels/pixels or joint angles, which will be denoted by *lower case* 'x' or 'j' subscripts, respectively. The remaining modes will represent the causal factors. Depending on the application, these will include people, views, illuminations, expressions, and/or actions, which will be denoted by *upper case* 'P', 'V', 'L', 'E', 'A' subscripts, respectively.

Before proceeding, however, the reader may wish to peruse Appendix A, which discusses the important issue of data representation in the context of multilinear methods.

5

# Multilinear Image Synthesis: TensorTextures

**Contents**

We first apply our multilinear framework to computer graphics, which is concerned with the forward problem of image synthesis.

The appearance of rendered surfaces in computer graphics is determined by a complex interaction of multiple causal factors related to scene geometry, illumination, and imaging. The *bidirectional texture function* or BTF [Dana et al. 1999] captures the appearance of extended, textured surfaces, including spatially varying reflectance, surface mesostructure (i.e., 3D texture caused by local height variation over rough surfaces), subsurface scattering, and other phenomena over a finite region of the surface. The BTF is a function of six variables:

$$\mathbf{f}(x, y, \theta_v, \phi_v, \theta_i, \phi_i), \tag{5.1}$$

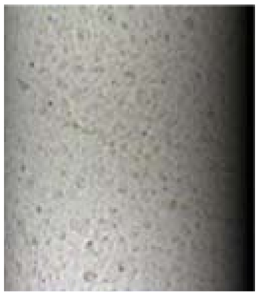
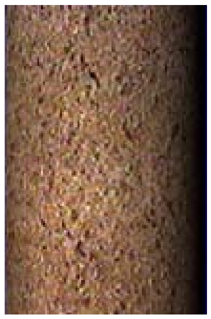
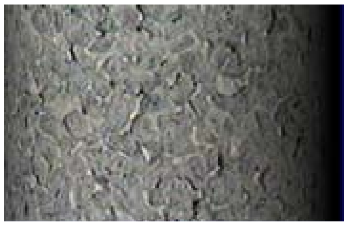
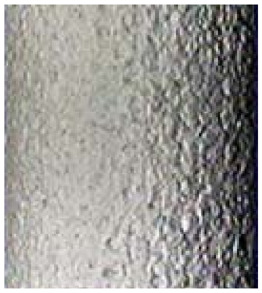
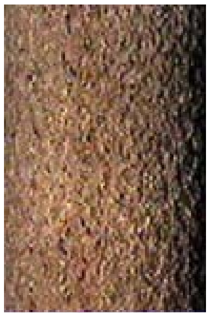
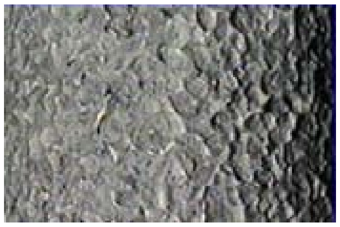
| | Concrete | Pebbles | Plaster |
|---|---|---|---|
| Standard Texture Mapping |  |  |  |
| BTF Texture Mapping |  |  |  |

*Figure 5.1: Standard texture mapping versus texture mapping using bidirectional texture function. Adopted from [Dana et al. 1999].*

where $(x, y)$ are surface parametric (texel) coordinates and where $(\theta_v, \phi_v)$ is the view direction and $(\theta_i, \phi_i)$ is the illumination direction (a.k.a. the photometric angles). Several BTF acquisition devices have been described in the literature (see, e.g., [Dana et al. 1999]). In essence, these devices sample the BTF by acquiring images of a surface of interest from several different views under several different illuminations. Given only sparsely sampled BTF data, the problem of rendering the appearance of a textured surface viewed from an arbitrary vantage point under arbitrary illumination is a problem in image-based rendering (IBR). Figure 5.1 illustrates the difference between standard textures and more realistic BTF textures showing relief effects.

*TensorTextures* is a new image-based technique for rendering textured surfaces from sparsely sampled BTF data, and it will serve the first instantiation of our multilinear framework. More specifically, from an ensemble of sample images of a textured surface, the offline analysis stage of our TensorTextures algorithm learns a generative model that accurately approximates the BTF. Then, in the online synthesis stage, the trained generative model serves in rendering the appearance of the textured surface under arbitrary view and illumination conditions. Figure 5.2 shows an example TensorTexture. Although the coins in the treasure chest appear to have considerable 3D relief as we vary the view and illumination directions, this is in fact a TensorTexture mapped onto a perfectly planar surface. The TensorTextures model has learned a compact representation of the variation in appearance of the surface under changes in view and illumination, including complex details due to surface mesostructure,

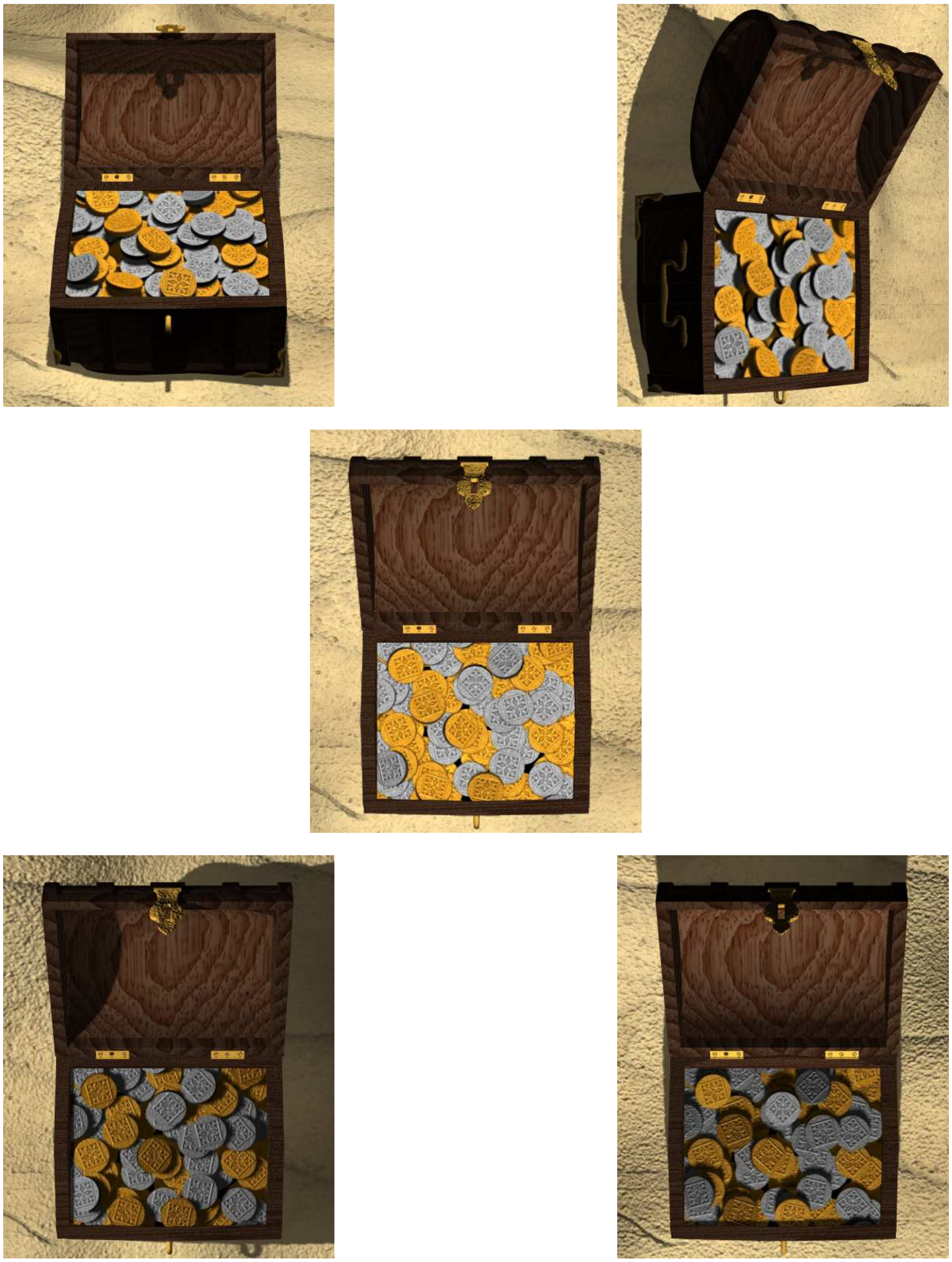


*Figure 5.2: The chest contains a TensorTexture mapped onto a planar surface, which appears to have considerable 3D relief when viewed from various directions (top and center) and under various illuminations (center and bottom).*

such as self-occlusion, interreflection, and self-shadowing. Figure 1.3 shows a system diagram of our TensorTextures analysis and rendering technique.

A major technical advantage of our multilinear framework for image-based rendering is that the underlying tensor formulation can disentangle and explicitly represent each of the multiple causal factors inherent to image formation. This stands in contrast to linear (matrix) PCA, which has so far been the standard BTF representation/compression method [Sattler et al. 2003] and is, in fact, subsumed by our more general multilinear framework. A major limitation of PCA is that it captures the overall variation in the image ensemble without explicitly distinguishing what proportion is attributable to each of the relevant causal factors—illumination change, view change, etc. Our method prescribes a more sophisticated tensor decomposition that further analyzes this overall variation into individually encoded causal factors using a novel set of basis functions. Harnessing the power of multilinear algebra, our approach contributes a novel, explicitly multimodal model with which to tackle the BTF modeling/rendering problem.

Unlike the approach of [Malzbender et al. 2001] and similar algorithms, ours estimates the complete BTF, including variations not only in illumination, but also in view. Unlike the BTF synthesis method of [Tong et al. 2002] and similar algorithms, our technique is purely image based—it avoids the nontrivial problem of 3D geometric microstructure estimation. Unlike all previous methods, the one that we develop in this thesis is a *nonlinear* BTF model.

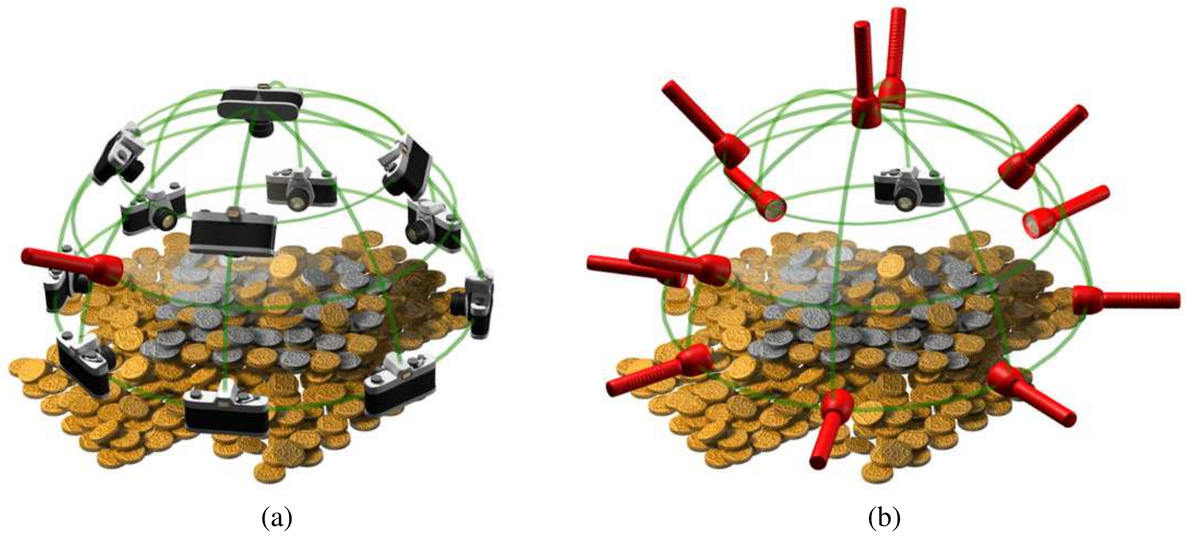
## 5.1 Multilinear BTF Analysis

Given an ensemble of images of a textured surface, we define an image data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_\text{x} \times I_\text{L} \times I_\text{V}}$, where $I_\text{V}$ and $I_\text{L}$, corresponding to the causal factor modes, are, respectively, the number of different view and illumination conditions associated with the image acquisition process, and $I_\text{x}$, corresponding to the measurement mode, is the number of texels in each texture image. As a concrete example, which we will use for illustrative purposes, consider the synthetic scene of scattered coins shown in Figure 5.3. A total of 777 RGB images of the scene are acquired from $I_\text{V} = 37$ different view directions over the viewing hemisphere (Figure 5.3(a)), each of which is illuminated by a light source oriented in $I_\text{L} = 21$ different directions over the illumination hemisphere (Figure 5.3(b)). The size of each RGB image is $I_\text{x} = 240 \times 320 \times 3 = 230,400$ texels.

In order to analyze the texture, images viewed from different angles must be brought into correspondence. The most general assumption we can make is that the texture is planar.[1] To align texture features across all images, we rectify images acquired from oblique angles to bring them into correspondence with the fronto-parallel image, thus undoing the perspective distortion (Figure 5.4(b)). We apply a planar homography [Hartley and Zisserman 2000; Zhang 1999]

$$\mathbf{x}'_i = \mathbf{H}\mathbf{x}, \tag{5.2}$$

[1] Clearly the textures considered are not planar, but our synthesis results do not suffer much from this weak assumption. A more accurate model might allow one to do better dimensionality reduction in the view mode.



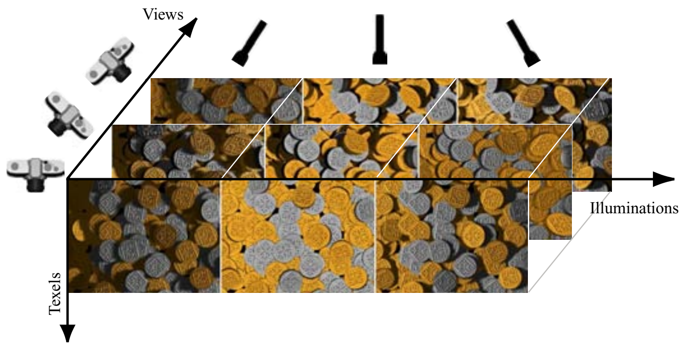


*Figure 5.3: Image acquisition and representation. Images are acquired from several different view directions over the viewing hemisphere (a) and, for each view, under several different illumination conditions over the illumination hemisphere (b). The ensemble of acquired images is organized in a third-order tensor (c) with view, illumination, and texel modes. Although the contents of the texel mode are vectors of RGB texel values, for clarity they are displayed as 2D images in this and subsequent figures.*

a 2D to 2D projective transformation that warps points from an image plane $\mathbf{x}_i$ to another image plane $\mathbf{x}'_i$. $H$ is a $3\times 3$ homogeneous matrix that applies a rotation, scale, and a translation. It contains 8 degrees of freedom; hence, $4$ pairs of $2D$ fiducials are sufficient to determine it.[2] The four fiducial pairs can be extracted from the actual texture or from a reference checkerboard that has been photographed from the same vantage points as the textures (Figure 5.4(a)). The Direct Linear Transformation algorithm is a least squares method for estimating $\mathbf{H}$ [Hartley and Zisserman 2000].

We organize the rectified images as a 3rd-order tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{230400\times 21\times 37}$, a portion of which is shown in Figure 5.3(c). We can apply the MPCA algorithm (Algorithm 4.2.1) to decompose this tensor as follows:

$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{T}} \times_{\text{L}} \mathbf{U}_{\text{L}} \times_{\text{V}} \mathbf{U}_{\text{V}}, \tag{5.3}$$

[2]Three collinear points in either image are a degenerate configuration preventing a unique solution.

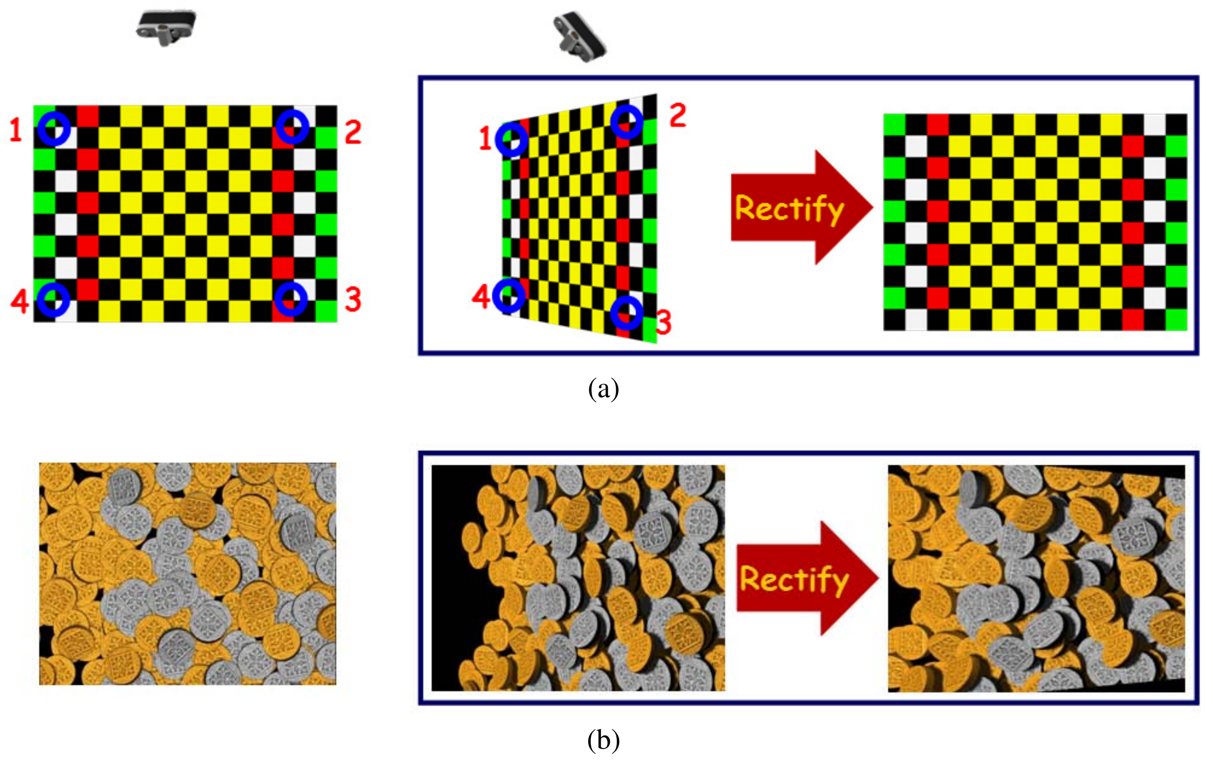


*Figure 5.4: (a) The rectifying homography is computed using at least 4 pairs of fiducials from a checkerboard texture. (b) The texture is unwarped and brought into correspondence to the frontal image by applying the computed homography.*

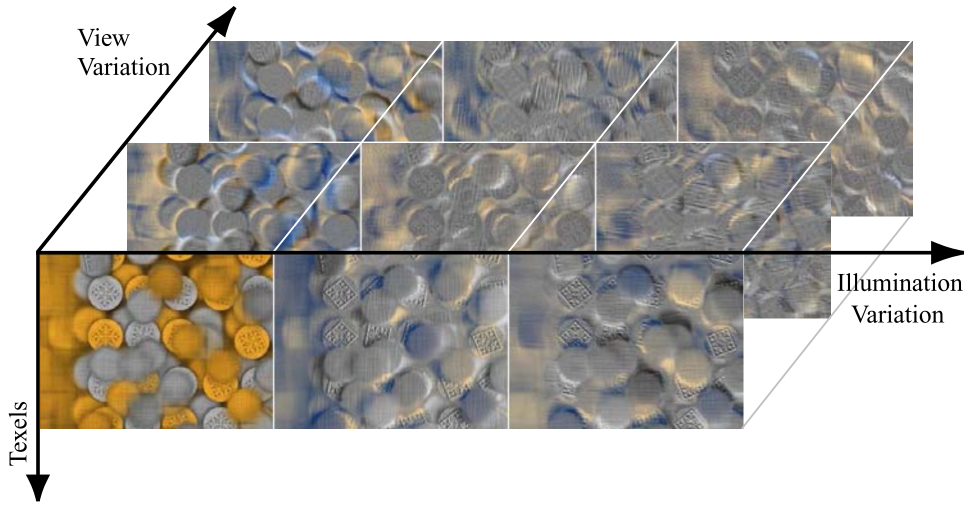


*Figure 5.5: A partial visualization of the* $\mathcal{T} \in \mathbb{R}^{230400 \times 21 \times 37}$ *TensorTextures bases of the coins image ensemble.*

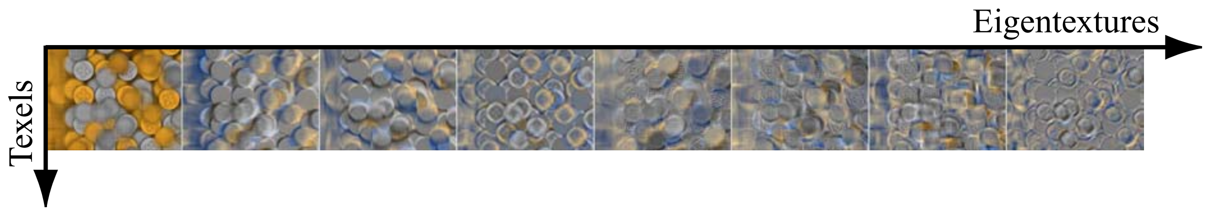


*Figure 5.6:* $\mathbf{U}_{\mathrm{x}}$ *contains the PCA eigenvectors, which are the principal axes of variation across all images.*

the product of two orthonormal mode matrices associated with the causal factors and an extended core tensor $\boldsymbol{\mathcal{T}}$, which is the TensorTextures basis that governs the interaction between the mode matrices. The rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{V}} \in \mathbb{R}^{37\times37}$ are the different illumination-invariant view representations (the subscript "V" denotes the *view* mode). The rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{L}} \in \mathbb{R}^{21\times21}$ are the different view-invariant illumination representations (the subscript "L" denotes the *illumination* mode). An image $\mathbf{d}_{l,v}$ in the ensemble $\boldsymbol{\mathcal{D}}$ is represented as

$$\mathbf{d}_{l,v} = \boldsymbol{\mathcal{T}} \times_{\mathrm{L}} \mathbf{l}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}}, \tag{5.4}$$

where $\mathbf{l}$ and $\mathbf{v}$ are light and view representations, which are rows in $\mathbf{U}_{\mathrm{L}}, \mathbf{U}_{\mathrm{V}}$.

## 5.2 Computing TensorTextures

TensorTextures $\boldsymbol{\mathcal{T}}$ models how the appearance of a textured surface varies with view and illumination. The TensorTextures representation, which we illustrate in Figure 5.5, is the product

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{Z}} \times_{\mathrm{x}} \mathbf{U}_{\mathrm{x}} \tag{5.5}$$
$$= \boldsymbol{\mathcal{D}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}}^{\mathrm{T}}. \tag{5.6}$$

From a computational standpoint, equation (5.6) is preferable to (5.5) in practice, since it prescribes computing the relatively small matrices $\mathbf{U}_{\mathrm{V}}$ and $\mathbf{U}_{\mathrm{L}}$ using the MPCA algorithm rather than the generally large measurement mode matrix $\mathbf{U}_{\mathrm{x}} \in \mathbb{R}^{230400\times777}$ and core tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{777\times21\times37}$.

Figure 5.6 shows the column vectors of the measurement mode matrix $\mathbf{U}_{\mathrm{x}}$, which span the texture space (the subscript "x" denotes the *texels* mode). These are, in fact, the PCA eigenvectors (i.e., "eigenimages" or "eigentextures"). Our multilinear TensorTextures model subsumes the linear eigentextures (PCA) model, as we showed in Section 4.2. The TensorTextures representation $\boldsymbol{\mathcal{T}}$ in (5.5) transforms the eigentextures into a tensorial representation of the variation of the causal factors (view and illumination). It characterizes how the view parameters and illumination parameters interact and multiplicatively modulate the appearance of a surface under variation in view direction $(\theta_v, \phi_v)$, illumination direction $(\theta_i, \phi_i)$, and position $(x, y)$ over the surface.

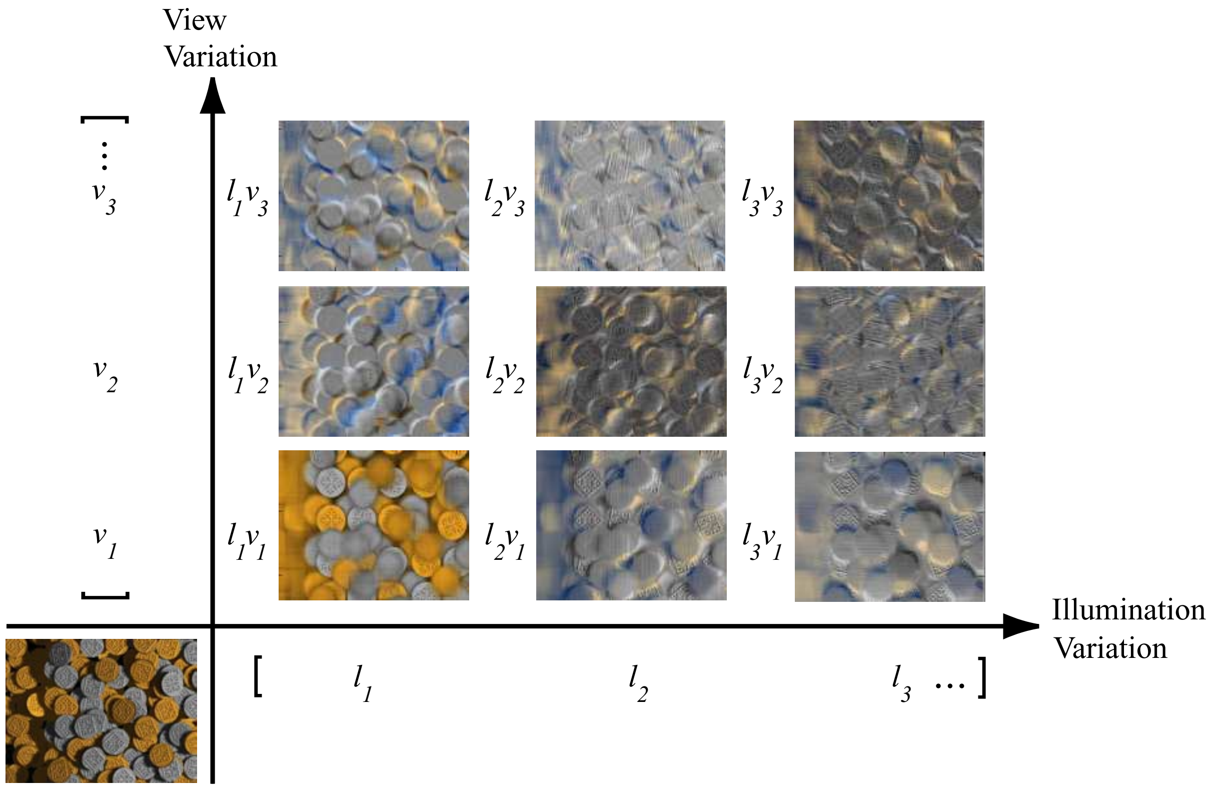


*Figure 5.7: The lower left image is rendered by multiplicatively modulating each of the TensorTextures basis vectors with the coefficients in the view coefficient vector* $\mathbf{v}$ *and illumination coefficient vector* $\mathbf{l}$.

## 5.3 Texture Representation and Dimensionality Reduction

The dimensionality-reduced PCA image representation is a coefficient vector of length $\tilde{R}_{\text{x}} \leq I_{\text{x}}$, whereas an image in our dimensionality-reduced TensorTextures representation comprises a coefficient vector of length $\tilde{R}_{\text{L}} \leq I_{\text{L}}$, equal to the number of columns in $\mathbf{U}_{\text{L}} \in \mathbb{R}^{I_{\text{L}} \times \tilde{R}_{\text{L}}}$, plus a coefficient vector of length $\tilde{R}_{\text{V}} \leq I_{\text{V}}$, equal to the number of columns in $\mathbf{U}_{\text{V}} \in \mathbb{R}^{I_{\text{V}} \times \tilde{R}_{\text{V}}}$. In our example, PCA would decompose the image ensemble into 777 basis vectors (eigentextures), each of dimension 230,400, and represent each image by a coefficient vector of length 777, which specifies what proportion of each basis vector to accumulate in order to obtain that image. By contrast, TensorTextures decomposes the image ensemble into $37 \times 21$ basis vectors of the same dimension, and represents each image by two coefficient vectors, one of length 37 to encode the view and the other of length 21 to encode the illumination. Thus, each image is represented by $37 + 21 = 58$ coefficients. Figure 5.7 shows how these coefficients multiplicatively modulate the TensorTextures basis vectors according to (5.4) in order to approximate a texture image (which we will soon exploit in multilinear image synthesis or rendering).

Through application of the MPCA algorithm (Algorithm 4.2.1), our multilinear analysis enables a *strategic* dimensionality reduction, which is a mode-specific version of the conventional linear dimensionality reduction of PCA. Whereas dimensionality reduction in PCA results in unpredictable image degradation, multilinear models yield predictable image degradation that can be controlled indepen-

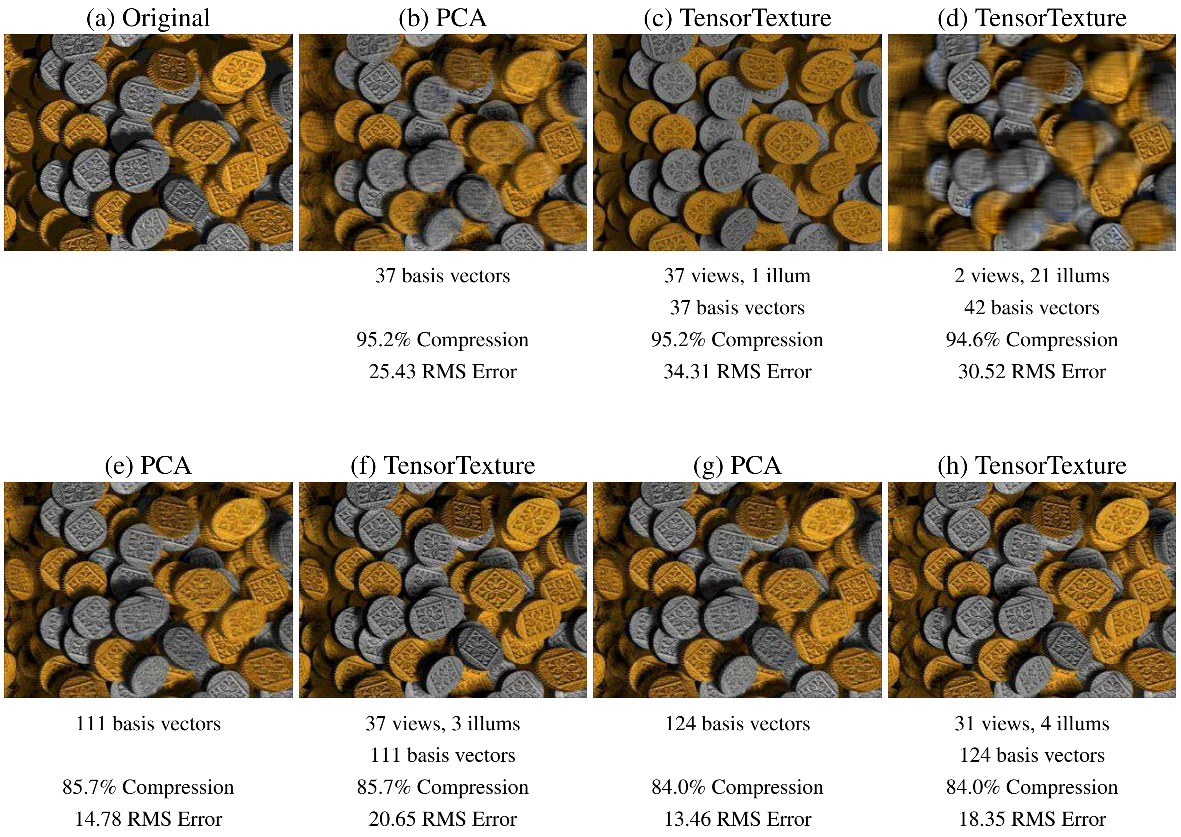


*Figure 5.8: The "perceptual error" incurred by compressing the illumination representation of the TensorTextures model is smaller than that of indiscriminate PCA compression in a subspace of comparable dimension, despite a higher RMS error incurred by the TensorTextures model for these images in the database. However, the overall RMS error for the entire dataset is lower for the TensorTextures model. (a) Original image. (b)–(h) PCA and TensorTexture compressions of image (a) using various numbers of basis vectors. The label above each image indicates the type of compression, while the annotations below indicate the basis set, the compression rate, and the root mean squared (RMS) error relative to the original image (a). For example, the PCA compression (e) retains 111 of the 777 most dominant eigentexture basis vectors, while the TensorTexture image compression (f) retains 111 TensorTextures bases associated with $\mathbf{U}_{\text{V}} \in \mathbb{R}^{37\times 37}$ and $\mathbf{U}_{\text{L}} \in \mathbb{R}^{21\times 3}$, which reduces the illumination representation from 21 dimensions to 3 ($\tilde{R}_{\text{L}} = 3$). In this instance, the RMS error of the PCA-compressed images are lower, yet comparable TensorTexture compressions have the better perceptual quality.*

dently in viewing and illumination.

Figure 5.8 compares TensorTexture image compression against PCA compression. Note in Figure 5.8(c) that the $95.2\%$ reduction of the illumination dimensionality suppresses illumination effects such as shadows and highlights, but that it does not substantially degrade the clarity of the texture, since the rank of the view mode matrix has not been reduced. However, a comparable compression using PCA results in the blurred texture of Figure 5.8(b). Since the multilinear model is less constrained than the linear model, the RMS error of TensorTexture compression is generally smaller than PCA compression. However, there is no guarantee that every individual image has a lower RMS error than the PCA RMS error, since the error function tries to find the best model fit for the entire data set rather than for each image. In this particular instance, however, the RMS error of the TensorTexture compression relative to the original image (Figure 5.8(a)) happens to be larger than the PCA compression, yet its "*perceptual error*" is smaller, yielding a substantially better image quality than comparable PCA compressions.

Figure 5.8(d) shows the degradation of the TensorTexture if we drastically compress in the view mode. Applying PCA compression in Figure 5.8(e), we retain the 111 (out of 777) most dominant eigentextures. Applying TensorTextures, we compress the dimensionality of the illumination mode from 21 to 3 ($\tilde{R}_{\text{L}} = 3$) in Figure 5.8(f). Since $R_{\text{V}} = 37$, we retain $37 \times 3$ TensorTexture basis vectors, equaling the number of retained PCA basis vectors. The total number of coefficients representing the compressed images is $37 + 3$. Figure 5.8(d)–(e) illustrate the same scenario with $31 \times 4$ TensorTexture basis vectors.

## 5.4 Multilinear Image-Based Rendering

Our TensorTextures basis (equation (5.6) and Figure 5.5) leads to a straightforward multilinear rendering algorithm, which is illustrated in Figure 5.7. To render an image $\mathbf{d}$, we compute (cf., (5.4))

$$\mathbf{d} = \boldsymbol{\mathcal{T}} \times_{\text{L}} \mathbf{l}^{\text{T}} \times_{\text{V}} \mathbf{v}^{\text{T}}, \tag{5.7}$$

where $\mathbf{v}$ and $\mathbf{l}$ are, respectively, the view and illumination representation vectors associated with the desired view and illumination directions. These will in general be *novel* directions, in the sense that they will differ from the *observed* directions associated with sample images in the ensemble. Given a novel view (illumination) direction, we first find the three nearest observed view (illumination) directions which form a triangle on the view (illumination) hemisphere that contains this novel direction. We then compute the novel view (illumination) representation vector $\mathbf{v}$ ($\mathbf{l}$) as a convex combination, using homogeneous barycentric coordinates, of the view (illumination) representation vectors associated with the three observed view (illumination) directions (Figure 5.9). Note that this algorithm is appropriate for a planar surface, since every texel of the rendered texture shares the same view/illumination representation. The algorithm (5.7) was applied to render the coins TensorTexture on a planar surface in the treasure chest under continuously varying view and illumination directions (Figure 5.2).

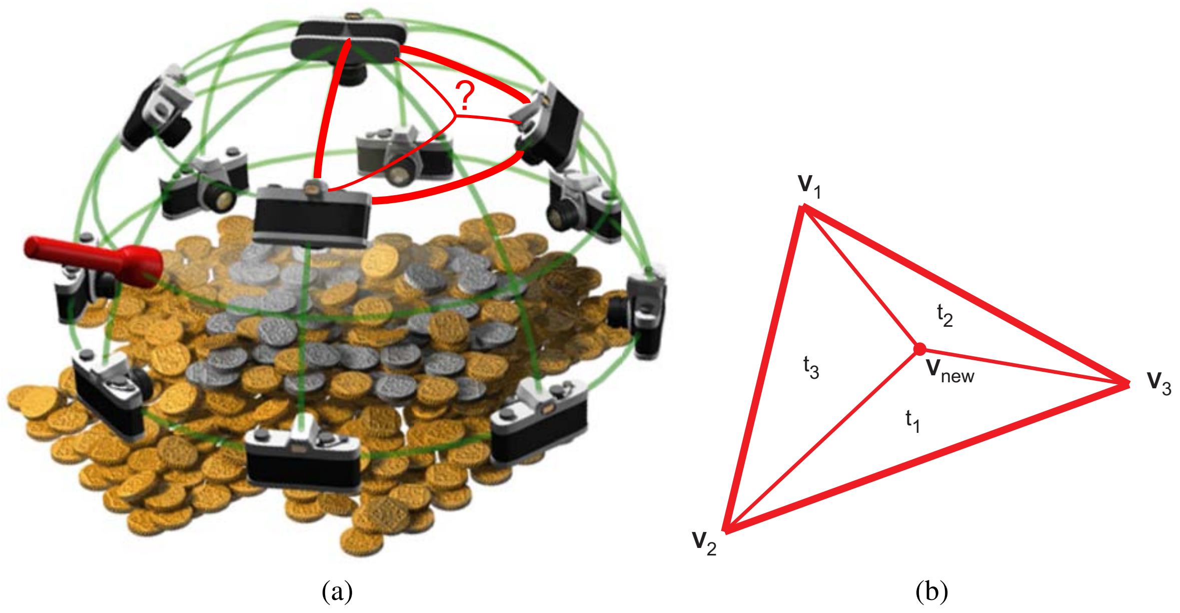
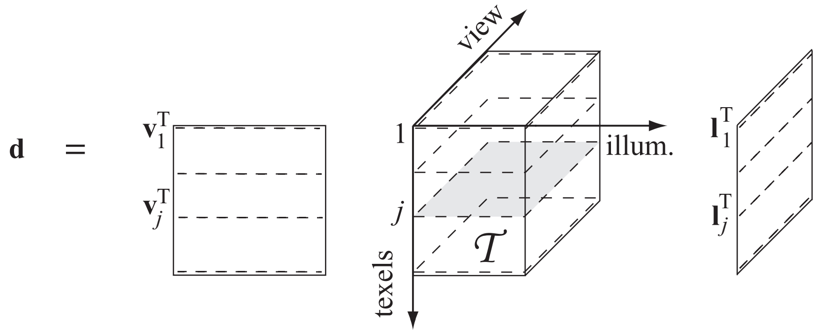


*Figure 5.9: Computing a new view representation using barycentric coordinates.*



$\mathbf{d}$ =
$\mathbf{v}_1^{\mathrm{T}}$
$\mathbf{v}_j^{\mathrm{T}}$
view
1
illum.
$j$
$\mathcal{T}$
texels
$\mathbf{l}_1^{\mathrm{T}}$
$\mathbf{l}_j^{\mathrm{T}}$

*Figure 5.10: TensorTexture rendering when every texel $j$ has a different associated view $\mathbf{v}_j$ and illumination $\mathbf{l}_j$ direction.*

## 5.5 Rendering on Curved Surfaces

When rendering a TensorTexture $\mathbf{d}$ on a curved surface, the view $\mathbf{v}_j$ and illumination $\mathbf{l}_j$ representation vectors associated with each texel $j$ of $\mathbf{d}$ are computed with respect to the given view and illumination directions as well as the direction of the surface normal at the center of texel $j$. The RGB value $d_j$ for texel $j$ is then computed as follows:

$$d_j = \boldsymbol{\mathcal{T}}_j \times_{\mathrm{L}} \mathbf{l}_j^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}_j^{\mathrm{T}}, \tag{5.8}$$

where $\boldsymbol{\mathcal{T}}_j$ is a subtensor of the TensorTexture which governs the interaction between view and illumination for texel $j$ (Figure 5.10).

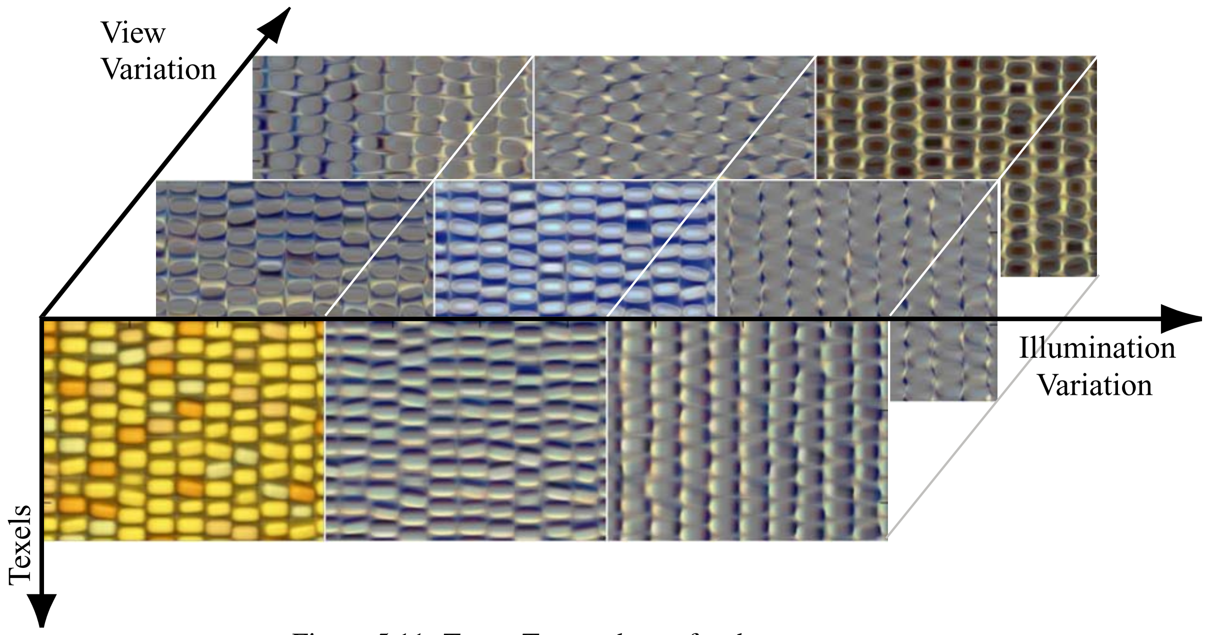


*Figure 5.11: TensorTexture bases for the corn texture.*

## 5.6 Additional Results

We have applied the TensorTextures algorithm to two synthetic image ensembles: The "coins" ensemble, which has served to illustrate our TensorTextures algorithm, and a "corn" image ensemble whose TensorTextures representation is illustrated in Figure 5.11. Figure 5.12 compares standard texture mapping to TensorTexture mapping using the corn texture. Figure 5.13 demonstrates the application of the algorithm of equation (5.8) to render the corn TensorTexture onto a perfect cylinder that forms the head of a scarecrow, lit from two different directions. As the cylinder is rotated, the TensorTexture shows the desired 3D effects, including self-occlusion and self-shadowing between the corn kernels, as well as an appropriately radial pattern of corn kernel orientations around cylinder. Figure 5.14 shows the closing shot of an animated short called *Scarecrows' Quarterly* in which we have employed the TensorTextured scarecrow head.

Both of our synthetic image datasets were acquired by rendering 3D graphics models of surfaces featuring considerable mesostructure. As was the case of the coins, the images of the corn surface were also acquired by rendering the surface from 37 different view and 21 different illumination directions. Both the coins and the corn TensorTexture models retain $37 \times 11 = 407$ TensorTexture basis vectors by reducing the illumination mode from 21 to 11, while retaining all of the basis vectors of the view mode in order to maintain the sharpness of the rendered images.

It takes considerable time to render each of the original sample images because of the nontrivial scene geometries and rendering methods employed. In particular, Alias' *Maya* consumed around 180 seconds on average to render the coins images and around 20 seconds on average to render the corn images on a 2GHz P4 with 1GB of RAM. After our TensorTextures model has been computed offline, the online rendering of the TensorTextures is significantly more efficient. For the coins example, the

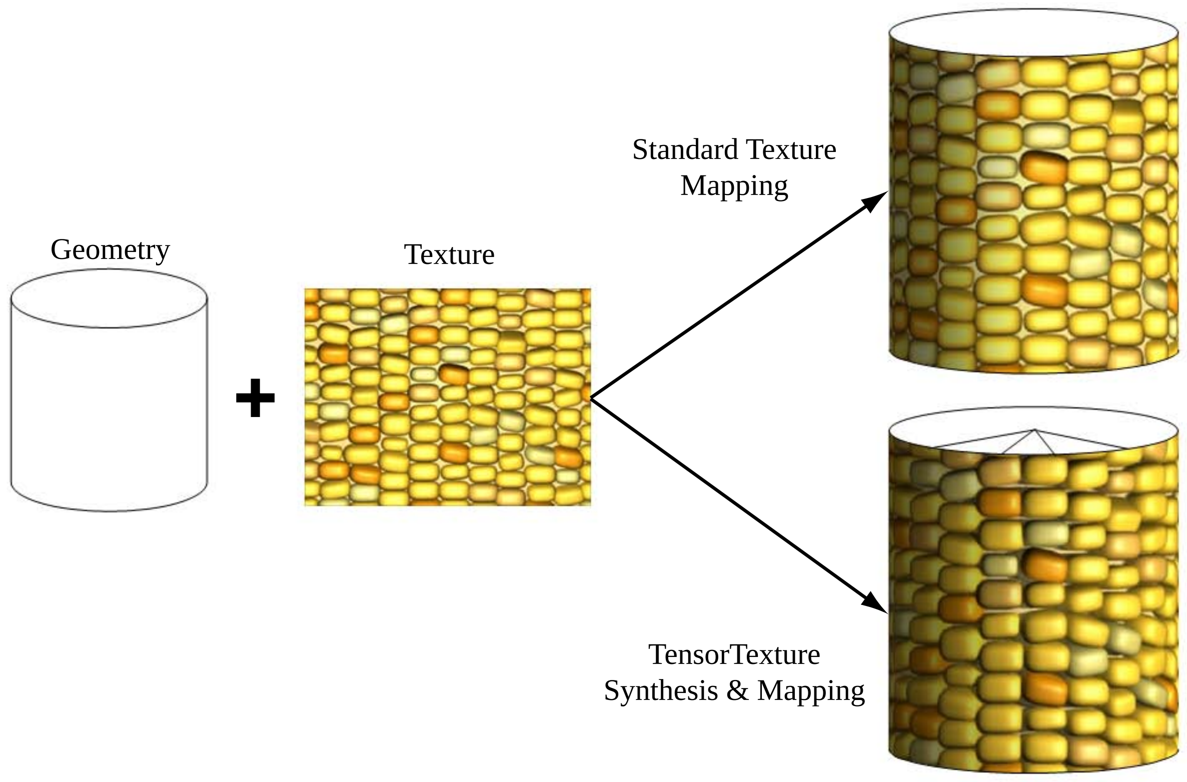


*Figure 5.12: Standard texture mapping versus TensorTexture mapping. TensorTextures is a generative model that learns the bidirectional texture function.*

rendering of the TensorTextured surfaces for arbitrary views and illuminations, implemented in not especially well-optimized Matlab code, took on average 1.6 seconds per image on the same workstation. Furthermore, because it is image-based, the TensorTextures online rendering speed is independent of the scene complexity.

We have also applied our TensorTextures algorithm to images of natural textured surfaces from the University of Bonn BTF database. The materials and methods for the image acquisition are detailed in [Sattler et al. 2003]. Figure 5.15 shows "Impalla" (a stone) and "Corduroy" TensorTextures mapped onto spheres using the rendering algorithm of equation (5.8). The TensorTextures bases were computed from ensembles of RGB sample images, each of size $256 \times 256 \times 3$, acquired under 81 view and 81 illumination directions. The image data were organized as tensors $\mathcal{D} \in \mathbb{R}^{196608 \times 81 \times 81}$. The view and illumination mode matrices were computed in accordance with Step 1 of the MPCA algorithm (Algorithm 4.2.1), and their dimensionality was reduced from 81 to 61 ($\tilde{R}_{\mathrm{V}} = 61$) and from 81 to 27 ($\tilde{R}_{\mathrm{L}} = 61$), respectively, yielding the reduced mode matrices $\mathbf{U}_{\mathrm{V}} \in \mathbb{R}^{81 \times 61}$ and $\mathbf{U}_{\mathrm{L}} \in \mathbb{R}^{81 \times 27}$. The $\mathcal{T} \in \mathbb{R}^{196608 \times 61 \times 27}$ TensorTextures bases vectors were computed according to (5.6). As a final demonstration, we have created a *Flintstone Phonograph* animation which maps the Impalla TensorTexture on the planar "turntable" surface (Figure 5.16).

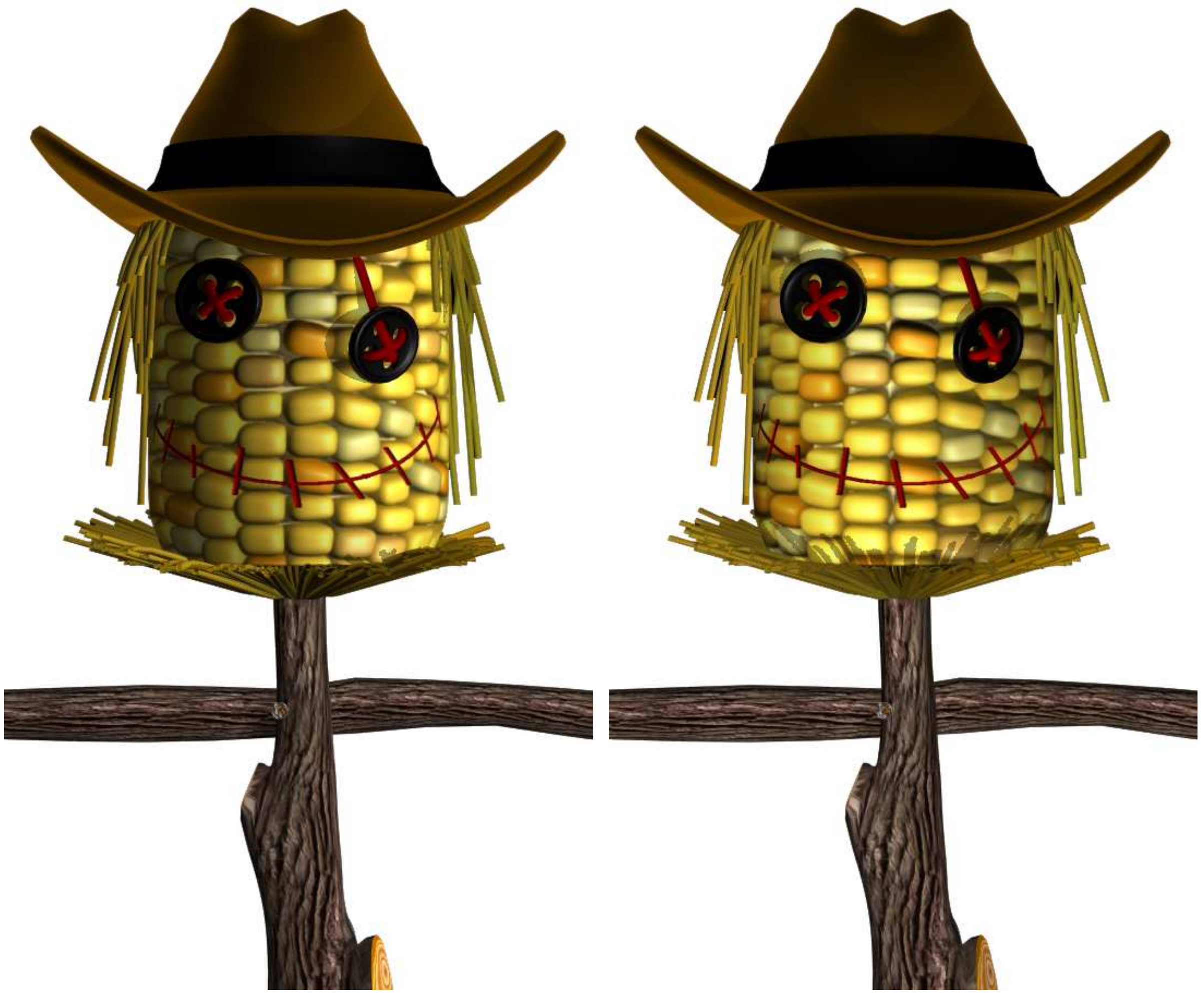


*Figure 5.13: Renderings, with different light source directions, of the corn TensorTexture mapped onto a cylinder that forms the scarecrow's head.*

*Figure 5.14: A still image from the Scarecrows' Quarterly animation.*

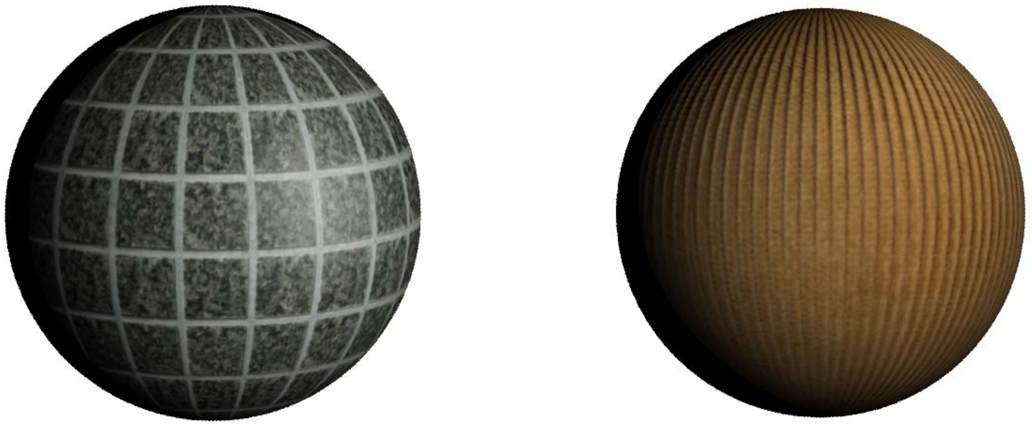


*Figure 5.15: "Impalla" (left) and "Corduroy" (right) TensorTextures rendered on spheres. A third order image data tensor associated with 81 view directions, 81 illumination directions, and 196608 texels (256 rows $\times$ 256 columns $\times$ 3 channels) was employed to the TensorTextures bases $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{D}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}}^{\mathrm{T}}$. Image compression was obtained by retaining $61 \times 27$ TensorTextures basis associated with $\mathbf{U}_{\mathrm{V}} \in \mathbb{R}^{81\times 61}$ and $\mathbf{U}_{\mathrm{L}} \in \mathbb{R}^{81\times 27}$, which reduces the view representation from 81 dimensions to 61 ($\tilde{R}_{\mathrm{V}} = 61$) and the illumination representation from 81 dimensions to 27 ($\tilde{R}_{\mathrm{L}} = 27$).*

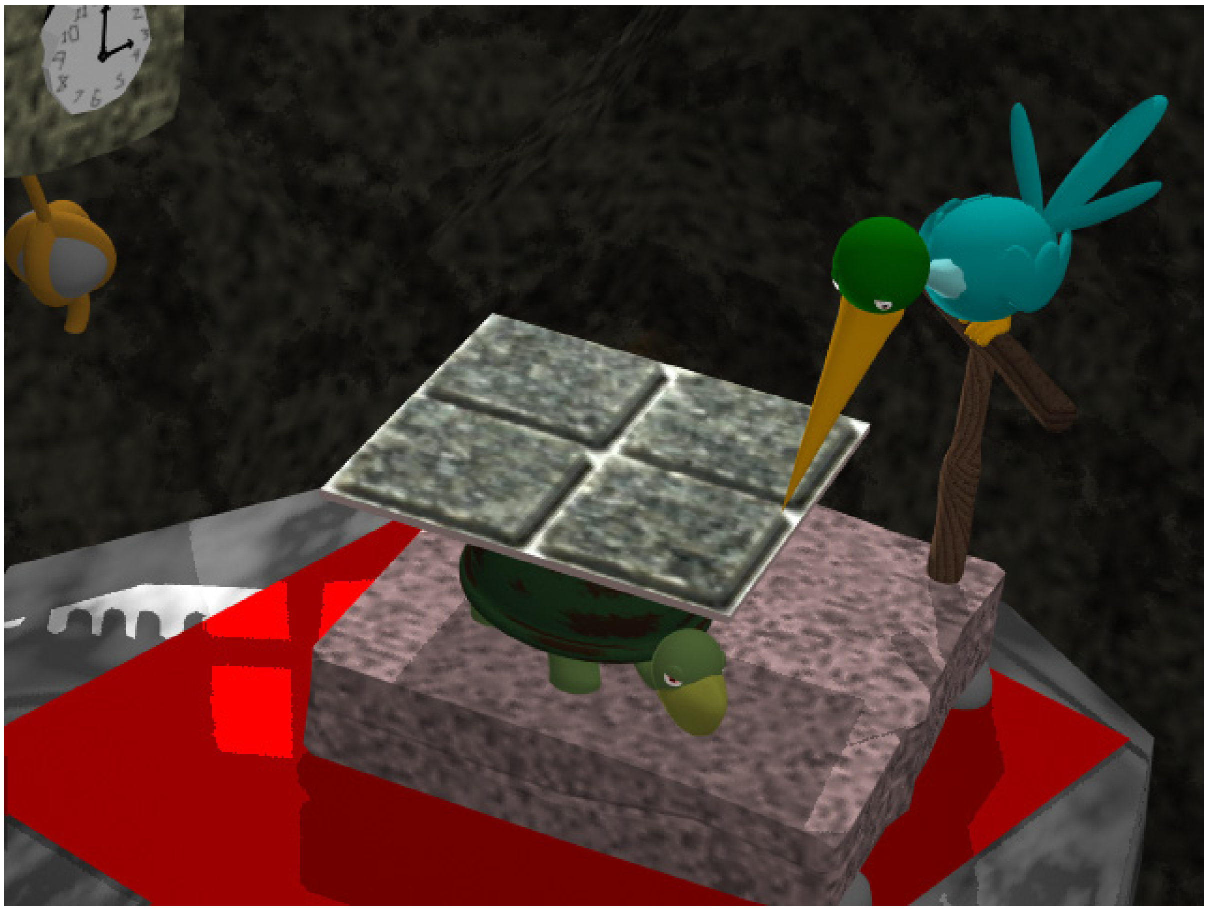


*Figure 5.16: A still image from the Flintstone Phonograph animation.*

# 6 Multilinear Image Analysis: TensorFaces

**Contents**

We will now apply our multilinear framework in the context of computer vision, which is concerned with image analysis, conceptually the inverse problem to computer graphics. A highly-researched class of methods in computer vision is known as *appearance-based methods*. They have been applied to the recognition of images of arbitrary objects, but have attracted the greatest attention in the context of human facial images. Given a database of suitably labeled training images of numerous individuals, our multilinear approach aspires to learn parsimonious appearance-based representations of the image ensemble, which may be used for facial image compression or for facial image recognition, a subject that we will investigate in the next chapter. Note, however, that in no way is our multilinear approach limited to appearance-based methods; it is also applicable to range data, 3D vertices, and other types of geometric models that are commonly used in *model-based methods* in computer vision [Terzopoulos et al. 2004].

The conventional appearance-based approach addresses the problem of facial representation by taking advantage of linear algebra. In particular, PCA has been at the core of the dominant appearance-based methods, such as the well-known “eigenfaces” face recognition method [Sirovich and Kirby 1987; Turk and Pentland 1991a]. As we have stated, however, PCA accounts only for single-factor variations in image ensembles. In our appearance-based recognition work, we confront the fact that natural images result from the interaction of *multiple* causal factors related to scene structure, illumination, and imaging. For facial images, these causal factors include different facial geometries and expressions, head poses, and lighting conditions.

Applied to facial images, our multilinear representation is called *TensorFaces*. TensorFaces represents the image ensemble as a higher-dimensional tensor. This image data tensor $\boldsymbol{\mathcal{D}}$ must be decomposed in order to separate and parsimoniously represent the causal factors related to scene structure, illumination, and imaging.

## 6.1 The TensorFaces Representation

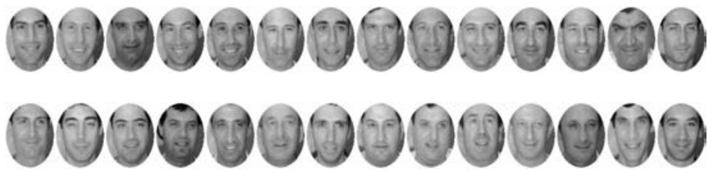
We first apply our multilinear analysis technique to ensembles of natural facial images, employing a portion of the Weizmann face image database [Moses et al. 1996] of 28 male subjects imaged in 15 different views, under 4 different illuminations, performing 3 different expressions.[1] For the sake of concreteness in the ensuing discussion, consider an image ensemble of the 28 subjects acquired from 5 views, under 3 illuminations, and performing 3 expressions, for a total of 1260 images. Using a global rigid optical flow algorithm, we roughly aligned the original $512 \times 352$ pixel images relative to one reference image. The images were then decimated by a factor of 3 and cropped as shown in Figure 6.1, yielding a total of 7943 pixels per image within the elliptical cropping window.

Thus, the number of modes is $M = 5$ and our facial image data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{7943\times 28\times 5\times 3\times 3}$ (Figure 6.1(c)). Applying the MPCA algorithm (Algorithm 4.2.1) to $\boldsymbol{\mathcal{D}}$, we obtain the decomposition

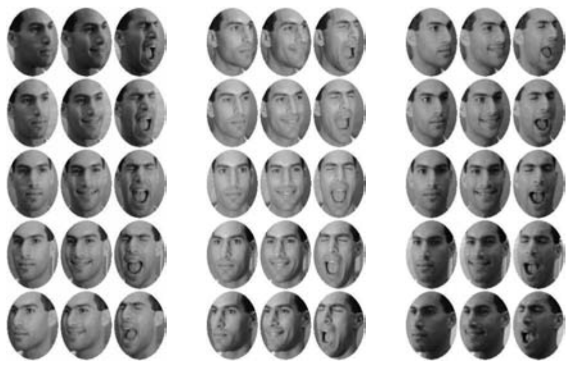
$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{T}} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}} \times_{\mathrm{E}} \mathbf{U}_{\mathrm{E}}. \tag{6.1}$$

The extended core tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{7943\times 28\times 5\times 3\times 3}$, which we call *TensorFaces* (Figure 6.2), captures the interaction between the four causal factors: The rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{P}} \in \mathbb{R}^{28\times 28}$ are the representations for different people whose images comprise the data tensor (the subscript “P” denotes the *people* mode), the rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{V}} \in \mathbb{R}^{5\times 5}$ represent different views (the subscript “V” denotes the *view* mode), the rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{L}} \in \mathbb{R}^{3\times 3}$ represent the illuminations (the subscript “L” denotes the *illumination* mode), and the rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{E}} \in \mathbb{R}^{3\times 3}$ represent different expressions (the subscript “E” denotes the *expression* mode).

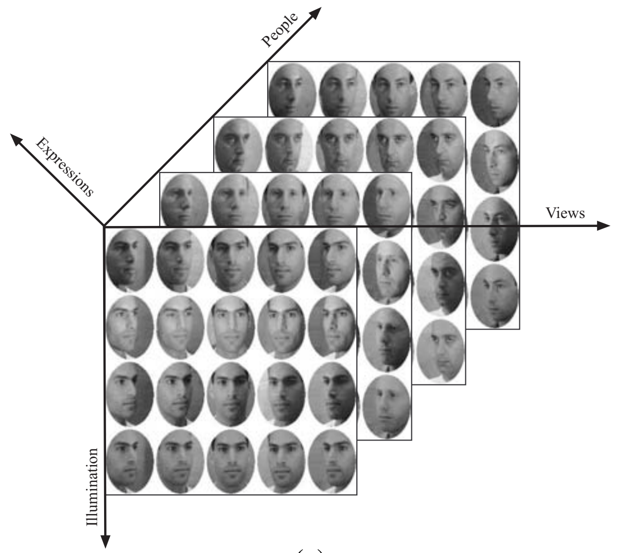
[1] A computer-controlled robot arm positioned the camera to $\pm 34^\circ$, $\pm 17^\circ$, and $0^\circ$, the frontal view in the horizontal plane. The face was illuminated by turning on and off three light sources fixed at the same height as the face and positioned to the left, center, and right of the face. For additional details, see [Moses et al. 1996].



(a)



(b)



(c)

*Figure 6.1: The Weizmann facial image database (28 subjects, 45 images per subject). (a) The 28 subjects shown in expression 2 (smile), view 3 (frontal), and illumination 2 (frontal). (b) Part of the image set for subject 1. Left to right, the three panels show images captured in illuminations 1, 2, and 3. Within each panel, images of expressions 1, 2, and 3 (neural, smile, yawn) are shown horizontally while images from views 1, 2, 3, 4, and 5 are shown vertically. The image of subject 1 in (a) is the image situated at the center of (b). (c) The* $5^{th}$*-order data tensor* $\mathcal{D}$ *for the image ensemble; only images in expression 1 (neutral) are shown.*

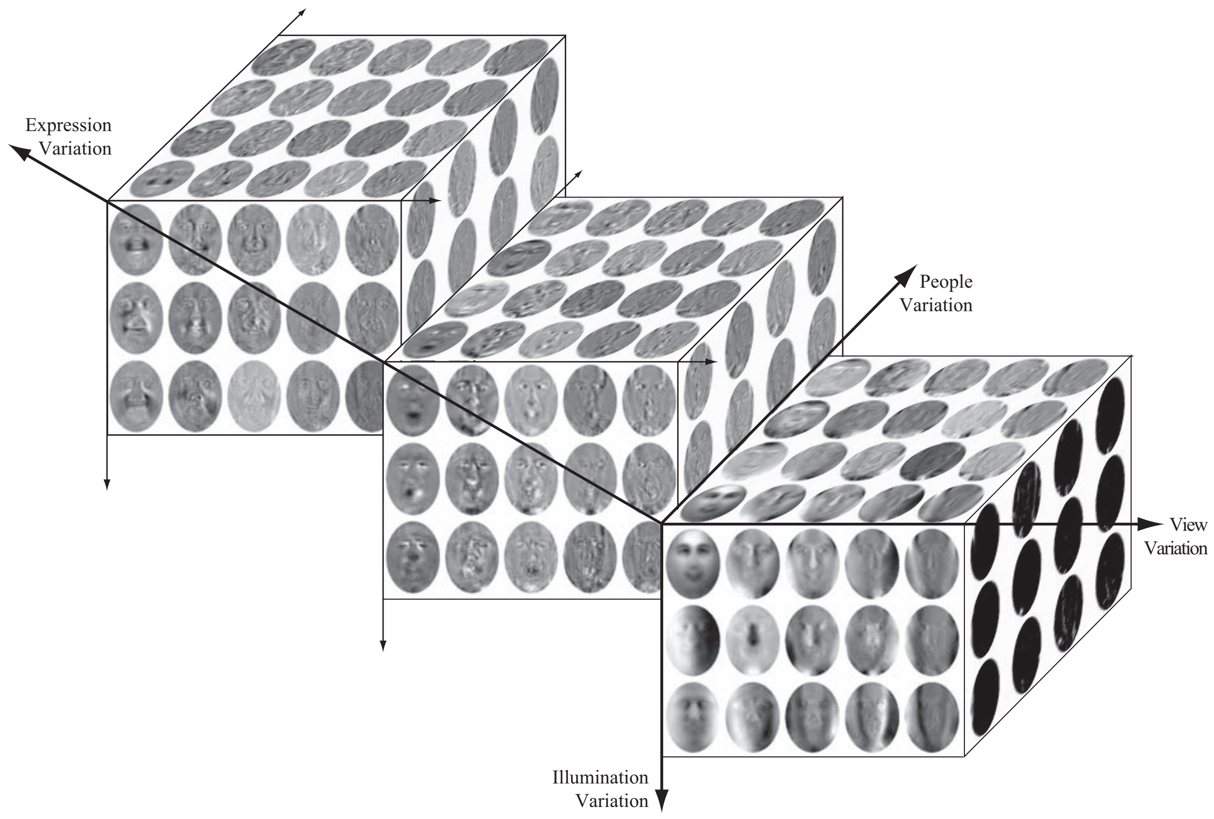


*Figure 6.2: A partial visualization of the $\mathcal{T} \in \mathbb{R}^{7943\times 28\times 5\times 3\times 3}$ TensorFaces representation of $\mathcal{D}$, obtained as $\mathcal{T} = \mathcal{D} \times_{\text{P}} \mathbf{U}_{\text{P}}^{\text{T}} \times_{\text{V}} \mathbf{U}_{\text{V}}^{\text{T}} \times_{\text{L}} \mathbf{U}_{\text{L}}^{\text{T}} \times_{\text{E}} \mathbf{U}_{\text{E}}^{\text{T}}$.*

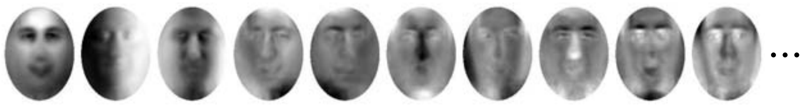


*Figure 6.3: The measurement mode matrix $\mathbf{U}_{\text{x}}$ contains the conventional PCA eigenvectors ("eigenfaces"), which are the principal axes of variation across all the images. The first 10 eigenvectors are shown.*

TensorFaces is expressed as

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{Z}} \times_{\text{x}} \mathbf{U}_{\text{x}}, \tag{6.2}$$

where $\mathbf{U}_{\text{x}} \in \mathbb{R}^{7943\times 1260}$ is the measurement mode matrix whose orthonormal column vectors span the image space (the lower case “x” subscript denotes the *pixel* mode). Note that neither $\mathbf{U}_{\text{x}}$ nor the core tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{1260\times 28\times 5\times 3\times 3}$ are needed in our multilinear analysis and they are not explicitly computed in practice. Instead, the MPCA algorithm computes TensorFaces $\boldsymbol{\mathcal{T}}$ and the 4 causal factor mode matrices by forming the product

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{D}} \times_{\text{P}} \mathbf{U}_{\text{P}}^{\text{T}} \times_{\text{V}} \mathbf{U}_{\text{V}}^{\text{T}} \times_{\text{L}} \mathbf{U}_{\text{L}}^{\text{T}} \times_{\text{E}} \mathbf{U}_{\text{E}}^{\text{T}}. \tag{6.3}$$

From a computational standpoint, equation (6.3) is preferable to (6.2) since it prescribes computing the relatively small matrices $\mathbf{U}_{\text{P}}$, $\mathbf{U}_{\text{V}}$, $\mathbf{U}_{\text{L}}$, and $\mathbf{U}_{\text{E}}$ rather than the generally large matrix $\mathbf{U}_{\text{x}}$ that PCA must compute. This efficiency is exploited by the MPCA algorithm.

In Section 4.2 we showed that multilinear analysis subsumes linear, PCA analysis. Each column vector in $\mathbf{U}_{\text{x}}$ is an “eigenimage” (Figure 6.3), since $\mathbf{U}_{\text{x}}$ is computed by performing an SVD of the matrix $\mathbf{D}_{[\text{x}]}$, the mode-1 matrixized data tensor $\boldsymbol{\mathcal{D}}$, whose columns comprise all the vectorized facial images. The resulting eigenimages are identical to the conventional eigenfaces [Sirovich and Kirby 1987; Turk and Pentland 1991b; Turk and Pentland 1991a]. Eigenimages represent merely the principal axes of variation of the entire ensemble of images in the dataset. The big advantage of our multilinear analysis beyond linear PCA is that TensorFaces explicitly represent each of the causal factors and encode how they interact to produce facial images.

## 6.2 Strategic Dimensionality Reduction

To illustrate the dimensionality reduction abilities of our MPCA algorithm (Algorithm 4.2.1), we employ from the Weizmann facial image database an ensemble of images of 11 people, each photographed in neutral expression from a frontal view under 16 different illuminations. Figure 6.4(a) shows three of the 176 original 7943-pixel images for one of the subjects.

Applied to the ensemble tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{7943\times 11\times 16}$, the Step 1 of the algorithm yields mode matrices $\mathbf{U}_{\text{P}} \in \mathbb{R}^{11\times 11}$ and $\mathbf{U}_{\text{L}} \in \mathbb{R}^{16\times 16}$. We then truncate the illumination mode matrix $\mathbf{U}_{\text{L}}$ from 16 columns to 3 ($\tilde{R}_{\text{L}} = 3$), thus obtaining the reduced-rank matrix $\mathbf{U}_{\text{L}} \in \mathbb{R}^{16\times 3}$. We iterate in Step 2, updating $\mathbf{U}_{\text{L}}$ and the (non-truncated) mode matrix $\mathbf{U}_{\text{P}}$, until convergence (3 iterations).

Figure 6.4(b) shows illumination-compressed images of the subject extracted from the dimensionality-reduced multilinear representation $\boldsymbol{\mathcal{D}} \simeq \boldsymbol{\mathcal{T}} \times_{\text{P}} \mathbf{U}_{\text{P}} \times_{\text{L}} \mathbf{U}_{\text{L}}$. Note that the $81.25\%$ reduction of the illumination dimensionality suppresses illumination effects such as shadows and highlights, but that it does not substantially degrade the appearance of the person, since the rank of the person mode matrix was not reduced. When we increase the illumination dimensionality to 6, the shadows and highlights begin to reappear, as shown in Figure 6.4(c).

Thus, our multilinear model enables a *strategic* dimensionality reduction, which is more targeted

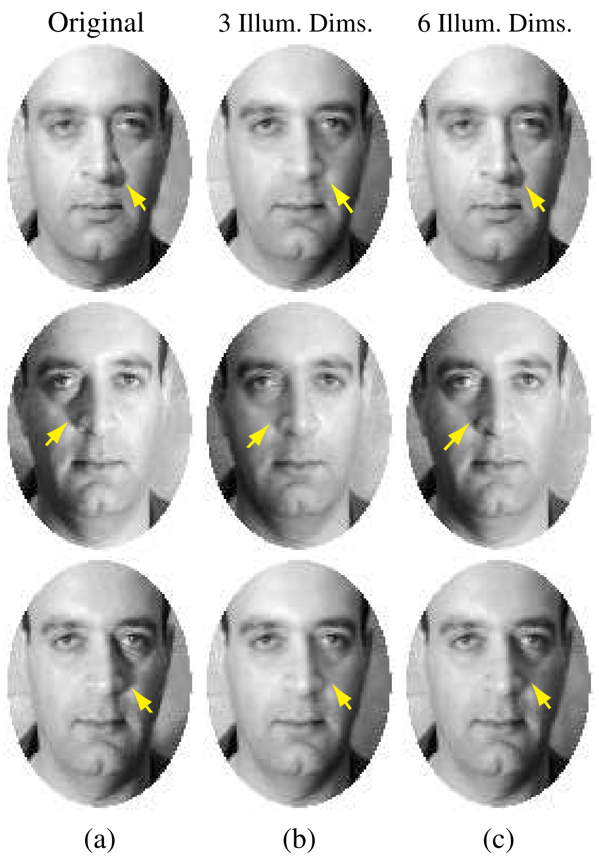


*Figure 6.4: A subject was imaged under 16 different illuminations. (a) Three original images displaying different illumination conditions. (b) Compression of the images in (a) by reducing the illumination representation from 16 dimensions to 3 ($\tilde{R}_{\text{L}} = 3$); i.e., $\mathbf{U}_{\text{L}} \in \mathbb{R}^{16\times 3}$. This degrades the illumination effects (cast shadows, highlights). Arrows indicate the shadow cast by the nose in the original images (a) and the attenuated shadow in the compressed images (b). The shadow begins to reappear when the illumination dimensionality is increased to 6 ($\tilde{R}_{\text{L}} = 6$) in (c); i.e., $\mathbf{U}_{\text{L}} \in \mathbb{R}^{16\times 6}$. Image sharpness and detailed facial features are well-preserved in both (b) and (c).*

than linear (PCA) dimensionality reduction. Figure 6.5 compares TensorFaces image compression against PCA compression. Applying PCA compression, we retain in Figure 6.5(b) the 11 (out of 176) most dominant eigenfaces and in Figure 6.5(d) the 33 most dominant eigenfaces. Applying TensorFaces, we compress the dimensionality of the illumination mode from 16 to 1 ($\tilde{R}_{\text{L}} = 1$) in Figure 6.5(c) and from 16 to 3 ($\tilde{R}_{\text{L}} = 3$) in Figure 6.5(e). Since $\tilde{R}_{\text{P}} = 11$, in the first instance we retain $11 \times 1$ TensorFaces, while in the second we retain $11 \times 3$ TensorFaces, each time equaling the number of retained eigenfaces. Note that the total number of coefficients representing the compressed images is $11 + 1$ and $11 + 3$, respectively. The figure shows shows an unusual case in which the root mean squared errors (RMSE) relative to the original images, which are indicated in the figure, are larger for the TensorFaces compressions than they are for the PCA compressions. However, the "*perceptual error*" [Teo and Heeger 1994] of the TensorFaces compressions are significantly smaller, yielding substantially better image quality than PCA in subspaces of comparable dimension.

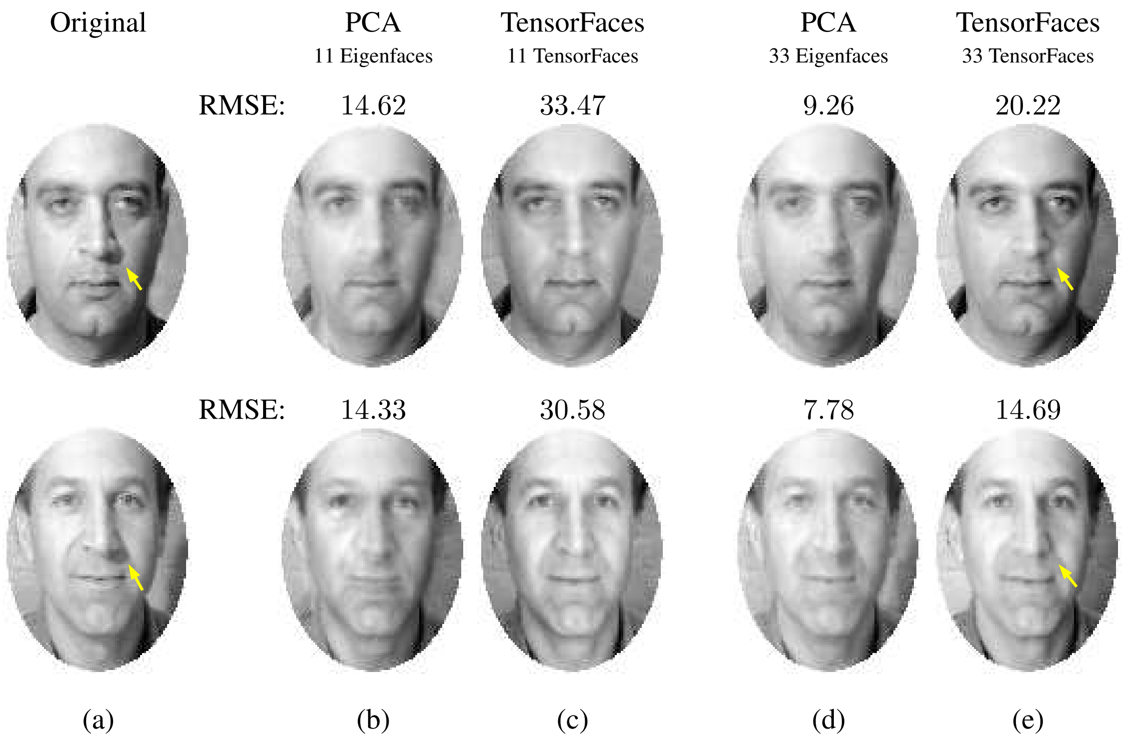
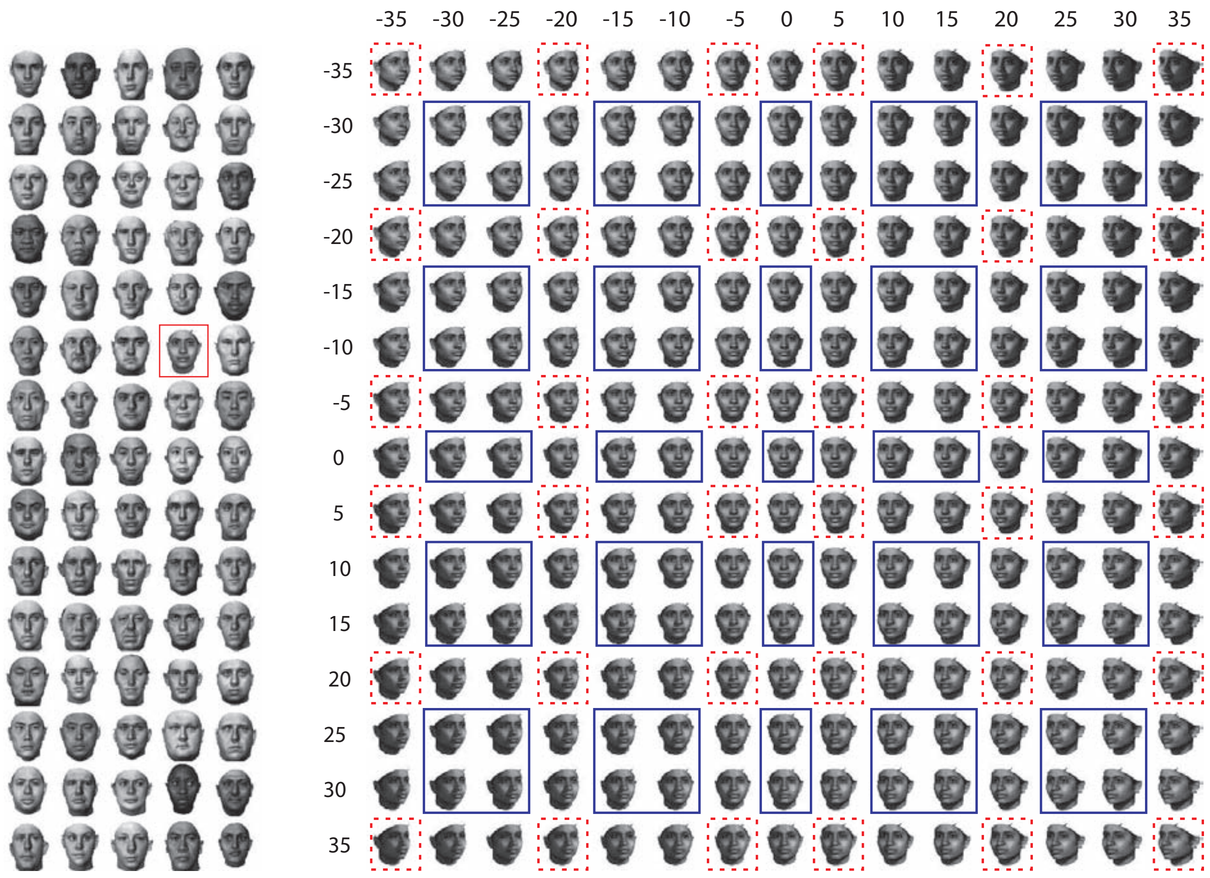


*Figure 6.5: The "perceptual error" of TensorFaces compression of illumination is smaller than indiscriminate PCA compression in a subspace of comparable dimension. (a) Original images. (b) PCA image compression obtained by retaining the 11 most dominant eigenfaces. (c) TensorFaces image compression obtained by retaining 11 TensorFaces associated with $\mathbf{U}_{\text{P}} \in \mathbb{R}^{11\times 11}$ and $\mathbf{U}_{\text{L}} \in \mathbb{R}^{16\times 1}$, which reduces the illumination representation from 16 dimensions to 1 ($\tilde{R}_{\text{L}} = 1$). (d) PCA image compression obtained by retaining the 33 most dominant eigenfaces. (e) TensorFaces image compression obtained by retaining 33 TensorFaces associated with $\mathbf{U}_{\text{P}} \in \mathbb{R}^{11\times 11}$ and $\mathbf{U}_{\text{L}} \in \mathbb{R}^{16\times 3}$, which reduces the illumination representation from 16 dimensions to 3 ($\tilde{R}_{\text{L}} = 3$). Compared to the original images, the root mean squared errors (RMSE) of the PCA-compressed images are lower, yet the TensorFaces-compressed images have significantly better perceptual quality.*

## 6.3 Application to a Morphable Faces Database

We next illustrate our technique using gray-level facial images of 75 subjects (Figure 6.6). Each subject is imaged from 15 different views ($\theta = -35^\circ$ to $+35^\circ$ in $5^\circ$ steps on the horizontal plane $\phi = 0^\circ$) under 15 different illuminations ($\theta = -35^\circ$ to $+35^\circ$ in $5^\circ$ steps on an inclined plane $\phi = 45^\circ$). Figure 6.6(b) shows the set of 225 images for one of the subjects with views arrayed horizontally and illuminations arrayed vertically. The image set was rendered from a 3D scan of the subject shown framed in Figure 6.6(a). Each image is $80 \times 107 = 8560$ pixels in size. The 75 scans shown in the figure were recorded using a Cyberware$^{\text{TM}}$ 3030PS laser scanner and are part of the 3D morphable faces database created at the University of Freiburg [Blanz and Vetter 1999].

We select an ensemble of 2700 images from the dataset of Figure 6.6 comprising the dash-framed images for each person. Thus, $M = 4$ and our facial image data tensor is $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{8560\times 75\times 6\times 6}$ (Fig-



(a) (b)

*Figure 6.6: A facial image dataset. (a) 3D scans of 75 subjects, recorded using a Cyberware*$^{\mathrm{TM}}$ *3030PS laser scanner as part of the University of Freiburg 3D morphable faces database [Blanz and Vetter 1999]. (b) Facial images for a subject (framed head in (a)), viewed from 15 different views (across) under 15 different illuminations (down). In our recognition experiments, the dash-framed images served as training images; the solid-framed images served as test images.*

ure 6.7). Applying multilinear analysis to $\mathcal{D}$, we obtain

$$\mathcal{D} = \mathcal{T} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}. \tag{6.4}$$

As before, TensorFaces $\mathcal{T} \in \mathbb{R}^{8560\times 75\times 6\times 6}$ governs the interaction between the now 3 causal factors: The rows of the causal factor mode matrix $\mathbf{U}_{\mathrm{P}} \in \mathbb{R}^{75\times 75}$ represent the different people, those of the causal factor mode matrix $\mathbf{U}_{\mathrm{V}} \in \mathbb{R}^{6\times 6}$ represent the different views, and those of the causal factor mode matrix $\mathbf{U}_{\mathrm{L}} \in \mathbb{R}^{6\times 6}$ represent the different illuminations. Tensor $\mathcal{T} = \mathcal{Z} \times_{\mathrm{x}} \mathbf{U}_{\mathrm{x}}$, where the orthonormal column vectors of the measurement mode matrix $\mathbf{U}_{\mathrm{x}} \in \mathbb{R}^{8560\times 2700}$ span the image space of images and the core tensor $\mathcal{Z} \in \mathbb{R}^{2700\times 75\times 6\times 6}$ contains normalizing parameters associated with the 4 mode matrices. Figure 6.8(a) illustrates the eigenfaces, or column vectors of $\mathbf{U}_{\mathrm{x}}$, for this dataset, while Figure 6.8(b) illustrates the TensorFaces efficiently computed by MPCA as $\mathcal{T} = \mathcal{D} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}}^{\mathrm{T}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}^{\mathrm{T}}$.

To illustrate dimensionality reduction using the MPCA algorithm (Algorithm 4.2.1), we employed

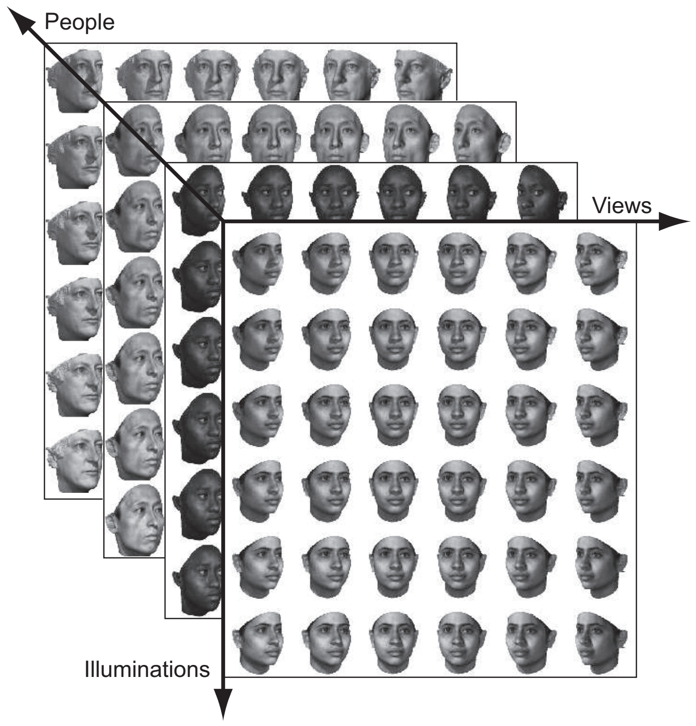


*Figure 6.7: A portion of the* $4^{th}$*-order data tensor* $\boldsymbol{\mathcal{D}}$ *for the image ensemble formed from the dash-framed images of each person in Figure 6.6. Only 4 of the 75 people are shown.*

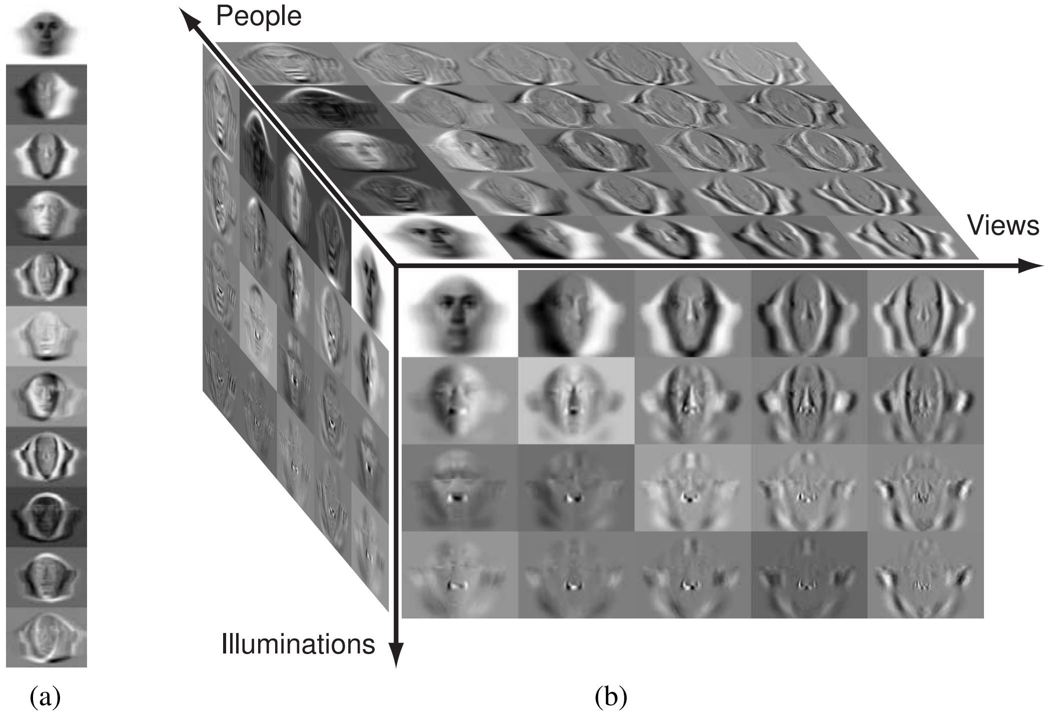


*Figure 6.8: Eigenfaces and TensorFaces bases for an ensemble of 2,700 facial images spanning 75 people, each imaged under 6 viewing and 6 illumination conditions. (a) PCA eigenvectors (eigenfaces)* $\mathbf{U}_{\text{x}}$*, which are the principal axes of variation across all images. (b) A partial visualization of the TensorFaces* $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{8560\times75\times6\times6}$ *representation obtained from* $\boldsymbol{\mathcal{D}}$ *as* $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{D}} \times_{\text{P}} \mathbf{U}_{\text{P}}^{\text{T}} \times_{\text{V}} \mathbf{U}_{\text{V}}^{\text{T}} \times_{\text{L}} \mathbf{U}_{\text{L}}^{\text{T}}$.

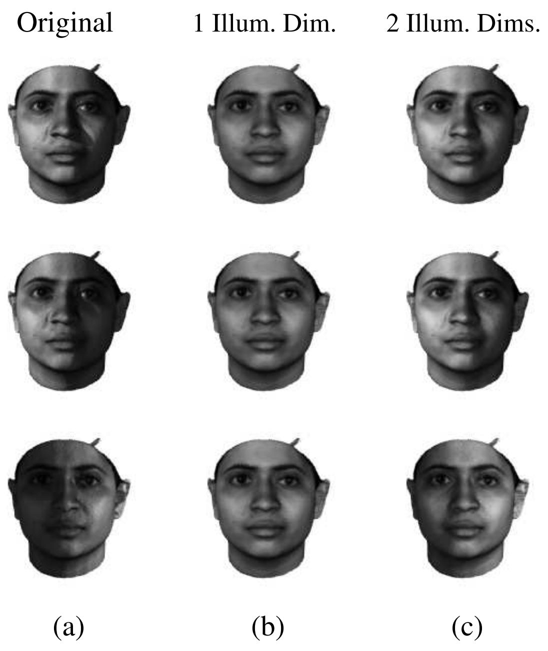


*Figure 6.9: 16 subjects were imaged under 7 view and 20 illumination conditions. (a) Three original images displaying different illumination conditions. (b) Compression of the images in (a) by reducing the illumination representation from 20 dimensions to 1 ($\tilde{R}_L = 1$); i.e., $\mathbf{U}_L \in \mathbb{R}^{20\times1}$. This degrades the illumination effects (cast shadows, highlights). The shadow begins to reappear when the illumination dimensionality is increased to 2 ($\tilde{R}_L = 2$) in (c); i.e., $\mathbf{U}_L \in \mathbb{R}^{20\times2}$ a matrix. Image sharpness and detailed facial features are well-preserved in both (b) and (c).*

an image ensemble in which each of 16 subjects was imaged from 7 different views under 20 different illumination conditions. Figure 6.9(a) shows three of the 2240 original 8560-pixel images for one of the subjects.

Applying our MPCA algorithm (Algorithm 4.2.1) to $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{8560\times16\times7\times20}$, Step 1 yields mode matrices $\mathbf{U}_P \in \mathbb{R}^{16\times16}$, $\mathbf{U}_V \in \mathbb{R}^{7\times7}$, and $\mathbf{U}_L \in \mathbb{R}^{20\times20}$. We then truncate the illumination mode matrix $\mathbf{U}_L$ from 20 columns to $\tilde{R}_L = 1$, thus obtaining the reduced-rank matrix $\mathbf{U}_L \in \mathbb{R}^{20\times1}$. We iterate in Step 2, updating $\mathbf{U}_L$ along with the other (non-truncated) mode matrices $\mathbf{U}_P$ and $\mathbf{U}_V$ until convergence (3 iterations).

Figure 6.9(b) shows illumination-compressed images of the subject extracted from the dimensionality-reduced multilinear representation $\boldsymbol{\mathcal{D}} \simeq \boldsymbol{\mathcal{T}} \times_P \mathbf{U}_P \times_V \mathbf{U}_V \times_L \mathbf{U}_L$. Note that the 95% reduction of the illumination dimensionality suppresses illumination effects such as shadows and highlights, but that it does not substantially degrade the appearance of the person, since the rank of the person mode matrix was not reduced. Increasing the illumination dimensionality to 2, $\mathbf{U}_L \in \mathbb{R}^{20\times2}$, the shadows and highlights begin to reappear, as shown in Figure 6.9(c).

Thus, our multilinear model again enables a *strategic* dimensionality reduction, which is more targeted than linear (PCA) dimensionality reduction. Figure 6.10 compares TensorFaces image compression against PCA compression. Applying PCA compression, we retain in Figure 6.10(b) the 112 (out of 2240) most dominant eigenfaces and in Figure 6.10(d) the 224 most dominant eigenfaces. Applying

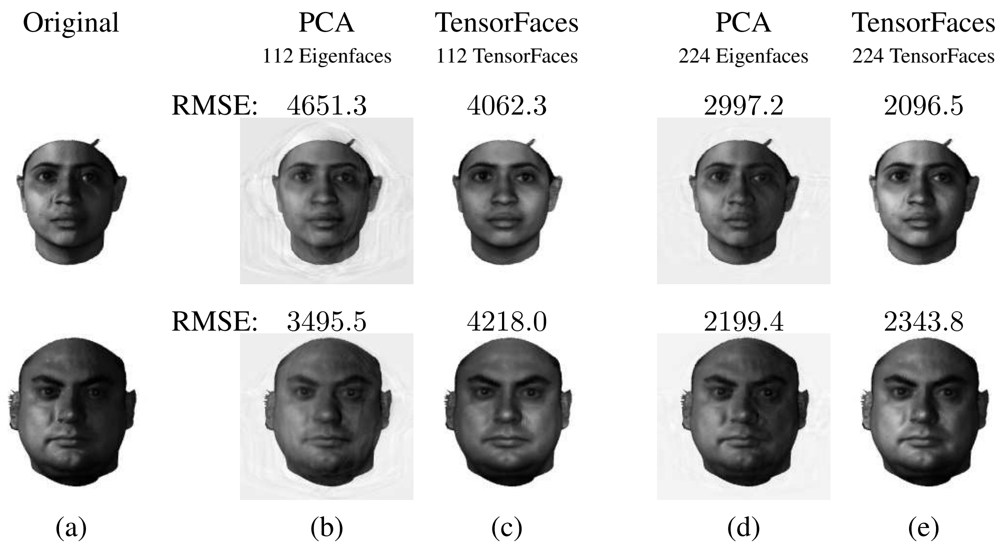


*Figure 6.10: The "perceptual error" of TensorFaces compression of illumination is smaller than indiscriminate PCA compression in a subspace of comparable dimension. (a) Original image. (b) PCA image compression obtained by retaining the 112 most dominant eigenfaces. (c) TensorFaces image compression by reducing the illumination representation from 20 dimensions to 1 ($\tilde{R}_{\text{L}} = 1$). (d) PCA image compression obtained by retaining the 224 most dominant eigenfaces. (e) TensorFaces image compression by reducing the illumination representation from 20 dimensions to 2 ($\tilde{R}_{\text{L}} = 2$). Note that in the images of the male, the root mean squared errors (RMSE) of the PCA-compressed images are lower, yet the TensorFaces-compressed images have significantly better perceptual quality.*

TensorFaces, we compress the dimensionality of the illumination mode from 20 to 1 ($\tilde{R}_{\text{L}} = 1$) in Figure 6.10(c) by retaining 112 TensorFaces associated with $\mathbf{U}_{\text{P}} \in \mathbb{R}^{16\times16}$, $\mathbf{U}_{\text{V}} \in \mathbb{R}^{7\times7}$, and $\mathbf{U}_{\text{L}} \in \mathbb{R}^{20\times1}$, and from 20 to 2 ($\tilde{R}_{\text{L}} = 2$) in Figure 6.10(e) obtained by retaining 224 TensorFaces associated with $\mathbf{U}_{\text{P}} \in \mathbb{R}^{16\times16}$, $\mathbf{U}_{\text{V}} \in \mathbb{R}^{7\times7}$, and $\mathbf{U}_{\text{L}} \in \mathbb{R}^{20\times2}$. Since $R_{\text{P}} = 16$, in the first instance we retain $16 \times 7 \times 1$ TensorFaces, while in the second we retain $16 \times 7 \times 2$ TensorFaces, each time equaling the number of retained eigenfaces. Note that the total number of coefficients representing the compressed images is $16 + 7 + 1$ and $16 + 7 + 2$, respectively. For the compressed images of the male, the root mean squared errors (RMSE) relative to the original image, which are indicated in the figure, are larger for the TensorFaces compressions than they are for the PCA compressions. However, the "*perceptual error*" of the TensorFaces compressions are significantly smaller, yielding substantially better image quality than PCA in subspaces of comparable dimension.

## 6.4 The Independent TensorFaces Representation

We will now demonstrate the application of our multilinear independent components analysis (MICA) method (Section 4.3) to the image dataset in Figure 6.6.

Our approach performs a multilinear ICA decomposition of the tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{8560\times75\times6\times6}$ of vec-

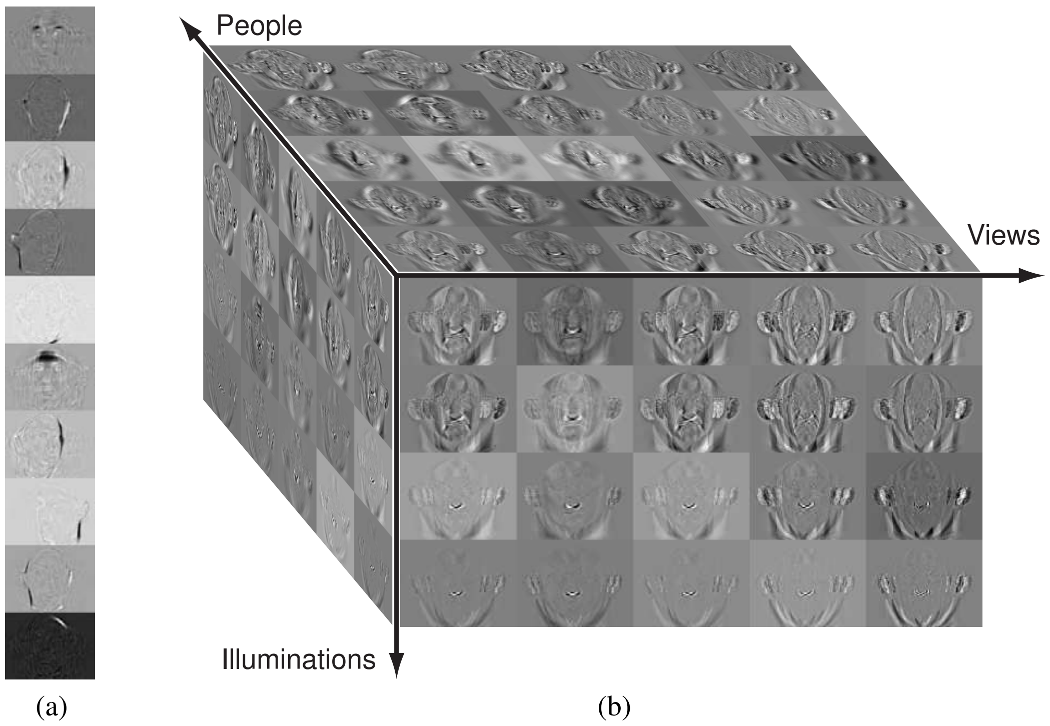


*Figure 6.11: ICA and MICA bases for an ensemble of 2,700 8,560-pixel facial images (Figure 6.6) comprising 75 people, each imaged under 6 viewing and 6 illumination conditions. (a) Independent components $\mathbf{C}_{\text{x}}$. (b) A partial visualization of the MICA representation of $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{8560\times75\times6\times6}$, obtained as $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{S}} \times_{\text{x}} \mathbf{C}_{\text{x}}$.*

torized training images per (4.17):

$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{S}} \times_{\text{x}} \mathbf{C}_{\text{x}} \times_{\text{P}} \mathbf{C}_{\text{P}} \times_{\text{V}} \mathbf{C}_{\text{V}} \times_{\text{L}} \mathbf{C}_{\text{L}}, \tag{6.5}$$

thereby extracting a set of causal factor mode matrices—the matrix $\mathbf{C}_{\text{P}} \in \mathbb{R}^{75\times75}$ whose column vectors span the people space, the matrix $\mathbf{C}_{\text{V}} \in \mathbb{R}^{6\times6}$ whose column vectors span the view space, and the matrix $\mathbf{C}_{\text{L}} \in \mathbb{R}^{6\times6}$ whose column vectors span the illumination space—and a MICA basis tensor

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{S}} \times_{\text{x}} \boldsymbol{\mathcal{C}}_{\text{x}}, \tag{6.6}$$

which we can call *Independent TensorFaces*, that governs the interaction between the different mode matrices. The independent TensorFaces representation is illustrated in Figure 6.11(b). For comparison, the ordinary ICA basis vectors $\mathbf{C}_{\text{x}}$ are shown in Figure 6.11(a).

## 6.5 Structure and Decomposition of Facial Image Manifolds

Manifold learning methods aspire to recover a low-dimensional parameterization associated with a low-dimensional submanifold that is embedded in a high-dimensional measurement space. PCA or multidimensional scaling (MDS) extract a linear submanifold that is embedded in the measurement space.

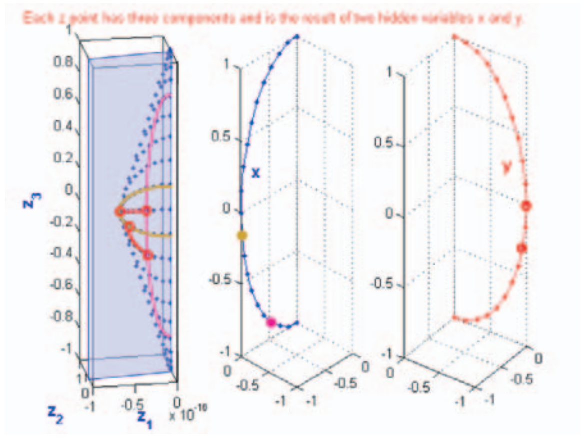


*Figure 6.12:* $\mathbf{z}(\mathbf{x}, \mathbf{y})$, *a function of two hidden variables,* $\mathbf{x}$ *and* $\mathbf{y}$, *is sampled to generate measurement data confined to a plane in 3D space. Standard manifold learning methods will miss the curvilinear structure of the data. One should extract the underlying variables and separately map each of their manifolds.*

When densely sampled training data are available, techniques such as LLE [Roweis and Saul 2000] and Isomap [Tenenbaum et al. 2000] can recover the intrinsic geometry of a nonlinear submanifold. Although the well-known toy examples, such as the "swiss roll" spiral [Tenenbaum et al. 2000], that have been employed to illustrate the latter methods have visually pleasing geometric structures, they tend to be gross over-simplifications of the manifold structure of real measurement data. In important real-world situations, particularly involving multimodal measurement data, the data manifold of interest is actually the result of the interaction of several component manifolds. Furthermore, these multiple manifolds can be embedded into one another such that the measurement data points that lie on one manifold are not clearly separated from those lying on the other manifolds, thus destroying any recognizable geometric structure, especially when the measurement data are very sparse.

Figure 6.12 displays a toy example in which the measurement data points were generated by the interaction of two nonlinear component manifolds. The resulting planar manifold is the result of the interaction of two different factors, where each factor lies on a separate manifold that has a low-dimensional parameterization. Clearly, in such a scenario, unsupervised manifold learning methods such as the aforementioned ones, lack the power to extract a meaningful parameterization.

For a non-toy example, consider facial image ensembles that sparsely sample several constituent manifolds; e.g., the view manifold, the illumination manifold, the person manifold, and the expression manifold. These manifolds interact with one another nontrivially, yielding a seemingly unstructured cloud of measurement data. In Figure 6.13, we display a slice through the data cloud. This slice lies on a subspace that is made up of several manifolds which are embedded within each other, with the data points lying on one manifold *not* clearly separated from the data points on other manifolds.

Our multilinear analysis can be regarded as a *manifold decomposition* approach that extracts from a compound manifold constituent component manifolds, each of which can separately be described by low dimensional parameterizations. Unlike the aforementioned methods, our method is supervised, but

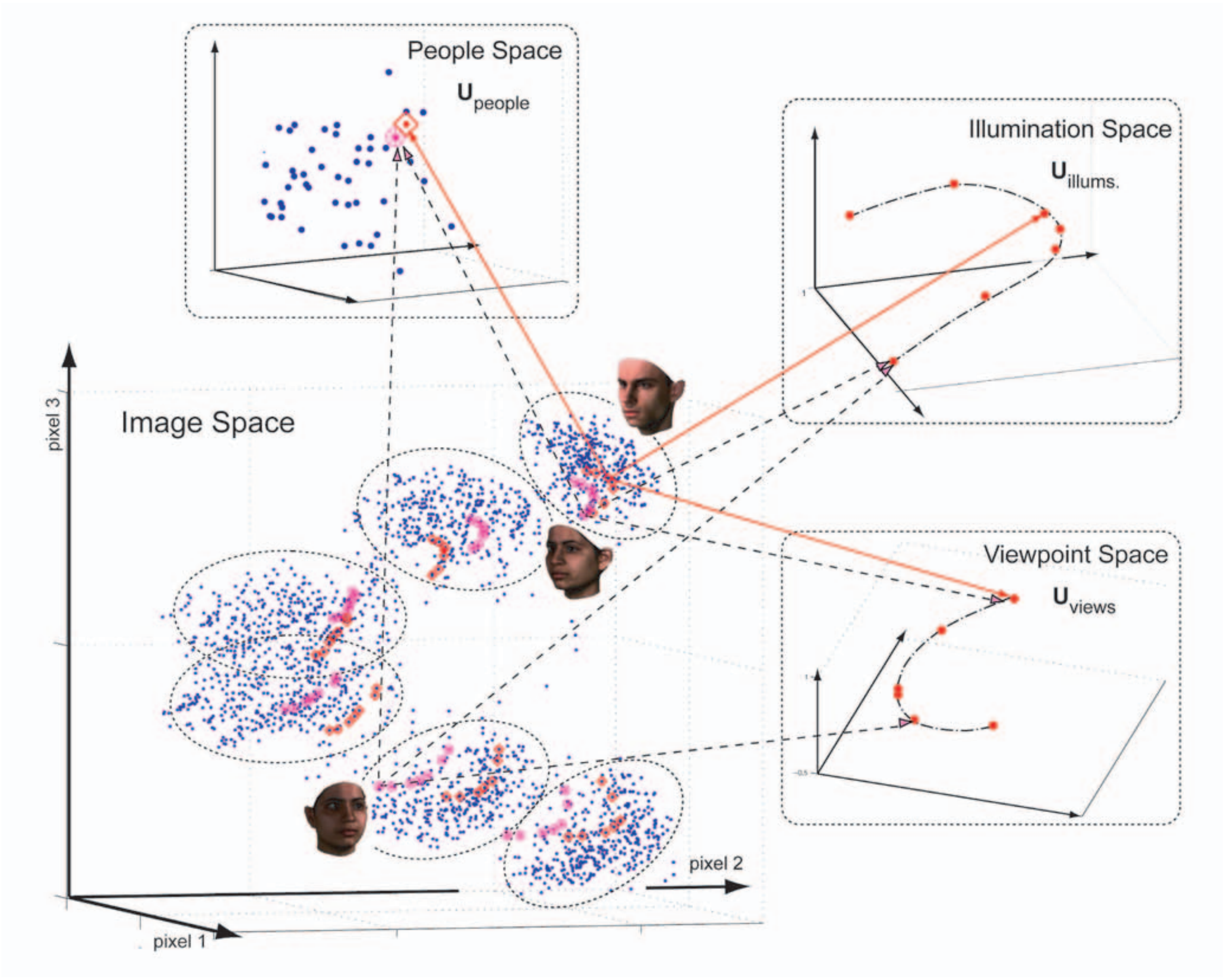


*Figure 6.13: Manifold structure of facial image ensembles. Each blue dot in the image space represents an image. A total of 1050 images are represented. (Each image is represented by its three most representative pixels.) 25 people were photographed under 7 illumination conditions and from 6 viewing directions. The images analyzed form 6 major clusters, where a cluster contains all the images photographed from the same viewing direction. As shown in the above diagram, within the image space, the image clusters are organized into a semi-circular structure that can be parameterized by the viewing direction. Similarly, within each cluster, images for each individual are also organized into a semi-circular structure that can be parameterized by the illumination direction. The red diamonds are images of one person under various imaging conditions. Similarly, the purple circles are images of a second person under various imaging conditions.*

it can decompose compound data manifolds that are very sparsely sampled. The parameterizations of the extracted component submanifolds can then be learned using methods such as LLE and Isomap [Vasilescu 2006a].

Given the image ensemble, our method can map each image from the measurement (pixel) space simultaneously into three other constituent spaces—the person space, the view space, and the illumination space. Hence, it can discover the semi-circular structure associated with the view manifold and the semi-circular structure associated with the illumination manifold embedded in the people/view subspace. Each constituent manifold can then be parameterized by its own specific low-dimensional parameterization.

# 7 Multilinear Image Recognition: The Multilinear Projection

## Contents

## Algorithms

We will now apply our multilinear framework to pattern recognition. Facial image recognition is a classic problem in the field, and *appearance-based face recognition* has been an active area of biometrics research in recent years [Chellappa et al. 1995]. Given a database of suitably labeled training images of numerous individuals, this supervised recognition technique aspires either to recognize the faces of these individuals in previously unseen test (probe) images or to identify the test images as new faces. The conventional approach addresses the problem by taking advantage of linear algebra. In particular,

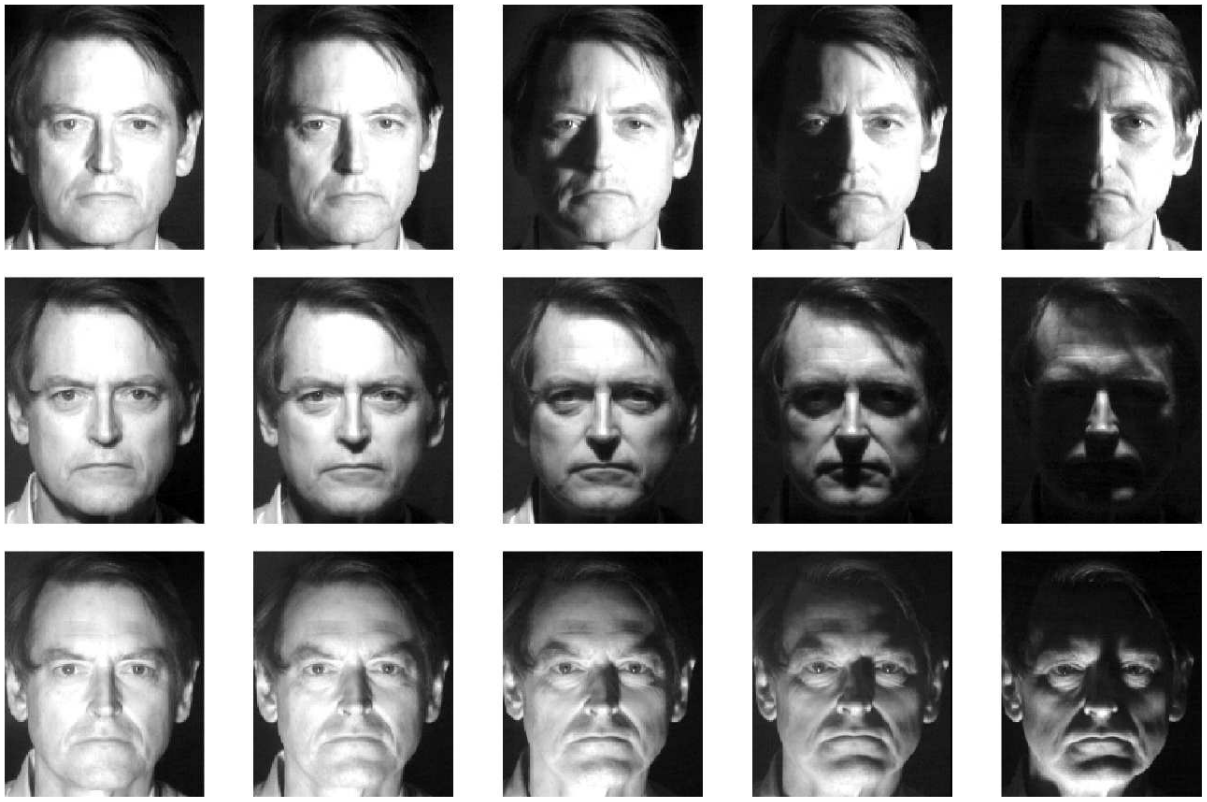


*Figure 7.1: A face imaged under multiple illumination conditions. The image appearance changes dramatically. (From [Belhumeur et al. 1997].)*

PCA (a.k.a. "eigenfaces") has been the dominant method for appearance-based facial image recognition [Sirovich and Kirby 1987; Turk and Pentland 1991a]. This linear method and its variants adequately address face recognition under constrained conditions—e.g., frontal mugshots, fixed lightsources, fixed expression—where person identity is the only causal factor that is allowed to vary.

However, people possess a remarkable ability to recognize faces under unconstrained conditions; that is, despite changes in view, illumination, and facial expression (i.e., facial geometry). Figure 7.1 illustrates the dramatic change in the image that is possible under varying illumination conditions. Sometimes two images of the same face acquired under different illumination conditions can differ more dramatically than images of two different faces acquired under the same lighting condition.

Our work confronts the challenge of learning tractable, nonlinear models of image ensembles useful in difficult appearance-based recognition problems, such as facial recognition under varying conditions. Our multilinear method is able to deal with variation in multiple causal factors; hence, it can recognize faces under unconstrained pose, illumination, and expression (PIE) conditions. In facial recognition scenarios that involve varying view and illumination, multilinear recognition algorithms based on TensorFaces yield dramatically improved recognition rates relative to standard, linear recognition algorithms based on eigenfaces and ICA.

We propose two approaches for recognition in the context of our multilinear framework. The first approach employs a set of linear projections. In the second approach, which is significantly better, we introduce a multilinear (tensor) projection operator that generalizes the linear projections.

Figure 1.4 illustrates the architecture of a facial image recognition system within our multilinear framework. Appendix B discusses some important issues regarding the acquisition of facial image datasets for the purposes of training multilinear recognition algorithms.

## 7.1 Multiple Linear Projections (MLP)

For concreteness, consider the Weizmann facial image database from Section 6.1, which comprises 60 images per person that vary with view, illumination, and expression. PCA represents each person as a set of 45 vector-valued coefficients, one from each image in which the person appears. The length of each PCA coefficient vector is $28 \times 5 \times 3 \times 3 = 1260$. By contrast, multilinear analysis enables us to represent each person, regardless of view, illumination, and expression, with the same coefficient vector of dimension 28 relative to the bases comprising the tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{7943\times28\times5\times3\times3}$, where

$$\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{Z}} \times_{\text{x}} \mathbf{U}_{\text{x}} \times_{\text{V}} \mathbf{U}_{\text{V}} \times_{\text{L}} \mathbf{U}_{\text{L}} \times_{\text{E}} \mathbf{U}_{\text{E}} \tag{7.1}$$

$$= \boldsymbol{\mathcal{T}} \times_{\text{V}} \mathbf{U}_{\text{V}} \times_{\text{L}} \mathbf{U}_{\text{L}} \times_{\text{E}} \mathbf{U}_{\text{E}}. \tag{7.2}$$

some of which are shown in Figure 7.2. Each column in the figure is a basis matrix that comprises 28 basis vectors. In any column, the first basis vector depicts the average person and the remaining basis vectors capture the variability over people, for the particular combination of view, illumination, and expression associated with that column. Each image is represented by a set of coefficient vectors for the person, view, illumination, and expression causal factors that generated the image. This is an important distinction between our multilinear PCA approach to facial recognition and that for linear PCA.

In the PCA or eigenfaces technique, one decomposes a data matrix $\mathbf{D}$ of labeled training facial images $\mathbf{d}_i$ into a reduced-dimensional basis matrix $\mathbf{B}_{\text{PCA}} = \mathbf{U}_{\text{x}}\boldsymbol{\Sigma}$ and a coefficient matrix $\mathbf{U}_{\text{I}}$ whose rows are coefficient vectors $\mathbf{c}_i^{\text{T}}$ associated with each vectorized image $\mathbf{d}_i$. Given an unlabeled test image (probe) $\mathbf{d}$, the projection operator $\mathbf{B}^{+}_{\text{PCA}}$ linearly projects this $\mathbf{d}$ into the reduced-dimensional space of image coefficient vectors. The nearest coefficient vector may be used to label (i.e., recognize) probe $\mathbf{d}$.

By contrast, our facial recognition algorithm performs the TensorFaces decomposition of the tensor $\boldsymbol{\mathcal{D}}$ of vectorized training images $\mathbf{d}_{pvle}$, extracts the matrix $\mathbf{U}_{\text{P}}$, which contains row vectors $\mathbf{p}_p^{\text{T}}$ of coefficients for each person $p$, and constructs the basis tensor $\boldsymbol{\mathcal{B}}$, which contains a collection of basis matrices, which are view $v$, illumination $l$, and expression $e$ specific. We can think of the $\mathbf{p}_p^{\text{T}}$ as *person signatures*.

For the Weizmann database example, we index into the basis tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{7943\times28\times5\times3\times3}$ for a particular view $v$, illumination $l$, and expression $e$ to obtain a subtensor $\boldsymbol{\mathcal{B}}_{vle} \in \mathbb{R}^{7943\times28\times1\times1\times1}$. We matrixize $\boldsymbol{\mathcal{B}}_{vle}$ in the people mode to obtain the matrix $\mathbf{B}_{vle[\text{P}]} \in \mathbb{R}^{28\times7943}$. Note that a specific training image $\mathbf{d}_{pvle} \in \mathbb{R}^{7943\times1\times1\times1\times1}$ of person $p$ in view $v$, illumination $l$, and expression $e$ can be written as

$$\mathbf{d}_{pvle} = \boldsymbol{\mathcal{B}}_{vle} \times_{\text{P}} \mathbf{p}_p^{\text{T}} \tag{7.3}$$

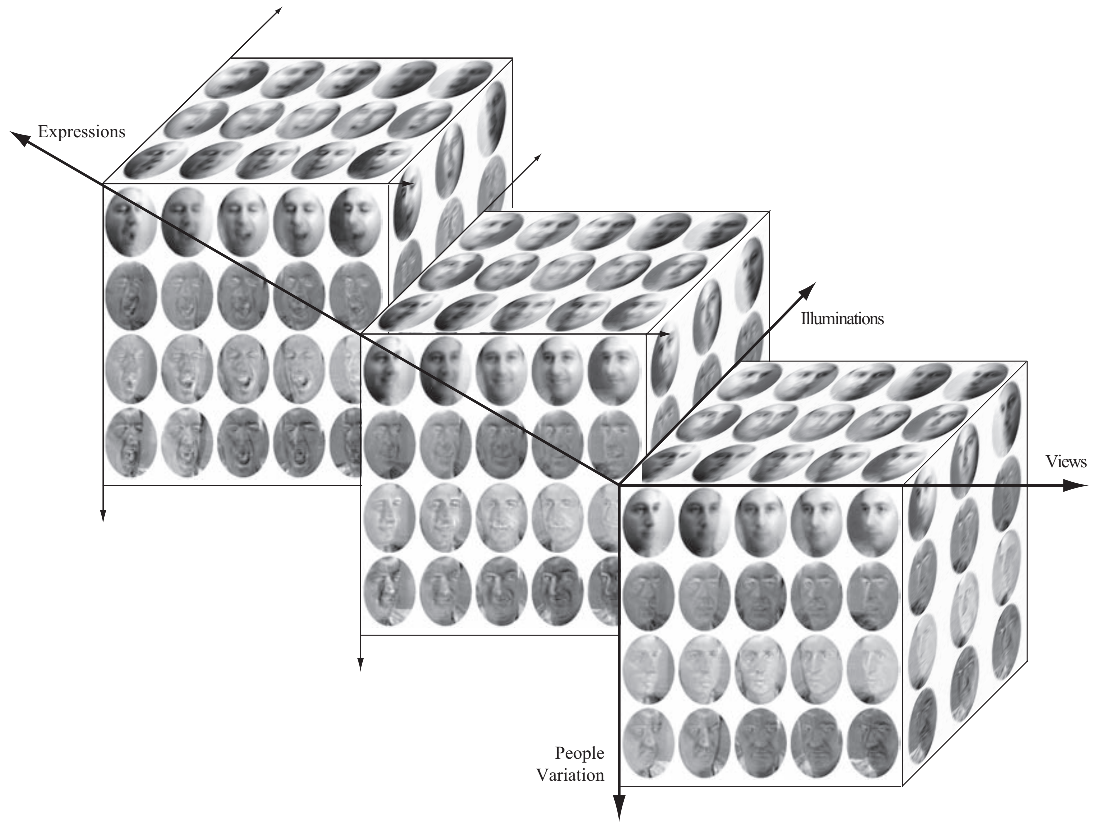


*Figure 7.2: A partial visualization of the bases tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{7943\times 28\times 5\times 3\times 3} = \boldsymbol{\mathcal{T}} \times_{\text{V}} \mathbf{U}_{\text{V}} \times_{\text{L}} \mathbf{U}_{\text{L}} \times_{\text{E}} \mathbf{U}_{\text{E}}$ (only the subtensor associated with the neutral expression is shown), which defines 45 different bases for each combination of views, illuminations and expressions. These bases have 28 basis vectors that span the people space. The basis vectors in any particular row play the same role in each column. The topmost plane depicts the average person, while the basis vectors in the remaining planes capture the variability across people in the various view, illumination, and expression combinations.*

or, using (3.12), in matrix form as

$$\mathbf{d}_{pvle_{[\text{P}]}} = \mathbf{p}_p^{\text{T}} \mathbf{B}_{vle\,[\text{P}]}; \tag{7.4}$$

hence,

$$\mathbf{p}_p = \mathbf{B}_{vle_{[\text{P}]}}^{+\text{T}} \mathbf{d}_{pvle_{[\text{P}]}}^{\text{T}} \tag{7.5}$$

$$= \mathbf{B}_{vle_{[\text{P}]}}^{+\text{T}} \mathbf{d}_{pvle}. \tag{7.6}$$

Now, given an unlabeled test image $\mathbf{d}$, we can use the projection operator $\mathbf{B}_{vle_{[\text{P}]}}^{+\text{T}}$ to project $\mathbf{d}$ into a set of candidate coefficient vectors

$$\mathbf{r}_{vle} = \mathbf{B}_{vle_{[\text{P}]}}^{+\text{T}} \mathbf{d} \tag{7.7}$$

for every $v$, $l$, $e$ combination. Our recognition algorithm compares every $\mathbf{r}_{vle}$ against the person signatures $\mathbf{p}_p^{\mathrm{T}}$. The best matching $\mathbf{r}_{vle}$ and $\mathbf{p}_p$ vector pair, i.e., those vectors that yield the smallest value of

$$\arg\min_{p;\,vle} \|\mathbf{p}_p - \mathbf{r}_{vle}\|, \tag{7.8}$$

result in the unlabeled test image $\mathbf{d}$ acquiring the person label of person signature $\mathbf{p}_p$ and the view, illumination, and expression labels of $\mathbf{r}_{vle}$, thereby identifying $\mathbf{d}$ as portraying person $p$ in view $v$, illumination $l$, and expression $e$.

## 7.2 The Multilinear Projection (MP)

The TensorFaces recognition algorithm proposed in the previous section was based on an approach involving multiple linear projections. It computed a set of linear projection operators, which yielded a set of candidate coefficient vectors. However, this is less than ideal. We will now develop a multilinear projection method that simultaneously infers the identity, view, illumination, etc., coefficient vectors of an unlabeled test image. Our multilinear projection maps an unlabeled test image from the measurement, pixel space to the multiple causal factor spaces (Figure 6.13). We thus obtain an improved recognition algorithm that fully exploits the multilinear structure of our tensor framework. For concreteness, we will continue the development of this superior approach in the context of the facial image dataset of Figure 6.6.

Given the data tensor $\boldsymbol{\mathcal{D}}$ of labeled, vectorized training images $\mathbf{d}_{pvle}$, where the subscripts denote person $p$, view $v$, illumination $l$, and expression $e$ labels, we can apply the MPCA algorithm (Algorithm 4.2.1) to compute causal mode matrices $\mathbf{U}_{\mathrm{P}}$, $\mathbf{U}_{\mathrm{V}}$, $\mathbf{U}_{\mathrm{L}}$, and $\mathbf{U}_{\mathrm{E}}$ as well as the TensorFaces basis $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{D}} \times_{\mathrm{P}} \mathbf{U}_{\mathrm{P}}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{U}_{\mathrm{V}}^{\mathrm{T}} \times_{\mathrm{L}} \mathbf{U}_{\mathrm{L}}^{\mathrm{T}} \times_{\mathrm{E}} \mathbf{U}_{\mathrm{E}}^{\mathrm{T}}$ (6.2) that governs the interaction between them (Figure 6.2(b)). Then the method represents an image $\mathbf{d}_{pvl}$ by the relevant set of person, view, and illumination coefficient vectors as follows:

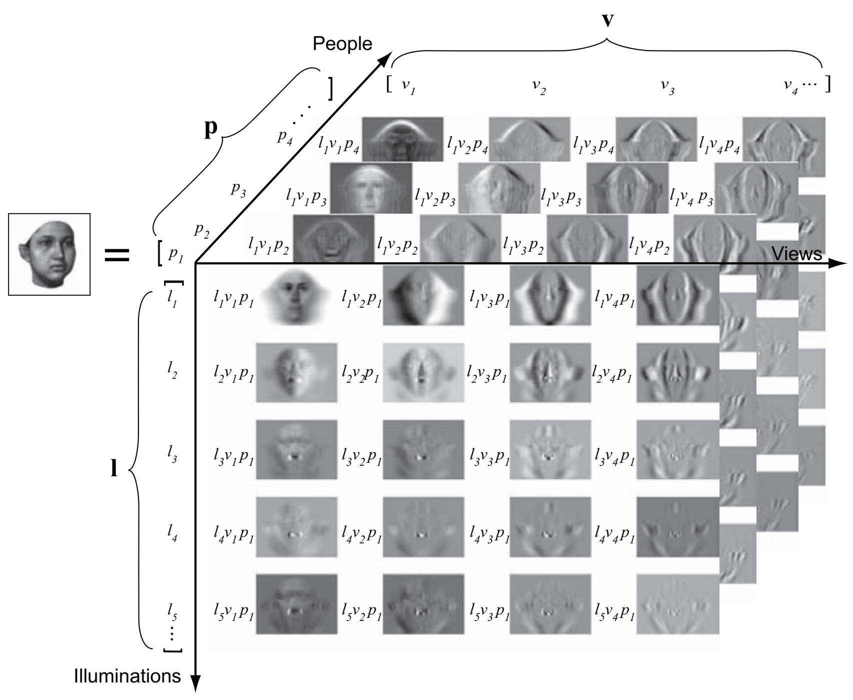
$$\mathbf{d}_{pvle} = \boldsymbol{\mathcal{T}} \times_{\mathrm{P}} \mathbf{p}_p^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}_v^{\mathrm{T}} \times_{\mathrm{L}} \mathbf{l}_l^{\mathrm{T}} \times_{\mathrm{E}} \mathbf{e}_e^{\mathrm{T}}. \tag{7.9}$$

This multilinear representation is illustrated in Figure 7.3(a) for the data tensor in Figure 6.7, which lacks the expression mode. Alternatively, we can apply the MICA algorithm (Algorithm 4.3.1), which employs higher-order statistics to compute a basis tensor $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{D}} \times_{\mathrm{P}} \mathbf{C}_{\mathrm{P}}^{+} \times_{\mathrm{V}} \mathbf{C}_{\mathrm{V}}^{+} \times_{\mathrm{L}} \mathbf{C}_{\mathrm{L}}^{+} \times_{\mathrm{E}} \mathbf{C}_{\mathrm{E}}^{+}$. Analogous to the MPCA case, an image can be represented with respect to the MICA basis, as follows:

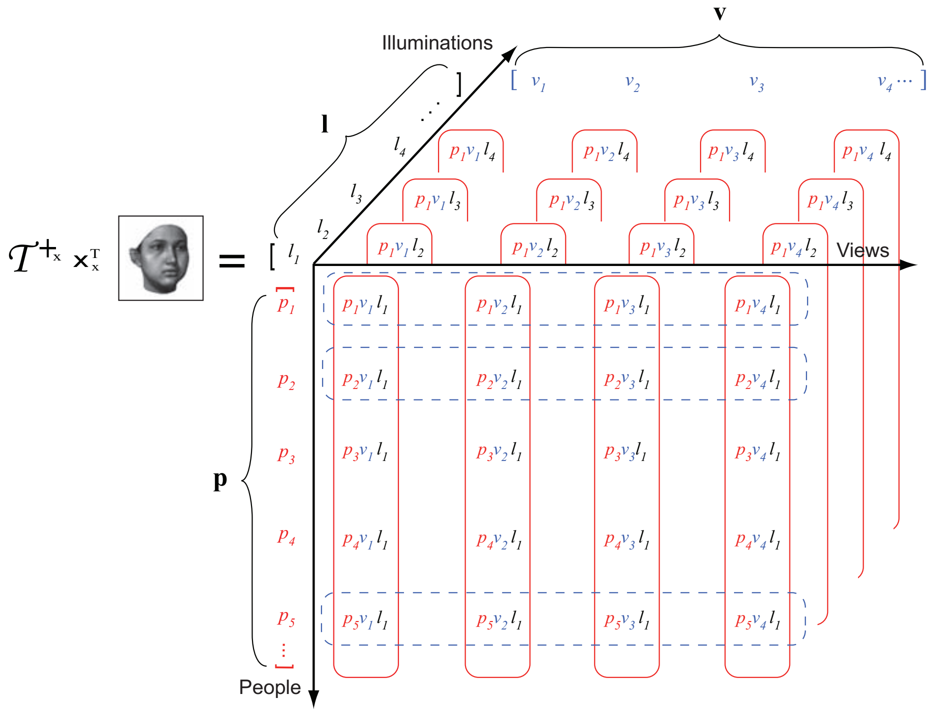
$$\mathbf{d}_{pvle} = \boldsymbol{\mathcal{M}} \times_{\mathrm{P}} \mathbf{p}_p^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}_v^{\mathrm{T}} \times_{\mathrm{L}} \mathbf{l}_l^{\mathrm{T}} \times_{\mathrm{E}} \mathbf{e}_e^{\mathrm{T}}, \tag{7.10}$$

which again for the data tensor in Figure 6.7 is illustrated in Figure 7.4.

Given an unlabeled test image (probe) $\mathbf{d}$ and $\boldsymbol{\mathcal{T}}$ or $\boldsymbol{\mathcal{M}}$, we must determine the unknown coefficient vectors, $\mathbf{p}_p$, $\mathbf{v}_v$, $\mathbf{l}_l$, and $\mathbf{e}_e$ in order to recognize the person, view, illumination, and expression associated with the test image. Solving for these vectors in (7.9) or (7.10) will, in principle, require the computation of a pseudo-inverse of tensor $\boldsymbol{\mathcal{T}}$ or $\boldsymbol{\mathcal{M}}$. In analogy with matrix algebra, this raises the following questions: How does one "invert" a tensor? When one "multiplies" a tensor with its "inverse tensor",

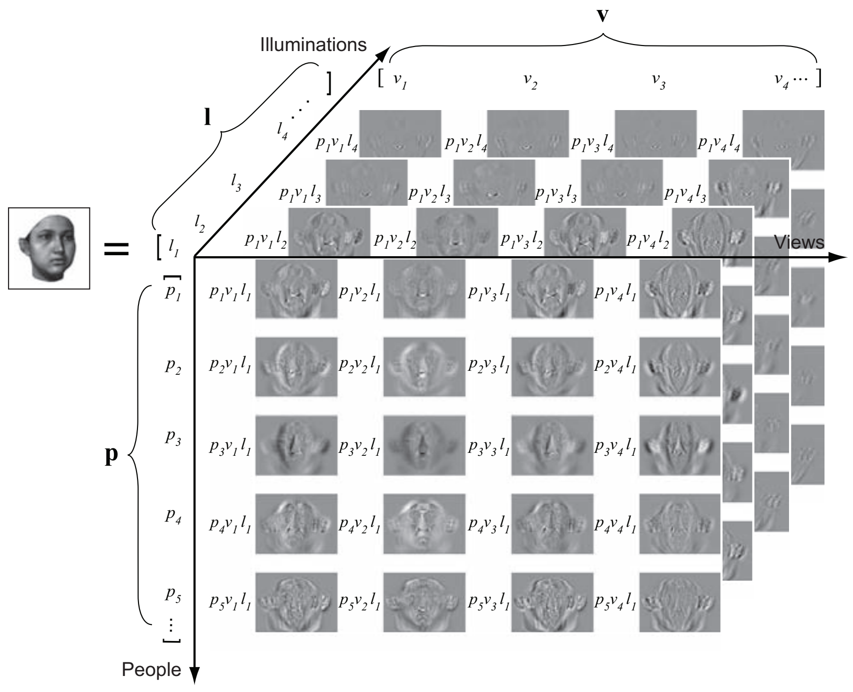


(a)

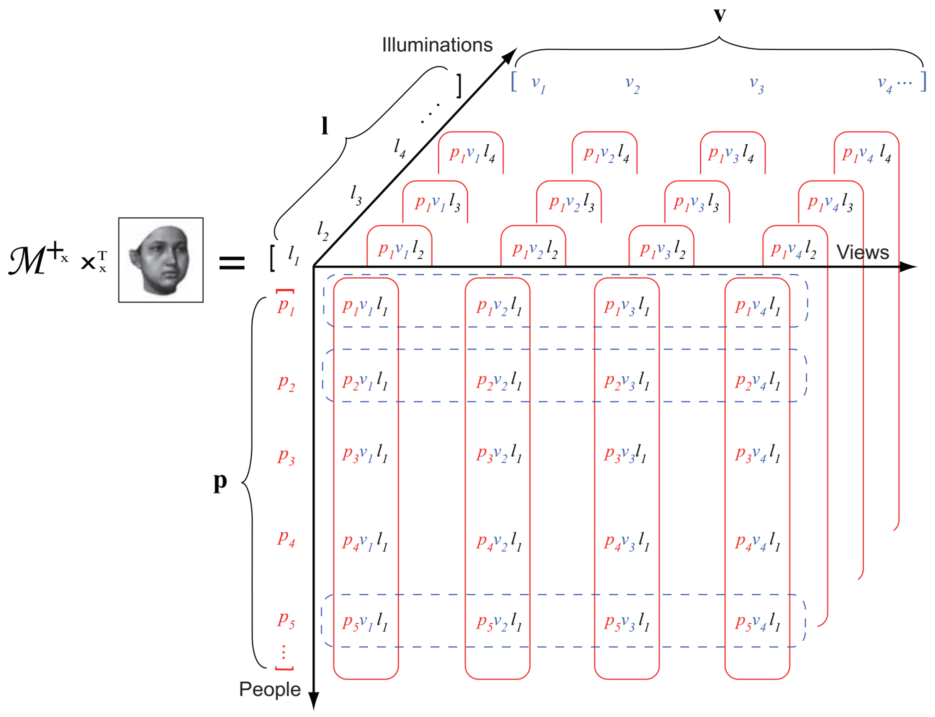


(b)

*Figure 7.3: (a) MPCA image representation* $\mathbf{d} = \mathcal{T} \times_{\text{P}} \mathbf{p}^{\text{T}} \times_{\text{V}} \mathbf{v}^{\text{T}} \times_{\text{L}} \mathbf{l}^{\text{T}}$. *(b) Given an unlabeled test image* $\mathbf{d}$*, the associated coefficient vectors* $\mathbf{p}$, $\mathbf{v}$, $\mathbf{l}$ *are estimated by decomposing the response tensor* $\mathcal{R} = \mathcal{T}^{+_{\text{x}}} \times_{\text{x}}^{\text{T}} \mathbf{d}$ *using a multilinear projection algorithm.*



(a)



(b)

*Figure 7.4: (a) MICA image representation* $\mathbf{d} = \boldsymbol{\mathcal{M}} \times_{\mathrm{P}} \mathbf{p}^{\mathrm{T}} \times_{\mathrm{V}} \mathbf{v}^{\mathrm{T}} \times_{\mathrm{L}} \mathbf{l}^{\mathrm{T}}$*. (b) Given an unlabeled test image* $\mathbf{d}$*, the associated coefficient vectors* $\mathbf{p}$, $\mathbf{v}$, $\mathbf{l}$ *are estimated by decomposing the response tensor* $\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{M}}^{+_{\mathrm{x}}} \times_{\mathrm{x}}^{\mathrm{T}} \mathbf{d}$ *using a multilinear projection algorithm.*

what should be the resulting "identity tensor"? We will next show that an $M^{\text{th}}$-order tensor has $M$ *pseudo-inverse tensors*, one with respect to each mode, and that there are $M$ *identity tensors*, one per mode, whose structure is not diagonal with ones along the main diagonal.

### 7.2.1 Identity and Pseudo-Inverse Tensors

First, we generalize Definition 3.2.6 of the mode-$m$ product of a tensor and a matrix to two tensors:[1]

**Definition 7.2.1 (Generalized Mode-$m$ Product)** *The generalized mode-$m$ product between two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_m \times \ldots \times I_M}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_m \times \ldots \times J_M}$ is expressed as follows:*

- $\boldsymbol{\mathcal{A}} \times_m \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times J_m \times I_{m+1} \times \ldots \times I_M}$*, where* $I_m = J_1 \ldots J_{m-1} J_{m+1} \ldots J_M$*, can be expressed in matrix form as* $\mathbf{C}_{[m]} = \mathbf{B}_{[m]} \mathbf{A}_{[m]}$.
- $\boldsymbol{\mathcal{A}} \times_m^{\mathrm{T}} \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times K_m \times I_{m+1} \times \ldots \times I_M}$*, where* $K_m = J_1 \ldots J_{m-1} J_{m+1} \ldots J_M$ *and* $I_m = J_m$*, can be expressed in matrix form as* $\mathbf{C}_{[m]} = \mathbf{B}_{[m]}^{\mathrm{T}} \mathbf{A}_{[m]}$.
- $\boldsymbol{\mathcal{A}}^{\mathrm{T}} \times_m \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_m \times J_m}$*, where* $I_1 \ldots I_{m-1} I_{m+1} \ldots I_M = J_1 \ldots J_{m-1} J_{m+1} \ldots J_M$*, can be expressed in matrix form as* $\mathbf{C}_{[m]}^{\mathrm{T}} = \mathbf{B}_{[m]} \mathbf{A}_{[m]}^{\mathrm{T}}$.
- $\boldsymbol{\mathcal{A}}\ {}^{\mathrm{T}}\times_m^{\mathrm{T}} \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{C}} \in \mathbb{R}^{J_1 \times \ldots \times J_{m-1} \times I_m \times J_{m+1} \times \ldots \times J_M}$*, where* $J_m = I_1 \ldots I_{m-1} I_{m+1} \ldots I_M$*, can be expressed in matrix form as* $\mathbf{C}_{[m]}^{\mathrm{T}} = \mathbf{B}_{[m]}^{\mathrm{T}} \mathbf{A}_{[m]}^{\mathrm{T}}$.

■

With the above generalization, we define the mode-$m$ identity tensor as follows:

**Definition 7.2.2 (Mode-$m$ Identity Tensor)** *Tensor $\boldsymbol{\mathcal{I}}_m$ is a mode-$m$ multiplicative identity tensor if and only if $\boldsymbol{\mathcal{I}}_m \times_m \boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}$, where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots \times I_m \times \ldots \times I_M}$ and $\boldsymbol{\mathcal{I}}_m \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times J_m \times I_{m+1} \times \ldots \times I_M}$, where $J_m = I_1 I_2 \ldots I_{m-1} I_{m+1} \ldots I_M$.* ■

While a mode-wise identity tensor might seem to be a construct peculiar to multilinear algebra, one should recall that in linear algebra there exist left and right identity matrices for every rectangular matrix $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$. Whereas the left and right identity matrices have different dimensions, they share the same diagonal structure. By contrast, the mode-$m$ identity tensors are not diagonal tensors. Figure 7.5 illustrates the structure of the three identity tensors of order 3.

The mode-$m$ identity tensor can be used to tensorize a matrix or a row vector via a mode-$m$ product. It does not change the values of the matrix/vector but simply reorders its elements. In particular, it can re-tensorize a matrix obtained by matrixizing a tensor; i.e., given a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \ldots \times I_m \times \ldots \times I_M}$ and an identity tensor, $\boldsymbol{\mathcal{I}}_m \in \mathbb{R}^{I_1 \times \ldots \times I_{m-1} \times J_m \times I_{m+1} \times \ldots \times I_M}$ with $J_m = I_1 I_2 \ldots I_{m-1} I_{m+1} \ldots I_M$, then

$$\boldsymbol{\mathcal{I}}_m \times_m \mathbf{A}_{[m]} = \boldsymbol{\mathcal{A}}. \tag{7.11}$$

[1]Note that there have been two previous attempts at such a generalization [Bader and Kolda 2007; Vasilescu and Terzopoulos 2007a], which were informal and/or incomplete.

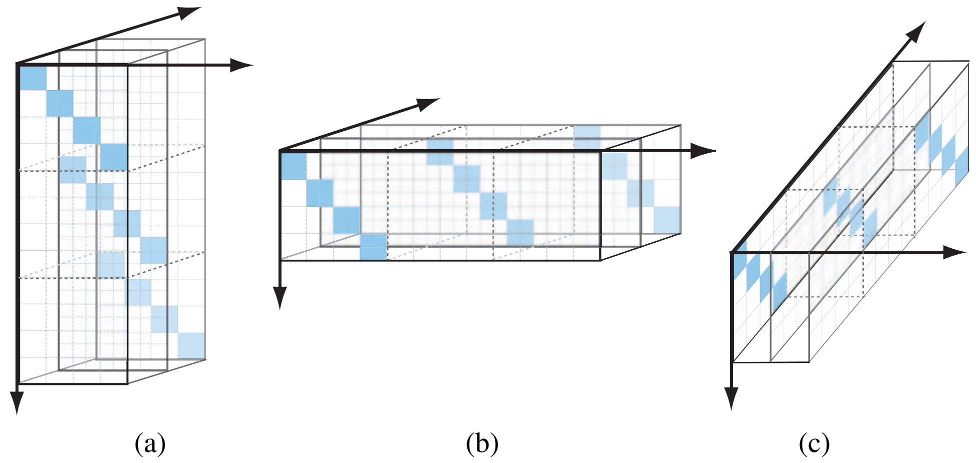


Figure 7.5: The three identity tensors of order 3; (a) mode-1 identity tensor; (b) mode-2 identity tensor; (c) mode-3 identity tensor.

We now define a mode-$m$ pseudo-inverse tensor that generalizes the pseudo-inverse matrix from linear algebra.

**Definition 7.2.3 (Mode-$m$ Pseudo-Inverse Tensor)** *The mode-$m$ pseudoinverse tensor $\mathcal{A}^{+_m}$ of tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ satisfies:*

1. $(\mathcal{A} \times^{\mathrm{T}}_m \mathcal{A}^{+_m}) \times_m \mathcal{A} = \mathcal{A}$

2. $(\mathcal{A}^{+_m}\, {}^{\mathrm{T}}\!\times_m \mathcal{A})\ {}^{\mathrm{T}}\!\times^{\mathrm{T}}_m \mathcal{A}^{+_m} = \mathcal{A}^{+_m}$

*The mode-$m$ pseudoinverse tensor $\mathcal{A}^{+_m}$ of $\mathcal{A}$ is the tensorized version of $\mathbf{A}^{+\mathrm{T}}_{[m]}$; i.e., $\mathbf{A}^{+\mathrm{T}}_{[m]} = [\mathcal{A}^{+_m}]_{[m]}$.* ■

### 7.2.2 Multilinear Projection Algorithms

To determine the coefficient vectors that represent an unlabeled observation (image), which is a point (vector) in the (pixel) measurement space, we must map the observation from the measurement space to the causal factor spaces (Figure 7.3). Given an unlabeled test (probe) image $\mathbf{d}$ and a learned TensorFaces model $\mathcal{T}$, the image is represented as follows:

$$\mathbf{d} = \mathcal{T} \times_{\mathrm{P}} \mathbf{r}^{\mathrm{T}}_{\mathrm{P}} \times_{\mathrm{V}} \mathbf{r}^{\mathrm{T}}_{\mathrm{V}} \times_{\mathrm{L}} \mathbf{r}^{\mathrm{T}}_{\mathrm{L}} \times_{\mathrm{E}} \mathbf{r}^{\mathrm{T}}_{\mathrm{E}} + \boldsymbol{\rho}, \tag{7.12}$$

where $\mathbf{d} \in \mathbb{R}^{I_{\mathrm{x}} \times 1 \times \ldots \times 1}$ and $\boldsymbol{\rho}$ is a residual vector that lies outside the range of the multilinear generative model. Thus, $\boldsymbol{\rho}$ is orthogonal to the TensorFaces basis $\mathcal{T}$ and $\boldsymbol{\rho} = \mathbf{0}$ when $\mathbf{d}$ lies in the subspace spanned by the basis. Thus, to compute the coefficient vector representations, $\mathbf{r}_{\mathrm{P}}$, $\mathbf{r}_{\mathrm{V}}$, $\mathbf{r}_{\mathrm{L}}$, and $\mathbf{r}_{\mathrm{E}}$, needed to recognize the person, view, illumination, and expression depicted in test image $\mathbf{d}$, we must pseudo-invert $\mathcal{T}$ with respect to the (pixel) measurement mode—i.e., compute $\mathcal{T}^{+_{\mathrm{x}}}$.

In view of the above considerations, we will now derive a general multilinear projection algorithm. To this end, we will temporarily revert back to numbering modes for full generality and assume that mode 1 is the measurement (e.g., pixel) mode. The general, $M$-mode form of (7.12) is

$$\mathbf{d} = \boldsymbol{\mathcal{T}} \times_2 \mathbf{r}_2^{\mathrm{T}} \ldots \times_M \mathbf{r}_M^{\mathrm{T}} + \boldsymbol{\rho}. \tag{7.13}$$

Performing a mode-1 product of both sides of this equation by the mode-1 pseudo-inverse of the TensorFaces bases, we obtain a *response tensor*

$$\begin{aligned}
\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{T}}^{+_1} \times_1^{\mathrm{T}} \mathbf{d} &= \boldsymbol{\mathcal{T}}^{+_1} \times_1^{\mathrm{T}} (\boldsymbol{\mathcal{T}} \times_2 \mathbf{r}_2^{\mathrm{T}} \ldots \times_m \mathbf{r}_m^{\mathrm{T}} \ldots \times_M \mathbf{r}_M^{\mathrm{T}} + \boldsymbol{\rho}) && (7.14)\\
&= (\boldsymbol{\mathcal{T}}^{+_1} \times_1^{\mathrm{T}} \boldsymbol{\mathcal{T}}) \times_2 \mathbf{r}_2^{\mathrm{T}} \ldots \times_m \mathbf{r}_m^{\mathrm{T}} \ldots \times_M \mathbf{r}_M^{\mathrm{T}} + (\boldsymbol{\mathcal{T}}^{+_1} \times_1^{\mathrm{T}} \boldsymbol{\rho}) && (7.15)\\
&\simeq \boldsymbol{\mathcal{I}}_1 \times_2 \mathbf{r}_2^{\mathrm{T}} \ldots \times_m \mathbf{r}_m^{\mathrm{T}} \ldots \times_M \mathbf{c}_M^{\mathrm{T}} + \mathbf{0} && (7.16)\\
&= \boldsymbol{\mathit{1}} \times_2 \mathbf{r}_2 \ldots \times_m \mathbf{r}_m \ldots \times_M \mathbf{r}_M \qquad \text{Rank-}(1,\ldots,1) && (7.17)\\
&= \mathbf{r}_2 \circ \mathbf{r}_3 \ldots \circ \mathbf{r}_M, \qquad \text{Rank-1} && (7.18)
\end{aligned}$$

where $\mathbf{d} \in \mathbb{R}^{I_1 \times 1 \times \ldots \times 1}$ and $\mathbf{d}_{[1]}^{\mathrm{T}} = \mathbf{d}^{\mathrm{T}}$, where $(\boldsymbol{\mathcal{T}}^{+_1} \times_1^{\mathrm{T}} \boldsymbol{\mathcal{T}}) \simeq \boldsymbol{\mathcal{I}}_1$ when $I_1 < I_2 I_3 \ldots I_M$, otherwise $(\boldsymbol{\mathcal{T}}^{+_1} \times_1^{\mathrm{T}} \boldsymbol{\mathcal{T}}) = \boldsymbol{\mathcal{I}}_1$, and where $\boldsymbol{\mathcal{I}}_1 \in \mathbb{R}^{(I_2 I_3 \ldots I_M) \times I_2 \times \ldots \times I_M}$. The three equalities (7.16)-(7.18) can be derived using the definition of the mode-$m$ product (3.18) and the vec-Kronecker property $\mathrm{vec}(\mathbf{a} \circ \mathbf{b}) = \mathrm{vec}(\mathbf{a}\mathbf{b}^{\mathrm{T}}) = \mathbf{b} \otimes \mathbf{a}$. The rank-1/rank-$(1,\ldots,1)$ structure of the response tensor $\boldsymbol{\mathcal{R}}$ is amenable to a tensor decomposition using the MPCA algorithm or a modified CP algorithm in order to determine the $\mathbf{r}_m$ coefficient vector representations.

We will first develop a multilinear projection (MP) algorithm that computes the coefficient vectors by applying the MPCA algorithm (Algorithm 4.2.1). Since, in principle, the mode-$m$ vectors of $\boldsymbol{\mathcal{R}}$ are multiples of the $\mathbf{r}_m$ coefficient vectors (e.g., for facial image recognition, $\mathbf{r}_{\mathrm{P}}$, $\mathbf{r}_{\mathrm{V}}$, $\mathbf{r}_{\mathrm{L}}$, $\mathbf{r}_{\mathrm{E}}$; cf. the framed rows/columns in Figure 7.3(b)), matrixizing $\boldsymbol{\mathcal{R}}$ in each mode yields rank-1 matrices, enabling the MPCA algorithm to compute the corresponding coefficient vector. The MPCA thus maps $\boldsymbol{\mathcal{R}}$ into $M-1$ different mode spaces that explicitly account for the contribution of each causal factor $2 \le m \le M$ (e.g., person, view, illumination, and expression, when dealing with facial images; in particular, the person coefficient vector $\mathbf{r}_{\mathrm{P}}$ is the leading left-singular vector of the SVD of $\mathbf{R}_{[\mathrm{P}]}$).

Algorithm 7.2.1 is a *Rank-*$(1,\ldots,1)$ *Multilinear Projection (MP) Algorithm*. It exploits the MPCA algorithm (Algorithm 4.2.1) which achieves dimensionality reduction through alternating least squares. In practice, the decomposition of $\boldsymbol{\mathcal{R}}$ may not result in a rank-$(1,\ldots,1)$ decomposition.

Alternatively, the multilinear projection algorithm can employ a modified CANDECOMP/PARAFAC (CP) algorithm to compute the best fitting rank-1 term for the response tensor. Like the MPCA algorithm, the CP algorithm takes advantage of the structure of $\boldsymbol{\mathcal{R}}$. The mode-$m$ vectors of $\boldsymbol{\mathcal{R}}$ are multiples of $\mathbf{r}_m$ and the scalar multiples have a well defined structure that the CP algorithm exploits. Given the structure of $\boldsymbol{\mathcal{R}}$, the outer product of coefficient vectors $\mathbf{r}_m$ may be expressed in matrix form as:

$$\begin{aligned}
\mathbf{R}_{[m]} &\simeq (\mathbf{r}_2 \circ \mathbf{r}_3 \ldots \circ \mathbf{r}_M)_{[m]} && (7.19)\\
&= \mathbf{r}_m (\mathbf{r}_M \otimes \ldots \otimes \mathbf{c}_{m+1} \odot \mathbf{r}_{m-1} \otimes \ldots \otimes \mathbf{r}_2)^{\mathrm{T}} && (7.20)\\
&= \mathbf{r}_m (\mathbf{r}_M \odot \ldots \odot \mathbf{r}_{m+1} \odot \mathbf{r}_{m-1} \odot \ldots \odot \mathbf{r}_2)^{\mathrm{T}} && (7.21)
\end{aligned}$$

**Algorithm 7.2.1** Multilinear projection (MP) algorithm with MPCA, rank-$(1, \dots, 1)$ decomposition

**Input** a TensorFaces basis tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \dots \times I_M}$,[a] where mode $m = 1$ is the measurement mode, and an unlabeled test observation (image) $\mathbf{d}$.

```
1. Compute the pseudo-inverse $\boldsymbol{\mathcal{T}}^{+_1}$ (in matrix form, $\mathbf{T}_{[1]}^{+\mathrm{T}}$).
2. Compute the response tensor $\boldsymbol{\mathcal{R}} := \boldsymbol{\mathcal{T}}^{+_1} \times_1 \mathbf{d}^{\mathrm{T}}$.
3. Apply the $M$-mode dimensionality reduction algorithm (Algorithm 3.2.3) to $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{1 \times I_2 \times \dots \times I_M}$
   with reduced ranks $\tilde{R}_1 := \dots := \tilde{R}_M := 1$, obtaining mode vectors $\mathbf{r}_1 = r_1, \mathbf{r}_2, \dots, \mathbf{r}_m$.
```

**Output** the causal factor representation vectors $\mathbf{r}_2 \dots \mathbf{r}_M$.

[a]Or given a MICA basis tensor $\boldsymbol{\mathcal{M}}$.

$$= \mathbf{r}_m \mathbf{y}_m^{\mathrm{T}}, \tag{7.22}$$

where $\mathbf{y}_m = (\mathbf{r}_M \odot \dots \odot \mathbf{r}_{m+1} \odot \mathbf{r}_{m-1} \odot \dots \odot \mathbf{r}_2)$. Therefore, each coefficient vector representation is given by

$$\mathbf{r}_m = \mathbf{R}_{[m]} \mathbf{y}_m (\mathbf{y}_m^{\mathrm{T}} \mathbf{y}_m)^{-1} = \mathbf{R}_{[m]} \mathbf{y}_m / \|\mathbf{y}_m\|^2. \tag{7.23}$$

Given the form of $\mathbf{y}_m$, we can compute its norm efficiently using the relationships (3.24) and (3.25) between the Khatri-Rao and Hadamard products, as follows:

$$\|\mathbf{y}_m\|^2 = \mathbf{y}_m^{\mathrm{T}} \mathbf{y}_m = \mathbf{r}_M^{\mathrm{T}} \mathbf{r}_M \circledast \dots \circledast \mathbf{r}_{m+1}^{\mathrm{T}} \mathbf{r}_{m+1} \circledast \mathbf{r}_{m-1}^{\mathrm{T}} \mathbf{r}_{m-1} \circledast \dots \circledast \mathbf{r}_2^{\mathrm{T}} \mathbf{r}_2 \tag{7.24}$$

$$= (\mathbf{r}_M^{\mathrm{T}} \mathbf{r}_M) \dots (\mathbf{r}_{m+1}^{\mathrm{T}} \mathbf{r}_{m+1})(\mathbf{r}_{m-1}^{\mathrm{T}} \mathbf{r}_{m-1}) \dots (\mathbf{r}_2^{\mathrm{T}} \mathbf{r}_2) \tag{7.25}$$

$$= \|\mathbf{r}_M\|^2 \dots \|\mathbf{r}_{m+1}\|^2 \|\mathbf{r}_{m-1}\|^2 \dots \|\mathbf{r}_2\|^2. \tag{7.26}$$

Algorithm 7.2.2 is a *multilinear projection algorithm using Rank-1 decomposition*, a modified CP algorithm (Algorithm 7.2.2), which computes the best fitting rank-1 decomposition of $\boldsymbol{\mathcal{R}}$. Each $\mathbf{r}_m$ is computed efficiently according to (7.23) with (7.26) and in an iterative manner by holding all other coefficient vectors fixed.

Note that the main difference between the MP-MPCA and MP-CP algorithms is the initialization step. The MPCA initialization does not constrain the core tensor to be rank-$(1, \dots, 1)$, whereas the CP initialization does. The latter is a better initial condition because $\boldsymbol{\mathcal{R}}$ is a rank-1 tensor.

The MP-MPCA method is sensitive to the sign indeterminacy of the decomposition of the response tensor; i.e., for any pair of factor representation vectors,

$$\boldsymbol{\mathcal{R}} \simeq \mathbf{r}_2 \circ \dots \circ \mathbf{r}_i \circ \dots \circ \mathbf{r}_j \circ \dots \circ \mathbf{r}_M \tag{7.27}$$

$$= \mathbf{r}_2 \circ \dots \circ -\mathbf{r}_i \circ \dots \circ -\mathbf{r}_j \circ \dots \circ \mathbf{r}_M, \tag{7.28}$$

and alternative decompositions can be obtained by flipping the signs of any number of vector pairs. Sign consistency in the MP-MPCA can be achieved analogously to how one might achieve consistency

**Algorithm 7.2.2** Multilinear projection (MP) algorithm with CP, rank-1 decomposition

```
Input a TensorFaces basis tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$,[a] where mode $m = 1$ is the measurement mode, and
an unlabeled test observation (image) $\mathbf{d}$.

1. Compute the pseudo-inverse $\boldsymbol{\mathcal{T}}^{+_1}$ (in matrix form, $\mathbf{T}^{+\mathrm{T}}_{[1]}$).
2. Compute the response tensor $\boldsymbol{\mathcal{R}} := \boldsymbol{\mathcal{T}}^{+_1} \times_1 \mathbf{d}^{\mathrm{T}}$.
3. Initialize $\mathbf{y}_m$ to the column norms of $\mathbf{R}_{[m]}$.
4. For $m := 2, \ldots, M$, set $\mathbf{r}_m := \mathbf{R}_{[m]}\mathbf{y}_m / \|\mathbf{y}_m\|^2$.
5. Local optimization via alternating least squares:
   Iterate for $n := 1, \ldots, N$
       For $m := 2, \ldots, M$,
           Set $\mathbf{r}_m := \mathbf{R}_{[m]} (\mathbf{r}_M \odot \ldots \odot \mathbf{r}_{m-1} \odot \mathbf{r}_{m+1} \odot \ldots \odot \mathbf{r}_2) /$
               $\|\mathbf{r}_M\|^2 \ldots \|\mathbf{r}_{m+1}\|^2 \|\mathbf{r}_{m-1}\|^2 \ldots \|\mathbf{r}_2\|^2$.
       Set $\boldsymbol{\mathcal{Z}} := \boldsymbol{\mathcal{R}} \times_2 \mathbf{r}_2^{\mathrm{T}} \ldots \times_M \mathbf{r}_M^{\mathrm{T}}$.[b]
   until convergence.[c]

Output the converged causal factor representation vectors $\mathbf{r}_2 \ldots \mathbf{r}_M$.
```

[a] Or given a MICA basis tensor $\boldsymbol{\mathcal{M}}$.
[b] Note that $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{1^M}$ is a degenerate tensor of order $M$; i.e., a scalar.
[c] See Footnote *a* in Algorithm 3.2.1.

in choosing PCA basis vectors [Bro et al. 2008]. Note, however, that the MP-CP method starts with a consistent initialization condition, so it is less prone to sign indeterminacy.

The application of Algorithm 7.2.1 or Algorithm 7.2.2 on an unlabeled test image $\mathbf{d}$ yields causal factor representation vectors $\mathbf{r}_2, \ldots, \mathbf{r}_M$. For recognition, we assign causal mode labels to $\mathbf{d}$ by computing a cosine similarity measure between $\mathbf{r}_m$ and each of the $I_m$ rows $\mathbf{c}_i^{\mathrm{T}}$ of $\mathbf{U}_m$:

$$\arg\max_i \frac{\mathbf{c}_i^{\mathrm{T}} \mathbf{r}_m}{\|\mathbf{c}_i\| \|\mathbf{r}_m\|}. \tag{7.29}$$

The probe $\mathbf{d}$ is assigned the label $i$, where $1 \leq i \leq I_m$, of the signature $\mathbf{c}_i^{\mathrm{T}}$ that maximizes (7.29). In the particular context of facial image recognition, we denote the $\mathbf{r}_m$ vectors computed by the MP algorithms as $\mathbf{r}_{\mathrm{P}}$, $\mathbf{r}_{\mathrm{V}}$, $\mathbf{r}_{\mathrm{L}}$, and $\mathbf{r}_{\mathrm{E}}$, in association with the people, views, illuminations, and expressions modes, respectively. To recognize the unlabeled test image $\mathbf{d}$, we maximize the set of similarity measures

$$\arg\max_p \frac{\mathbf{p}_p^{\mathrm{T}} \mathbf{r}_{\mathrm{P}}}{\|\mathbf{p}_p\| \|\mathbf{r}_{\mathrm{P}}\|}; \quad \arg\max_v \frac{\mathbf{v}_v^{\mathrm{T}} \mathbf{r}_{\mathrm{V}}}{\|\mathbf{v}_v\| \|\mathbf{r}_{\mathrm{V}}\|}; \quad \arg\max_l \frac{\mathbf{l}_l^{\mathrm{T}} \mathbf{r}_{\mathrm{L}}}{\|\mathbf{l}_l\| \|\mathbf{r}_{\mathrm{L}}\|}; \quad \arg\max_e \frac{\mathbf{e}_e^{\mathrm{T}} \mathbf{r}_{\mathrm{E}}}{\|\mathbf{e}_e\| \|\mathbf{r}_{\mathrm{E}}\|}; \tag{7.30}$$

that for $\mathbf{r}_{\mathrm{P}}$, $\mathbf{r}_{\mathrm{V}}$, $\mathbf{r}_{\mathrm{L}}$, and $\mathbf{r}_{\mathrm{E}}$ find the best matching signatures; i.e., rows $\mathbf{p}_p^{\mathrm{T}}$, $\mathbf{v}_v^{\mathrm{T}}$, $\mathbf{l}_l^{\mathrm{T}}$, and $\mathbf{e}_e^{\mathrm{T}}$ of the causal mode matrices $\mathbf{U}_\text{P}$, $\mathbf{U}_\text{V}$, $\mathbf{U}_\text{L}$, and $\mathbf{U}_\text{E}$, respectively. Evaluating the set of similarity measures together enables us to recognize the probe image $\mathbf{d}$ as depicting person $p$ in view $v$, illumination $l$, and expression $e$.

Figure 1.4 illustrates the architecture of our multilinear recognition system, showing the TensorFaces (MPCA) model trained on the Weizmann facial image database and using (7.30). Of course, if we are interested only in recognizing the person depicted in $\mathbf{d}$, we can achieve savings by storing only the person signatures $\mathbf{U}_\text{P}$ and performing only the first similarity optimization in (7.30).

## 7.3 Facial Image Recognition Experiments

We will now evaluate the recognition algorithms that we have developed in this chapter. There are a number of meaningful experimental scenarios: We employ either (A) the Weizmann face image database (Figure 6.1) or (B) the Freiburg image database (Figure 6.6). Using either of these datasets, we train (1) an MPCA (TensorFaces) model (Figure 6.8) and (2) an MICA (independent TensorFaces) model (Figure 6.11). Then, in the testing phase, we recognize unlabeled test images (probes) by first inferring their coefficient vector representations and then using a similarity measure to label the probes, thus achieving recognition. The inference step may be accomplished using (i) the multiple linear projection (MLP) method of Section 7.1, or (ii) the multilinear projection (MP) method of Section 7.2 implemented either (a) by MP-MPCA (Algorithm 7.2.1) or (b) by MP-CP (Algorithm 7.2.2).

We compare the multilinear face recognition results obtained using trained MPCA and MICA models relative to each other and against those obtained using trained linear PCA [Sirovich and Kirby 1987; Turk and Pentland 1991a] and ICA [Bartlett 2001; Bartlett et al. 2002] models. In all the experiments reported below, people depicted in unlabeled test images (probes), which were not part of the training set, are recognized by inferring the person representation associated with the test image and choosing the person label using the similarity methods of (7.8) for MLP and (7.29) for MP.

Our first set of experiments employed the Weizmann dataset and the data tensor $\boldsymbol{\mathcal{D}}$ illustrated in Figure 6.1(c). The ensemble of training images were modeled using PCA (eigenfaces) and MPCA (TensorFaces) and the learned basis vectors are illustrated in Figure 6.3 and Figure 6.2, respectively. The PCA test image representation was computed by using the conventional linear projection. The MPCA person representation for test images was computed using the multiple linear projections (MLP) method (Figure 7.2). The person recognition rates show that MPCA yielded significantly better results than PCA in scenarios involving the recognition of people imaged in previously unseen views (Table 7.1, Row 1) and under a previously unseen illuminations (Table 7.1, Row 2).

Our next experiments employed the Freiburg facial image dataset of 16,875 images illustrated in Figure 6.6 and the data tensor $\boldsymbol{\mathcal{D}}$ illustrated in Figure 6.7. We trained PCA, ICA, MPCA, and MICA models. The trained PCA and MPCA bases are illustrated in Figure 6.3 and Figure 6.2, respectively. The trained ICA and MICA bases are shown in Figure 6.11. The trained PCA basis matrix (eigenfaces) has dimensionality $\mathbf{U}_\text{x} \in \mathbb{R}^{8560\times 222}$, as does the ICA basis matrix, while the MPCA (TensorFaces) basis and mode matrices have dimensions $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{8560\times 74\times 3\times 1}$, $\mathbf{U}_\text{P} \in \mathbb{R}^{75\times 74}$, $\mathbf{U}_\text{V} \in \mathbb{R}^{6\times 3}$, and $\mathbf{U}_\text{L} \in \mathbb{R}^{6\times 1}$,

| **PCA vs. MPCA Recognition Experiments** | **PCA** | **MPCA** |
|---|---|---|
| **Training:** 23 people, 3 views ($\theta = 0, \pm 34$), 4 illuminations (center, left, right, left+right)<br>**Testing:** 23 people, 2 views ($\theta = \pm 17$), 4 illuminations (center, left, right, left+right) | $61\%$ | $80\%$ |
| **Training:** 23 people, 5 views ($\theta = 0, \pm 17, \pm 34$), 3 illuminations (center, left, right)<br>**Testing:** 23 people, 5 views ($\theta = 0, \pm 17, \pm 34$), 1 illumination (left+right) | $27\%$ | $88\%$ |

*Table 7.1: Experimental results with the Weizmann image dataset in Figure 6.1 comparing PCA (eigenfaces) against MPCA (TensorFaces) with the multiple linear projection (MLP) recognition method.*

| **PCA & ICA vs. MPCA & MICA Recognition Experiment** | **PCA** | **ICA** | **MPCA** | **MICA** |
|---|---|---|---|---|
| **Training:** 75 people, 6 views ($\theta = \pm 35, \pm 20, \pm 5, \phi = 0$), 6 illuminations ($\theta = 45$, $\phi = 90 + \delta$, $\delta = \pm 35, \pm 20, \pm 5$)<br>**Testing:** 75 people, 9 views ($\phi = 0 \pm 10, \pm 15, \pm 25, \pm 30$), 9 illuminations ($\theta = 90 + \delta$, $\delta = \pm 35, \pm 20, \pm 5, \theta = 0$) | $83.90\%$ | $89.50\%$ | $92.65\%$ | $98.14\%$ |

*Table 7.2: Face recognition experiment using PCA, ICA, MPCA, and MICA models trained on the dataset in Figure 6.6. The MPCA and MICA recognition results were obtained using the MP-MPCA algorithm.*

as do the trained MICA basis tensor and mode matrices. Note that, for a fair comparison, all the models employ 222 basis vectors. The PCA/ICA image representation has 222 parameters, whereas our MPCA/MICA image representation has $74 + 3 + 1 = 78$ parameters. The MPCA and MICA image representations and response tensors are shown in Figure 7.3 and Figure 7.4, respectively.

Table 7.2 compares the recognition rates that we obtained in an experiment using these trained models. The MICA and MPCA models, using the MP-MPCA algorithm, yielded better recognition rates than PCA (eigenfaces) and ICA in scenarios involving the recognition of people imaged in previously unseen views and illuminations, with MICA performing significantly better than MPCA in this experiment.

Next, we trained the MPCA (TensorFaces) model and obtained recognition results employing our different projection algorithms to compute the person representations of unlabeled test images (probes). Table 7.3 compares the recognition rates obtained when applying the multiple linear projections (MLP) method and when applying the multilinear projection (MP) method with the MP-MPCA algorithm or with the MP-CP algorithm. Note that the MP-CP algorithm outperforms the MP-MPCA algorithm used in the previous recognition experiment.

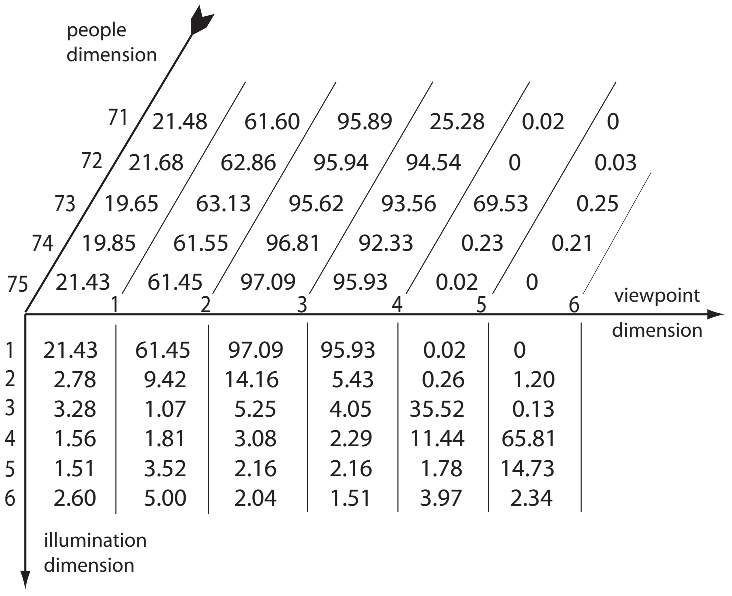
Table 7.4 provides a detailed study of how dimensionality reduction in the trained MPCA model affects recognition rates when using the MP-CP algorithm. The table shows recognition percentage rates obtained for the number of people, view, and illumination basis vectors retained as indicated along each axis.

Finally, we also experimented with the 2DPCA approach of [Ye 2005; Frangi and Yang 2004],[2]

[2]The name 2DSVD chosen by Ye [2005] is a misnomer. The acronym 2DSVD stands for 2D Singular Value Decomposition. (Note that the 2DSVD is known in the literature as a Tucker2 or a 2-mode SVD of a third-order data tensor.) A decomposition, by definition, should not include a dimensionality reduction step. However, the method described employs

| **MPCA: MLP vs. MP-MPCA vs. MP-CP Recognition Experiment** | **MLP** | **MP-MPCA** | **MP-CP** |
|---|---|---|---|
| **Training:** 75 people, 6 views ($\theta = \pm 35, \pm 20, \pm 5$, $\phi = 0$), 6 illuminations ($\theta = 45$, $\phi = 90 + \delta$, $\delta = \pm 35, \pm 20, \pm 5$) **Testing:** 75 people, 9 views ($\phi = 0 \pm 10, \pm 15, \pm 25, \pm = 30$), 9 illuminations ($\theta = 90 + \delta$, $\delta = \pm 35, \pm 20, \pm 5, \theta = 0$) | $92.67\%$ | $92.65\%$ | $96.81\%$ |

*Table 7.3: Facial recognition rates when using the image dataset in Figure 6.6 to train an MPCA model and using the MLP and MP recognition methods.*


people dimension
viewpoint dimension
illumination dimension

| people dimension | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 71 | 21.48 | 61.60 | 95.89 | 25.28 | 0.02 | 0 |
| 72 | 21.68 | 62.86 | 95.94 | 94.54 | 0 | 0.03 |
| 73 | 19.65 | 63.13 | 95.62 | 93.56 | 69.53 | 0.25 |
| 74 | 19.85 | 61.55 | 96.81 | 92.33 | 0.23 | 0.21 |
| 75 | 21.43 | 61.45 | 97.09 | 95.93 | 0.02 | 0 |

| illumination dimension | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 21.43 | 61.45 | 97.09 | 95.93 | 0.02 | 0 |
| 2 | 2.78 | 9.42 | 14.16 | 5.43 | 0.26 | 1.20 |
| 3 | 3.28 | 1.07 | 5.25 | 4.05 | 35.52 | 0.13 |
| 4 | 1.56 | 1.81 | 3.08 | 2.29 | 11.44 | 65.81 |
| 5 | 1.51 | 3.52 | 2.16 | 2.16 | 1.78 | 14.73 |
| 6 | 2.60 | 5.00 | 2.04 | 1.51 | 3.97 | 2.34 |

*Table 7.4: Recognition rates obtained by the MP-CP, rank-1 recognition algorithm with the MPCA (TensorFaces) model subject to various dimensionality reductions. The table shows percentage recognition rates obtained for the number of people, view, and illumination basis vectors retained that are indicated along each axis.*

which advocates the organization of the 2700 training images into a third-order tensor, $\boldsymbol{\mathcal{D}}_{\text{2DPCA}} \in \mathbb{R}^{80\times107\times2700}$, where each image $\mathbf{D}_i \in \mathbb{R}^{80\times107}$ is treated as a matrix (Figure A.1(b)); hence, it becomes an ensemble of row/column observations rather than the single observation vector that each image is in our approach. This data tensor can be decomposed according to Equation (A.4). For a fair comparison, we should use the same number of parameters to represent each image in 2DPCA as in PCA/ICA. Unfortunately, we cannot directly compare with the recognition rates reported in Table 7.2, which correspond to 222 PCA/ICA coefficients per image, because for 2DPCA the number of coefficients representing each image can be reduced either to 221 or to 224.[3] 2DPCA achieved recognition rates of 88.48% with 221 coefficients and 89.84% with 224 coefficients, which are comparable to ICA,

dimensionality reduction; hence, 2DPCA is a more appropriate name, and we will use it to refer to both methods.

[3]For the case at hand, in Equation (A.4) the tensor $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{13\times17\times2700}$ (or $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{14\times16\times2700}$) contains a set of $\mathbf{R}_i \in \mathbb{R}^{13\times17}$ (or $\mathbf{R}_i \in \mathbb{R}^{14\times16}$), which is the coefficient representation matrix for each image $\mathbf{D}_i$, $1 \leq i \leq 2700$. There are a total of 221 (or 224) coefficients per image. The image representation $\mathbf{R}_i$ is the response to $\mathbf{U}_{\text{xc}} \in \mathbb{R}^{107\times13}$ reduced from $\mathbf{U}_{\text{xc}} \in \mathbb{R}^{107\times107}$, and $\tilde{\mathbf{U}}_{\text{xr}} \in \mathbb{R}^{80\times17}$ reduced from $\mathbf{U}_{\text{xr}} \in \mathbb{R}^{80\times80}$.

but significantly lower than the recognition rates achieved by our MPCA model with the MP-CP algorithm (96.81%) and our MICA model with the MP-MPCA algorithm (98.14%). Appendix A provides a more detailed discussion of the drawbacks of the "image-as-a-matrix" approach in general and, in the final section, a more detailed comparison of the 2DPCA method relative to the other methods in the above experiment.

## 7.4 Discussion

We will now discuss the assumptions and limitations of our approach to (facial) object recognition.

As we stated in the introductory chapter, our models make the assumption that the multiple causal factors combine with one another in a multiplicative manner. Thus, although our multifactor model is nonlinear, the nonlinearity is not arbitrary. The tacit assumption (cf. (1.4)) is that the variability in the measurement data due to changes in any single factor can be adequately modeled in a linear manner when all other factors remain fixed. This assumption can be viewed as a limitation in terms of the complexity of measurement data with which our models can accurately deal, relative to arbitrarily nonlinear manifold learning models such as Locally Linear Embedding (LLE) [Roweis and Saul 2000]. However, as we discussed in Section 4.4, such piecewise, local models require dense training data, whereas our global model can be fitted to rather sparse training data. Furthermore, while a multilinear analysis may not be able to model arbitrarily nonlinear data accurately, we can *decompose* the data in terms of their explanatory variables using a multilinear decomposition and then further analyze and model each of those variables by a fully nonlinear method such as LLE.

It is more meaningful to compare our multilinear model against the kernel PCA [Schölkoph et al. 1998] and kernel ICA [Yang et al. 2005] techniques that were also discussed in Section 4.4. While these models make no specific nonlinearity assumptions, they model the distribution of the observed data from which they try to infer causal information, whereas our multilinear approach explicitly models the true multifactor causal structure of the measurement data generation process. As we discussed earlier, our model shares the same Gaussian (non-Gaussian) assumptions made by the PCA (ICA) models, in their conventional or kernel versions, and as was noted in Section 4.4, we can also further generalize our multilinear models using kernels.

Compared to unsupervised learning methods, a drawback of our multilinear models is that they must be trained in a supervised manner using properly labeled multimodal training data that have been collected in a systematic and comprehensive manner. This can be cumbersome and tedious relative to, say, training the standard PCA model on a simple collection of images. However, our multifactor approach has significantly loftier aspirations in terms of data analysis and ultimate recognition rates. Moreover, the offline labor-intensive data collection and training phase is compensated by the fact that there exists a computationally inexpensive online classification step. Furthermore, the training data acquisition and labeling process can be facilitated or even fully automated using special equipment, such as automated robotic devices to move cameras and lightsources [Moses et al. 1996; Dana et al. 1999], or multi-camera/multi-illumination systems [Debevec et al. 2000], or kaleidoscopic imaging devices [Han and Perlin 2003]. However, collecting large quantities of image data can be tedious and perhaps even prohibitively expensive, and our current model assumes the availability of complete data sets. It can and should be expanded to deal with missing data.

Finally, perhaps the biggest limitation of our approach to face recognition is that it is a strictly appearance-based method and, hence, it shares the well-known advantages and drawbacks of all appearance-based methods. We do not attempt to extract features from the image or apply any so-called "model-based techniques". Nor have we attempted to use any alternative sources of information about the face, such as 3D data. However, we believe that our approach can be extended to exploit additional sources of information, but of course at additional data collection costs.

# 8 Multilinear Motion Analysis, Synthesis, and Recognition: Human Motion Signatures

**Contents**

Our development has until now focused exclusively image data. In this chapter, we will consider the application of our tensor algebraic framework to motion data.

To motivate our motion application, consider the following questions: In analogy with handwritten signatures, do people have characteristic motion signatures that individualize their movements? If so, can these signatures be extracted from example motions? If so, can extracted signatures be used to recognize, say, a particular individual's walk subsequent to observing examples of other movements produced by this individual? An ability to perceive motion signatures would seem to be well-grounded from an evolutionary perspective, since survival depends on recognizing the movements of predator or prey, or of friend or foe.

In the 1960s, the psychologist Gunnar Johansson performed a series of famous experiments in which he attached point light sources to people's limbs and recorded videos of them performing different activities, such as walking, running, and dancing [Johansson 1974]. Observers of these moving light displays, videos in which only the light sources are visible, were asked to classify the activity performed and to note certain characteristics of the movements, such as a limp or an energetic/tired walk. Observers can usually perform this task with ease and they could sometimes determine gender and even recognize specific individuals in this way. This may corroborate the hypothesis that the motion signature is a perceptible element of human movement.

Our work with motion data has three goals:

1. To develop new algorithms that can analyze everyday human motions and separate out distinctive motion signatures.
2. To synthesize novel motions that are in accord with extracted motion signatures.
3. To recognize specific individuals performing new motions, using extracted motion signatures associated with these individuals.

Our approach exploits corpora of motion data which are now reasonably easy to acquire through a variety of modern motion capture technologies developed for use in the entertainment industry (see, e.g., [Gleicher 2001]). Motion synthesis through the analysis of motion capture data is currently attracting a great deal of attention within the computer graphics community as a means of animating graphical characters. Several authors have developed generative motion models for this purpose. Recent papers report the use of neural network learning models [Grzeszczuk et al. 1998] and hidden Markov models [Brand and Hertzmann 2000].

Within our multilinear framework, corpora of motion capture data including multiple people and actions are organized as higher-order arrays, or tensors. We apply our MPCA/MICA algorithms to extract human motion signatures among the other causal factors inherent to human movement.

## 8.1 Motion Data Acquisition

Human limb motion was recorded using a VICON system that employs four video cameras (Figure 8.1). The cameras detect infrared light reflected from retroreflective markers placed on each leg of a human subject. The system computes the 3D position of the markers relative to a fixed lab coordinate frame. The video cameras are positioned on one side of a 12 meter long walkway such that each marker can be observed by at least two cameras during motion.

To extract the three angles of a human joint, we must define a plane for each limb whose motion can be measured relative to the sagittal, frontal, and transverse planes through the body (Figure 8.2). The motion of each limb is the combined motion carried out in these three planes. The sagittal plane divides the body into left and right halves. Thus, motion in this plane is front to back. The frontal plane divides the body into front and back halves. Thus motion in this plane is side to side. The transverse plane divides the body into upper and lower halves. Thus, motion in this plane is an in-out twisting motion.

*Figure 8.1: The motion capture facility.*

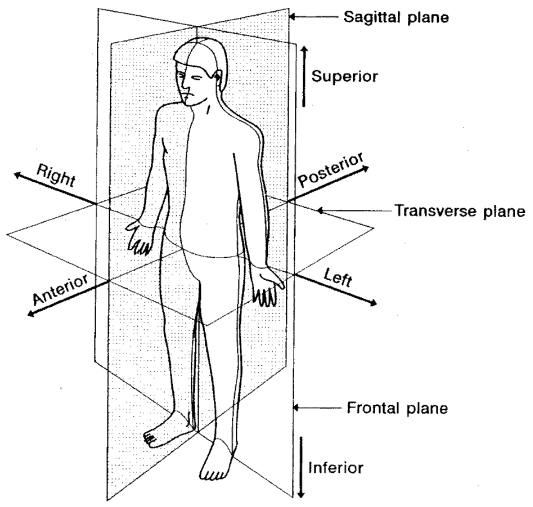


*Figure 8.2: Body with 3 reference planes and 6 fundamental directions (from [Whittle 1996]).*

A total of 18 markers were used, 9 to measure the motion of each leg, as illustrated in Figure 8.3(a). In general, we must define a plane for each limb, using three markers. We placed a marker at each end of the limb (at the joints) and we attached a third marker to a 15cm long stick strapped halfway down the limb and extending away from the body, such that the plane formed with the other markers is parallel to the frontal plane. For the foot plane, markers were placed on the ankle joint (at the base of the fifth metatarsal), laterally on the heel and on the lateral malleolus (right before the toes). The pelvis was defined by markers placed on the iliac crest, anterior superior spine, and greater trochanter.

To suppress noise, the collected motion data were low-pass filtered by a fourth-order Butterworth filter at a cut off frequency of 6 Hz and missing data were cubic spline interpolated from nearby data. After the marker positions were measured, filtered, and interpolated, the time-varying rotations of each of the limbs were calculated with respect to the lab frame, $F_L$ (Figure 8.3(b)). These transformations are used to compute intermediate transformations between the limb segments from which joint angles can be computed. Pelvic rotation is the rotation with respect to lab frame, $F_L$. The rotation of the thigh

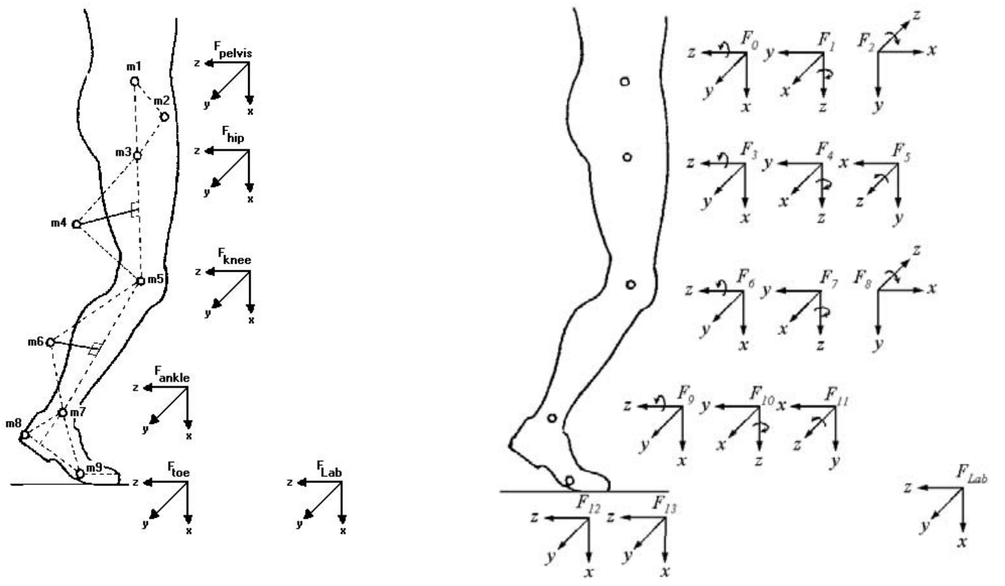


*Figure 8.3: (a) Marker configuration and coordinate frames. (b) Coordinate systems assigned to the 3 measured parts of the leg.*

is the rotation with respect to the pelvis through which the hip angles are obtained. The knee angles are defined as the rotations of the shank with respect to the thigh. The ankle angles are defined as the rotations of the foot with respect to the shank.

To compute the joint angles, we first calculate the frame coordinate transformation for each limb with respect to the lab, we then calculate the relative orientation of each limb in the kinematic chain, and we finally solve the inverse kinematic equations. These three steps are explained in Appendix C.

The result of each motion data acquisition run is a set of time series of the pelvis orientation plus the joint angles for the thigh, knee, and ankle joints of each leg. A training corpus of motion data was collected from 6 subjects. Three motions were acquired for each person—walk, ascend-stairs, descend stairs. Each motion was repeated 10 times. A motion cycle was segmented from each motion sequence. As shown in Figure 8.4, a motion cycle comprises the following motion events: Foot contact, opposite toe off the ground, opposite foot contact, toe off the ground, foot contact.

## 8.2 Motion Analysis: TensorMotions

Given motion sequences acquired from several people, we assemble a training data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_\mathrm{j} \times I_\mathrm{P} \times I_\mathrm{A}}$, where $I_\mathrm{j}$ is the number of joint angle time samples, $I_\mathrm{P}$ is the number of people, and $I_\mathrm{A}$ is the number of action classes. We apply the MPCA algorithm (Algorithm 4.2.1) to $\boldsymbol{\mathcal{D}}$ to obtain the model

$$\tilde{\boldsymbol{\mathcal{D}}} = \boldsymbol{\mathcal{T}} \times_\mathrm{P} \mathbf{U}_\mathrm{P} \times_\mathrm{A} \mathbf{U}_\mathrm{A}, \tag{8.1}$$

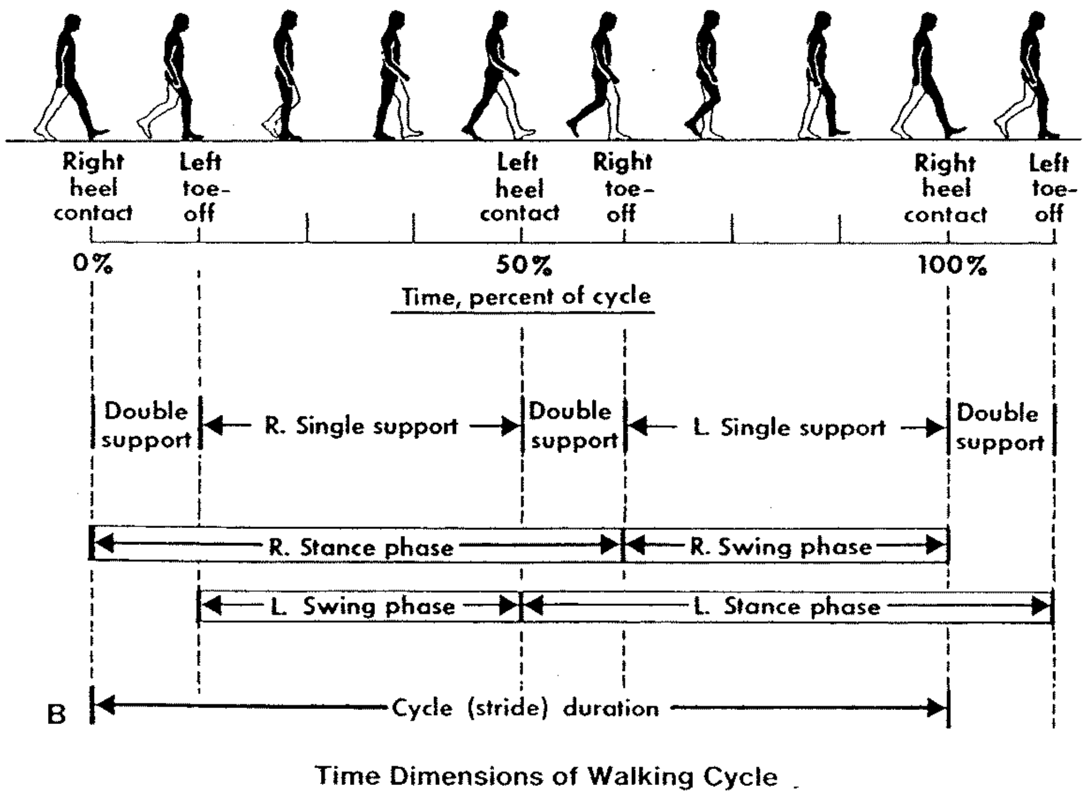


*Figure 8.4: Normal walking cycle illustrating the events of gait (from [Inman et al. 1981]).*

the product of an extended core tensor $\mathcal{T} = \mathcal{Z} \times_{\mathrm{J}} \mathbf{U}_{\mathrm{J}}$, a people basis matrix $\mathbf{U}_{\mathrm{P}}$, and an action basis matrix $\mathbf{U}_{\mathrm{A}}$. The column vectors of $\mathbf{U}_{\mathrm{J}}$ span the measurement space of joint angles and are the *eigenmotions* that are normally computed by PCA. The core tensor $\mathcal{Z}$ transforms the eigenmotions into *TensorMotions* $\mathcal{T}$, which govern how the causal factors (people and actions) interact to produce the observed motions. The row vectors of $\mathbf{U}_{\mathrm{A}}$, which we denote $\mathbf{a}_a^{\mathrm{T}}$, comprise coefficients for each action $a$, where $1 \leq a \leq I_{\mathrm{A}}$. Each of these vectors is an action-specific invariant across people. The row vectors of $\mathbf{U}_{\mathrm{P}}$, which we denote $\mathbf{p}_p^{\mathrm{T}}$, comprise coefficients for each person $p$, where $1 \leq p \leq I_{\mathrm{P}}$. Each of these vectors is a person-specific invariant across actions. We call them the *human motion signatures*. Figure 8.5 illustrates the motion data representation and analysis. Given the trained model (8.1), a motion can be synthesized as

$$\mathbf{d} = \mathcal{T} \times_{\mathrm{P}} \mathbf{r}_{\mathrm{P}}^{\mathrm{T}} \times_{\mathrm{A}} \mathbf{r}_{\mathrm{A}}^{\mathrm{T}}. \tag{8.2}$$

Comparing our multilinear technique to conventional PCA, the latter would decompose a motion data matrix whose columns are observed motions $\mathbf{d}_i$ into $\mathbf{D}_{[\mathrm{j}]} = \mathbf{U}_{\mathrm{J}}\mathbf{C}$, a basis matrix $\mathbf{U}_{\mathrm{J}}$ comprising the most significant eigenmotions times a coefficient matrix $\mathbf{C}$ containing a vector of coefficients $\mathbf{c}_i$ for every observed motion. Thus, PCA represents each person as a set of $I_{\mathrm{A}}$ coefficient vectors, one for each action. By contrast, our multilinear analysis enables us to represent each person $p$ with a single vector

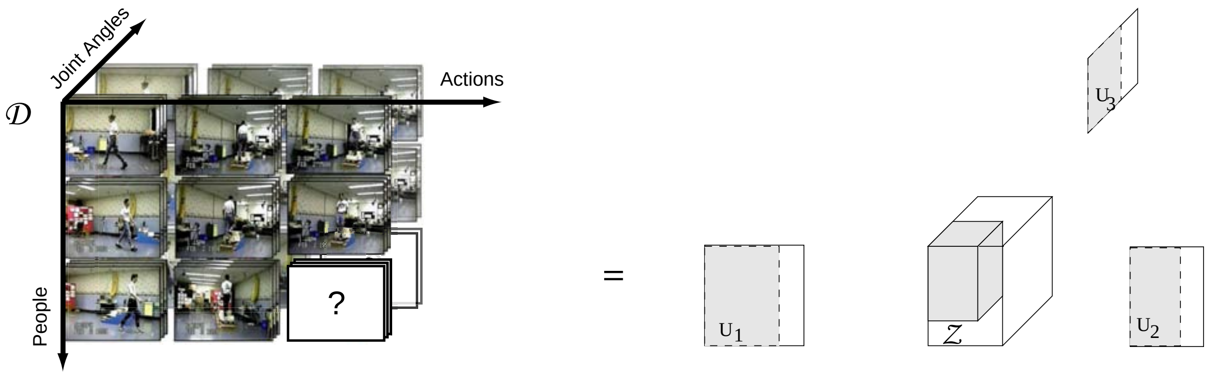


*Figure 8.5: Motion data collection and analysis model.*

of coefficients $\mathbf{p}_p^{\mathrm{T}}$ that is invariant of the action performed.

## 8.3 Motion Synthesis

By computing the decomposition (8.1), we analyze the training corpus of motion data $\boldsymbol{\mathcal{D}}$ acquired from a group of people and extract the human motion signatures $\mathbf{U}_{\mathrm{P}}$, the action representations $\mathbf{U}_{\mathrm{A}}$, and the TensorMotions basis $\boldsymbol{\mathcal{T}}$. We thus learn a generative model with which, given motion data for a new subject performing one or more, but not all, of the actions previously observed for the $I_{\mathrm{P}}$ people in the motion database, we can synthesize the remaining, unobserved actions in the style particular to this new subject.

Let $\breve{\boldsymbol{\mathcal{D}}}_n \in \mathbb{R}^{I_{\mathrm{J}} \times 1 \times \breve{I}_{\mathrm{A}}}$ be the motions of a new subject, where $\breve{I}_{\mathrm{A}}$ is the number of observed actions, which are a subset of all the $I_{\mathrm{A}}$ actions previously observed and analyzed for the group of people. From (8.1), we have

$$\breve{\boldsymbol{\mathcal{D}}}_n = \boldsymbol{\mathcal{T}} \times_{\mathrm{P}} \mathbf{p}_n^{\mathrm{T}} \times_{\mathrm{A}} \breve{\mathbf{U}}_{\mathrm{A}} \tag{8.3}$$

$$= \boldsymbol{\mathcal{A}} \times_{\mathrm{P}} \mathbf{p}_n^{\mathrm{T}}, \tag{8.4}$$

where $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{T}} \times_{\mathrm{A}} \breve{\mathbf{U}}_{\mathrm{A}}$ is the action-specific basis, $\breve{\mathbf{U}}_{\mathrm{A}}$ comprises a subset of the rows of $\mathbf{U}_{\mathrm{A}}$, and $\mathbf{p}_n^{\mathrm{T}}$ is the motion signature of the new person. Therefore, the synthesis algorithm first computes

$$\mathbf{p}_n^{\mathrm{T}} = \boldsymbol{\mathcal{A}}^{+_{\mathrm{P}}} \times_{\mathrm{P}}^{\mathrm{T}} \breve{\boldsymbol{\mathcal{D}}}_n, \tag{8.5}$$

where $\boldsymbol{\mathcal{A}}^{+_{\mathrm{P}}}$ is the people-mode pseudoinverse of tensor $\boldsymbol{\mathcal{A}}$. This model makes the assumption that the motion signature lies in the subspace spanned by $\mathbf{U}_{\mathrm{P}}$. We can then synthesize a complete set of motions $\boldsymbol{\mathcal{D}}_n$ for the new person as follows:

$$\boldsymbol{\mathcal{D}}_n = \boldsymbol{\mathcal{T}} \times_{\mathrm{A}} \mathbf{U}_{\mathrm{A}} \times_{\mathrm{P}} \mathbf{p}_n^{\mathrm{T}}. \tag{8.6}$$

Similarly, if we observe a known person or people who contributed motions to the training corpus performing a new action $\breve{\boldsymbol{\mathcal{D}}}_n \in \mathbb{R}^{I_j \times \breve{I}_{\text{P}} \times 1}$, where $\breve{I}_{\text{P}}$ is the number of people observed performing the new action, from (8.1), we have

$$\breve{\boldsymbol{\mathcal{D}}}_n = \boldsymbol{\mathcal{T}} \times \breve{\mathbf{U}}_{\text{P}} \times_{\text{A}} \mathbf{a}_n^{\text{T}} \tag{8.7}$$

$$= \boldsymbol{\mathcal{P}} \times_{\text{A}} \mathbf{a}_n^{\text{T}}, \tag{8.8}$$

where $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{T}} \times_{\text{P}} \breve{\mathbf{U}}_{\text{P}}$ is the people-specific basis and $\breve{\mathbf{U}}_{\text{P}}$ comprises a subset of the rows of $\mathbf{U}_{\text{P}}$. The associated action parameters $\mathbf{a}_n^{\text{T}}$ are computed as

$$\mathbf{a}_n^{\text{T}} = \boldsymbol{\mathcal{P}}^{+_{\text{A}}} \times_{\text{A}}^{\text{T}} \breve{\boldsymbol{\mathcal{D}}}_n, \tag{8.9}$$

where $\boldsymbol{\mathcal{P}}^{+_{\text{A}}}$ is the action-mode pseudoinverse of tensor $\boldsymbol{\mathcal{P}}$. We can then synthesize the new action in the style of every person in the database using

$$\boldsymbol{\mathcal{D}}_n = \boldsymbol{\mathcal{T}} \times_{\text{P}} \mathbf{U}_{\text{P}} \times_{\text{A}} \mathbf{a}_n^{\text{T}}. \tag{8.10}$$

## 8.4 Motion Recognition

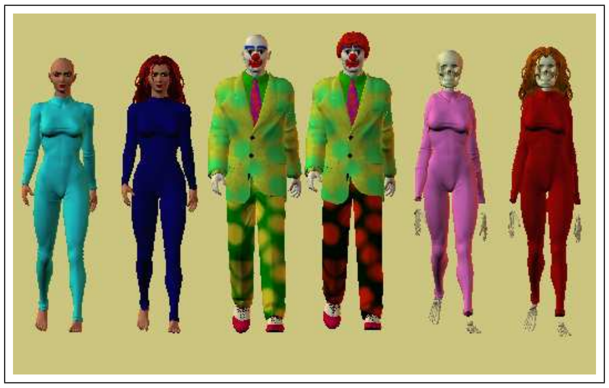
Our multilinear analysis yields the basis tensor $\boldsymbol{\mathcal{T}}$ whose pseudoinverse maps observed motions from the measurement space—joint angles over time—into the underlying causal factor spaces—the people space and action space—thereby enabling the recognition of people and actions from the motion data. Given a test (probe) motion $\mathbf{d}$ and the trained TensorMotions $\boldsymbol{\mathcal{T}}$, we can apply the multilinear projection algorithms of Section 7.2 to compute the causal factor representation vectors $\mathbf{r}_{\text{P}}$ and $\mathbf{r}_{\text{A}}$ and then simultaneously recognize the person $p$ and action $a$ via similarity maximizations like those in (7.30).

However, if we can identify by inspection the action depicted in the probe motion $\mathbf{d}$, the task of recognizing the person who produced $\mathbf{d}$ becomes easier. We map $\mathbf{d}$ into the people signature space by first computing the action-specific basis $\boldsymbol{\mathcal{B}}_a = \boldsymbol{\mathcal{T}} \times_{\text{A}} \mathbf{a}_a^{\text{T}}$, for the identified action $a$, where $\mathbf{a}_a^{\text{T}}$ is the $a$th row in $\mathbf{U}_{\text{A}}$. We then compare the signature $\mathbf{r}_{\text{P}}^{\text{T}} = \boldsymbol{\mathcal{B}}_a^{+_{\text{P}}} \times_{\text{P}} \mathbf{d}^{\text{T}}$ against the person signatures $\mathbf{p}_p^{\text{T}}$ for $1 \leq p \leq I_{\text{P}}$, which are the rows in $\mathbf{U}_{\text{P}}$, and the recognized person label assigned to $\mathbf{d}$ is the label $p$ of the best matching signature vector $\mathbf{p}_p$; i.e.,

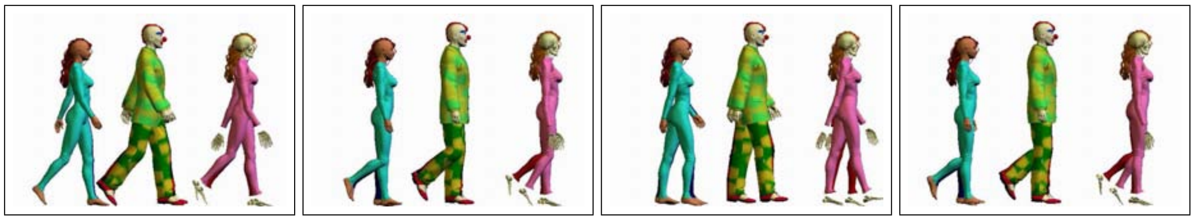
$$\arg\max_p \frac{\mathbf{p}_p^{\text{T}} \mathbf{r}_{\text{P}}}{\|\mathbf{p}_p\| \|\mathbf{r}_{\text{P}}\|}. \tag{8.11}$$

Conversely, to recognize the action depicted in $\mathbf{d}$ generated by an identified person $p$, we map $\mathbf{d}$ into the action parameter space by computing the person-specific basis tensor $\boldsymbol{\mathcal{B}}_p = \boldsymbol{\mathcal{T}} \times_{\text{P}} \mathbf{p}_p^{\text{T}}$ for the identified person $p$ and then compare the signature $\mathbf{r}_{\text{A}}^{\text{T}} = \boldsymbol{\mathcal{B}}^{+_{\text{A}}} \times_{\text{A}} \mathbf{d}^{\text{T}}$ against the action signatures $\mathbf{a}_a^{\text{T}}$ for $1 \leq a \leq I_{\text{A}}$, which are the rows in $\mathbf{U}_{\text{A}}$, and the recognized action label assigned to $\mathbf{d}$ is the label $a$ of the best matching signature vector $\mathbf{a}_a$; i.e.,

$$\arg\max_a \frac{\mathbf{a}_a^{\text{T}} \mathbf{r}_{\text{A}}}{\|\mathbf{a}_a\| \|\mathbf{r}_{\text{A}}\|}. \tag{8.12}$$



(a)



(b)

*Figure 8.6: Synthesizing 3 styles of walking motions from example motions of ascending stairs in those corresponding styles. (a) Comparing synthesized walking motion data against ground truth (the synthesized data is depicted by the characters without hair), our method captures stylistic differences in motion such as pigeon-toed walking, knocked-knees or strutting. (b) The synthesized motions are depicted by the characters in the foreground and, for comparison, the captured walking motions are depicted by the characters in the background.*

## 8.5 Experiments and Results

Our experiments indicate that, given a sufficient quantity of motion training data, our human motion signature extraction algorithm can consistently produce walks and stair ascend/descend motions in the styles of particular individuals included in the training corpus.

To determine whether people have motion signatures that are invariant of action classes, we extracted a motion signature from a subset of actions for a new individual (8.5) and synthesized the remainder of the actions using the extracted motion signature (8.6). We then validated the synthetic motions by classifying them against a database of all the real motions, and the classification accuracy was 100%.

In a "leave-one-out" validation study, we verified that our algorithm was able to compute motion signatures sufficiently well to synthesize all three types of motions in the distinctive style of each individual compared against ground-truth motion capture data of that individual. If the motion signature $\mathbf{p}_n$ of a new person captures the distinctive pattern of movement, the synthesized walk would best match the actual walk of the person. Using our classifier, the synthesized walk was indeed recognized against a complete database that included the actual walk data for the new person.

Figure 8.6(a) shows, in frontal view, the synthesis of three different styles of walking motion given only examples of descending stairs in those corresponding styles. Note that the walking styles differ

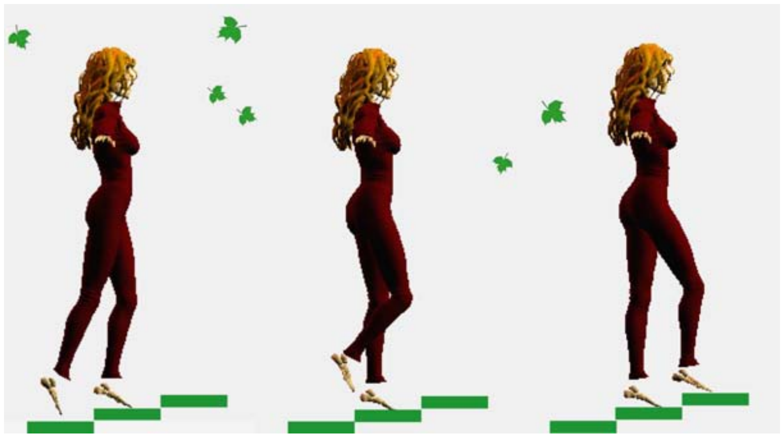


*Figure 8.7: A synthesized stair-ascending motion.*

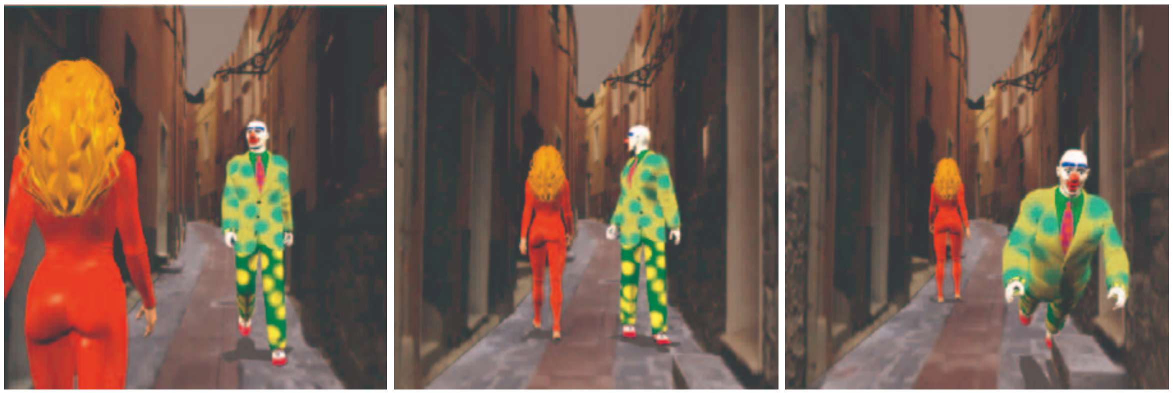


*Figure 8.8: Frames from an animation short created with synthesized motion data.*

subtly: The woman on the left walks in a pigeon-toed style, the clown struts, and the skeleton on the right walks with knocked knees. Figure 8.6(b) shows a side view of the motions; the figures animated using synthesized motions are in the foreground. Figure 8.7 shows a stair ascending motion synthesized for one of the individuals. Our algorithm extracted the motion signature from a sample walk from this individual. We then used the extracted motion signature to synthesize the stair-ascending motion for this individual. The motion signature was combined with general stair ascending parameters that were previously extracted from our database.

We have created an animation short using motion data synthesized by our algorithm (Figure 8.8). The graphical characters shown are modeled and rendered using the *MetaCreations Poser* software package.

## 8.6 Discussion

The synthesis results that we obtained were adequate to animate our characters in simple ways, and the observed motion recognition rates were 100% accurate. There exists a substantial literature on motion analysis and synthesis in computer graphics and on motion analysis and recognition in computer vision, some of which was covered elsewhere in this thesis. A comparison of our recognition rates against those of related algorithms found in the literature would probably not be meaningful because we employed a rather small dataset in our experiments. Systematic experiments need to be performed on much more extensive datasets, but the data collection issues discussed in Appendix B in the context of acquiring facial image datasets apply analogously to motion capture data. To our knowledge, extensive motion databases that would be suitable for fully training our multilinear model do not yet exist.

The limitations of our approach to motion recognition are analogous to those that were discussed in Section 7.4 in the context of face recognition and we refer the reader to that discussion. To summarize, the limitations include the multilinearity assumption, which can be relaxed to more general nonlinearity using kernel functions, and the aforementioned training data collection requirements (since we are using motion capture data, the appearance-based modeling issues do not apply). However, we point out again that, despite the limitations, our multilinear framework offers significant advantages that are unique among existing methods, most significantly among them the ability to model the essential multimodal causal structure of the observational data—in this case, human motion capture data.

# 9 Conclusion

**Contents**

## 9.1 Summary

We have introduced and developed a tensor framework for visual computing. Within this framework, the analysis, synthesis, and recognition of image ensembles resulting from the confluence of multiple causal factors related to scene structure, illumination, and imaging are regarded as problems in multilinear algebra wherein the image ensemble is represented as a higher-order tensor. This image data tensor is decomposed in order to separate and parsimoniously represent the causal factors.

We pursued a tensor decomposition approach to tackling these problems based on the $M$-mode singular value decomposition for multilinear principal components analysis with truncation and an alternating least squares iteration for dimensionality reduction. One of our novel contributions was a complimentary multilinear generalization of independent components analysis. Another contribution was the introduction of multilinear projection, with associated novel definitions of the identity and pseudo-inverse tensors, for the purposes of recognition in our tensor framework.

We demonstrated our approach to be of value in tackling problems of current importance in computer graphics, computer vision, and pattern recognition. In particular, we presented a multilinear image-based rendering method that we call TensorTextures, as well as a multilinear facial image representation and recognition method that we call TensorFaces. These new multilinear algebraic (tensor) methods outperform their conventional linear algebraic (matrix) counterparts.

We also applied our multilinear framework to the analysis, synthesis and recognition of human motion data with promising results. We anticipate numerous other applications to data-intensive domains where PCA/ICA have been employed in the past.

## 9.2 Future Work

Our research opens up numerous avenues for future work. Some of the major ones are as follows:

- **Kernel MPCA/MICA and Multifactor Generalizations of LLE/Isomap/LE Manifold Learning Methods:** As we discussed in Section 4.4, although it is fundamentally nonlinear, our multilinear framework can be straightforwardly generalized to deal with nonlinear data through "kernelization". This leads to Kernel Multilinear PCA (K-MPCA) and Kernel Multilinear ICA (K-MICA) methods. In future work, we will apply and and evaluate these methods on our datasets. Further exploiting kernelization within our multilinear framework, we also proposed in Section 4.4 multifactor generalizations of the Locally Linear Embedding (LLE), Isomap (IM), and Laplacian Eigenmap (LE) manifold learning algorithms. We will implement these generalizations and evaluate their performance in our image synthesis/analysis/recognition applications.

- **Applying the Multilinear Framework to Other Fields:** We have motivated and demonstrated our multilinear framework in the domains of computer graphics, computer vision, and machine learning. However, or framework should, in principle, also be applicable to any other domain where PCA and ICA have been employed in the past. Interestingly, the application of these techniques seems to be more or less ubiquitous. Examples include: The analysis of microarrays in bioinformatics [Alter et al. 2000; Lee and Batzoglou 2003]; in finance for modeling yield curves and other uses [Alexander 2001; Back and Weigend 1997]; in information technology for data mining; in the social sciences for discovering social networks; etc. All of these domains can potentially benefit by generalizing existing domain-specific PCA/ICA-based algorithms to their natural multilinear extensions.

- **Multilinear Texture Synthesis:** In our TensorTextures work, we introduced a multilinear approach to the image-based rendering of textured surfaces via a parsimonious, explicitly multifactor generative approximation to the BTF. We believe that our approach can handle data sets that result from the variation of additional causal factors, such as BTF scale (i.e., zooming into or away from a textured surface), high dynamic range (HDR) BTF acquisition at multiple exposures, or temporally varying BTFs, such as aging skin or leaves changing colors in the fall. In future work, one could also incorporate into TensorTextures a variant of view-dependent displacement maps [Wang et al. 2003]. Another avenue for future work is multilinear texture synthesis, which would generalize existing PCA-based texture synthesis approaches [Lefebvre and Hoppe 2006; Liang et al. 2001].

- **Multilinear Deformable Models:** Multilinear analysis is ripe for future exploitation in area of deformable models in computer vision and graphics. An important category of such models in vision is known as *active shape/appearance models* [Cootes et al. 1995; Cootes et al. 1998]. The deformation modes of these models are learned from a set of training examples by applying PCA and dimensionality reduction. Among other applications, such models have proven themselves useful for medical image segmentation. Multilinear analysis promises to generalize and augment the power of such deformable models, enabling them to learn multiple modes of deformation arising from different sources—e.g., apparent shape deformations due to view change, versus true deformations due to pathological conditions, versus true deformations due to non-pathological conditions, such as aging. In graphics, multilinear analysis should be readily applicable in the area of *morphable models* [Blanz and Vetter 1999; Allen et al. 2003] where PCA has been used.

- **Computing the $M$-Mode SVD Incrementally and When Data Are Missing:** Our current work has assumed that we can assemble complete data tensors in order to carry out multilinear analyses. Unfortunately, this may not be a realistic assumption in real world applications. A promising statistical technique for dealing with missing data is known as *imputation*. Furthermore, since in real world applications data may be acquired sequentially, a multilinear model would best be updated with every new data acquisition; thus, there is a need for an incremental $M$-mode SVD/ICA. An example is biomedical applications that might have a time component, such as experiments that need to be run at different stages of a chronic illness. Computing the matrix SVD in an incremental manner while estimating missing data was considered by Brand [2002]. The incremental computation and imputation of missing data in a multilinear framework is in need of further development.

- **Multilinear Local Feature Decomposition:** In appearance-based object recognition, it is important to be able to decompose and represent an object in terms of local topographic features [Penev and Atick 1996]. Currently the TensorFaces basis vectors contain global features. In a multilinear local feature decomposition, the basis vectors for the people mode would contain topographic representations of the face such as the eyes, nose, and mouth, while the basis vectors for the illumination mode would contain topographic representation associated with illumination, such as shadow under the eye cast by the brow, the shadow cast by the nose, the shadow cast by the chin, etc. In principle, this may be achieved by adding constraints to our multilinear models analogous to the type of constraints that have been added to linear models, such as non-negativity constraints [Lee and Seung 1999; Shen and Israel 1989] or sparsity constraints [Cadima and Jolliffe 1995; Chennubhotla and Jepson 2001; d'Aspremont et al. 2004; Zou et al. 2006; Moghaddam et al. 2006]. This should be a promising direction for future work.

- **A Multilinear Face Recognition System:** Although our multilinear framework is relevant to biometric systems, it has been beyond the scope of this thesis to implement a practical face recognition system. A big challenge is the recognition of an individual under unconstrained imaging conditions. Such a system might be structured as shown in Figure 1.4. First, a multilinear model is learned from a (hopefully large) set of cooperative participants, each of whom supply a complete set of training images acquired from multiple views, under multiple illuminations, in multiple expressions, etc. An uncooperative subject whose face is detected in one or more surveillance images can then be enrolled into the system, by representing his/her facial image(s) relative to the statistics encoded in the learned model. This should make it possible, in principle, to synthesize a complete image set of training images for the subject. The image database can then be augmented with these new training images and the multilinear model incrementally updated. This should enable the recognition of the subject from an arbitrary, unlabeled image, as illustrated at the bottom of the figure. Implementing a prototype recognition system of this kind, especially one that would support the multilinear fusion of multimodality biometric data (e.g., video and speech) would be a worthy future goal.

# A
# On the Treatment of Observational Data as Vectors, Matrices, or Tensors

Raw observational data usually comprise sets of random variables. The conventional way of organizing a multivariate observation for statistical analysis is as a vector of measurement variables. In recent years, however, several authors have advocated leaving the data elements associated with a single observation organized in the manner that the acquisition device provides them; in particular, organizing a CCD image as a matrix of pixel variables, rather than "vectorizing" it. In this context, bilinear and multilinear models for vision, graphics, and learning fall under four different categories:

1. Multilinear, or rank-$(R_1, R_2, \dots, R_M)$, decomposition and different generalizations for higher-order data tensors that contain vectorized measurement data, as in our work and the work of other authors [Wang et al. 2003; Vlasic et al. 2005; Hsu et al. 2005].

2. Like Category 1 above, but with rank-$R$ decomposition.

3. Multilinear, or rank-$(R_1, R_2, \dots, R_M)$, decomposition of higher-order data tensors where each observation is a matrix or higher-order tensor; e.g., an image is treated as a matrix and a unimodal image ensemble is a third-order tensor [Ye 2005; Xu et al. 2005; Wang and Ahuja 2005; Wang et al. 2005; Wang and Ahuja 2008].[1]

[1]Some authors [Ye 2005; Wang and Ahuja 2005] claim that they perform a generalized rank-$R$ approximation of tensors; however, according to the standard definition of the rank-$R$ and rank-$(R_1, R_2, \dots, R_M)$ decompositions, they actually perform a multilinear decomposition of a third-order tensor with dimensionality reduction in only two of the modes—the image row/column modes. This multilinear decomposition is also known as a Tucker2 decomposition or a $(M-1)$-mode SVD.

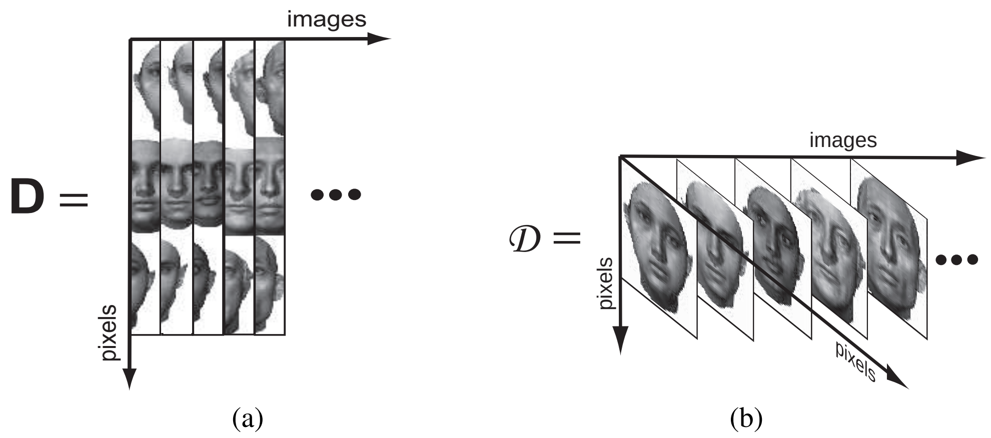


*Figure A.1: Image ensemble organizations. (a) Images are vectorized, (i.e., the pixels of an image are organized into a 1-way array). An ensemble of images is organized as a 2-way array or a "data matrix" (a multimodal image ensemble is a higher-order "data tensor"). The thick strips represent single columns. (b) The pixels of an image are organized as a 2-way array. A unimodal image ensemble is organized as a 3-way data tensor.*

4. Like Category 3 above, but with rank-$R$ decomposition [Shashua and Levin 2001; Furukawa et al. 2002; Wang and Ahuja 2004; Shashua and Hazan 2005].

We will now evaluate some of the arguments found in recent publications in favor of treating an image as a matrix rather than vectorizing it (Figure A.1), and conclude that it is preferable to vectorize images. The following are arguments for treating an image as a matrix, followed by rebuttals:

- *"An image (video) is intrinsically a matrix (tensor)."*

  Technically, a matrix corresponds to a transformation from one vector space to another, so a more appropriate statement would be that an image (video) is a 2D (3D) array of numbers. However, this statement merely presupposes a CCD imaging sensor with a rectangular array of receptors; the photoreceptors in biological eyes are by no means organized as regular matrices. Yet one can readily treat both cases in a uniform manner by vectorizing the collection of receptor responses, thus treating the *entire* image as a point in a high-dimensional image space.

- *"An inherent problem of the image-as-a-vector representation is that the spatial redundancy within each image matrix is not fully utilized, and some local spatial relationships are lost."*

  On the contrary, the subdivision of the image into rows and columns, suffers from the very problem incorrectly attributed to the conventional, vectorized-image representation. PCA applied to an ensemble of vectorized images encodes the pixel covariances or second-order statistics of the ensemble. In other words, treating an $N_1 \times N_2$ image as a vector leads to the computation for each pixel of *all* its pairwise covariances with *every other* pixel in the image (there are $N_1 N_2$ such covariances for each pixel) (Figure A.2). By contrast, when one regards an image as a matrix, it becomes a collection of row (column) observations and PCA explicitly computes how pixels covary with other pixels within the same row (column). There are only $N_1 + N_2$ such covariances

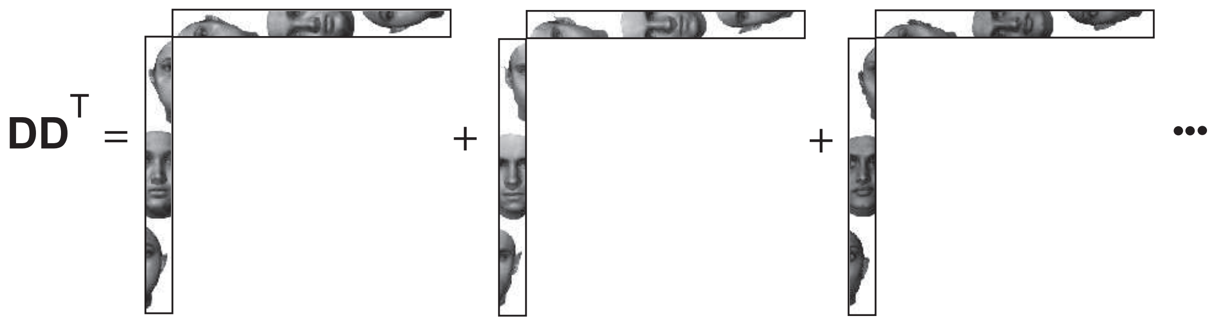


*Figure A.2: When the pixels associated with an image are organized into a 1-way array, PCA computes all possible pairwise pixel covariances.*

(Figure A.3). In particular, the covariances of pixels separated diagonally from one another are ignored (there are $N_1N_2 - N_1 - N_2$ diagonal covariances for each pixel) (Figure A.3(c)). Thus, PCA computed on an image represented as a matrix encodes much less information than PCA computed on an image represented as a vector. We examine the mathematical relationship between the two representations in more detail below.

- *"Representing images as 2D matrices instead of vectors allows covariances between both rows and columns to be exploited for dimensionality reduction, ..."*

  This statement is misleading. The decomposition creates a separate representation for rows and columns. Covariances between rows and columns are not exploited.

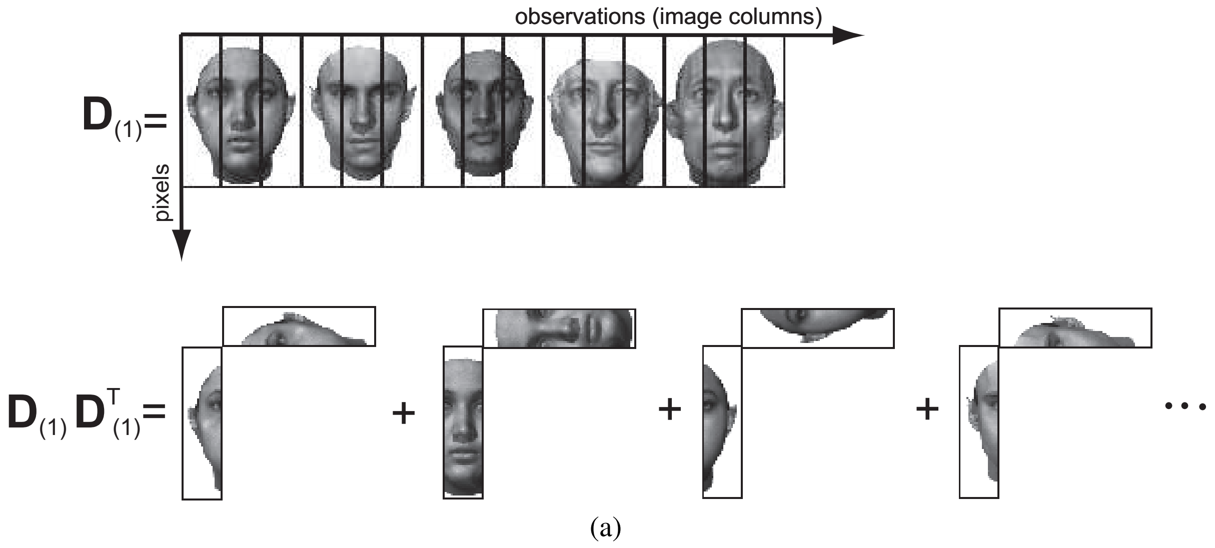
- *"We overcome the curse of dimensionality by treating images as matrices" (i.e., as a set of separate column measurements rather than a single measurement vector).*

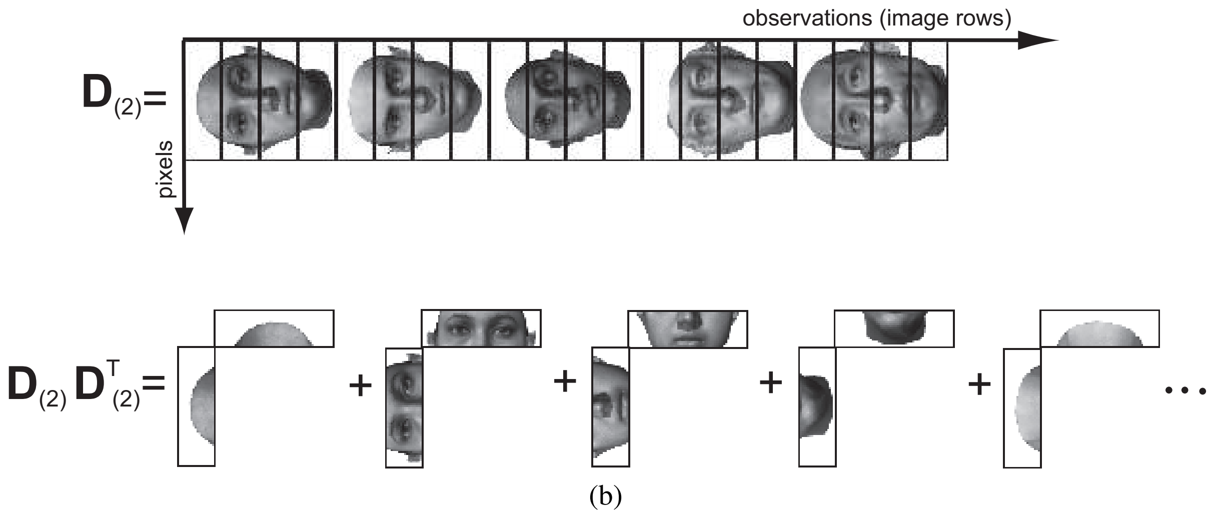
  *"First, in real applications such as face recognition, a very limited number of sample images are typically available for each subject, resulting in the well-known small-sample-size problem. In such contexts, an image-as-matrix representation is more appropriate, since the smaller number of data entries along each data dimension facilitates subspace learning from little training data."*

  Treating an image as matrix does not resolve or directly deal with the curse-of-dimensionality and/or small-sample-size problems. Consider the extreme case where the sample size is a single image. Clearly, one image provides no statistically significant information about the facial image space, and treating an image as a matrix—i.e. the image as a set of observations, where each row/column is a measurement—will not provide any further information about the facial image space. It simply computes a set of statistics associated with the rows/columns of that single image.

- *"Treating an image as a matrix, results in a model that uses less storage."*

  The storage requirements of both PCA and 2DPCA [Frangi and Yang 2004] grow linearly with the number of images, while the amount of storage needed to store the basis vectors is constant. This statement applies both to 2DPCA which performs a 1-mode SVD of a third-order tensor, and to 2DSVD [Ye 2005], which performs a 2-mode SVD of a third-order tensor; in both instances,


observations (image columns)
$\mathbf{D}_{(1)}=$
pixels
$\mathbf{D}_{(1)}\mathbf{D}^{T}_{(1)}=$
+
+
+
...
(a)


observations (image rows)
$\mathbf{D}_{(2)}=$
pixels
$\mathbf{D}_{(2)}\mathbf{D}^{T}_{(2)}=$
+
+
+
+
...
(b)

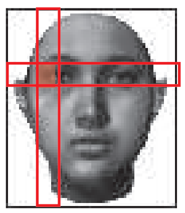

(c)

*Figure A.3: Matrixizing a data tensor and covariance computation. (a) The data tensor is matrixized in the first mode and pixel covariances are computed between pixels within a column. (b) The data tensor is matrixized in the second mode and pixel covariances are computed between pixels within a row. (c) Diagonal covariances are not computed.*

the tensor is an ensemble of 2D images. The upper bound on the PCA basis storage requirements (corresponding to no dimensionality reduction) is the number of pixels squared, while that of the 2DPCA is the number of rows squared plus the number of columns squared. While the basis vector storage requirements for 2DPCA is less than that of PCA, the dominant factor is the amount of storage needed to represent the images in the database. By contrast, the storage requirements of MPCA and MICA grow linearly with the number of people, which is usually only a fraction of the number of images.

- *"Computing PCA on a dataset that treats an image as a matrix is computationally more efficient."*

  The PCA of a data matrix of vectorized $N_1 \times N_2$ images computes the SVD of the pixel covariance matrix whose dimensionality is $(N_1 N_2)^2$. When treating images as "matrices" and an image ensemble as a third-order tensor, one obtains a column-pixel covariance matrix and a row-pixel covariance matrix, of dimensionality $N_1^2$ and $N_2^2$, respectively. Computing the SVDs of two smaller covariance matrices for 2DSVD is less expensive than performing an SVD on a large covariance matrix for PCA, but one should bear in mind that this is an offline computational task. Our MPCA/MICA algorithms treat images as vectors, but they compute SVDs not of the pixel covariance matrix, but of the covariance matrices associated with each causal factor. Although this can be more expensive than PCA or 2DSVD, depending on the size of the causal factor matrices, it is an offline computational cost that is offset by a less expensive online recognition computation. Several papers in the literature advocate the computation of the MPCA by treating images as matrices (e.g., [Wang and Ahuja 2005; Wang et al. 2005]), which additionally necessitates computing SVDs of the row-pixel covariance and column-pixel covariance matrices.

We now examine the mathematical relationship between image-as-a-vector and image-as-a-matrix representations in more detail. A data matrix $\mathbf{D} \in \mathbb{R}^{I_\text{x} \times I_\text{I}}$ whose columns are vectorized images (Figure A.1(a)) can be decomposed using standard matrix SVD. The SVD is expressed in terms of the mode-$m$ product and matrix multiplication as follows:

$$\mathbf{D} = \mathbf{S} \times_\text{x} \mathbf{U}_\text{x} \times_\text{I} \mathbf{U}_\text{I} \qquad \text{Tensor Notation} \tag{A.1}$$

$$\mathbf{D}_{[\text{x}]} = \mathbf{D}_{[\text{I}]}^\text{T} = \mathbf{U}_\text{x} \underbrace{\mathbf{S}\mathbf{U}_\text{I}^\text{T}}_{\mathbf{R}^\text{T}}, \qquad \text{SVD Notation} \tag{A.2}$$

where $\mathbf{U}_\text{x}$ contains the PCA basis vectors. These basis vectors are computed from the pixel covariance matrix, which contains all possible pairwise pixel covariances (Figure A.2). Matrix $\mathbf{R}$ is the response/coefficient matrix, where the $i$th column in $\mathbf{R}^\text{T}$ is the coefficient representation relative to the PCA basis of $\mathbf{d}_i$, which is the $i$th observation/column in $\mathbf{D}$.

When an $I_\text{xc} \times I_\text{xr}$ image is organized as a data matrix $\mathbf{D}_i$ and an ensemble of such images is organized into a third order data tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{I_\text{xc} \times I_\text{xr} \times I_\text{I}}$ (Figure A.1(b)), the data tensor can be decomposed using the $M$-mode SVD as follows:

$$\boldsymbol{\mathcal{D}} = \boldsymbol{\mathcal{Z}} \times_\text{xc} \mathbf{U}_\text{xc} \times_\text{xr} \mathbf{U}_\text{xr} \times_\text{I} \mathbf{U}_\text{I} \tag{A.3}$$

$$= \boldsymbol{\mathcal{R}} \times_{\text{xc}} \mathbf{U}_{\text{xc}} \times_{\text{xr}} \mathbf{U}_{\text{xr}}, \tag{A.4}$$

where $\mathbf{U}_{\text{xc}}$ is computed by matrixizing the data tensor along the row (Figure A.3(a)) and computing the basis vectors associated with the covariance matrix between pixels within image columns, where each column is considered a single measurement. Matrix $\mathbf{U}_{\text{xr}}$ contains the basis vectors associated with the covariance matrix between pixels within image rows, where each row is considered a single measurement (Figure A.3(b)). The product $\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{Z}} \times_{\text{I}} \mathbf{U}_{\text{I}}$ contains the response for each image when projected onto $\mathbf{U}_{\text{xc}}$ and $\mathbf{U}_{\text{xr}}$. $\mathbf{R}_i$ is the $i^{\text{th}}$ slice of tensor $\boldsymbol{\mathcal{R}}$ and it contains the response or coefficients associated with image $\mathbf{D}_i$.

Matrixizing the data tensor along the image mode results in a data matrix $\mathbf{D}_{[\text{I}]}$ whose transpose is the same matrix used by PCA in Equation (A.2), thus giving us an opportunity to compare terms: From Equations (A.3) and (A.2), we have

$$\mathbf{D}_{[\text{I}]}^{\text{T}} = \mathbf{U}_{\text{x}}\mathbf{S}\mathbf{U}_{\text{I}}^{\text{T}} \tag{A.5}$$

$$= \underbrace{(\mathbf{U}_{\text{xc}} \otimes \mathbf{U}_{\text{xr}})\mathbf{Z}_{[\text{I}]}^{\text{T}}}_{\substack{\text{Unnormalized}\\ \text{PCA Basis Matrix}}} \quad \underbrace{\mathbf{U}_{\text{I}}^{\text{T}}}_{\substack{\text{Normalized}\\ \text{PCA Coefficient Matrix}}} \tag{A.6}$$

Essentially, $M$-mode SVD has decomposed the unnormalized PCA basis $(\mathbf{U}_{\text{x}}\mathbf{S})$ into two basis matrices, $\mathbf{U}_{\text{xc}}$ and $\mathbf{U}_{\text{xr}}$, whose columns span column/row pixel covariances. By contrast, when images are vectorized and organized into a multifactor tensor (akin to the TensorFaces approach), $M$-mode SVD further decomposes the PCA coefficient matrix.

Tensor $\boldsymbol{\mathcal{Z}}$ contains normalizing parameters associated with $\mathbf{U}_{\text{I}}$, $\mathbf{U}_{\text{xr}}$ and $\mathbf{U}_{\text{xc}}$. It may be tempting to think of $\mathbf{Z}_{[\text{I}]}^{\text{T}}$ as a matrix that spans cross-diagonal pixel covariance from most important to least important; however, cross-diagonal pixel covariance is not computed, and the $M$-mode SVD can only model the data with respect to $\mathbf{U}_{\text{xc}}$ and $\mathbf{U}_{\text{xr}}$. Thus, dimensionality reduction cannot discard the least important redundancies between cross-diagonal pixels. Note that there is an exact relationship *only* if no dimensionality reduction is performed.

The same counterargument holds true for organizing a sensory input data into a higher-order tensor. To summarize, as we stated above, it is preferable to treat images as vectors.

# B

# On Data Acquisition and PIE Image Datasets

Data collection for machine learning approaches to object analysis and recognition has evolved over the years from the haphazard aquisition of images to more systematic data collection methodologies. However, many of the most popular datasets suffer from data capture and processing shortcomings which hinder their use for the purposes of training machine learning algorithms, especially multimodal ones like ours. In this appendix, we first discuss important issues in data aquisition that one should consider in the context of data analysis and inference. Then we evaluate some popular image databases with regard to the issues and explain the suitability of the datasets that we have used in our work.

A successful statistical analysis usually requires a careful plan for the acquisition of data intended to answer the specific questions of interest. First, a representative sample of the population is needed. Second, in order to establish not only the causality, but also the actual mathematical relationship between a hypothesized explanatory variables and the response variable, the response variable of interest must be measured while only a single hypothesized explanatory variable is changed and all other variables are held constant.[1] Moreover, the same response variable must be measured while any explanatory variable is varied. While this point may seem obvious, surprisingly most popular image databases fail

[1]Modeling image formation or the generative process of any data requires identifying key causal factors. A high correlation between two variables need not imply causation between the variables. Some observed associations between two variables are due to a cause-and-effect relationship between these variables, but others may be explained by so called lurking variables. (Note that if a variable $x$ explains or causes changes in another variable $y$, the variable $x$ is known as an *explanatory variable* and $y$ is known as a *response variable*. A response variable is directly measurable, whereas the explanatory variable is not, and it is sometimes referred to as a *hidden variable* in machine learning. A *lurking variable* is one not considered among the explanatory or response variables, but which nonetheless impacts the relationship between explanatory and response variables.) An observed correlation between variables may be the result of a *common response* to a lurking variable that causes changes in both the hypothesized explanatory variable and the response variable. *Confounding* between two variables, either explanatory or lurking, means that we cannot distinguish between the effects of the two variables on the response variable.

to maintain such rigor.

In our multilinear approach, our training method can in principle be extended to deal with incomplete datasets, say, through imputation. However, we believe that any data acquisition method that does not measure how a response variable changes with respect to each of the explanatory variables is flawed and facial recognition results obtained from the use of such a database are suspect.

In the case of facial data, most so-called PIE (pose (view), illumination, and expression) databases are either incomplete or suffer serious acquisition flaws. For example, the CMU PIE database [Sim et al. 2001] is an incomplete database of facial images of 68 subjects in front of varying backgrounds. It contains two major partitions, the first with 13 pose and 4 expression variations only, the second with 13 pose and 43 illumination variations. Thus, there is no consistent variation in illumination and expression. The biggest problem with this database, however, is that the background (a lurking variable) can change from one pose to the next. In an attempt to mitigate this problem, tight cropping windows have been placed around each face. Unfortunately, this creates additional complexity. When placing a tight window around each face, features are no longer in correspondence. Therefore, a pixel in the middle of the image measures information associated with nose shape for frontal images, while in another image where an individual is photographed from a side view, the same pixel might now measure eye information. Thus, the effects of face geometry and the effects of pose cannot be distinguished. This problem can be ameliorated by pre-processing each image, segmenting each face in an image, bringing facial features back into correspondence and replacing the background.[2]

The Yale facial image database [Georghiades et al. 2001] contains images of 10 individuals photographed under 9 poses and 64 illuminations, but the background again changes for each pose. Therefore, like the CMU dataset, this dataset requires preprocessing to mitigate the confounding.

The Weizmann Institute facial image dataset [Moses et al. 1996], which we use in our work, is the only true PIE database that captures images of every person under every possible combination of pose, illumination, and expression. It includes images for 28 people, 5 views, 4 illuminations and 3 expressions, for a total of 1680 images. Facial images were captured by varying each explanatory variable—the person, the pose, the illumination, and the expression—in turn while keeping the other explanatory variables fixed, and without modifying any other lurking variables that would affect the response variable—the acquired facial image. All the images have an identical and uniform background. Furthermore, these images were captured such that the eyes are located vertically along the same scan line in an image and the midpoint between the eyes is horizontally centered in the image regardless of view or person. Thus, facial features are in near correspondence across the different people and poses.

Another properly acquired facial image database, which we also use, is one that we synthesized using the full 3D scans of the heads of 100 subjects, recorded using a Cyberware$^{\text{TM}}$ 3030PS laser scanner, which was collected by the University of Freiburg [Blanz and Vetter 1999]. Using the Maya renderer, we rendered images for each of these 100 subjects under 15 poses and 15 illumination conditions. All the images were generated such that the eyes are aligned along the same scan line and the midpoint

[2] A background that changes from one person to another, or whose appearance changes when the pose changes, introduces spurious variance into the observational data.

between the eyes is horizontally centered in the image. Each subject is imaged from 15 different views ($\theta = -35°$ to $+35°$ in $5°$ steps on the horizontal plane $\phi = 0°$) under 15 different illuminations ($\theta = -35°$ to $+35°$ in $5°$ steps on an inclined plane $\phi = 45°$), yielding a total of 22,500 images.

# C
# On Motion Capture Data Processing

This appendix presents the details of the frame coordinate transformations of the limbs with respect to to lab coordinate system, the relative orientation of limbs in the kinematic chain, and the solution of the inverse kinematic equations to compute the joint angles (see [Apkarian et al. 1989]).

## Global Coordinate Transformations

The transformation matrix that represents the pelvis frame $\mathbf{F}_p$ with respect to the lab frame $\mathbf{F}_L$ is defined as

$$\mathbf{T}_{\text{pelvis}}^{\text{lab}} = \begin{bmatrix} \hat{\mathbf{x}}_p & \hat{\mathbf{y}}_p & \hat{\mathbf{z}}_p & \mathbf{t}_p \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.1}$$

where the overstruck carets denote unit vectors and where

$$\begin{aligned} \hat{\mathbf{x}}_p &= (\mathbf{m}_3 - \mathbf{m}_1)/\|\mathbf{m}_3 - \mathbf{m}_1\|, \\ \hat{\mathbf{y}}_p &= \hat{\mathbf{x}}_p \times (\mathbf{m}_2 - \mathbf{m}_1)/\|\mathbf{m}_2 - \mathbf{m}_1\|, \\ \hat{\mathbf{z}}_p &= \hat{\mathbf{x}}_p \times \hat{\mathbf{y}}_p, \\ \mathbf{t}_p &= \mathbf{m}_3. \end{aligned}$$

Similarly, we define the matrix that transforms the thigh frame $\mathbf{F}_t$ with respect to the lab frame $\mathbf{F}_L$ as

$$\mathbf{T}_{\text{thigh}}^{\text{lab}} = \begin{bmatrix} \hat{\mathbf{x}}_t & \hat{\mathbf{y}}_t & \hat{\mathbf{z}}_t & \mathbf{t}_t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.2}$$

where

$$\begin{aligned}
\hat{\mathbf{x}}_t &= (\mathbf{m}_5 - \mathbf{m}_3)/\|\mathbf{m}_5 - \mathbf{m}_3\|,\\
\hat{\mathbf{z}}_t &= \hat{\mathbf{x}}_t \times (\mathbf{m}_4 - \mathbf{m}_5)/\|\mathbf{m}_4 - \mathbf{m}_5\|,\\
\hat{\mathbf{y}}_t &= \hat{\mathbf{z}}_t \times \hat{\mathbf{x}}_t,\\
\mathbf{t}_t &= \mathbf{m}_5.
\end{aligned}$$

We define the matrix that transforms the shank frame $\mathbf{F}_s$ with respect to the lab frame $\mathbf{F}_L$ as

$$\mathbf{T}_{\text{shank}}^{\text{lab}} = \begin{bmatrix} \hat{\mathbf{x}}_s & \hat{\mathbf{y}}_s & \hat{\mathbf{z}}_s & \mathbf{t}_s \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.3}$$

where

$$\begin{aligned}
\hat{\mathbf{x}}_s &= (\mathbf{m}_7 - \mathbf{m}_5)/\|\mathbf{m}_7 - \mathbf{m}_5\|,\\
\hat{\mathbf{y}}_s &= \hat{\mathbf{x}}_s \times (\mathbf{m}_6 - \mathbf{m}_5)/\|\mathbf{m}_6 - \mathbf{m}_5\|,\\
\hat{\mathbf{z}}_s &= \hat{\mathbf{x}}_s \times \hat{\mathbf{y}}_s,\\
\mathbf{t}_s &= \mathbf{m}_7.
\end{aligned}$$

We define the matrix that transforms the foot frame with respect to the lab frame $\mathbf{F}_L$, through which the ankle angles are obtained, as

$$\mathbf{T}_{\text{foot}}^{\text{lab}} = \begin{bmatrix} \hat{\mathbf{x}}_f & \hat{\mathbf{y}}_f & \hat{\mathbf{z}}_f & \mathbf{t}_f \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.4}$$

where

$$\begin{aligned}
\hat{\mathbf{z}}_f &= (\mathbf{m}_8 - \mathbf{m}_9)/\|\mathbf{m}_8 - \mathbf{m}_9\|,\\
\hat{\mathbf{y}}_f &= \hat{\mathbf{z}}_f \times (\mathbf{m}_8 - \mathbf{m}_7)/\|\mathbf{m}_8 - \mathbf{m}_7\|,\\
\hat{\mathbf{x}}_f &= \hat{\mathbf{y}}_f \times \hat{\mathbf{z}}_f,\\
\mathbf{t}_f &= \mathbf{m}_8.
\end{aligned}$$

Finally, we define the transformation matrix of the toe frame with respect to the lab frame $\mathbf{F}_L$ as

$$\mathbf{T}_{\text{toe}}^{\text{lab}} = \begin{bmatrix} \hat{\mathbf{x}}_{\text{o}} & \hat{\mathbf{y}}_{\text{o}} & \hat{\mathbf{z}}_{\text{o}} & \mathbf{t}_{\text{o}} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{C.5}$$

where

$$\begin{aligned}
\hat{\mathbf{z}}_{\text{o}} &= (\mathbf{m}_8 - \mathbf{m}_9)/\|\mathbf{m}_8 - \mathbf{m}_9\|,\\
\hat{\mathbf{y}}_{\text{o}} &= \hat{\mathbf{z}}_{\text{o}} \times (\mathbf{m}_8 - \mathbf{m}_7)/\|\mathbf{m}_8 - \mathbf{m}_7\|,
\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{x}}_\mathrm{o} &= \hat{\mathbf{y}}_\mathrm{o} \times \hat{\mathbf{z}}_\mathrm{o},\\ \mathbf{t}_\mathrm{o} &= \mathbf{m}_9.\end{aligned}$$

## Relative Coordinate Transformations

The orientation of each segment relative to other segments is computed as follows:

$$\mathbf{T}_\mathrm{pelvis}^\mathrm{thigh} = \mathbf{T}_\mathrm{pelvis}^\mathrm{lab}\mathbf{T}_\mathrm{lab}^\mathrm{thigh}, \tag{C.6}$$

$$\mathbf{T}_\mathrm{thigh}^\mathrm{shank} = \mathbf{T}_\mathrm{thigh}^\mathrm{lab}\mathbf{T}_\mathrm{lab}^\mathrm{shank}, \tag{C.7}$$

$$\mathbf{T}_\mathrm{shank}^\mathrm{foot} = \mathbf{T}_\mathrm{shank}^\mathrm{lab}\mathbf{T}_\mathrm{lab}^\mathrm{foot}, \tag{C.8}$$

$$\mathbf{T}_\mathrm{foot}^\mathrm{toe} = \mathbf{T}_\mathrm{foot}^\mathrm{lab}\mathbf{T}_\mathrm{lab}^\mathrm{toe}. \tag{C.9}$$

## Inverse Kinematics

The human body is modeled as an open kinematic mechanism consisting of a sequence of 10 rigid links connected by 3 spherical joints (Table C.1). The spherical joint allows for rotations around three axis and is thus modeled as a sequence of three single-axis revolute joints. The sequence of rotations for each joint is abduction/adduction, rotation in the frontal plane, internal/external rotation, rotation in the transverse plane, flexion/extension, and rotation in the sagittal plane.

Once the coordinate transformations have been obtained according to (C.6)–(C.9), the inverse kinematic solutions are required to compute the joint angles that would give rise to the transformations. Let $\theta^\mathrm{m}$ denote the measured angle $\theta$. Then, the generalized rotary joint, $\mathbf{A}_{i-1}^i$, is defined as

$$\mathbf{A}_{i-1}^i = \begin{bmatrix} \cos\theta_i^\mathrm{m} & -\sin\theta_i^\mathrm{m}\cos\phi_i & \sin\theta_i^\mathrm{m}\sin\phi_i & a_i\cos\theta_i^\mathrm{m} \\ \sin\theta_i^\mathrm{m} & \cos\theta_i^\mathrm{m}\cos\phi_i & -\cos\theta_i^\mathrm{m}\sin\phi_i & a_i\sin\theta_i^\mathrm{m} \\ 0 & \sin\phi_i & \cos\phi_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.10}$$

and

$$\begin{aligned}\mathbf{T}_\mathrm{lab}^\mathrm{pelvis} &= \mathbf{T}_0^3\\ &= \mathbf{A}_0^1\mathbf{A}_1^2\mathbf{A}_2^3, \qquad (C.11)\\ \mathbf{T}_\mathrm{pelvis}^\mathrm{thigh} &= \mathbf{T}_3^6\\ &= \mathbf{A}_3^4\mathbf{A}_4^5\mathbf{A}_5^6, \qquad (C.12)\\ \mathbf{T}_\mathrm{thigh}^\mathrm{shank} &= \mathbf{T}_6^9\\ &= \mathbf{A}_6^7\mathbf{A}_7^8\mathbf{A}_8^9, \qquad (C.13)\\ \mathbf{T}_\mathrm{shank}^\mathrm{foot} &= \mathbf{T}_9^{12}\end{aligned}$$

| Link $i$ | Frame $i-1$ | $\phi_i$ | $\theta_i^{\mathrm{m}}$ | $a_i$ | $d_i$ |
|---|---|---|---|---|---|
| 1 | 0 | 90 | $90+\theta_1$ | 0 | 0 |
| 2 | 1 | 90 | $-90+\theta_2$ | 0 | 0 |
| 3 | 2 | -90 | $90+\theta_3$ | $\\|\mathbf{m}_1-\mathbf{m}_3\\|$ | 0 |
| 4 | 3 | 90 | $90+\theta_4$ | 0 | 0 |
| 5 | 4 | 90 | $90+\theta_5$ | 0 | 0 |
| 6 | 5 | 90 | $90+\theta_6$ | $\\|\mathbf{m}_3-\mathbf{m}_5\\|$ | 0 |
| 7 | 6 | 90 | $90+\theta_7$ | 0 | 0 |
| 8 | 7 | 90 | $-90+\theta_8$ | 0 | 0 |
| 9 | 8 | -90 | $90+\theta_9$ | $\\|\mathbf{m}_5-\mathbf{m}_7\\|$ | 0 |
| 10 | 9 | 90 | $90+\theta_{10}$ | 0 | 0 |
| 11 | 10 | 90 | $90+\theta_{11}$ | 0 | 0 |
| 12 | 11 | 90 | $90+\theta_{12}$ | $\\|\mathbf{m}-\mathbf{m}_9\\|$ | 0 |
| 13 | 12 | 0 | 0 | 0 | $\\|\mathbf{m}-\mathbf{m}_8\\|$ |
| 14 | 13 | 0 | 0 | 0 | $-\\|\mathbf{m}_8-\mathbf{m}_9\\|$ |
| 15 | 14 | 0 | 0 | 0 | 0 |

*Table C.1: Denavit-Hattenberg notation for the kinematic chain of a leg. The common normal between two joint axes $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ is $\mathbf{x}_i = \mathbf{z}_{i-1} \times \mathbf{z}_i$; the angle $\theta_i^{\mathrm{offset}} = \theta_i^{\mathrm{m}} - \theta_i$ is from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$ around $\mathbf{z}_{i-1}$; the angle $\theta_i$ is the actual angle rotated by a joint; $\phi$ is the angle from $\mathbf{z}_{i-1}$ to $\mathbf{z}_i$ around $\mathbf{x}_i$; the displacement $a_i$ is from $\mathbf{F}_{i-1}$ to $\mathbf{F}_i$ along the common normal; $d_i$ is the joint displacement, i.e., the distance along $\mathbf{z}_{i-1}$ between two consecutive common normals; $\mathbf{m} = [(\mathbf{m}_7 - \mathbf{m}_8)\cdot\hat{\mathbf{m}}_{98}]\hat{\mathbf{m}}_{98} + \mathbf{m}_8$, where $\hat{\mathbf{m}}_{98} = (\mathbf{m}_9 - \mathbf{m}_8)/\|\mathbf{m}_9 - \mathbf{m}_8\|$.*

$$= \mathbf{A}_9^{10}\mathbf{A}_{10}^{11}\mathbf{A}_{11}^{12}. \tag{C.14}$$

For example, the pelvis angles $\theta_1, \theta_2, \theta_3$ can be solved for as follows:

$$\theta_1 = \arctan\left(\frac{-\mathbf{T}_{\mathrm{lab}}^{\mathrm{pelvis}}(1,2)}{\mathbf{T}_{\mathrm{lab}}^{\mathrm{pelvis}}(2,2)}\right), \tag{C.15}$$

$$\theta_3 = \arctan\left(\frac{\mathbf{T}_{\mathrm{lab}}^{\mathrm{pelvis}}(3,1)}{\mathbf{T}_{\mathrm{lab}}^{\mathrm{pelvis}}(3,3)}\right), \tag{C.16}$$

$$\theta_2 = \arctan\left(\frac{\mathbf{T}_{\mathrm{lab}}^{\mathrm{pelvis}}(3,2)}{\mathbf{T}_{\mathrm{lab}}^{\mathrm{pelvis}}(3,1)}\sin\theta_3\right). \tag{C.17}$$

Similar solutions can be obtained for the other joint angles.

# Bibliography

ALEXANDER, C. 2001. *Market Models: A Guide to Financial Data Analysis*. Wiley, Chichester, UK. 107

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphics 22*, 3 (July), 587–594. 108

ALTER, O., BROWN, P. O., AND BOTSTEIN, D. 2000. Singular value decomposition for genome-wide expression data processing and modeling. *Proc. National Academy of Sciences 9*, 18 (August), 10101–10106. 107

APKARIAN, J., NAUMANN, S., AND CAIRNS, B. 1989. A three-dimensional kinematic and dynamic model of the lower limb. *Journal of Biomechanics 22*, 2, 143–155. 119

BACK, A. D., AND WEIGEND, A. S. 1997. A first application of independent component analysis to extracting structure from stock returns. *International Journal on Neural Systems 8*, 4, 473–484. 107

BADER, B. W., AND KOLDA, T. G. 2007. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing 30*, 1 (December), 205–231. 86

BARTLETT, M. S., MOVELLAN, J. R., AND SEJNOWSKI, T. J. 2002. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks 13*, 6, 1450–1464. 17, 34, 91

BARTLETT, M. S. 2001. *Face Image Analysis by Unsupervised Learning*. Kluwer Academic, Boston. 17, 91

BELHUMEUR, P. N., HESPANHA, J., AND KRIEGMAN, D. J. 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7 (July), 711–720. 16, 80

BELKIN, M., AND NIYOGI, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15*, 6, 1373–1396. 45

BELTRAMI, E. 1873. Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Universita 11*, 98–106. 11

BENABDELKADER, C., CUTLER, R., AND DAVIS, L. 2002. Motion-based recognition of people in eigengait space. In *Proc. 5th IEEE International Conf. on Automatic Face and Gesture Recognition*, 267–274. 17

BLANZ, V., AND VETTER, T. A. 1999. Morphable model for the synthesis of 3D faces. In *Proc. ACM SIGGRAPH 99 Conf.*, 187–194. 70, 71, 108, 117

BRAND, M., AND HERTZMANN, A. 2000. Style machines. In *Proc. ACM SIGGRAPH 2000 Conf.*, New Orleans, LA, 183–192. 18, 97

BRAND, M. 2002. Incremental singular value decomposition of uncertain data with missing values. In *Proc. 7th European Conf. on Computer Vision*, Springer, vol. 2350 of *Lecture Notes in Computer Science*, 707–720. 108

BREGLER, C., AND MALIK, J. 1998. Tracking people with twists and exponential maps. In *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, 8–15. 17

BRO, R., ACAR, E., AND KOLDA, T. G. 2008. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics 22* (February), 135–140. 90

CADIMA, J., AND JOLLIFFE, I. 1995. Loadings and correlations in the interpretation of principal components. *Applied Statistics 22*, 203–214. 108

CAI, D., HE, X., WEN, J.-R., HAN, J., AND MA., W.-Y. 2006. Support tensor machines for text categorization. Tech. Rep. UIUCDCS-R-2006-2714, University of Illinois at Urbana-Champaign, Department of Computer Science, April. 15

CARROLL, J. D., AND CHANG, J. J. 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition. *Psychometrika 35*, 283–319. 13, 27

CARROLL, J. D., PRUZANSKY, S., AND KRUSKAL, J. B. 1980. CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika 45*, 3–24. 22, 23

CATELL, R. B. 1944. Parallel proportional profiles and other principles for determining the choice of factors by rotation. *Psychometrika 9*, 267–283. 13

CHAM, D. D. T. J., AND REHG, J. 1999. Recovery of 3D articulated motion from 2D correspondences. Tech. rep., Compaq Cambridge Research Lab, Cambridge, MA. 17

CHELLAPPA, R., WILSON, C. L., AND SIROHEY, S. 1995. Human and machine recognition of faces: A survey. *Proceedings of the IEEE 83*, 5 (May), 705–740. 1, 16, 79

CHEN, S. E., AND WILLIAMS, L. 1993. View interpolation for image synthesis. In *Proc. of ACM SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 279–288. 15

CHEN, S. E. 1995. Quicktime VR: An image-based approach to virtual environment navigation. In *Proc. of ACM SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 29–38. 15

CHENNUBHOTLA, C., AND JEPSON, A. 2001. Sparse PCA (S-PCA): Extracting multi-scale structure from data. In *IEEE Proc. International Conf. on Computer Vision (ICCV)*, Vancouver, Canada, 641–647. 108

COOTES, T. F., TAYLOR, C. J., COOPER, D. H., AND GRAHAM, J. 1995. Active shape models: Their training and application. *Computer Vision and Image Understanding 61*, 1 (January), 38–59. 108

COOTES, T. F., EDWARDS, G. J., AND TAYLOR, C. J. 1998. Active appearance models. In *Proc. Fifth European Conf. on Computer Vision*, Springer, vol. 1406 of *Lecture Notes in Computer Science*, 484–498. 108

COOTES, T., EDWARDS, G., AND TAYLOR, C. 2001. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 6, 681–685. 38

COPPI, R., AND BOLASCO, S., Eds. 1989. *Multiway Data Analysis*. North-Holland, Amsterdam. 11

CUTTING, J. E., AND KOZLOWSKI, L. T. 1977. Recognizing friends by their walk: Gait perception without familiarity cues. *Bull. Psychonometric Soc 9*, 353–356. 17

CUTTING, J. E., PROFFITT, D. R., AND KOZLOWSKI, L. T. 1978. A biomechanical invariant for gait perception. *Journal of Experimental Psychology: Human Perception and Performance 4*, 3, 357–372. 17

DAI, G., AND YEUNG, D. Y. 2006. Tensor embedding methods. In *Proc. of the 21st National Conf. on Artificial Intelligence (AAAI)*, Boston, MA, 330–335. 15

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Trans. on Graphics 18*, 1–34. 16, 48, 49, 94

D'ASPREMONT, A., GHAOUI, L. E., JORDAN, M. I., AND LANCKRIET, G. R. G. 2004. A direct formulation for sparse PCA using semidefinite programming. In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, BC, Canada, vol. 17. 108

D'AUBIGNY G., AND E., P. 1989. Some optimality properties of the generalization of the Tucker method to the analysis of $n$-way tables with specified metrics. In *Multiway Data Analysis*, R. Coppi and S. Bolasco, Eds. North-Holland, Amsterdam, 53–64. 11

DAVIS, J., AND GAO, H. 2003. Recognizing human action efforts: An adaptive three-mode PCA framework. In *Proc. IEEE International Conf. on Computer Vision, (ICCV)*, Nice, France, 1463–1469. 14

DE LATHAUWER, L., DE MOOR, B., AND VANDEWALLE, J. 2000. A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications 21*, 4, 1253–1278. 11, 22, 23

DE LATHAUWER, L., DE MOOR, B., AND VANDEWALLE, J. 2000. On the best rank-1 and rank-$(R_1, R_2, \ldots, R_n)$ approximation of higher-order tensors. *SIAM Journal of Matrix Analysis and Applications 21*, 4, 1324–1342. 11, 28, 29

DE LATHAUWER, L. 1997. *Signal Processing Based on Multilinear Algebra*. PhD thesis, Katholieke Univ. Leuven, Belgium. 11, 13, 40

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. of ACM SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, 11–20. 16

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., W., S., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. *Computer Graphics (ACM SIGGRAPH 00)*, 145–156. 16, 94

DENIS, J. B., AND DHORNE, T. 1989. Orthogonal tensor decompositon of 3-way tables. In *Multiway Data Analysis*, R. Coppi and S. Bolasco, Eds. North-Holland, Amsterdam, 31–37. 13, 14, 25, 27

ECKART, C., AND YOUNG, G. 1936. The approximation of one matrix by another of lower rank. *Psychometrika 1*, 211–218. 11, 20

ELGAMMAL, A., AND LEE, C. S. 2004. Separating style and content on a nonlinear manifold. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 04)*, vol. I, 478–485. 14

FRANC, A. 1989. Multiway arrays: Some algebraic remarks. In *Multiway Data Analysis*, R. Coppi and S. Bolasco, Eds. North-Holland, Amsterdam, 19–29. 11

FRANC, A. 1992. *Etude Algebrique des Multitableaux: Apports de l'Algebre Tensorielle*. These de doctorat, specialite statistique, Univ. de Montpellier II, Montpellier. 11

FRANGI, A., AND YANG, J. 2004. Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 1, 131–137. 92, 112

FREEMAN, W., AND TENENBAUM, J. 1997. Learning bilinear models for two-factor problems in vision. In *Proc. IEEE International Conf. on Computer Vision and Pattern Recognition*, 554–560. 13

FURUKAWA, R., KAWASAKI, H., IKEUCHI, K., AND SAKAUCHI, M. 2002. Appearance based object modelling using texture database: Acquisition, compression and rendering. In *Proc. 13th Eurographics Workshop on Rendering*, 257–266. 14, 16, 111

GEORGHIADES, A. S., BELHUMEUR, P. N., AND KRIEGMAN, D. J. 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence 23*, 6, 643–660. 117

GLEICHER, M., Ed. 2001. Making Motion Capture Useful, ACM SIGGRAPH 2001 Course 51. 97

GOLUB, G. H., AND KAHN, W. 1965. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis 2*, 205–224. 11

GOLUB, G. H., AND REINSCH, C. 1970. Singular value decomposition and least squares solution. *Numerische Mathematik 14*, 403–420. (Also appeared in vol. 53 pages 134-151). 11

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. *Computer Graphics 30*, 43–45. (Proc. of ACM SIGGRAPH 96). 16

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *ACM Transactions on Graphics 23*, 3 (Aug.), 522–531. 18

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. NeuroAnimator: Fast neural network emulation and control of physics-based models. In *Proc. ACM SIGGRAPH 98 Conf.*, 9–20. 18, 97

HAM, J., LEE, D. D., MIKA, S., AND SCHÖLKOPF, B. 2004. A kernel view of the dimensionality reduction of manifolds. In *Proc. 21st International Conf. on Machine learning*, 369–376. 46, 47

HAN, J. Y., AND PERLIN, K. 2003. Measuring bidirectional texture reflectance with a kaleidoscope. In *ACM Trans. on Graphics*, ACM, vol. 22, 741–748. Proc. of ACM SIGGRAPH 2003. 16, 95

HARSHMAN, R. 1970. Foundations of the PARAFAC procedure: Model and conditions for an explanatory factor analysis. Tech. Rep. UCLA Working Papers in Phonetics 16, University of California, Los Angeles, Los Angeles, CA, December. 13, 27

HARTLEY, R. I., AND ZISSERMAN, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press. 51, 52

HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York. 34

HE, X., CAI, D., AND NIYOGI, P. 2005. Tensor subspace analysis. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, Vancouver, Canada. 15

HSU, E., PULLI, K., AND POPOVIC, J. 2005. Style translation for human motion. *ACM Transactions on Graphics 24*, 3, 1082–1089. 14, 110

HUANG, P. S., HARRIS, C. J., AND NIXON, M. S. 1999. Recognising humans by gait via parametric canonical space. *Artificial Intelligence in Engineering 13*, 4 (October), 359–366. 17

HYVÄRINEN, A., KARHUNEN, J., AND OJA, E. 2001. *Independent Component Analysis*. Wiley, New York. 35

INMAN, V. T., RALSTON, H. J., AND TODD, F. 1981. *Human Walking*. Williams & Wilkins, Baltimore, MD. 100

JOHANSSON, G. 1974. Visual motion perception. *Scientific American* (June), 76–88. 17, 97

JORDAN, C. 1874. Memoire sur les formes bilineaires. *Journal de Mathematiques Pures et Appliquees, Deuxieme Series 19*, 35–54. 11

KAPTEYN, A., NEUDECKER, H., AND WANSBEEK, T. 1986. An approach to $n$-mode component analysis. *Psychometrika 51*, 2 (June), 269–275. 11, 28, 29

KOENDERINK, J. J., AND VAN DOORN, A. 1996. Illuminance texture due to surface mesostructure. *Journal of the Optical Society of America 13*, 3, 452–463. 16

KOLDA, T. G. 2001. Orthogonal tensor decompositions. *SIAM J. on Matrix Analysis and Applications 23*, 1, 243–255. 13, 14, 25, 27

KOUDELKA, M. L., MAGDA, S., BELHUMEUR, P. N., AND KRIEGMAN, D. J. 2003. Acquisition, compression, and synthesis of bidirectional texture functions. In *Proc. 3rd Int. Workshop on Texture Analysis and Synthesis*, Nice, France, 59–64. 16

KOZLOWSKI, L. T., AND CUTTING, J. E. 1977. Recognizing the sex of a walker from a dynamic point-light display. *Perception and Psychophysics 21*, 3, 575–580. 17

KRAMER, M. A. 1991. Nonlinear principal components analysis using autoassociative neural networks. *AIChE Journal 32*, 2 (February), 233–243. 17

KROONENBERG, P. M., AND DE LEEUW, J. 1980. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika 45*, 69–97. 11, 28, 29

KRUSKAL, J. B. 1989. Rank, decomposition, and uniqueness for 3-way and $n$-way arrays. In *Multiway Data Analysis*, R. Coppi and S. Bolasco, Eds. North-Holland, Amsterdam, 7–18. 14, 27

LARSON, G. J. W. 1992. Measuring and modeling anisotropic reflection. *Computer Graphics (Proc. of ACM SIGGRAPH 92) 26*, 2 (July), 265–272. 16

LAW, H. G., JR., C. W. S., HATTIE, J. A., AND MCDONALD, R. P., Eds. 1984. *Research Methods for Multimode Data Analysis*. Praeger Publishers. 11

LEE, S.-I., AND BATZOGLOU, S. 2003. Application of independent component analysis to microarrays. *Genome Biology 4*, 11, R76. 107

LEE, D. D., AND SEUNG, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature 401*, 6755 (October), 788–791. 108

LEE, T. W., LEWICKI, M. S., AND SEJNOWSKI, T. J. 2000. ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 10, 1078–1089. 43

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics 25*, 3 (July), 541–548. 107

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. *Computer Graphics 30*, 31–42. (Proc. of ACM SIGGRAPH 96). 16

LIANG, L., LIU, C., XU, Y.-Q., GUO, B., AND SHUM, H.-Y. 2001. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics 20*, 3 (July), 127–150. 107

LIU, X., YU, Y., AND SHUM, H.-Y. 2001. Synthesizing bidirectional texture functions for real-world surfaces. *ACM Trans. Graphics*, 97–106. (Proc. ACM SIGGRAPH 01). 16

MAGNUS, J. R., AND NEUDECKER, H. 1988. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons, New York, New York. 13

MALZBENDER, T., GELB, D., AND WOLTERS, H. 2001. Polynomial texture maps. *Computer Graphics (ACM SIGGRAPH 01)* (August), 519–528. 16, 51

MARIMONT, D. H., AND WANDELL, B. A. 1992. Linear models of surface and illuminance spectra. *J. Optical Society of America, A. 9*, 11, 1905–1913. 13

MATUSIK, W., PFISTER, H.-P., BRAND, M., AND MCMILLAN, L. 2003. A data driven reflectance model. *Computer Graphics (ACM SIGGRAPH 03) 22*, 3 (July), 759–769. 16

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: An image-based rendering system. In *Proc. of ACM SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, 39–46. 16

MESETH, J., MULLER, G., SATTLER, M., AND KLEIN, R. 2003. BTF rendering for virtual environments. In *Proc. Virtual Concepts 2003*, 356–363. 16

MOGHADDAM, B., AND PENTLAND, A. 1997. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 7 (July), 696–710. 16

MOGHADDAM, B., JEBARA, T., AND PENTLAND, A. 2000. Bayesian face recognition. *Pattern Recognition 33*, 11 (November), 1771–1782. 17

MOGHADDAM, B., WEISS, Y., AND AVIDAN, S. 2006. Spectral bounds for sparse PCA: Exact and greedy algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, vol. 18, 915–922. 108

MOSES, Y., EDELMAN, S., AND ULLMAN, S. 1996. Generalization to novel images in upright and inverted faces. *Perception 25*, 443–461. 65, 94, 117

MURASE, H., AND NAYAR, S. 1995. Visual learning and recognition of 3D objects from appearance. *Int. Journal of Computer Vision 14*, 1. 16, 17

NICODEMUS, F. E., RICHMON, J. C., HSIA, J. J., GINSBERG, I. W., AND LIMPERIS, T. 1977. *Geometric considerations and nomenclature for reflectance*, vol. 160 of *NBS Monograph.* National Bureau of Standards, Washington, DC, October. 16

NIYOGI, S., AND ADELSON, E. 1994. Analying and recognizing walking figures in xyt. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 469–474. 17

PENEV, P., AND ATICK, J. 1996. Local feature analysis: A general statistical theory for object representation. *Neural Systems 7*, 477–500. 108

RAO, C. R., AND MITRA, S. K. 1971. *Generalized Inverse of Matrices and its Applications.* Wiley, New York. 24

ROWEIS, S. T., AND SAUL, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (December), 2323–2326. 45, 76, 94

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2003. Efficient and realistic visualization of cloth. In *Proc. of the 14th Eurographics Workshop on Rendering*, Leuven, Belgium, 167–177. 16, 51, 60

SCHMIDT, E. 1907. Zur theorie de linearen und nichtlinearen integralgleichungen. 1. Entwicklung willkurlichen functionen nach system vorgeschriebener. *Mathematische Annalen 63*, 433–476. 20

SCHÖLKOPH, B., SMOLA, A., AND MULLER, K.-R. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation 10*, 5, 1299–1319. 17, 43, 94

SEITZ, S. M., AND DYER, C. R. 1996. View morphing: Synthesizing 3D metamorphoses using image transforms. In *Proc. of ACM SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, 21–30. 16

SHAKHNAROVICH, G., AND MOGHADDAM, B. 2004. Face recognition in subspaces. In *Handbook of Face Recognition*, S. Z. Li and A. K. Jain, Eds. Springer, New York, 141–168. 15, 16, 17

SHASHUA, A., AND HAZAN, T. 2005. Non-negative tensor factorization with applications to statistics and computer vision. In *Proc. International Conf. on Machine Learning (ICML'05)*, ACM Press, New York, NY, ACM Press, New York, NY, 792–799. 14, 111

SHASHUA, A., AND LEVIN, A. 2001. Linear image coding for regression and classification using the tensor-rank principle. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Hawai, I–42–49. 13, 111

SHEN, J., AND ISRAEL, G. 1989. A receptor model using a specific non-negative transformation technique for ambient aerosol. *Atmospheric Environment 23*, 10, 2289–2298. 108

SHUM, H. Y., CHAN, S. C., AND KANG, S. B. 2007. *Image-Based Rendering.* Springer, New York. 15

SIDENBLADH, H., BLACK, M., AND FLEET, D. 2000. Stochastic tracking of 3D human figures using 2D image motion. In *European Conf. on Computer Vision (ECCV)*, vol. 2, 702–718. 17

SIM, T., BAKER, S., AND BSAT, M. 2001. The CMU pose, illumination, and expression (PIE) database of human faces. Tech. Rep. CMU-RI-TR-01-02, Robotics Institute, Pittsburgh, PA, January. 117

SIROVICH, L., AND KIRBY, M. 1987. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A. 4*, 519–524. 16, 65, 68, 80, 91

STEWART, G. W. 1993. On the early history of the singular value decomposition. *SIAM Review 35*, 4, 551–566. 11

SUYKENS, F., VON BERGE, K., LAGAE, A., AND KLEIN, R. 2003. Interactive rendering with bidirectional texture functions. *Comp. Graphics Forum 22*, 3, 463–472. Proc. Eurographics. 16

TANAWONGSUWAN, R., AND BOBICK, A. 2001. Gait recognition from time-normalized joint-angle trajectories in the walking plane. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 726–731. 17

TENENBAUM, J. B., AND FREEMAN, W. T. 2000. Separating style and content with bilinear models. *Neural Computation 12*, 1247–1283. 13

TENENBAUM, J. B., DE SILVA, V., AND LANGFORD, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (December), 2319–2323. 45, 76

TEO, P., AND HEEGER, D. 1994. Perceptual image distortion. In *IEEE Conf. on Image Processing*, 982–986. 69

TERZOPOULOS, D., LEE, Y., AND VASILESCU, M. 2004. Model-based and image-based methods for facial image synthesis, analysis and recognition. In *Proc. Sixth IEEE International Conf. on Automatic Face and Gesture Recognition*, 3–8. 64

TIPPING, M. E., AND BISHOP, C. M. 1999. Mixtures of probabilistic principal component analysers. *Neural Computation 11*, 2, 443–482. 43

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. on Graphics 21*, 665–672. Proc. of ACM SIGGRAPH 2002. 16, 51

TUCKER, L. R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika 31*, 279–311. 11, 28

TURK, M. A., AND PENTLAND, A. P. 1991. Eigenfaces for recognition. *Journal of Cognitive Neuroscience 3*, 1, 71–86. 16, 65, 68, 80, 91

TURK, M. A., AND PENTLAND, A. P. 1991. Face recognition using eigenfaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Hawai, 586–590. 16, 68

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2002. Multilinear analysis for facial image recognition. In *Proc. Int. Conf. on Pattern Recognition*, Quebec City, vol. 2, 511–514. 8, 14, 15

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2002. Multilinear analysis of image ensembles: TensorFaces. In *Proc. European Conf. on Computer Vision (ECCV 2002)*, Copenhagen, Denmark, 447–460. 8, 14, 15

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2003. Learning multilinear models of image ensembles. In *Proc. Learning Workshop*, Snowbird, UT. 8, 14

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2003. Multilinear subspace analysis of image ensembles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Madison, WI, vol. II, 93–99. 8, 14

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2003. TensorTextures. In *ACM SIGGRAPH 2003 Conf. Abstracts and Applications*, San Diego, CA. 8, 14

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2004. Multilinear independent components analysis. In *Proc. Learning Workshop*, Snowbird, UT. 8, 14

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2004. TensorTextures: Multilinear image-based rendering. *ACM Transactions on Graphics 23*, 3 (August), 336–342. Proc. ACM SIGGRAPH 2004 Conf., Los Angeles, CA. 8, 14

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2005. Multilinear independent components analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Diego, CA, vol. I, 547–553. 8, 13, 14

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2005. Multilinear independent components analysis and multilinear projection operator for face recognition. In *Proc. Workshop on Tensor Decompositions and Applications*, Marseille, France. 8

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2007. Multilinear projection for appearance-based recognition in the tensor framework. In *Proc. 11th IEEE International Conf. on Computer Vision (ICCV'07)*, Rio de Janeiro, Brazil, 1–8. 8, 14, 86

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2007. Multilinear (tensor) ICA and dimensionality reduction. In *Independent Component Analysis and Signal Separation, Proc. 7th International Conf. on Independent Component Analysis (ICA 2007)*, Springer, vol. 4666 of *Lecture Notes in Computer Science*, 818–826. 8

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2007. Multilinear (tensor) image synthesis, analysis, and recognition [exploratory DSP]. *IEEE Signal Processing Magazine 24*, 6, 118–123. 8

VASILESCU, M. A. O. 2001. An algorithm for extracting human motion signatures. In *IEEE Conf. on Computer Vision and Pattern Recognition*, Hawai. 8, 14

VASILESCU, M. A. O. 2001. Human motion signatures for character animation. In *ACM SIGGRAPH 2001 Conf. Abstracts and Applications*, Los Angeles, 200. 8, 14

VASILESCU, M. A. O. 2002. Human motion signatures: Analysis, synthesis, recognition. In *Proc. Int. Conf. on Pattern Recognition*, Quebec City, vol. 3, 456–460. 8, 14

VASILESCU, M. A. O. 2006. Manifold decomposition and low dimensional parameterization. In *Proc. Learning Workshop*, Snowbird, UT. 8, 14, 78

VASILESCU, M. 2006. Incremental multilinear SVD. In *Proc. Conf. on ThRee-way methods In Chemistry And Psychology (TRICAP 06)*. 8

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIC, J. 2005. Face transfer with multilinear models. *ACM Transactions on Graphics (TOG) 24*, 3 (July), 426–433. 14, 110

WANG, H., AND AHUJA, N. 2004. Compact representation of multidimensional data using tensor rank-one decomposition. In *IEEE, International Conf. on Pattern Recognition (ICPR)*, I–44–47. 14, 111

WANG, H., AND AHUJA, N. 2005. Rank-R approximation of tensors using image-as-matrix representation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 346–353. 15, 110, 114

WANG, H., AND AHUJA, N. 2008. A tensor approximation approach to dimensionality reduction. *International Journal of Computer Vision 6*, 3 (March), 217–229. 110

WANG, L., WANG, X., TONG, X., LIN, S., HU, S., GUO, B., AND SHUM, H. 2003. View displacement mapping. *ACM Trans. Graphics 22*, 3, 334–339. Proc. ACM SIGGRAPH 03. 14, 107, 110

WANG, H., WU, Q., SHI, L., YU, Y., AND AHUJA, N. 2005. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics 24*, 3, 527–535. 14, 110, 114

WHITTLE, M. W. 1996. *Gait Analysis: An introduction*. Butterworth-Heinemann, Oxford. 98

XIA, J., YEUNG, D. Y., AND DAI, G. 2006. Local discriminant embedding with tensor representation. In *Proc. of the IEEE International Conf. on Image Processing (ICIP)*, Atlanta, GA, 929–932. 15

XU, D., YAN, S., ZHANG, L., TANG, X., ZHANG, H., AND LIU, Z. 2005. Concurrent subspaces analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA. 14, 15, 110

YANG, J., GAO, X., ZHANG, D., AND YANG, J. 2005. Kernel ICA: An alternative formulation and its application to face recognition. *Pattern Recognition 38*, 10, 1784–1787. 43, 94

YANG, M.-H. 2002. Kernel eigenfaces vs kernel fisherfaces: Face recognition using kernel methods. In *Proc. IEEE International Conf. on Automatic Face and Gesture Recognition*, Washington, DC, 215–220. 17

YE, J. 2005. Generalized low rank approximations of matrices. *Machine Learning 61*, 1, 167–191. 15, 92, 110, 112

ZHANG, T., AND GOLUB, G. H. 2002. Rank-one approximation to high order tensors. *SIAM Journal on Matrix Analysis and Applications 23*, 2, 534–550. 13

ZHANG, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. International Conf. on Computer Vision*. 51

ZHAO, W., CHELLAPPA, R., PHILLIPS, P., AND ROSENFELD, A. Face recognition: A literature survey. *ACM Computing Surveys 35*, 4 (Dec.), 399–458. 16

ZOU, H., HASTIE, T., AND TIBSHIRANI, R. 2006. Sparse principal component analysis. *Journal of Computational and Graphical Statistics 15*, 265–286. 108