Discriminant Nonnegative Tensor Factorization Algorithms

Stefanos Zafeiriou

***Abstract*—Nonnegative matrix factorization (NMF) has proven to be very successful for image analysis, especially for object representation and recognition. NMF requires the object tensor (with valence more than one) to be vectorized. This procedure may result in information loss since the local object structure is lost due to vectorization. Recently, in order to remedy this disadvantage of NMF methods, nonnegative tensor factorizations (NTF) algorithms that can be applied directly to the tensor representation of object collections have been introduced. In this paper, we propose a series of unsupervised and supervised NTF methods. That is, we extend several NMF methods using arbitrary valence tensors. Moreover, by incorporating discriminant constraints inside the NTF decompositions, we present a series of discriminant NTF methods. The proposed approaches are tested for face verification and facial expression recognition, where it is shown that they outperform other popular subspace approaches.**

***Index Terms*—Face verification, facial expression recognition, linear discriminant analysis, nonnegative matrix factorization (NMF), nonnegative tensor factorization (NTF), subspace techniques.**

I. Introduction

WHEN an object is represented using a linear combination of bases, there are two main directions. The first refers to dense coding. In computer vision, dense coding is related to object representation using a combination of dense bases (i.e., in case the image representations dense bases correspond to dense 2-D matrices). A very popular dense coding is the one that corresponds to bases images that are found by the application of principal component analysis (PCA) to facial images [1].

The other direction refers to sparse coding. In computer vision, sparse coding corresponds to object representation using bases with components that are spatially distributed without any connectivity. The representation using sparse bases was a first step towards the implementation of part-based representation [2]–[7]. Moreover, in [2], it is showed that linear sparse coding of natural images yielded features qualitatively very similar to the receptive fields of simple cells in primary visual cortex. Subsequently, the very closely related model of independent component analysis (ICA) [4] was introduced to give similar results [2], [3]. The problem with the above models [2], [4] is that they permit the existence of negative entries to the representation. This is in contrast with the fact that the firing rates of the simple cells in primary visual cortex are nonnegative [6].

The nonnegativity of the firing rates and the fact that the representation of an object into its parts is more naturally coded by using only additions between the different bases [8]–[12] lead to the introduction of nonnegative matrix factorization (NMF), proposed in [5] and [13] using either the least squares error or the Kullback–Leibler (KL) divergence for measuring the cost of the approximation. Over the past few years, the NMF algorithm and its alternatives have proven to be very useful for several problems, especially for facial image characterization and representation problems [13]–[20]. In [19], a general NMF method has been proposed using generalized Bregman distances and general multiplicative update rules for the factorization have been introduced.

NMF, like PCA [21], represents a facial image as a linear combination of basis images. NMF does not allow negative elements in either the basis images or the representation coefficients used in the linear combination of the basis images. Thus, it represents a facial image only by additions of weighted basis images. The nonnegativity constraints correspond better to the intuitive notion of combining facial parts to create a complete facial image. The bases of PCA are the Eigenfaces, resembling distorted versions of the entire face, while the bases of NMF are localized features that correspond better to the notion of facial parts. The belief that NMF produces local representations is mainly intuitive (i.e., addition of different nonnegative bases using nonnegative weights). Recently, theoretical work has been done [22] in order to determine whether NMF provides a correct decomposition into parts and at the same time a set of requirements has been defined. This set of requirements is quite restrictive and cannot be satisfied by all kinds of image databases (e.g., facial image databases) [20]. Nevertheless, the sparsity of NMF in various facial image characterization problems has been in many cases verified [13], [15], [20].

The NMF algorithm has attracted a significant interest in the scientific community [13]–[20]. Methods for enhancing the sparsity (i.e., produce more local bases) of NMF have been proposed in [7], [14], and [18]. One disadvantage of NMF is that the object images should be vectorized in order to find the nonnegative decomposition. This vectorization leads to information loss, since the local structure of the images is lost. In order to remedy this drawback, the 3-D nonnegative tensor factorization (NTF) has been proposed [23]–[25]. In [23] and [25], an image database is represented as a 3-valence tensor, i.e., a 3-D cube that has as slices the 2-D images. Update rules

Manuscript received January 09, 2008; revised May 30, 2008; accepted August 05, 2008. First published January 13, 2009; current version published February 06, 2009.

The author is with the Imperial College London, Department of Electrical and Electronic Engineering, Communications and Signal Processing Research Group, South Kensington Campus, London SW7 2AZ, U.K. (e-mail: s.zafeiriou@imperial.ac.uk).

Digital Object Identifier 10.1109/TNN.2008.2005293

for the factors, used in the decomposition, that guarantee a nonincreasing behavior for the costs of the nonnegative decomposition have been proposed. An other drawback of NMF that is remedied through NTF is that, in general, NMF decompositions are nonunique in contrast to NTF decompositions, which are essentially unique under some mild conditions, for 3-valence tensors [26], [27], while the uniqueness of tensor factorization, with valence greater than 3 is even easier to satisfy. In this paper, we extend the 3-D NTF to n-valence NTF, providing proofs of convergence.

NMF has been further extended to supervised alternatives, the so-called discriminant-NMF (DNMF) or Fisher-NMF (FNMF) methods [16], [17], [20], [28], [29] by incorporating discriminant costs in the decomposition (for simplicity reasons, we will refer to all these methods [16], [17], [20], [28], [29] as DNMF variants). The intuitive motivation behind DNMF methods is to extract bases that correspond to discriminant facial regions for facial expression recognition [16], face recognition [17], and facial identity verification [20]. The incorporation of constraints in the NMF optimization problem is an active research topic [30], [31].

In this paper, we start by generalizing the 3-valence NTF framework proposed in [25] to arbitrary valence NTF decomposition. Afterwards, we generalize the NMF method with generalized Bregman distances [19] using arbitrary valence nonnegative tensors. This results in the derivation of a set of general multiplicative update rules for NTF. When applying the proposed approach to a number of Bregman distances, we come up with closed-form solutions for the update rules of the factorization. Afterwards, we propose a novel supervised feature extraction and object representation method by extending the DNMF method using tensors. To do so, we present two frameworks. In the first one, we use the KL divergence for measuring the cost of the NTF approximation, which leads to the generalization of the DNMF method proposed in [20] using arbitrary valence tensors. Afterwards, we apply discriminant constraints in the NTF framework that uses generalized Bregman distances and we propose a general framework for discriminant nonnegative tensor factorization (DNTF). Moreover, we provide closed-form DNTF solutions for a variety of Bregman distances. Recently, tensor-based description of popular feature extraction methods, such as linear discriminant analysis (LDA), has gained much attention by the research community and the advantages of tensorization have been discussed [32], [33]. The proposed methods can be used for supervised decomposition of arbitrary nonnegative tensor representations. The presented DNTF methods have both the advantages of DNMF and NTF methods. Finally, following the reasoning in [20], we propose another novel discriminant scheme, i.e., NTF plus LDA.

Summarizing the contributions of this paper, they are as follows.

- The generalization of the NTF framework proposed in [25] using arbitrary valence tensors. This is treated in Section II-C. This generalization will help us derive the DNTF update rules with KL divergence.
- The presentation of a novel general NTF framework using generalized Bregman distances (in Section II-D). A very similar method to this has been independently developed in [49] and has been applied to musical instrument identification.
- The presentation of a novel DNTF framework by generalizing the method in [20] using arbitrary valence tensors (in Section III-B).
- The presentation of a novel general DNMF framework using generalized Bregman distances (in Section III-C).
- Novel feature extraction method by combing NTF with LDA (in Section IV).

The rest of this paper is organized as follows. The NMF and the n-valence NTF methods are outlined in Section II. The formulation of the DNTF algorithms is described in Section III. The way DNTF and NTF can be used in order to extract low-dimensional features is presented in Section IV. In the same section, NTF plus LDA schemes are also proposed. Experimental results are presented in Section V. Finally, conclusions are drawn in Section VI.

II. From Nonnegative Matrix Factorization to Nonnegative Tensor Factorization

In this section, we will briefly describe the way NMF is formulated and how it can be afterwards extended to NTF using Kruskal's tensors. In the following, let there be a database $\mathcal{U}$ of nonnegative vectors $\mathbf{x} \in \Re_+^F$ with a total of L vectors (or objects). Let also each vector $\mathbf{x}$ be a vectorized representation of a tensor $\mathbf{A} \in \Re_+^{d_1 \times, \ldots \times d_n}$. In the simple case when $\mathbf{A}$ is a tensor representation of an image, i.e., $\mathbf{A} \in \Re_+^{d_1 \times d_2}$, then $\mathbf{x}$ is a columnwise vectorization of $\mathbf{A}$.

A. Nonnegative Matrix Factorization

For two vectors $\mathbf{x} = [x_1 \ldots x_F]^T$ and $\mathbf{q} = [q_1 \ldots q_F]^T$, the KL divergence (or relative entropy) between them is defined as [5]

$$\mathrm{KL}(\mathbf{x}\|\mathbf{q}) \triangleq \sum_i \left(x_i \ln \frac{x_i}{q_i} + q_i - x_i \right). \quad (1)$$

It can be shown that, in general, KL divergence is nonnegative and equal to zero if and only if its two arguments are equal. The basic idea behind NMF is to approximate the image $\mathbf{x}$ by a linear combination of the elements of $\mathbf{h} \in \Re_+^M$ such that $\mathbf{x} \approx \mathbf{Zh}$, where $\mathbf{Z} \in \Re_+^{F \times M}$ is a nonnegative matrix, whose columns sum up to one. In order to measure the error of the approximation $\mathbf{x} \approx \mathbf{Zh}$, the $\mathrm{KL}(\mathbf{x}\|\mathbf{Zh})$ divergence can been used [5].

In order to apply NMF in the database $\mathcal{U}$, the matrix $\mathbf{X} \in \Re_+^{F \times L} = [x_{i,j}]$ should be constructed, where $x_{i,j}$ is the ith element of the jth vector in the database. In other words, the jth column of $\mathbf{X}$ is the $\mathbf{x}_j$ vector. NMF aims at finding two matrices $\mathbf{Z} \in \Re_+^{F \times M} = [z_{i,k}]$ and $\mathbf{H} \in \Re_+^{M \times L} = [h_{k,j}]$ such that

$$\mathbf{X} \approx \mathbf{ZH}. \quad (2)$$

The vector $\mathbf{x}_j$ after the NMF decomposition can be written as $\mathbf{x}_j \approx \mathbf{Zh}_j$, where $\mathbf{h}_j$ is the jth column of $\mathbf{H}$. Thus, the columns of the matrix $\mathbf{Z}$ can be considered as basis images and the vectors $\mathbf{h}_j$ as the corresponding weight vectors. The $\mathbf{h}_j$ vectors can also be considered as the projected vectors of a lower dimensional feature space for the original facial vector $\mathbf{x}_j$ [20].

The defined cost for the decomposition (2) is the sum of all KL divergences for all images in the database. This way, the following metric can be formed:

$$\begin{aligned} D_n(\mathbf{X}||\mathbf{ZH}) &= \sum_j \mathrm{KL}(\mathbf{x}_j||\mathbf{Zh}_j) \\ &= \sum_{i,j}\left(x_{i,j}\ln\left(\frac{x_{i,j}}{\sum_k z_{i,k}h_{k,j}}\right)+\sum_k z_{i,k}h_{k,j}-x_{i,j}\right) \end{aligned} \quad (3)$$

as the measure of the cost for factoring $\mathbf{X}$ into $\mathbf{ZH}$ [5].

The NMF factorization is the outcome of the following optimization problem:

$$\min_{\mathbf{Z},\mathbf{H}} D_n(\mathbf{X}||\mathbf{ZH}) \quad \text{subject to} \quad z_{i,k}\geq 0, h_{k,j}\geq 0, \qquad \sum_i z_{i,j}=1 \qquad \forall j. \quad (4)$$

NMF has nonnegative constraints on both the elements of $\mathbf{Z}$ and $\mathbf{H}$; these nonnegativity constraints permit the combination of multiple basis images in order to represent an image (e.g., a face) using only additions between the different bases. The constraint $\sum_i z_{i,j}=1$ is a convenient way of eliminating the degeneracy associated with the invariance of $\mathbf{ZH}$ under the transformation $\mathbf{Z}\rightarrow\mathbf{ZB}$, $\mathbf{H}\rightarrow\mathbf{B}^{-1}\mathbf{H}$ [13], for all the positive diagonal matrices $\mathbf{B}$. By using an auxiliary function and the expectation–maximization (EM) algorithm, update rules for $h_{k,j}$ and $z_{i,k}$ that guarantee a nonincreasing behavior of (3) can be found in [5].

B. Nonnegative Matrix Factorization With Generalized Bregman Distances

In [19], a general method for NMF using generalized Bregman distances has been proposed. Let $\phi:\mathcal{D}\rightarrow\Re$ be a continuously differentiable and strictly convex function defined on a closed, convex set $\mathcal{D}\subseteq\Re_+$. The Bregman distance between p and q is defined as[1]

$$D_\phi(p,q)=\phi(p)-\phi(q)-\psi(q)(p-q) \quad (5)$$

where $\psi(q)=d\phi(q)/dq$. The generalized Bregman distance for the decomposition $\mathbf{x}_i\approx\mathbf{Zh}_i$ is

$$\begin{aligned} &D_\phi(\mathbf{Zh}_i,\mathbf{x}_i) \\ &=\sum_j\left(\phi\left([\mathbf{Zh}_i]_j\right)-\phi(x_{i,j})-\psi(x_{i,j})\left([\mathbf{Zh}_i]_j-x_{i,j}\right)\right) \end{aligned} \quad (6)$$

[1]For notation compactness, we will use the same notation D_ϕ for denoting the generalized Bregman distances between scalars, i.e., $D_\phi(p,q)$ defined on $\mathcal{D}\subseteq\Re_+$, between vectors $D_\phi(\mathbf{p},\mathbf{q})$ defined on $\mathcal{D}\subseteq\Re_+^K$ and between arbitrary valence tensors $D_\phi(\mathbf{P},\mathbf{Q})$ defined on $\mathcal{D}\subseteq\Re_+^{d_1\times\ldots\times d_n}$.

and the total distance for the decomposition $\mathbf{X}\approx\mathbf{ZH}$

$$D_\phi(\mathbf{ZH},\mathbf{X})=\sum_i D_\phi(\mathbf{Zh}_i,\mathbf{x}_i). \quad (7)$$

Let us have $F(\mathbf{h}_i)=D_\phi(\mathbf{Zh}_i,\mathbf{x}_i)$. In [19], it has been proven that the following function:

$$\begin{aligned} W(\mathbf{h}_i,\tilde{\mathbf{h}}_i)=\sum_{k,j}\Bigg(&\lambda_{k,j}\phi\left(\frac{z_{k,j}h_{i,j}}{\lambda_{k,j}}\right)-\phi(x_{i,k}) \\ &+\psi(x_{i,k})\left([\mathbf{Zh}_i]_j-x_{i,k}\right)\Bigg) \end{aligned} \quad (8)$$

with $\lambda_{k,j}=z_{k,j}\tilde{h}_{i,j}/\sum_l z_{k,l}\tilde{h}_{i,l}$ is an auxiliary function for $F(\mathbf{h})$.

Now by letting $\partial W(\mathbf{h}_k,\tilde{\mathbf{h}}_k)/\partial h_{k,j}=0$ $\partial W(\mathbf{z}_k,\tilde{\mathbf{z}}_k)/\partial z_{i,k}=0$ and assuming a multiplicative separable ψ [i.e., $\psi(xy)=\psi(x)\psi(y)$], then the following update rules:

$$h_{k,j}^{(t)}=h_{k,j}{}^{(t-1)}\psi^{-1}\left(\frac{\sum_i\psi(x_{i,j})z_{i,k}{}^{(t-1)}}{\sum_i z_{i,k}{}^{(t-1)}\psi\left(\sum_l z_{i,l}{}^{(t-1)}h_{l,j}{}^{(t-1)}\right)}\right) \quad (9)$$

$$z_{i,k}^{(t)}=z_{i,k}{}^{(t-1)}\psi^{-1}\left(\frac{\sum_j\psi(x_{i,j})h_{k,j}{}^{(t)}}{\sum_i z_{i,k}{}^{(t-1)}\psi\left(\sum_l z_{i,l}{}^{(t-1)}h_{l,j}{}^{(t)}\right)}\right) \quad (10)$$

guarantee a nonincreasing behavior of the cost function while satisfying the nonnegativity constraints. Moreover, when the function ψ is not separable in a multiplicative way, then in this case, update rules for the decomposition can be also derived (one such example is the KL divergence [19]).

C. Nonnegative Tensor Factorization

In order to extend NMF to NTF, the concept of a matrix $\mathbf{X}\in\Re_+^{F\times L}$ (which is a 2-valence tensor) is extended, using Kruskal's tensor notation, to n-valence tensors. Every object $\mathbf{A}_i$ has a nonnegative representation as a $(n-1)$-valence tensor, i.e., $\mathbf{A}_i\in\Re_+^{d_1\times\ldots\times d_{n-1}}$. Thus, the object database is n-valence tensor $\mathbf{G}\in\Re_+^{d_1\times\ldots\times d_n}$ with $d_n=L$. The dimension d_j is indexed by i_j with $0\leq i_j\leq d_j$. For example, the most natural way to model a facial image database using a tensor is by a 3-valence tensor $\mathbf{G}\in\Re_+^{d_1\times d_2\times d_3}$, where $d_1\times d_2$ is the resolution of an image (i.e., height and width) and $d_3=L$ is the number of images in the database. We will consider the general case of n-valence tensors. A n-valence tensor $\mathbf{F}$ is of rank at most K if it can be expressed as a sum of K rank-1 Kruskal tensors, i.e., a sum of K n-fold outer-products: $\mathbf{F}=\sum_{m=1}^K\mathbf{u}_1^m\otimes\mathbf{u}_2^m\ldots\otimes\mathbf{u}_n^m$, where $\mathbf{u}_i^m\in\Re^{d_i}$.

NMF can be now extended to NTF by finding a number of valence-1 tensors so that the tensor $\mathbf{G}$ can be decomposed as

$$\mathbf{G}\approx\sum_{m=1}^K\mathbf{u}_1^m\otimes\mathbf{u}_2^m\ldots\otimes\mathbf{u}_n^m \quad (11)$$

with the elements $u^m_{i_j,j} \geq 0$, $\mathbf{u}^m_{i_j} = [u^m_{i_j,1}, \dots, u^m_{i_j,d_j}]$. In order to measure the error of the above approximation, the KL divergence, in the general n-valence tensor case, is used as

$$D\left(\mathbf{G} \| \sum_{m=1}^{K} \mathbf{u}_1^m \otimes \mathbf{u}_2^m \ldots \otimes \mathbf{u}_n^m\right) = \sum_{i_1,\dots,i_n} \left(\mathbf{G}_{i_1,\dots,i_n} \ln\left(\frac{\mathbf{G}_{i_1,\dots,i_n}}{\sum_{m=1}^{K} u^m_{i_1,1} \cdots u^m_{i_n,n}}\right) - \mathbf{G}_{i_1,\dots,i_n} + \sum_{m=1}^{K} u^m_{i_1,1} \ldots u^m_{i_n,n}\right). \quad (12)$$

The optimization problem of the NTF decomposition is

$$\min_{\mathbf{u}^m_{i_j}} D\left(\mathbf{G} \| \sum_{m=1}^{K} \mathbf{u}_1^m \otimes \mathbf{u}_2^m \ldots \otimes \mathbf{u}_n^m\right)$$
$$\text{subject to} \quad u^m_{i_j,j} \geq 0 \quad \forall\, i_j, j, m \quad \text{and} \sum_{i_j} u^m_{i_j,j} = 1 \quad \forall\, i_j, \quad j \neq n, m. \quad (13)$$

The constraint $\sum_{i_j} u^m_{i_j,j} = 1, j \neq n$, is added by following the same reasoning as the constraint $\sum_i z_{i,j} = 1$ in the case of the NMF. That is, the normalization of the matrices $\mathbf{U}_1, \dots, \mathbf{U}_{n-1}$ is a generalization of the normalization of the bases matrix in the NMF algorithms for eliminating the degeneracy associated with the invariance of $\mathbf{ZH}$ under the transformation $\mathbf{Z} \to \mathbf{ZB}$, $\mathbf{H} \to \mathbf{B}^{-1}\mathbf{H}$, where $\mathbf{B}$ is a diagonal matrix with positive diagonal elements and proving the convergence of NMF algorithms to stationary limit points [13], [34]–[36]. Moreover, the normalization used is similar to the one considered in [37] for conic coding (the convex coding requires the normalization of the weights matrix as well; in our case, the weights matrix is the matrix $\mathbf{U}_n$). It is clear that all the proposed algorithms converge with or without normalization, since the proofs of convergence do not depend on the normalization of the various factors. Nevertheless, we have experimentally verified that when using this normalization the limit point is closest to stationary solutions. Additional comments about these constraints are given in Section III-B.

The update rules that can guarantee a nonincreasing behavior of the cost (12) for the factors $u^m_{i_j,j}$ with $j \neq n$ are shown in (14) at the bottom of the page. The corresponding update rules for the factors $u^m_{i_n,n}$ are given by

$$u^m_{i_n,n}{}^{(t)} = u^m_{i_n,n}{}^{(t-1)} \sum_{i_1,\dots,i_{n-1}} \mathbf{G}_{i_1,\dots,i_n} \times \frac{u^m_{i_1,1} \cdots u^m_{i_{n-1},n-1}}{\sum_{l=1}^{K} u^l_{i_1,1} \cdots u^l_{i_{n-1},n-1} u^l_{i_n,n}{}^{(t-1)}}. \quad (15)$$

For $j \neq n$, we also normalize the terms $\acute{u}^m_{i_j,j}$ so as

$$u^m_{i_j,j}{}^{(t)} = \frac{\left(\acute{u}^m_{i_j,j}\right)^{(t)}}{\sum_{i_j} \left(\acute{u}^m_{i_j,j}\right)^{(t)}} \quad (16)$$

in order to obey the summation to one constraint.

A proof of the above statements is given in Appendix I. This proof will help us calculate the update rules for the DNTF that will be presented in the following section. An implementation of the NTF algorithm based on the KL divergence for 3-valence tensors using only matrix operations can be found in Appendix II.

When choosing 3-valence tensors, we meet the decomposition proposed in [23]. The 3-D NTF decomposition has proved to be more suitable for part-based object representation than NMF [23]. Examples (such as the decomposition of the Swimmer data set [22]), which demonstrate the superiority of the 3-D NTF over the NMF, can be also found in [23]. In [22], it has been pointed out that one of the reasons for choosing NTF over NMF is the fact that tensor factorizations under some mild conditions are unique for rank $n \geq 3$, in contrast to matrix factorizations that are, in general, nonunique [23].

D. Nonnegative Tensor Factorization Using Generalized Bregman Distances

In this section, we will analyze the NTF problem using generalized Bregman distances and afterwards we will propose discriminant NTF methods using these distances. The approaches proposed here are the generalization of the NMF methods that have been proposed in [19] and briefly described in Section II-B. A very similar method to this has been independently developed in [49] and has been applied to musical instrument identification. The problem is the same as (11) but now we will not use KL divergence; instead, we consider the general form for a Bregman distance

$$\min_{u^l_{i_j,j} \geq 0} D_\phi\left(\sum_l \mathbf{u}_1^l \otimes \ldots \otimes \mathbf{u}_n^l, \mathbf{G}\right). \quad (17)$$

The generalized Bregman distance can be expanded as

$$D_\phi\left(\sum_l \mathbf{u}_1^l \otimes \ldots \otimes \mathbf{u}_n^l, \mathbf{G}\right) = \sum_{i_1,\dots,i_n} D_\phi\left(\sum_l u^l_{i_1,1} \ldots u^l_{i_n,n}, \mathbf{G}_{i_1,\dots,i_n}\right). \quad (18)$$

$$\left(\acute{u}^m_{i_j,j}\right)^{(t)} = u^m_{i_j,j}{}^{(t-1)} \frac{\sum_{i_1,\dots,i_{j-1},i_{j+1},\dots,i_n} \mathbf{G}_{i_1,\dots,i_n} \frac{u^m_{i_1,1} \cdots \cdots u^m_{i_{j-1},j-1} u^m_{i_{j+1},j+1} \cdots u^m_{i_n,n}}{\sum_{l=1}^{K} u^l_{i_1,1} \cdots u^l_{i_j,j}{}^{(t-1)} \cdots u^l_{i_n,n}}}{\sum_{i_n} u^m_{i_n,n}}. \quad (14)$$

By applying the rule in (5), the distance (18) can be expanded as

$$D_\phi\left(\sum_l \mathbf{u}_1^l \otimes \ldots \otimes \mathbf{u}_n^l, \mathbf{G}\right) = \sum_{i_1,\ldots,i_n} \phi\left(\sum_l u_{i_1,1}^l \ldots u_{i_n,n}^l\right) - \sum_{i_1,\ldots,i_n} \phi(\mathbf{G}_{i_1,\ldots,i_n}) - \sum_{i_1,\ldots,i_n} \psi(\mathbf{G}_{i_1,\ldots,i_n}) \times \left(\sum_l u_{i_1,1}^l \ldots u_{i_n,n}^l - \mathbf{G}_{i_1,\ldots,i_n}\right). \quad (19)$$

Before we proceed, let us define the matrices $\mathbf{U}_j = [u_{i_j,j}^l] \in \Re^{d_j \times K}$ (or, equivalently, $\mathbf{U}_j = [\mathbf{u}_j^1 \ldots \mathbf{u}_j^K]\ \forall j \in \{1,\ldots,n\}$). This notation will also help us in the next sections.

In order to find a multiplicative rule for the factors $u_{k,j}^l$, we will define an auxiliary function for general Bregman distances. To do so, we start by observing that for all $j = \{1,\ldots,n\}$, we can rewrite (19) as (20) shown at the bottom of the page, where $\mathbf{G}_{i_j=k} \in \Re_+^{i_1\times\ldots\times i_{j-1}\times i_{j+1}\times\ldots\times i_n}$ is a tensor of valence $n-1$, which is the kth slice tensor of the n-valence tensor $\mathbf{G}$ with $i_j = k$.

Now let us define the function

$$F(\mathbf{u}_{k,j}) = D_\phi\left(\sum_l u_{k,j}^l \mathbf{u}_1^l \otimes \ldots \otimes \mathbf{u}_{j-1}^l \otimes \mathbf{u}_{j+1}^l \otimes \ldots \otimes \mathbf{u}_n^l, \mathbf{G}_{i_j=k}\right) \quad (21)$$

where the notation $\mathbf{u}_{k,j}$ holds for the vector that corresponds to the kth row of the matrix $\mathbf{U}_j$ (the corresponding notation for the kth column of $\mathbf{U}_j$ is $\mathbf{u}_j^k$). Using the definition of generalized Bregman distance (5), the cost $F(\mathbf{u}_{k,j})$ can be written as (22) shown at the bottom of the page.

Now we will generalize the NMF framework [19] for defining auxiliary functions. That is, the auxiliary function of $F(\mathbf{u}_{i_j,j})$ is the following:

$$W\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right) = \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \times \left(\sum_l \lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} \times \phi\left(\frac{u_{i_j,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,n}^l}{\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n}}\right) - \phi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right)\right) - \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \psi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right) \times \left(\sum_l u_{i_j,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,n}^l - \mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right) \quad (23)$$

where the $\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n}$ is defined as follows:

$$\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} = \frac{\tilde{u}_{i_j,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,n}^l}{\sum_m \tilde{u}_{i_j,j}^m u_{i_1,1}^m \ldots u_{i_{j-1},j-1}^m . u_{i_{j+1},j+1}^m \ldots u_{i_n,n}^m}. \quad (24)$$

The proof of the fact that $W(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j})$ is an auxiliary function for $F(\mathbf{u}_{i_j,j})$ can be found in Appendix III.

$$D_\phi\left(\sum_l \mathbf{u}_1^l \otimes \ldots \otimes \mathbf{u}_n^l, \mathbf{G}\right) = \sum_{k=1}^{d_j}\left(\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \phi\left(\sum_l u_{k,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,n}^l\right) - \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \phi(\mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},i_n}) - \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \psi(\mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},i_n}) \left(\sum_l u_{k,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,1}^l - \mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},i_n}\right)\right) = \sum_{k=1}^{d_j} D_\phi\left(\sum_l u_{k,j}^l \mathbf{u}_1^l \otimes \ldots \otimes \mathbf{u}_{j-1}^l \otimes \mathbf{u}_{j+1}^l \otimes \ldots \otimes \mathbf{u}_n^l, \mathbf{G}_{i_j=k}\right) \quad (20)$$

$$F(\mathbf{u}_{k,j}) = \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \left(\phi\left(\sum_l u_{k,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,n}^l\right) - \phi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},\ldots,i_n}\right)\right) - \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \psi(\mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},\ldots,i_n}) \left(\sum_l u_{k,j}^l u_{i_1,1}^l \ldots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \ldots u_{i_n,n}^l - \mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},\ldots,i_n}\right) \quad (22)$$

In order to derive the update rules for the factors $u^l_{i_j,j}$, we need to calculate the partial derivatives $\partial W/\partial u^l_{i_j,j}$, as shown at the end of Appendix III. We have

$$\begin{aligned}\frac{\partial W}{\partial u^l_{i_j,j}} &\\ = &\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \left(u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right)\\ &\times\psi\left(\frac{u^l_{i_j,j}}{\tilde{u}^l_{i_j,j}}\sum_m \tilde{u}^m_{i_j,j}u^m_{i_1,1}\cdots u^m_{i_{j-1},j-1}u^m_{i_{j+1},j+1}\cdots u^m_{i_n,n}\right)\\ &-\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\psi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right)\\ &\times u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}. \end{aligned}\tag{25}$$

As in [19], we are mainly interested in investigating the Bregman distances for which ψ is multiplicative separable [i.e., $\psi(xy)=\psi(x)\psi(y)$]

$$\begin{aligned}\psi\left(\frac{u^l_{i_j,j}}{\tilde{u}^l_{i_j,j}}\right)&\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\left(u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right)\\ &\times\psi\left(\sum_m \tilde{u}^m_{i_j,j}u^m_{i_1,1}\cdots u^m_{i_{j-1},j-1}u^m_{i_{j+1},j+1}\cdots u^m_{i_n,n}\right)\\ =&\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\psi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right)\\ &\times u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\end{aligned}\tag{26}$$

then the update rules that connect the iteration $t+1$ with the previous iteration t, for a multiplicative separable ψ, are shown in (27) at the bottom of the page.

For some special Bregman distances (such as KL divergence, Frobenius and Itakura–Saito distance), we have the following update rules. For the Frobenius norm (i.e., $\phi(x)=x^2/2$), as shown in (28) at the bottom of the page, an implementation of the NTF algorithm based on Frobenius norm for 3-valence tensors using only matrix operations can be found in Appendix IV. For the Itakura–Saito distance [i.e., $\phi(x)=-\log(x)$], we have (29) shown at the bottom of the page.

For the KL divergence, ψ is not multiplicative separable and is treated a bit differently from the others that have a multiplicative separable ψ (as shown at the end of Appendix III). The update rules are shown in (30) at the bottom of the page.

For all the above update rules, the factors $u^l_{i_j,j}$ for $j\neq n$ can undergo a similar normalization as (16) in order to obey the summation to one constraint.

III. From Discriminant Matrix Factorization to Discriminant Tensor Factorization

Both the costs of NMF and NTF decomposition treat uniformly all the objects in the database. That is, there is no-class information incorporated in the costs of the decomposition. In [16], [17], [20], [28], and [29], an alternative discriminant cost has been formulated by incorporating discriminant costs inside the cost function to be minimized. This procedure has lead to the so-called DNMF method. This decomposition has been motivated by the intuition of finding basis images that correspond to discriminant parts of faces. That is, discriminant costs (which have been the minimization of within-class variance and the maximization of between-class variance) have been incorporated in a form that has been used in many NMF methods (e.g., local NMF (LNMF) [14]). The bases of discriminant NMF

$$u^l_{i_j,j}{}^{(t)}=u^l_{i_j,j}{}^{(t-1)}\psi^{-1}\left(\frac{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\psi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right)\prod^n_{r\neq j}u^l_{i_r,r}}{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\prod^n_{r\neq j}u^l_{i_r,r}\psi\left(\sum_m u^m_{i_j,j}{}^{(t-1)}\prod^n_{r\neq j}u^l_{i_r,r}\right)}\right).\tag{27}$$

$$u^l_{i_j,j}{}^{(t)}=u^l_{i_j,j}{}^{(t-1)}\frac{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\prod^n_{m\neq j}u^l_{i_m,m}}{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\prod^n_{r\neq j}u^l_{i_r,r}\sum_m u^m_{i_j,j}{}^{(t-1)}\prod^n_{r\neq j}u^l_{i_r,r}}\tag{28}$$

$$u^l_{i_j,j}{}^{(t)}=u^l_{j,i_j}{}^{(t-1)}\frac{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\frac{\prod^n_{m\neq j}u^l_{i_m,m}}{\sum_m u^m_{i_j,j}{}^{(t-1)}\prod^n_{r\neq j}u^l_{i_r,r}}}{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\frac{\prod^n_{m\neq j}u^l_{i_m,m}}{\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}}}\tag{29}$$

$$u^l_{i_j,j}{}^{(t)}=u^l_{i_j,j}{}^{(t-1)}e^{\frac{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\prod^n_{r\neq j}u^l_{i_r,r}\log\frac{\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}}{\sum_m u^m_{i_j,j}{}^{(t-1)}\prod^n_{r\neq j}u^l_{i_r,r}}}{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\prod^n_{m\neq j}u^l_{i_m,m}}}\tag{30}$$

methods have been proved very useful for face verification, face recognition, and facial expression recognition [16], [17], [20], [28], [29]. For more details concerning the motivation behind the discriminant NMF methods and how these methods have been formulated, the interested reader may refer to [16], [17], [20], [28], and [29]. Especially, for intuitive and experimental verification of the fact that DNMF methods are well suited for facial expression description and recognition, the interested reader should refer to [29].

For formulating the DNMF and DNTF methods, let the database $\mathcal{U}$ be separated in R different disjoint object classes. For the rth class, the notation $\mathcal{U}_r$ is used. The class $\mathcal{U}_r$ contains N_r samples. For example, in our experiments, the different classes were facial classes for face verification or facial expression classes for facial expression recognition.

A. Discriminant Nonnegative Matrix Factorization

In order to formulate the DNMF method, we will use as an example the facial image characterization problems (i.e., face verification and facial expression recognition) that have been considered in our experiments. Now, let the matrix $\mathbf{X}$ that contains all the facial images of the database $\mathcal{U}$ be organized as follows. The jth column of the database $\mathbf{X}$ is the ρth image of the rth class. Thus, $j = \sum_{i=1}^{r-1} N_i + \rho$. The vector $\mathbf{h}_j$ that corresponds to the jth column of the matrix $\mathbf{H}$ is the coefficient vector for the ρth facial image of the rth class and will be denoted as $\boldsymbol{\eta}_\rho^{(r)} = [\eta_{\rho,1}^{(r)} \dots \eta_{\rho,M}^{(r)}]^T$. The mean vector of the vectors $\boldsymbol{\eta}_\rho^{(r)}$ for the class r is denoted as $\boldsymbol{\mu}^{(r)} = [\mu_1^{(r)} \dots \mu_M^{(r)}]^T$ and the mean of all classes as $\boldsymbol{\mu} = [\mu_1 \dots \mu_M]^T$. Then, the within-class scatter and between-class scatter matrices for the coefficient vectors $\mathbf{h}_j$ are defined as

$$\mathbf{S}_w = \sum_{r=1}^{R} \sum_{\rho=1}^{N_r} \left(\boldsymbol{\eta}_\rho^{(r)} - \boldsymbol{\mu}^{(r)}\right)\left(\boldsymbol{\eta}_\rho^{(r)} - \boldsymbol{\mu}^{(r)}\right)^T$$

and

$$\mathbf{S}_b = \sum_{r=1}^{R} N_r \left(\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}\right)\left(\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu}\right)^T. \quad (31)$$

A modified divergence has been constructed inspired from the minimization of the Fisher criterion [20]. The discriminant costs include the $\mathrm{tr}[\mathbf{S}_w]$ to be as small as possible while $\mathrm{tr}[\mathbf{S}_b]$ to be as large as possible. The discriminant cost function is given by

$$D_d(\mathbf{X}||\mathbf{Z}_D\mathbf{H}) = D_n(\mathbf{X}||\mathbf{Z}_D\mathbf{H}) + \gamma \mathrm{tr}[\mathbf{S}_w] - \delta \mathrm{tr}[\mathbf{S}_b] \quad (32)$$

where $\gamma \geq 0$ and $\delta \geq 0$ with $\gamma\delta \neq 0$. Following the same EM approach used by NMF [5] and LNMF [14] techniques, update rules can be found in [20].

B. Discriminant Nonnegative Tensor Factorization

Using the notion of Kruskal tensors, we can extend the DNMF method to the DNTF method. In DNMF, the discriminant constraints concern the coefficients $h_{i,j}$ of the decomposition. The problem now involves the selection of proper coefficients of the NTF decomposition on which the discriminant analysis should be applied. In order to answer this question, let us examine the decomposition of a 3-valence tensor. In this case, the tensor $\mathbf{G} \in \Re_+^{d_1 \times d_2 \times d_3}$ is a 3-D matrix that is built of slices that are the images $\mathbf{A}_1, \dots, \mathbf{A}_{d_3}$ and every image $\mathbf{A}_i \in \Re_+^{d_1 \times d_2}$. Let $\mathbf{U}_1 = [\mathbf{u}_1^1, \dots, \mathbf{u}_1^K]$, $\mathbf{U}_2 = [\mathbf{u}_2^1, \dots, \mathbf{u}_2^K]$, and $\mathbf{U}_3 = [\mathbf{u}_3^1, \dots, \mathbf{u}_3^K]$. The Khatri–Rao product of the two matrices $\mathbf{U}_1$ and $\mathbf{U}_2$ is defined as $\mathbf{U}_1 \odot \mathbf{U}_2 = [\mathbf{vec}(\mathbf{u}_1^1 \otimes \mathbf{u}_2^1) \dots \mathbf{vec}(\mathbf{u}_1^K \otimes \mathbf{u}_2^K)]$. Let $(\mathbf{U}_1 \odot \mathbf{U}_2)\mathbf{U}_3^T = [\mathbf{a}_1, \dots, \mathbf{a}_{d_3}]$, then each vector $\mathbf{a}_i$ is the vectorized image $\mathbf{A}_i$, i.e., the image $\mathbf{A}_i$ scanned columnwise. That is, each vectorized image $\mathbf{a}_i$ is a linear combination of the $\mathrm{vec}(\mathbf{u}_1^j \mathbf{u}_2^{j^T})$ with the coefficients of the decomposition taken from the ith row of the matrix $\mathbf{U}_3$. Thus, the weights of the representation are stored in the matrix $\mathbf{U}_3$, while the bases are found by combining the two matrices $\mathbf{U}_1$ and $\mathbf{U}_2$.

In the same way as in the DNMF decomposition (or like other methods, e.g., PCA plus LDA [38]), the discriminant analysis should be incorporated in the coefficients of the decomposition, which in the 3-valence tensor case correspond to the elements of the matrix $\mathbf{U}_3$. The jth row $u_{j,3}^1, \dots, u_{j,3}^K$ of the matrix $\mathbf{U}_3$ corresponds to the coefficients of the image $\mathbf{A}_j$. Following the same reasoning, we can generalize for the case of n-valence tensors that the matrix $\mathbf{U}_n \in \Re_+^{d_n \times K}$ has rows that correspond to coefficients of the decomposition of the objects $\mathbf{A}_i \in \Re_+^{d_1 \times d_2 \dots \times d_{n-1}}$. Let $\mathbf{v}_i = [u_{i_n,n}^1, \dots, u_{i_n,n}^K]$ be a column vector with the elements ith row of the matrix $\mathbf{U}_n$. This vector contains the representation coefficients of the object $\mathbf{A}_i$. Using the above observations, it can be shown that the constraint $\sum_{i_1,\dots,i_{n-1}} u_{i_1,1}^m \dots u_{i_{n-1},n-1}^m = 1$ is the generalization of the NMF constraint $\sum_i z_{i,j} = 1$. The above constraint can be expanded

$$\sum_{i_1,\dots,i_{n-1}} u_{i_1,1}^m \dots u_{i_{n-1},n-1}^m = \sum_{i_1} u_{i_1,1}^m \dots \sum_{i_{n-1}} u_{i_{n-1},n-1}^m = 1 \quad (33)$$

where it can be simplified to the constraint $\sum_{i_j} u_{i_j,j}^m = 1, \forall j \neq n$.

The objects $\mathbf{A}_i$ are separated to R different object classes. The coefficient vectors $\mathbf{v}_i$ are separated to R classes $(\mathcal{V}_1, \dots, \mathcal{V}_R)$ as well. The within- and between-class scatter matrices for these vectors are defined as

$$\acute{\mathbf{S}}_w = \sum_{r=1}^{R} \sum_{\mathbf{v} \in \mathcal{V}_r} \left(\mathbf{v} - \acute{\boldsymbol{\mu}}^{(r)}\right)\left(\mathbf{v}_i - \acute{\boldsymbol{\mu}}^{(r)}\right)^T$$

and

$$\acute{\mathbf{S}}_b = \sum_{r=1}^{R} N_r \left(\acute{\boldsymbol{\mu}}^{(r)} - \acute{\boldsymbol{\mu}}\right)\left(\acute{\boldsymbol{\mu}}^{(r)} - \acute{\boldsymbol{\mu}}\right)^T \quad (34)$$

where $\acute{\boldsymbol{\mu}}^{(r)} = [\acute{\mu}_1^{(r)}, \cdots, \acute{\mu}_{d_n}^{(r)}]$ is the mean vector of the class of vectors $\mathbf{v} \in \mathcal{V}_r$ and the $\acute{\boldsymbol{\mu}} = [\acute{\mu}_1, \cdots, \acute{\mu}_{d_n}]$ is the grand mean of $\mathcal{V}$. The proposed divergence with the discriminant constraints is

$$D_d(\mathbf{G}||\mathbf{Z}_D\mathbf{H}) = D\left(\mathbf{G}|| \sum_{m=1}^{K} \mathbf{u}_1^m \otimes \mathbf{u}_2^m \dots \otimes \mathbf{u}_n^m\right) + \gamma \mathrm{tr}[\acute{\mathbf{S}}_w] - \delta \mathrm{tr}[\acute{\mathbf{S}}_b] \quad (35)$$

and the corresponding optimization problem of the factorization is

$$\min_{\mathbf{u}^m_{i_j}} D_d\left(\mathbf{G}\|\sum_{m=1}^{K}\mathbf{u}_1^m\otimes\mathbf{u}_2^m\ldots\otimes\mathbf{u}_n^m\right)$$
$$\text{subject to}\quad \sum_{i_j} u^m_{i_j,j}=1 \qquad \forall\, i_j, \qquad j\neq n,m$$
$$\text{and } u^m_{i_j,j}\geq 0 \qquad \forall\, i_j,j,m. \quad (36)$$

The update rules that guarantee a nonincreasing behavior of (35), under nonnegativity constraints, for $j\neq n$, are given by (14) and (16). For $j=n$, let $\mathcal{F}_r=\{\sum_{\rho=1}^{r-1}N_\rho+1,\cdots,\sum_{\rho=1}^{r}N_\rho\}$. Then, for the objects of the rth class (i.e., $i_n\in\mathcal{F}_r$), the update rules are given by (37) shown at the bottom of the page, where $\acute{T}$ is given by

$$\acute{T}=(2\gamma+2\delta)\left(\frac{1}{N_r}\sum_{\lambda\neq i_n,\lambda\in\mathcal{F}_r}\left(u^m_{\lambda,n}\right)\right)-2\delta\acute{\mu}_m-1. \quad (38)$$

The proof of convergence is provided in Appendix V. The above procedure when having a 2-valence tensor $\mathbf{G}$ degenerates to the DNMF method in [20].

C. Discriminant Nonnegative Tensor Factorization Using Generalized Bregman Distances

Let us define the following discriminant cost using generalized Bregman distances:

$$D_{d,\phi}\left(\sum_{l=1}^{K}\mathbf{u}_1^l\otimes\mathbf{u}_2^l\ldots\otimes\mathbf{u}_n^l,\mathbf{G}\right)$$
$$=D_\phi\left(\sum_{l=1}^{K}\mathbf{u}_1^l\otimes\ldots\otimes\mathbf{u}_n^l,\mathbf{G}\right)+\gamma\mathrm{tr}[\mathbf{S}_w]-\delta\mathrm{tr}[\mathbf{S}_b]. \quad (39)$$

The optimization problem of a DNTF decomposition with generalized Bregman distances is to minimize $D_{d,\phi}(\sum_l\mathbf{u}_1^l\otimes\mathbf{u}_2^l\ldots\otimes\mathbf{u}_n^l,\mathbf{G})$ subject to the nonnegativity constraints for all the factors.

The function $D_{d,\phi}$ can be expanded as

$$D_{d,\phi}\left(\sum_{l=1}^{K}\mathbf{u}_1^l\otimes\mathbf{u}_2^l\ldots\otimes\mathbf{u}_n^l,\mathbf{G}\right)$$
$$=\sum_{i_j}\left(F\left(\mathbf{u}_{i_j,j}\right)+\frac{\gamma d_j}{\sum_m d_m}\mathrm{tr}[\acute{\mathbf{S}}_w]-\frac{\delta d_j}{\sum_m d_m}\mathrm{tr}[\acute{\mathbf{S}}_b]\right). \quad (40)$$

Let us define $\gamma_j=\gamma d_j/\sum_m d_m$ and $\delta_j=\delta d_j/\sum_m d_m$. Then, we can define the discriminant cost

$$F_d\left(\mathbf{u}_{i_j,j}\right)=F\left(\mathbf{u}_{i_j,j}\right)+\gamma_j\mathrm{tr}[\acute{\mathbf{S}}_w]-\delta_j\mathrm{tr}[\acute{\mathbf{S}}_b] \quad (41)$$

for which we should identify an auxiliary function.

It can be proven by using the results of Appendixes III and V that the discriminant cost (41) has the following auxiliary function:

$$W_{d,\phi}\left(\mathbf{u}_{i_j,j},\tilde{\mathbf{u}}_{i_j,j}\right)=W\left(\mathbf{u}_{i_j,j},\tilde{\mathbf{u}}_{i_j,j}\right)+\gamma_j\mathrm{tr}[\acute{\mathbf{S}}_w]-\delta_j\mathrm{tr}[\acute{\mathbf{S}}_b]. \quad (42)$$

Since $\mathrm{tr}[\acute{\mathbf{S}}_w]$ and $\mathrm{tr}[\acute{\mathbf{S}}_b]$ depend only on $u^l_{i_n,n}$, the update rules for the factors $u^l_{i_j,j}$ with $j=2,\ldots,n$ are given by (27).

For the factors $u^l_{i_n,n}$ and when we have a multiplicative separable ψ, the update rules can be found by solving (43) shown at the bottom of the page, as shown in Appendix VI. Equation (43) cannot provide us closed-form solutions for update rules for every multiplicative separable Bregman distance. Moreover, for the case of KL divergence, a closed-form solution cannot be found by following the above framework. Thus, for the KL divergence, only the framework in Section III-B can give us closed-form update rules. The factors $u^l_{i_j,j}$ for $j\neq n$ can undergo a similar normalization as (16) in order to obey the summation to one constraint.

For the Itakura–Saito distance, the update rules are given by

$$u^{l\ (t)}_{i_n,n}=\frac{A_2-B_2+\sqrt{(B_2-A_2)^2+4A_1B_1u^{l\ (t-1)}_{i_n,n}}}{2A_1} \quad (44)$$

$$u^{m\ (t)}_{i_n,n}=\frac{\acute{T}+\sqrt{\acute{T}^2+4\left(2\gamma-(2\gamma+2\delta)\frac{1}{N_r}\right)\sum_{i_1,\ldots,i_{n-1}}\mathbf{G}_{i_1,\ldots,i_n}\frac{u^m_{i_1,1}\ldots u^{m\ (t-1)}_{i_n,n}}{\sum_{l=1}^{K}u^l_{i_1,1}\ldots u^{l\ (t-1)}_{i_n,n}}}}{2\left(2\gamma-(2\gamma+2\delta)\frac{1}{N_r}\right)} \quad (37)$$

$$\psi\left(\frac{u^l_{i_n,n}}{\tilde{u}^l_{i_n,n}}\right)\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\left(u^l_{i_1,1}\ldots u^l_{i_{j-1},j-1},u^l_{i_{j+1},j+1}\ldots u^l_{i_n,n}\right)\psi\left(\sum_m\tilde{u}^m_{i_n,n}u^m_{1,i_1}\ldots u^m_{i_{j-1},j-1}u^m_{i_{j+1},j+1}\ldots u^m_{i_n,n}\right)$$
$$=\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\psi\left(\mathbf{G}_{i_1,\ldots,i_{j-1},k,i_{j+1},\ldots,i_n}\right)u^l_{i_1,1}\ldots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\ldots u^l_{i_n,n}$$
$$-\left(2\gamma-(2\gamma+2\delta)\frac{1}{N_r}\right)u^l_{i_n,n}+(2\gamma+2\delta)\left(\frac{1}{N_r}\sum_{\lambda\in\mathcal{F}_r}u^l_{\lambda,n}\right)-2\delta\acute{\mu}_l \quad (43)$$

where A_1, A_2, B_1, and B_2 are defined in Appendix VI. In case that $\mathbf{G}$ is a 2-valence tensor, the above procedure describes a general method for discriminant NMF using generalized Bregman distances.

IV. Feature Extraction and NTF Plus LDA Schemes

In this section, we will describe how low-dimensional features can be derived from the NTF and DNTF decompositions in test and training tensor object representations. Moreover, we will complete the discriminant tensor decomposition framework by introducing the NTF plus LDA framework following the same reasoning as [20], where the NMF plus LDA scheme has been proposed.

A. Feature Extraction

For feature extractions, we have generalized the feature extraction in NMF [15], [20] using tensor-based notation and representation. That is, in [15] and [20], a low-dimensional representation $\acute{\mathbf{x}}$ of the image vector $\mathbf{x}$ has been retrieved using correlation with the image bases matrix $\mathbf{Z}$, i.e., $\acute{\mathbf{x}} = \mathbf{Z}^T\mathbf{x}$. In the same manner, we can retrieve a low-dimensional representation of the tensor representation. Let $\mathbf{A}_i \in \Re_+^{d_1\times\ldots\times d_{n-1}}$ be a tensor representation. Then, by using the tensor decomposition $\mathbf{u}_1^m, \mathbf{u}_2^m, \ldots, \mathbf{u}_{n-1}^m$ with $m = 1, \ldots, K$, derived from the NTF or the DNTF method, a low-dimensional representation can be formed as $\tilde{\mathbf{A}}_i \in \Re_+^K$. The mth feature of the vector $\tilde{\mathbf{A}}_i$ can be found as

$$[\tilde{\mathbf{A}}_i]_m = \langle \mathbf{u}_1^m \otimes \mathbf{u}_2^m \ldots \otimes \mathbf{u}_{n-1}^m, \mathbf{A}_i \rangle \tag{45}$$

where $\langle \mathbf{B}, \mathbf{C} \rangle$ for two tensors $\mathbf{B}, \mathbf{C} \in \Re_+^{d_1\times\ldots\times d_{n-1}}$ is the correlation

$$\langle \mathbf{B}, \mathbf{C} \rangle = \sum_{i_1,i_2,\ldots,i_{n-1}} \mathbf{B}_{i_1,i_2,\ldots,i_{n-1}} \mathbf{C}_{i_1,i_2,\ldots,i_{n-1}}. \tag{46}$$

Another way to extract features is by producing orthogonal bases from the NTF or the DNTF decomposition (as has been already proposed for NMF [50], [51]). Orthogonal bases have been proven very useful in enhancing the performance of well-known feature extraction techniques, such as the ones in [39] and [40], in classification problems. We will describe the method using the 3-valence tensor case and it can be easily generalized in the n-valence case. Let us have a 3-valence tensor $\mathbf{G} \in \Re_+^{d_1\times d_2\times d_3}$ where d_3 is the number of objects in the training database and every object is represented by a $d_1 \times d_2$ matrix (2-valence tensor). Let the decomposition $\mathbf{U}_1 \in \Re_+^{d_1\times k}$, $\mathbf{U}_2 \in \Re_+^{d_2\times k}$, and $\mathbf{U}_3 \in \Re_+^{d_3\times k}$. Let us create the matrix $\acute{\mathbf{G}} \in \Re^{(d_1 d_2)\times d_3}$ whose columns are the vectorized objects of the tensor $\mathbf{G}$, i.e.,

$$\acute{\mathbf{G}} \approx (\mathbf{U}_1 \odot \mathbf{U}_2)\mathbf{U}_3^T. \tag{47}$$

In order to create a set of orthonormal bases, we apply QR decomposition to

$$\mathbf{U}_1 \odot \mathbf{U}_2 = \mathbf{QR}. \tag{48}$$

The matrix $\mathbf{Q} \in \Re^{(d_1 d_2)\times k}$ contains orthonormal bases in its columns.

In the case that a tensor object $\mathbf{T} \in \Re^{d_1\times d_2}$ arrives, it is first vectorized. Afterwards, features are extracted as

$$\mathbf{t} = \mathbf{Q}^\dagger \acute{\mathbf{T}} \tag{49}$$

where $\mathbf{Q}^\dagger$ is the pseudoinverse of $\mathbf{Q}^\dagger$ given by $\mathbf{Q}^\dagger = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T$. For arbitrary valence tensors, the product $(\mathbf{U}_1 \odot \ldots \odot \mathbf{U}_n) = [\mathbf{vec}(\mathbf{u}_1^1 \otimes \ldots \otimes \mathbf{u}_n^1), \ldots, \mathbf{vec}(\mathbf{u}_1^K \otimes \ldots \otimes \mathbf{u}_n^K)]$ should be used in order to extract features. Then, the above procedure for extracting features via orthogonal bases can be applied in a straightforward manner.

B. NTF Plus LDA

Using the above feature extraction procedure, the NTF plus LDA discriminant method can be formulated in the same manner as the NMF plus LDA in [20]. That is, the training low-dimensional feature representation $\tilde{\mathbf{A}}_i$ can be used to define the within and between scatter matrices $\tilde{\mathbf{S}}_\mathbf{w}$ and $\tilde{\mathbf{S}}_\mathbf{b}$, respectively. These matrices can be calculated using the equations in (31), but now for the representations $\tilde{\mathbf{A}}_i$. Afterwards, the multiclass discriminant Fisher discriminant criterion should be defined and solved in order to find a linear discriminant transformation $\mathbf{W} \in \Re^{K\times p}$

$$\mathbf{W}_o = \arg\max_{\mathbf{W}} \frac{\mathrm{tr}[\mathbf{W}^T\tilde{\mathbf{S}}_\mathbf{b}\mathbf{W}]}{\mathrm{tr}[\mathbf{W}^T\tilde{\mathbf{S}}_\mathbf{b}\mathbf{W}]} \tag{50}$$

where $\mathrm{tr}[.]$ is the trace matrix operator and p is the number of discriminant bases. For finding the optimal transform $\mathbf{W}_o$, one may follow approaches like the ones proposed in [20], [38], and [41]. For a tensor representation $\mathbf{A}_i$, the feature vector $\tilde{\mathbf{A}}_i \in \Re^p$ derived from the NTF plus LDA approach is given by

$$\acute{\mathbf{A}}_i = \mathbf{W}_o^T \tilde{\mathbf{A}}_i. \tag{51}$$

V. Experimental Results

We have conducted two sets of experiments in order to compare the DNTF and NTF plus LDA approaches with other approaches such as NMF, LNMF, NTF, DNMF, LDA, multilinear LDA, etc. To do so, we have chosen the face verification problem, since DNMF and NMF plus LDA schemes have already been tested for it [20]. Moreover, we have applied the proposed approaches to the facial expression recognition problem since DNMF has shown superior performance in that problem [16].

Motivated by the experiments described in [23] and [25], we anticipate that NTF methods would outperform the corresponding NMF methods. Moreover, since an NMF plus LDA algorithm achieved the best equal error rate (EER) in [20], we anticipate that NTF plus LDA would have the best performance in face verification problems. Finally, we anticipate that DNTF algorithms would achieve the best performance in facial expression recognition problems, since DNMF algorithms outperformed all the other tested subspace algorithms in facial expression recognition [29], [42].

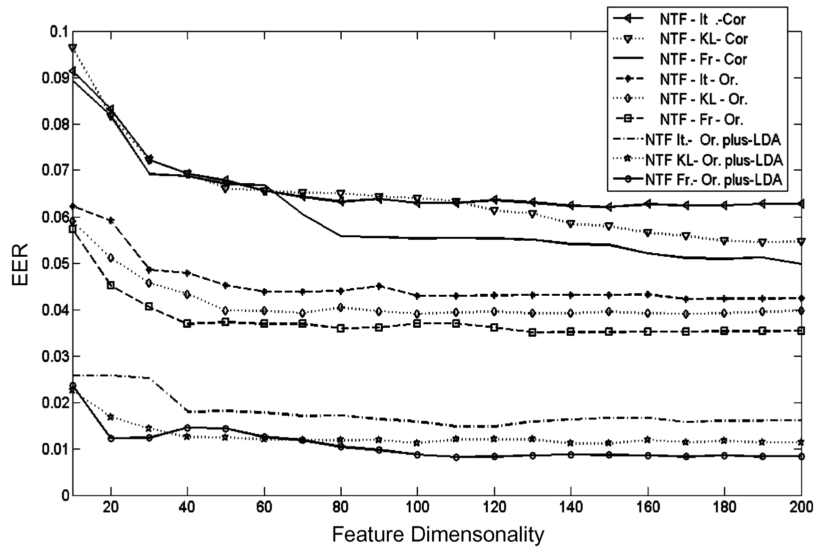


Fig. 1. EER for configuration I of XM2VTS versus dimensionality for NTF and NTF plus LDA methods. The NTF are derived using KL divergence, Frobenius (Fr.), and Itakura–Saito (It.) distance. Both correlation (Cor.) and orthogonal (Or.) feature extraction methods have been used.

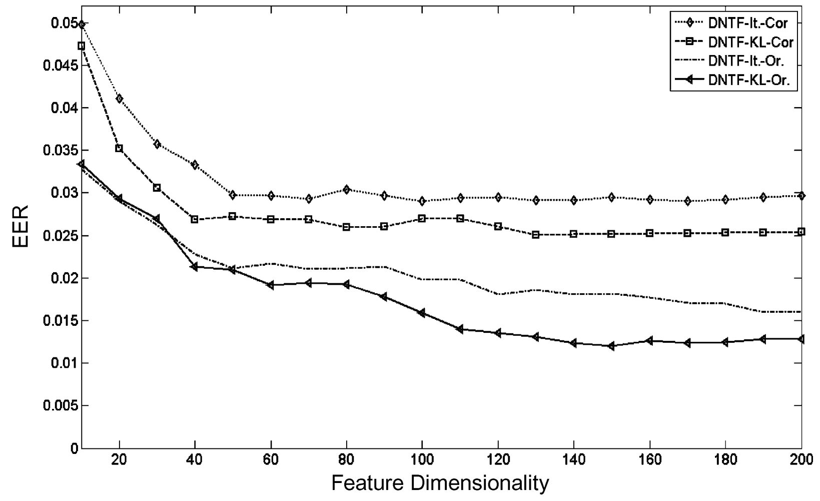


Fig. 2. EER for configuration I of XM2VTS versus dimensionality for DNTF. The DNTF are derived using KL divergence and Itakura–Saito (It.) distance. Both correlation (Cor.) and orthogonal (Or.) feature extraction methods have been used.

A. *Frontal Face Verification Experiments*

The experiments were conducted in the XM2VTS database using the protocol described in [43]. The images were aligned semiautomatically according to the eyes position of each facial image using the eye coordinates. The facial images were downscaled to a resolution of 64 $\times$ 64 pixels. Histogram equalization was used for the normalization of the facial image luminance. For the NTF-based methods and for multilinear LDA, an 64 $\times$ 64 $\times$ 600 tensor has been created in the training set.

The XM2VTS database contains 295 subjects, four recording sessions, and two shots (repetitions) per recording session. It provides two experimental setups, namely, configuration I and configuration II [43]. Each configuration is divided into three different sets: the training set, the evaluation set, and the test set. The training set is used to create client and impostor models for each person. The evaluation set is used to learn the verification decision thresholds. In case of multimodal systems, the evaluation set is also used to train the fusion manager [43]. For both configurations, the training set has 200 clients, 25 evaluation impostors, and 70 test impostors. The two configurations differ in the distribution of client training and client evaluation data. For additional details concerning the XM2VTS database, an interested reader can refer to [43].

The experimental procedure followed in the experiments was the one also used in [20]. For comparison reasons, the same methodology using the configuration I of the XM2VTS database was used. The performance of the algorithms is quoted by the EER, which is the scalar figure of merit that is often used to judge the performance of a verification algorithm. An interested reader may refer to [20] and [43] for more details concerning the XM2VTS protocol and the experimental procedure followed.

1) Comparing Various NTF and NTF Plus LDA Methods: We have performed experiments with the various distances used for building NTF methods. That is, KL divergence, Frobenius distance, and Itakura–Saito distance have been used for defining the various factorizations. Moreover, we have tested the two fea-

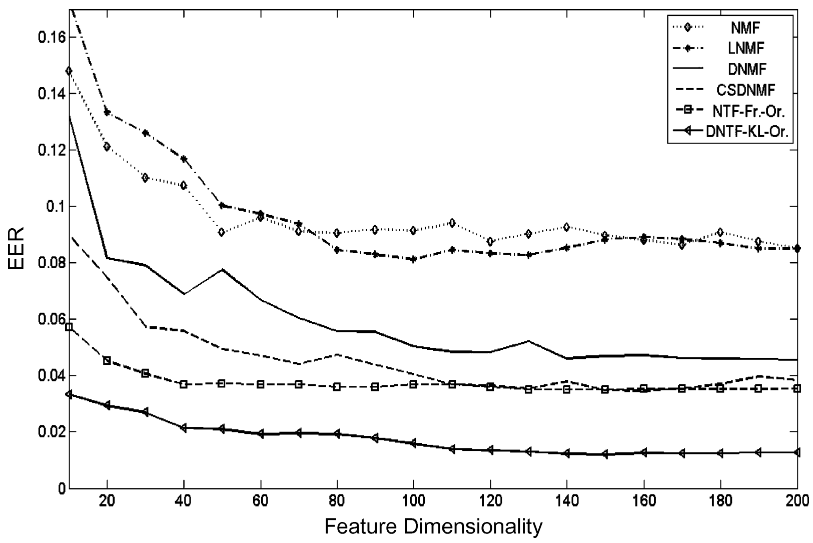


Fig. 3. EER for configuration I of XM2VTS versus dimensionality for NMF, LNMF, DNMF,CSDNMF, NTF, and DNTF.

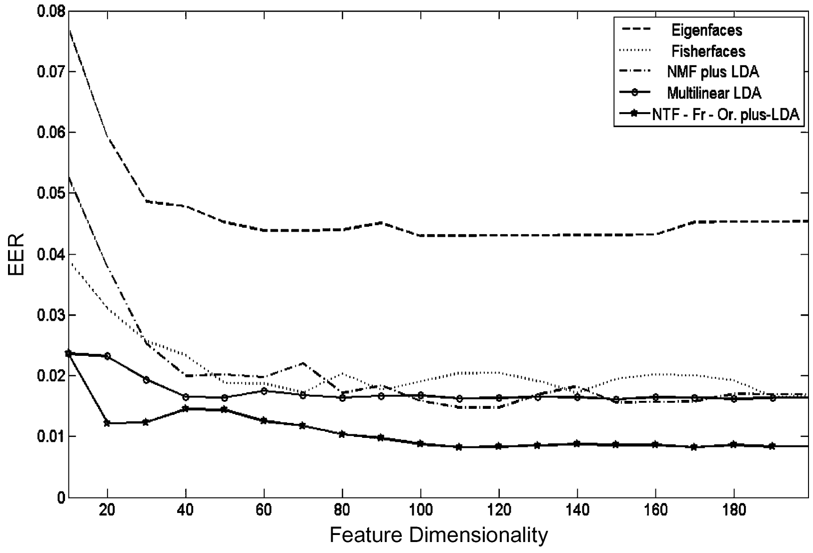


Fig. 4. EER for configuration I of XM2VTS versus dimensionality for PCA, PCA plus LDA, NMF plus LDA, NTF plus LDA, and multilinear LDA.

TABLE I
COMPARISON OF THE BEST EERS FOR ALL TESTED METHODS

Method	EER%	Method	EER%
DNTF	1.2	NTF plus LDA	0.8
NTF	3.5	NMF plus LDA	1.47
CSDNMF	3.5	Multilinear LDA	1.61
DNMF	4.4	Fisherfaces	1.7
LNMF	8	Eigenfaces	4.3
NMF	8.5		

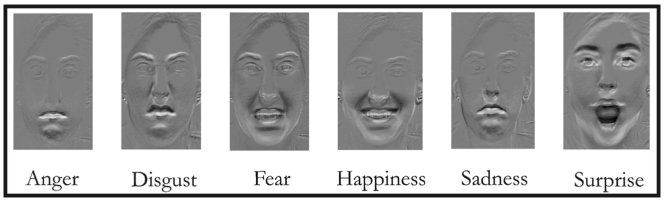


Fig. 5. Difference images for each facial expression for a poser from the Cohn–Kanade database.

ture extraction methods [i.e., the correlation based in (45) and the second using orthogonal bases in (49)]. Finally, we have applied LDA in all NTF methods. The EER versus the dimensionality of the decomposition for the above mentioned methods is shown in Fig. 1. As can be seen, the best verification performance, when comparing the various NTF methods, has been achieved when using Frobenius distance and orthogonal bases (EER equal to 3.5%). As can also be seen, the application of LDA on the features extracted by NTF highly increases the performance giving a very low EER of 0.8%.

2) Comparing the Various DNTF Methods: In the second set of experiments, the different DNTF methods proposed in this

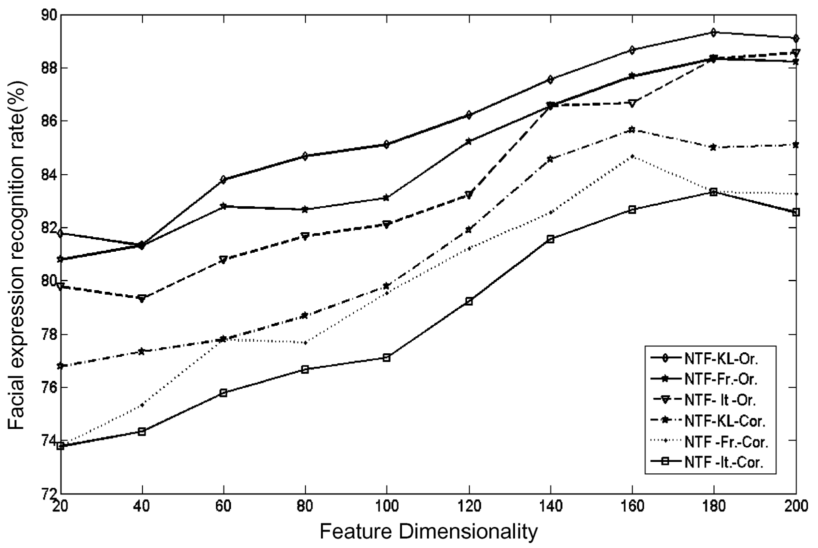


Fig. 6. Facial expression recognition rate versus dimensionality in Cohn–Kanade database for NTF methods.

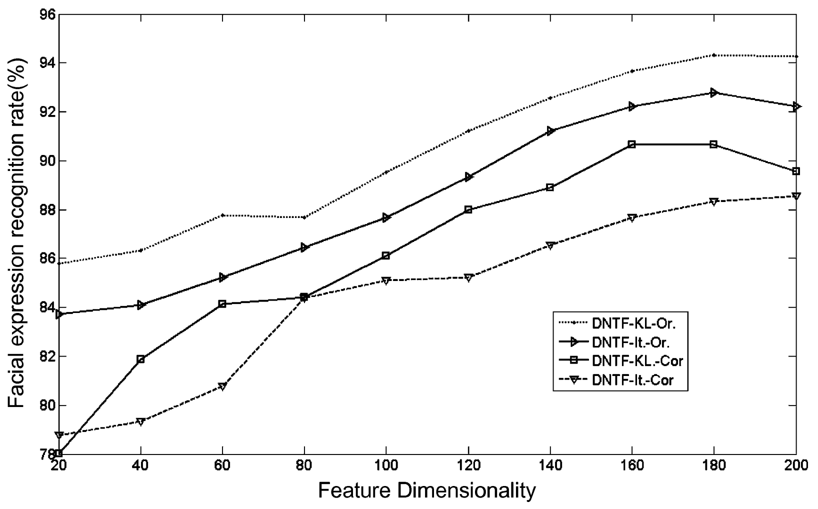


Fig. 7. Facial expression recognition rate versus dimensionality in Cohn–Kanade database for DNTF methods.

paper have been tested. That is, we have tested DNTF methods based on KL divergence and Itakura–Saito distance. We have also applied both feature extraction methods. The best performance has been achieved by DNTF with KL divergence, giving an EER equal to 1.2%. The EER versus the dimensionality for the above discriminant decomposition is shown in Fig. 2.

3) Comparing the Various Tested Methods: We have compared our method with the NMF method in [5], LNMF [14], DNMF [20], NTF [25], DNMF [20], class specific DNMF [20], PCA [1], PCA plus LDA [38], and multilinear LDA [32].

In Fig. 3, a comparison of EERs between the methods based on NMF and NTF is shown. That is, for NMF, LNMF, DNMF, CSDNMF, NTF, and DNTF, the EER is plotted versus the dimensionality. As can be seen, the best results are when using the proposed DNTF method with KL divergence.

In Fig. 4, the EER of the methods based on the addition of an LDA step (i.e., PCA plus LDA, multilinear LDA, NMF plus LDA, and NTF plus LDA) and PCA is plotted versus the dimensionality of the new lower dimensional space. The best performance was that of NTF plus LDA giving an EER $\approx 0.8\%$. The NMF plus LDA has led to EER $\approx 1.5\%$ and multilinear LDA EER $\approx 1.6\%$. Thus, NTF plus LDA had the best performance.

Finally, a comparison of the best EERs for the different tested approaches can be found in Table I.

B. Facial Expression Recognition Experiments

The database used for the facial expression recognition experiments was created using the Cohn–Kanade database [44]. This database is annotated with facial action units (FAUs). These combinations of FAUs were translated into facial expressions according to [45], in order to define the corresponding ground truth for the facial expressions. All the subjects were taken into consideration and their difference images, created by subtracting the neutral image intensity values from the corresponding values of the fully expressive facial expression image, were calculated. Each difference image was initially normalized, resulting in an image built only from positive values. These images form a tensor of $\mathbf{G} \in \Re^{90\times 60\times 372}$. The difference images are used instead of the original facial expressive images, due to the fact that in the difference images,

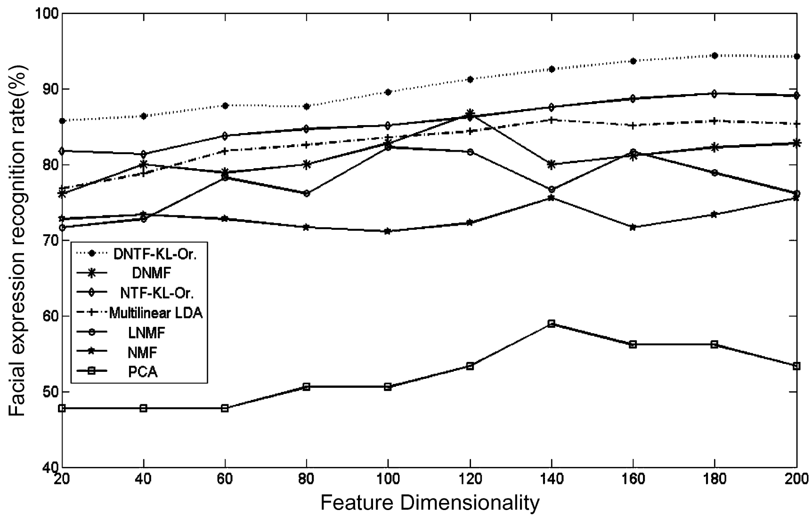


Fig. 8. Facial expression recognition rate versus dimensionality in Cohn–Kanade database for the tested approaches.

the facial parts in motion are emphasized [46], [47]. In Fig. 5, an example of the difference images for each facial expression is depicted.

In the experimental procedure, five sets containing 20% of the data for each of the six facial expression classes, chosen randomly, were created. One set containing 20% of the samples for each class is used as the test set, while the remaining sets form the training set. After the classification procedure is performed, the samples forming the test set are incorporated into the current training set while a new set of samples (20% of the samples for each class) is extracted to form the new test set. The remaining samples create the new training set. This procedure is repeated five times. The average classification accuracy is the mean value of the percentages of the correctly classified facial expressions. For all the tested methods, a simple nearest neighbor classifier [15], [16] has been used for classifying an expression.

1) Comparison of NTF Methods: We have tested various NTF methods proposed in this paper for facial expression recognition. We have also tested two different feature extraction methods. In Fig. 6, the facial expression recognition rate versus feature dimensionality is plotted. The best performance has been achieved by NTF using KL divergence with orthogonal bases giving a total of 89.1%.

2) Comparison of DNTF Methods: From the tested DNTF methods, the one with the best performance was the DNMF with KL divergence using orthogonal bases for feature extraction, achieving a total 94.33%. The facial expression recognition rate for various DNTF methods is plotted in Fig. 7.

3) Comparison of the Various Tested Methods: The other tested approaches were NMF, LNFM, DNMF, PCA, and multilinear LDA. In Fig. 8, the performance of tested approaches in facial expression recognition using 200 basis images in every approach is shown. PCA plus LDA, NMF plus LDA, and NTF plus LDA give a total of five features (six class problem) and had rather poor performance, thus being omitted from the plots. The multilinear LDA can extract more than five features [32]. As can be seen, the proposed DNTF method outperforms all the other tested approaches in facial expression recognition.

TABLE II
COMPARISON OF THE BEST EERS FOR ALL TESTED METHODS

Method	Recognition Rate%	Method	Recognition Rate%
DNTF	94.33	LNMF	82.2
NTF	89.1	NMF	75.6
DNMF	87	PCA	58.9
Multilinear LDA	85.7		

The best facial expression recognition accuracies achieved when using PCA, NMF, LNFM, DNMF, and multilinear LDA were equal to 58.9%, 75.6%, 82.2%, 87%, and 85.7%, respectively. Therefore, an increase of the recognition accuracy by more than 7% (in comparison with the DNMF results) is introduced due to the use of the proposed DNTF. The facial expression recognition rates for various tested methods are plotted in Fig. 8. A comparison of the best facial expression recognition rate for all the tested methods can be found in Table II.

Moreover, in order to understand if the proposed approach is statistically significantly better than other tested approaches, the McNemar's test [48] has been used for the facial expression recognition experiments. The McNemar's test is a null hypothesis statistical test based on a Bernoulli model. If the resulting value is below a desired significance level (for example, 0.02), the null hypothesis is rejected and the performance difference between two algorithms is considered to be statistically significantly better. Using this test, it has been verified that the proposed DNTF outperforms the other tested classifiers in the demonstrated experiments, at a significant level less that $p = 10^{-5}$.

VI. CONCLUSION

In this paper, we have proposed a series of unsupervised and supervised feature extraction methods for the decomposition of tensor objects with nonnegative representations such as facial images. The proposed methods do not require the vectorization of the representation. Thus, the local structure of the objects is not lost while the supervised methods increase the discrimination between different object classes. First, we have extended NMF algorithm using arbitrary valence Kruskal tensors. That is, we have proposed an NTF method using KL divergence. Afterwards, we proposed a general NTF framework using generalized Bregman distances. Moreover, we proposed a series of discriminant NTF methods by incorporating discriminant constraints inside various NTF decompositions, including a framework for DNTF with generalized Bregman distances. The presented experiments have shown that the features derived via the proposed procedures outperformed many other subspace representations in frontal face verification and in facial expression recognition problems.

Appendix I
Derivation of NTF Decomposition

Let W be an auxiliary function for $Y(\mathbf{F})$ if $W(\mathbf{F},\mathbf{F}^{(t-1)}) \geq Y(\mathbf{F})$ and $W(\mathbf{F},\mathbf{F}) = Y(\mathbf{F})$. If W is an auxiliary function of Y, then Y is nonincreasing under the update $\mathbf{F}^t = \arg\min_{\mathbf{F}} W(\mathbf{F},\mathbf{F}^{(t-1)})$ [5]. With the help of the auxiliary function, the update rules for $\mathbf{U}_1$ can be calculated. By fixing all rank-1 tensor matrices $\mathbf{U}_2,\cdots,\mathbf{U}_n$, the elements of the matrix $\mathbf{U}_1 = [\mathbf{u}_1^1,\ldots,\mathbf{u}_1^K]$ are updated by minimizing $\mathbf{Y}(\mathbf{U}_1) = D(\mathbf{G}\| \sum_{m=1}^K \mathbf{u}_1^m \otimes \mathbf{u}_2^m \ldots \otimes \mathbf{u}_n^m)$ defined in (32). For $\mathbf{U}_1$, we define the function

$$\begin{aligned} W\left(\mathbf{U}_1,\mathbf{U}_1^{(t-1)}\right) &= \sum_{i_1,\ldots,i_n} \left(\mathbf{G}_{i_1,\ldots,i_n} \ln\left(\mathbf{G}_{i_1,\ldots,i_n}\right) - \mathbf{G}_{i_1,\ldots,i_n}\right) \\ &+ \sum_{i_1,\ldots,i_n} \mathbf{G}_{i_1,\ldots,i_n} \sum_{m=1}^K \frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_{l=1}^K \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l} \\ &\times \left(\ln\left(u_{i_1,1}^m \ldots u_{i_n,n}^m\right) - \ln \frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_l \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l}\right) \\ &+ \sum_{i_1,\ldots,i_n} \sum_{m=1}^K u_{i_1,1}^m \ldots u_{i_n,n}^m. \end{aligned} \tag{52}$$

This function $W(\mathbf{U}_1,\mathbf{U}_1^{(t-1)})$ is an auxiliary function for $Y(\mathbf{U}_1)$. It is straightforward to show that $W(\mathbf{U}_1,\mathbf{U}_1) = Y(\mathbf{U}_1)$. In order to prove that $W(\mathbf{U}_1,\mathbf{U}_1^{(t-1)}) \geq Y(\mathbf{U}_1)$, since $\ln(\sum_{m=1}^K u_{i_1,1}^m \ldots u_{i_n,n}^m)$ is convex, the following inequality holds:

$$-\ln\left(\sum_{m=1}^K u_{i_1,1}^m \ldots u_{i_n,n}^m\right) \leq -\sum_{m=1}^K a_m \ln \frac{u_{i_1,1}^m \ldots u_{i_n,n}^m}{a_m} \tag{53}$$

for all nonnegative a_m that satisfy $\sum_m a_m = 1$. By letting $a_m = (u_{i_1,1}^m)^{(t-1)} \ldots u_{i_n,n}^m / \sum_{l=1}^K (u_{i_1,1}^l)^{(t-1)} \ldots u_{i_n,n}^l$, we obtain

$$\begin{aligned} -\ln\left(\sum_{m=1}^K u_{i_1,1}^m \ldots u_{i_n,n}^m\right) &\leq \sum_{m=1}^K \frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_{l=1}^K \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l} \\ &\times \left(\ln\left(u_{i_1,1}^m \ldots u_{i_n,n}^m\right) - \ln\left(\frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_{l=1}^K \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l}\right)\right). \end{aligned} \tag{54}$$

From (54), it is also straightforward to show that $W(\mathbf{U}_1,\mathbf{U}_1^{(t-1)}) \geq Y(\mathbf{U}_1)$. Thus, $W(\mathbf{U}_1,\mathbf{U}_1^{(t-1)})$ is an auxiliary function of $Y(\mathbf{U}_1)$.

Now, by letting $(\partial W(\mathbf{U}_1,\mathbf{U}_1^{(t-1)}))/\partial u_{i_1,1}^m = 0$

$$\begin{aligned} \frac{\partial W\left(\mathbf{U}_1,\mathbf{U}_1^{(t-1)}\right)}{\partial u_{i_1,1}^m} &= -\sum_{i_2,\ldots,i_n} \mathbf{G}_{i_1,\ldots,i_n} \frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_{l=1}^K \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l} \frac{1}{u_{i_1,1}^m} \\ &+ \sum_{i_2,\ldots,i_n} u_{i_2,2}^m \ldots u_{i_n,n}^m = 0 \end{aligned} \tag{55}$$

the update rule for $u_{i_1,1}^m$ can be calculated as

$$u_{i_1,1}^m = \frac{\sum_{i_2,\ldots,i_n} \mathbf{G}_{i_1,\ldots,i_n} \frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_{l=1}^K \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l}}{\sum_{i_2,\ldots,i_n} u_{i_2,2}^m \ldots u_{i_n,n}^m}. \tag{56}$$

Moreover, using the sum to one constraint, the above rules become

$$u_{i_1,1}^m = \frac{\sum_{i_2,\ldots,i_n} \mathbf{G}_{i_1,\ldots,i_n} \frac{\left(u_{i_1,1}^m\right)^{(t-1)} \ldots u_{i_n,n}^m}{\sum_{l=1}^K \left(u_{i_1,1}^l\right)^{(t-1)} \ldots u_{i_n,n}^l}}{\sum_{i_n} u_{i_n,n}^m}. \tag{57}$$

The update rules for all other tensors can be calculated in the same manner.

Appendix II
Implementation of the NTF Algorithm With KL Divergence Using Matrix Operations

At every iteration the following updates should be repeated for all matrices $\mathbf{U}_1$, $\mathbf{U}_2$, and $\mathbf{U}_3$. The matrix update rules presented in the following are equivalent to the update rules in (14)–(15).

For the matrix $\mathbf{U}_1 \in \Re_+^{d_1 \times K}$ we update, it as

$$\tilde{\mathbf{U}}_1^{(t)} = \mathbf{U}_1^{(t-1)} . \times (\mathbf{A}_1^{(t-1)} ./ \mathbf{B}^{(t-1)}) \tag{58}$$

where $\mathbf{A}_1^{(t-1)} = \mathbf{\Gamma}_1^{(t-1)}(\mathbf{U}_2^{(t-1)} \odot \mathbf{U}_3^{(t-1)})$, $.\times$ denotes the elementwise matrix multiplication and $./$ denotes elementwise matrix division. Matrix $\mathbf{A}_1$ is a $d_1 \times K$ matrix, a matrix with nonnegative elements. The $\mathbf{U}_2 \odot \mathbf{U}_3 \in \Re_+^{(d_2 d_3)\times K}$ is the Khatri–Rao product, i.e., $\mathbf{U}_2 \odot \mathbf{U}_3 = [\mathbf{vec}(\mathbf{u}_2^1 \otimes \mathbf{u}_3^1) \ldots \mathbf{vec}(\mathbf{u}_2^K \otimes \mathbf{u}_3^K)]$. Matrix $\mathbf{\Gamma}_1^{(t-1)}$ is equal to $\mathbf{\Lambda}_1 ./ \mathbf{Z}_1^{(t-1)}$. The $\mathbf{\Lambda}_1 \in \Re_+^{d1\times(d_2 d_3)}$ is the unfolding of the tensor $\mathbf{G}$ as

$$\mathbf{\Lambda}_1 = \begin{bmatrix} \mathbf{G}_{1,1,1} & \ldots & \mathbf{G}_{1,d_2,d_3} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_{d_1,1,1} & \ldots & \mathbf{G}_{d_1,d_2,d_3} \end{bmatrix} \tag{59}$$

and matrix

$$\mathbf{Z}_1^{(t-1)} = \mathbf{U}_1^{(t-1)}(\mathbf{U}_2^{(t-1)} \odot \mathbf{U}_3^{(t-1)})^T. \tag{60}$$

Finally, matrix $\mathbf{B}^{(t-1)} \in \Re_+^{d_1 \times K}$ is formulated by repeating the vector

$$\mathbf{b}^{(t-1)} = \left[\sum_{i_3} u_{i_3,3}^{1\ (t-1)} \cdots \sum_{i_3} u_{i_3,3}^{K\ (t-1)}\right]$$

as a row d_1 times, i.e.,

$$\mathbf{B}^{(t-1)} = \begin{bmatrix} \mathbf{b}^{(t-1)^T} \\ \vdots \\ \mathbf{b}^{(t-1)^T} \end{bmatrix}.$$

The estimation $\mathbf{U}_1^{(t)}$ is obtained from the matrix $\tilde{\mathbf{U}}_1^{(t)}$ when it is normalized so that every column sums to one.

Afterwards, for $\mathbf{U}_2 \in \Re_+^{d_2 \times K}$

$$\tilde{\mathbf{U}}_2^{(t)} = \mathbf{U}_2^{(t-1)}. \times \left(\mathbf{A}_2{}^{(t-1)}./\mathbf{B}^{(t-1)}\right) \tag{61}$$

where $\mathbf{A}_2^{(t-1)} = \mathbf{\Gamma}_2^{(t-1)}(\mathbf{U}_1^{(t)} \odot \mathbf{U}_3^{(t-1)})$. The Khatri–Rao product $\mathbf{U}_1 \odot \mathbf{U}_3$ is a $(d_1 d_3) \times K$ real nonnegative matrix. Matrix $\mathbf{\Gamma}_2^{(t-1)} = \mathbf{\Lambda}_2./\mathbf{Z}_2^{(t-1)}$. The tensor $\mathbf{G}$ is now unfolded as

$$\mathbf{\Lambda}_2 = \begin{bmatrix} \mathbf{G}_{1,1,1} & \cdots & \mathbf{G}_{d_1,1,d_3} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_{1,d_2,1} & \cdots & \mathbf{G}_{d_1,d_2,d_3} \end{bmatrix} \tag{62}$$

and $\mathbf{\Lambda}_2 \in \Re_+^{d_2 \times (d_1 d_3)}$. Matrix $\mathbf{Z}_2^{(t-1)} = \mathbf{U}_2^{(t-1)}(\mathbf{U}_1^{(t)} \odot \mathbf{U}_3^{(t-1)})^T$. The estimation $\mathbf{U}_2^{(t)}$ is obtained from the matrix $\tilde{\mathbf{U}}_2^{(t)}$ when it is normalized so that every column sums to one.

Finally, for matrix $\mathbf{U}_3 \in \Re_+^{d_3 \times K}$

$$\tilde{\mathbf{U}}_3 = \mathbf{U}_3^{(t-1)}. \times \mathbf{A}_3{}^{(t-1)} \tag{63}$$

where $\mathbf{A}_3^{(t-1)} = \mathbf{\Gamma}_3^{(t-1)}(\mathbf{U}_1^{(t)} \odot \mathbf{U}_2^{(t)})$. Matrix $\mathbf{\Gamma}_3$ is equal to $\mathbf{\Lambda}_3./\mathbf{Z}_3^{(t-1)}$. The tensor $\mathbf{G}$ is now unfolded as

$$\mathbf{\Lambda}_3 = \begin{bmatrix} \mathbf{G}_{1,1,1} & \cdots & \mathbf{G}_{d_1,d_2,1} \\ \vdots & \ddots & \vdots \\ \mathbf{G}_{1,1,d_3} & \cdots & \mathbf{G}_{d_1,d_2,d_3} \end{bmatrix} \tag{64}$$

and matrix $\mathbf{Z}_3^{(t-1)} = \mathbf{U}_3^{(t-1)}(\mathbf{U}_1^{(t)} \odot \mathbf{U}_2^{(t-1)})^T$.

The above procedure can be easily extended for arbitrary valence tensors by using the Khatri–Rao product between $n-1$ matrices.

Appendix III
Auxiliary Function of NTF With Generalized Bregman Distances

Here it will be proven that $W(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j})$ is an auxiliary function for $F(\mathbf{u}_{i_j,j})$. First, it is easy to verify that $W(\mathbf{u}_{i_j,j}, \mathbf{u}_{i_j,j}) = F(\mathbf{u}_{i_j,j})$, since $\sum_l \lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} = 1$. Now, in order to prove that $W(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}) \geq F(\mathbf{u}_{i_j,j})$, we calculate $W(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}) - F(\mathbf{u}_{i_j,j})$ as shown in (65) shown at the bottom of the page. Since function $\phi()$ is considered convex, $\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} \geq 0$, and $\sum_l \lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} = 1$, it is valid that (66) shown at the bottom of the page, holds.

Now, since we have proven that $W(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j})$ is an auxiliary function of $F(\mathbf{u}_{i_j,j})$ in order to derive update rules that guarantee a nonincreasing behavior of $F(\mathbf{u}_{i_j,j})$, we should calculate $\partial W/\partial u_{i_j,j}^l$ as shown in (67) at the top of the next page, and in case that we have $\psi(xy) = \psi(x)\psi(y)$, (68) shown at the top of the next page, holds.

For the special case of KL divergence, in which $\phi(x) = x\log(x)$ and hence $\psi(x) = \log(x) + 1$, we start with (67) and derive (69) shown at the top of the next page.

$$W\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right) - F\left(\mathbf{u}_{i_j,j}\right) = \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \sum_l \lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} \phi\left(\frac{u_{i_j,j}^l u_{i_1,1}^l \cdots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \cdots u_{i_n,n}^l}{\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n}}\right) - \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \phi\left(\sum_l u_{i_j,j}^l u_{i_1,1}^l \cdots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \cdots u_{i_n,n}^l\right). \tag{65}$$

$$\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \sum_l \lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} \phi\left(\frac{u_{i_j,j}^l u_{i_1,1}^l \cdots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \cdots u_{i_n,n}^l}{\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n}}\right) \geq \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \phi\left(\sum_l u_{i_j,j}^l u_{i_1,1}^l \cdots u_{i_{j-1},j-1}^l u_{i_{j+1},j+1}^l \cdots u_{i_n,n}^l\right) \Rightarrow W\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right) \geq F\left(\mathbf{u}_{i_j,j}\right). \tag{66}$$

$$\begin{aligned}\frac{\partial W}{\partial u^l_{i_j,j}} =& \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n} \psi\left(\frac{u^l_{i_j,j}u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}}{\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n}}\right)\\ &\times \frac{u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}}{\lambda_{i_1,\ldots,i_{j-1},l,i_{j+1},\ldots,i_n}} - \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \psi(\mathbf{G}_{i_1,\ldots,i_j,\ldots,i_n})\left(u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right)\\ =& \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\psi\left(\frac{u^l_{i_j,j}}{\tilde{u}^l_{i_j,j}}\sum_m u^m_{i_j,j}u^m_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right)\\ &- \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \psi\left(\mathbf{G}_{i_1,\ldots,i_j,\ldots,i_n}\right)\left(u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right) \qquad (67)\end{aligned}$$

$$\begin{aligned}\frac{\partial W}{\partial u^l_{i_j,j}} =& \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\left(u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right)\psi\left(\frac{u^l_{i_j,j}}{\tilde{u}^l_{i_j,j}}\right)\psi\left(\sum_m \tilde{u}^m_{i_j,j}u^m_{i_1,1}\cdots u^m_{i_{j-1},j-1}u^m_{i_{j+1},j+1}\cdots u^m_{i_n,n}\right)\\ &- \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} \psi(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n})\, u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}. \qquad (68)\end{aligned}$$

$$\begin{aligned}\frac{\partial W}{\partial u^l_{i_j,j}} = 0 \Leftrightarrow& \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\left(u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\right)\\ &\times\left(\log\left(\frac{u^l_{i_j,j}}{\tilde{u}^l_{i_j,j}}\right) + \log\left(\sum_m \tilde{u}^m_{i_j,j}u^m_{i_1,1}\cdots u^m_{i_{j-1},j-1}u^m_{i_{j+1},j+1}\cdots u^m_{i_n,n}\right)+1\right)\\ =& \sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n}\left(\log\left(\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}\right)+1\right)u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n} \Leftrightarrow \log\left(\frac{u^l_{i_j,j}}{\tilde{u}^l_{i_j,j}}\right)\\ =& \frac{\sum_{i_1,\ldots,i_{j-1},i_{j+1},\ldots,i_n} u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}\log\frac{\mathbf{G}_{i_1,\ldots,i_{j-1},i_j,i_{j+1},\ldots,i_n}}{\sum_m \tilde{u}^m_{i_j,j}u^m_{i_1,1}\cdots u^m_{i_{j-1},j-1}u^m_{i_{j+1},j+1}\cdots u^m_{i_n,n}}}{\sum_{i_{1q},\ldots,i_{j-1},i_{j+1},\ldots,i_n} u^l_{i_1,1}\cdots u^l_{i_{j-1},j-1}u^l_{i_{j+1},j+1}\cdots u^l_{i_n,n}}. \qquad (69)\end{aligned}$$

Appendix IV
Implementation of the NTF Algorithm in (28) Using Matrix Operations

The algorithm that is described in iteration form in (28) can be implemented using only matrix operations as follows. At every iteration, matrix $\mathbf{U}_1^{(t)}$ is updated as

$$\mathbf{U}_1^{(t)} = \mathbf{U}_1^{(t-1)}.\times\left(\mathbf{A}_1^{(t-1)}./\mathbf{M}_1^{(t-1)}\right) \qquad (70)$$

where $\mathbf{A}_1^{(t-1)} = \mathbf{\Lambda}_1\mathbf{K}_1^{(t-1)}$, where $\mathbf{K}_1^{(t-1)} = \mathbf{U}_2^{(t-1)}\odot\mathbf{U}_3^{(t-1)}$. The $\mathbf{\Lambda}_1$ is a matrix that contains the unfolding of the tensor $\mathbf{G}$ as in (59) and $\mathbf{M}_1^{(t-1)} = \mathbf{U}_1^{(t-1)}\mathbf{K}_1^{(t-1)^T}\mathbf{K}_1^{(t-1)}$.

For the second matrix $\mathbf{U}_2$, we have

$$\mathbf{U}_2^{(t)} = \mathbf{U}_2^{(t-1)}.\times\left(\mathbf{A}_2{}^{(t-1)}./\mathbf{M}_2{}^{(t-1)}\right) \qquad (71)$$

where $\mathbf{A}_2^{(t-1)} = \mathbf{\Lambda}_2\mathbf{K}_2^{(t-1)}$, where $\mathbf{K}_2^{(t-1)} = \mathbf{U}_1^{(t)}\odot\mathbf{U}_3^{(t-1)}$. Matrix $\mathbf{\Lambda}_2$ is the unfolding of the tensor $\mathbf{G}$ as in (62) and $\mathbf{M}_2^{(t-1)} = \mathbf{U}_2^{(t-1)}\mathbf{K}_2^{(t-1)^T}\mathbf{K}_2^{(t-1)}$.

For the third tensor

$$\mathbf{U}_3^{(t)} = \mathbf{U}_3^{(t-1)}.\times\left(\mathbf{A}_3^{(t-1)}./\mathbf{M}_3^{(t-1)}\right) \qquad (72)$$

where $\mathbf{A}_3^{(t-1)} = \mathbf{\Lambda}_3\mathbf{K}_3^{(t)}$, where $\mathbf{K}_3^{(t)} = \mathbf{U}_1^{(t)}\odot\mathbf{U}_2^{(t)}$. The $\mathbf{\Lambda}_3$ is the tensor unfolding of the tensor $\mathbf{G}$ as in (62) and $\mathbf{M}_3^{(t-1)} = \mathbf{U}_3^{(t-1)}\mathbf{K}_3^{T^{(t)}}\mathbf{K}_3^{(t)}$.

Appendix V
Derivation of the DNMF Decomposition

For the DNMF decomposition, the update rules for all the rank-1 tensors, except for the $\mathbf{u}_n^m$ tensors, are the same as in the NTF decomposition. For $\mathbf{u}_n^m$ (i.e., the coefficients of the decomposition), it can be easily proven, using the results of previous Appendix IV, that the function $W_d(\mathbf{U}_n, \mathbf{U}_n^{(t-1)})$ is an auxiliary function of $Y_d(\mathbf{U}_n)$. The function W_d is defined as

$$\begin{aligned}W_d&\left(\mathbf{U}_n, \mathbf{U}_n^{(t-1)}\right)\\ &= \sum_{i_1,\ldots,i_n}\left(\mathbf{G}_{i_1,\ldots,i_n}\ln\left(\mathbf{G}_{i_1,\ldots,i_n}\right) - \mathbf{G}_{i_1,\ldots,i_n}\right)\\ &\quad + \sum_{i_1,\ldots,i_n}\mathbf{G}_{i_1,\ldots,i_n}\sum_{m=1}^K\frac{u^m_{i_1,1}\cdots\left(u^m_{i_n,n}\right)^{(t-1)}}{\sum_{l=1}^K u^l_{i_1,1}\cdots\left(u^l_{i_n,n}\right)^{(t-1)}}\\ &\quad\times\left(\ln\left(u^m_{i_1,1}\cdots u^m_{i_n,n}\right) - \ln\frac{u^m_{i_1,1}\cdots\left(u^m_{i_n,n}\right)^{(t-1)}}{\sum_l u^l_{i_1,1}\cdots\left(u^l_{i_n,n}\right)^{(t-1)}}\right)\\ &\quad + \sum_{i_1,\ldots,i_n}\sum_{m=1}^K u^m_{i_1,1}\ldots u^m_{i_n,n} + \gamma\mathrm{tr}[\acute{\mathbf{S}}_w] - \delta\mathrm{tr}[\acute{\mathbf{S}}_b]. \qquad (73)\end{aligned}$$

From (54), it is straightforward to show that $W_d(\mathbf{U}_n, \mathbf{U}_n^{(t-1)}) \geq Y_d(\mathbf{U}_n)$. Thus, $W_d(\mathbf{U}_n, \mathbf{U}_n^{(t-1)})$ is an auxiliary function of $Y_d(\mathbf{U}_n)$.

The update rules are derived from setting $(\partial W_d(\mathbf{U}_n, \mathbf{U}_n^{(t-1)}))/\partial u_{i_n,n}^m$ equal to zero for all the $u_{i_n,n}^m$. Let $u_{i_n,n}^m$ be the mth element of the ρth object for the rth class. That is, $u_{i_n,n}^m$ is the element stored in the mth column and i_n row of the coefficients matrix $\mathbf{U}_n$ and, according to the organization of the database to different classes, $i_n = \sum_{j=1}^{r-1} N_j + \rho$. The partial derivatives of the $(\partial \mathrm{tr}[\acute{\mathbf{S}}_w])/u_{i_n,n}^m$ and $(\partial \mathrm{tr}[\acute{\mathbf{S}}_b])/\partial u_{i_n,n}^m$ are given by

$$\frac{\partial \mathrm{tr}[\acute{\mathbf{S}}_w]}{\partial u_{i_n,n}^m} = 2\left(u_{i_n,n}^m - \acute{\mu}_m^{(r)}\right) \quad \text{and} \quad \frac{\partial \mathrm{tr}[\acute{\mathbf{S}}_b]}{\partial u_{i_n,n}^m} = 2\left(\acute{\mu}_m^{(r)} - \acute{\mu}_m\right). \tag{74}$$

Using (74), we obtain the quadratic equation

$$\begin{aligned} &\frac{\partial W_d\left(\mathbf{U}_n, \mathbf{U}_n^{(t-1)}\right)}{\partial u_{i_n,n}^m} \\ &= -\sum_{i_1,\ldots,i_{n-1}} \mathbf{G}_{i_1,\ldots,i_n} \frac{u_{i_1,1}^m \cdots \left(u_{i_n,n}^m\right)^{(t-1)}}{\sum_{l=1}^K u_{i_1,1}^l \cdots \left(u_{i_n,n}^l\right)^{(t-1)}} \frac{1}{u_{i_n,n}^m} \\ &\quad + \sum_{i_2,\ldots,i_n} u_{i_2,2}^m \cdots u_{i_n,n}^m + 2\gamma\left(u_{i_n,n}^m - \acute{\mu}_m^{(r)}\right) \\ &\quad - 2\delta\left(\acute{\mu}_m^{(r)} - \acute{\mu}_m\right) = 0. \end{aligned} \tag{75}$$

The mean $\acute{\mu}_m^{(r)}$ can be expanded as $\acute{\mu}_m^{(r)} = (1/N_r)\sum_{\lambda \in \mathcal{F}_r} u_{\lambda,n}^m = (1/N_r)\sum_{\lambda \in \mathcal{F}_r, \lambda \neq i_n} u_{\lambda,n}^m + (1/N_r) u_{i_n,n}^m$. Now, (75) can be expanded as (76) shown at the bottom of the page. We could have expanded $\acute{\mu}_m$ as well, since $u_{i_n,n}^m$ participates in it. However, as in [20], in order to simplify the calculations, we assume that $u_{i_n,n}^m$ has very small contribution in the grand mean $\acute{\mu}_m$.

By solving the quadratic equation (75), the update rules for $i_n \in \mathcal{F}_r$ can be derived from (77) shown at the bottom of the page, where $\acute{T}$ is given by

$$\begin{aligned} \acute{T} &= (2\gamma + 2\delta)\left(\frac{1}{N_r}\sum_{\lambda \in \mathcal{F}_r, \lambda \neq i_n} u_{\lambda,n}^m\right) - 2\delta\acute{\mu}_m \\ &\quad - \sum_{i_1,\ldots,i_{n-1}} u_{i_1,1}^m \cdots u_{i_{n-1},n-1}^m \\ &= (2\gamma + 2\delta)\left(\frac{1}{N_r}\sum_{\lambda \in \mathcal{F}_r, \lambda \neq i_n} u_{\lambda,n}^m\right) - 2\delta\acute{\mu}_m - 1 \end{aligned} \tag{78}$$

since

$$\sum_{i_1,\ldots,i_{n-1}} u_{i_1,1}^m \cdots u_{i_{n-1},n-1}^m = \sum_{i_1} u_{i_1,1}^m \cdots \sum_{i_{n-1}} u_{i_{n-1},n-1}^m = 1.$$

APPENDIX VI
AUXILIARY FUNCTION FOR DISCRIMINANT NONNEGATIVE MATRIX FACTORIZATION WITH GENERALIZED BREGMAN DISTANCES

In this appendix, we calculate the update rules for the DNTF method with generalized Bregman distances. Let us define the DNTF cost function using generalized Bregman distance

$$F_d\left(\mathbf{u}_{i_j,j}\right) = F\left(\mathbf{u}_{i_j,j}\right) + \gamma_j \mathrm{tr}[\acute{\mathbf{S}}_w] - \delta_j \mathrm{tr}[\acute{\mathbf{S}}_b] \tag{79}$$

where $F(\mathbf{u}_{i_j,j})$ is cost defined in (22).

As commented in Section III-C, the auxiliary function for $F_d(\mathbf{u}_{i_j,j})$ is the following:

$$W_d\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right) = W\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right) + \gamma_j \mathrm{tr}[\acute{\mathbf{S}}_w] - \delta_j \mathrm{tr}[\acute{\mathbf{S}}_b]. \tag{80}$$

For all $j \in \{1, \ldots, n-1\}$, we have

$$\frac{\partial W_d\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right)}{\partial u_{i_j,j}^l} = \frac{\partial W\left(\mathbf{u}_{i_j,j}, \tilde{\mathbf{u}}_{i_j,j}\right)}{\partial u_{i_j,j}^l} \tag{81}$$

$$\begin{aligned} &-\sum_{i_1,\ldots,i_{n-1}} \mathbf{G}_{i_1,\ldots,i_n} \frac{u_{i_1,1}^m \cdots \left(u_{i_n,n}^m\right)^{(t-1)}}{\sum_{l=1}^K u_{i_1,1}^l \cdots \left(u_{i_n,n}^l\right)^{(t-1)}} + \left(\sum_{i_1,\ldots,i_{n-1}} u_{i_1,1}^m \cdots u_{i_{n-1},n-1}^m\right) u_{i_n,n}^m + 2\gamma\left(u_{i_n,n}^m\right)^2 \\ &\quad - (2\gamma + 2\delta)\left(\frac{1}{N_r}\sum_{\lambda \in \mathcal{F}_r, \lambda \neq i_n} u_{\lambda,n}^m + \frac{1}{N_r} u_{i_n,n}^m\right) u_{i_n,n}^m + 2\delta\mu_m u_{i_n,n}^m = 0 \\ &\Leftrightarrow \left(2\gamma - (2\gamma + 2\delta)\frac{1}{N_r}\right)\left(u_{i_n,n}^m\right)^2 + \left(\sum_{i_1,\ldots,i_{n-1}} u_{i_1,1}^m \cdots u_{i_{n-1},n-1}^m - (2\gamma + 2\delta) \times \left(\frac{1}{N_r}\sum_{\lambda \in \mathcal{F}_r, \lambda \neq i_n} u_{\lambda,n}^m\right) + 2\delta\acute{\mu}_m\right) u_{i_n,n}^m \\ &\quad - \sum_{i_1,\ldots,i_{n-1}} \mathbf{G}_{i_1,\ldots,i_n} \frac{u_{i_1,1}^m \cdots \left(u_{i_n,n}^m\right)^{(t-1)}}{\sum_{l=1}^K u_{i_1,1}^l \cdots u_{i_n,n}^{l\ (t-1)}} = 0. \end{aligned} \tag{76}$$

$$u_{i_n,n}^m = \frac{\acute{T} + \sqrt{\acute{T}^2 + 4\left(2\gamma - (2\gamma + 2\delta)\frac{1}{N_r}\right)\sum_{i_1,\ldots,i_{n-1}} \mathbf{G}_{i_1,\ldots,i_n} \frac{u_{i_1,1}^m \cdots \left(u_{i_n,n}^m\right)^{(t-1)}}{\sum_{l=1}^K u_{i_1,1}^l \cdots \left(u_{i_n,n}^l\right)^{(t-1)}}}}{2\left(2\gamma - (2\gamma + 2\delta)\frac{1}{N_r}\right)} \tag{77}$$

thus, for all $j \in \{1, \ldots, n-1\}$, the update rules for $u^l_{i_j,j}$ are given by (27). In the case that $j = n$ and ψ is multiplicative separable, we have

$$\begin{aligned}\psi\left(\frac{u^l_{i_n,n}}{\tilde{u}^l_{i_n,n}}\right) &\sum_{i_1,\ldots,i_{n-1}}\left(u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}\right)\\ &\times\psi\left(\sum_m \tilde{u}^m_{i_n,n}u^m_{i_1,1}\cdots u^m_{i_{n-1},n-1}\right)\\ &=\sum_{i_1,\ldots,i_{n-1}}\psi\left(\mathbf{G}_{i_1,\ldots,i_{n-1},i_n}\right)u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}\\ &\quad+\frac{\partial\gamma_n\mathrm{tr}[\acute{\mathbf{S}}_w]}{\partial u^l_{i_n,n}}-\frac{\partial\delta_n\mathrm{tr}[\acute{\mathbf{S}}_b]}{\partial u^l_{i_n,n}}\end{aligned} \tag{82}$$

which after some minor calculations becomes

$$\begin{aligned}\psi\left(\frac{u^l_{i_n,n}}{\tilde{u}^l_{i_n,n}}\right) &\sum_{i_1,\ldots,i_{n-1}}\left(u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}\right)\\ &\times\psi\left(\sum_m \tilde{u}^m_{i_n,n}u^m_{1,i_1}\cdots u^m_{i_{n-1},n-1}\right)\\ &=\sum_{i_1,\ldots,i_{n-1}}\psi\left(\mathbf{G}_{i_1,\ldots,i_{n-1},i_n}\right)u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}\\ &\quad-\left(2\gamma_n-(2\gamma_n+2\delta_n)\frac{1}{N_r}\right)u^l_{i_n,n}+(2\gamma_n+2\delta_n)\\ &\quad\times\left(\frac{1}{N_r}\sum_{\lambda\in\mathcal{F}_r}u^l_{\lambda,n}\right)-2\delta\acute{\mu}_l.\end{aligned} \tag{83}$$

In the case of Itakura–Saito distance, (83) becomes

$$\begin{aligned}-\frac{\tilde{u}^l_{i_n,n}}{u^l_{i_n,n}}&\sum_{i_1,\ldots,i_{n-1}}\frac{u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}}{\sum_m\tilde{u}^m_{i_n,n}u^m_{i_1,1}\cdots u^m_{i_{n-1},n-1}}\\ &=-\sum_{i_1,\ldots,i_{n-1}}\frac{u^l_{1,i_1}\cdots u^l_{i_{n-1},n-1}}{\mathbf{G}_{i_1,\ldots,i_{n-1},i_n}}\\ &\quad-\left(2\gamma_n-(2\gamma_n+2\delta_n)\frac{1}{N_r}\right)u^l_{i_n,n}\\ &\quad+(2\gamma_n+2\delta_n)\left(\frac{1}{N_r}\sum_{\lambda\in\mathcal{F}_r}u^l_{\lambda,n}\right)-2\delta_n\acute{\mu}_l.\end{aligned} \tag{84}$$

In order to simplify the notation, let

$$\begin{aligned}A_1&=\left(2\gamma_n-(2\gamma_n+2\delta_n)\frac{1}{N_r}\right)\\ A_2&=(2\gamma_n+2\delta_n)\left(\frac{1}{N_r}\sum_{\lambda\in\mathcal{F}_r}u^l_{\lambda,n}\right)-2\delta_n\acute{\mu}_l\\ B_1&=\sum_{i_1,\ldots,i_{n-1}}\frac{u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}}{\sum_m\tilde{u}^m_{i_n,n}u^m_{1,i_1}\cdots u^m_{i_{n-1},n-1}}\\ B_2&=\sum_{i_1,\ldots,i_{n-1}}\frac{u^l_{i_1,1}\cdots u^l_{i_{n-1},n-1}}{\mathbf{G}_{i_1,\ldots,i_{n-1},i_n}}.\end{aligned} \tag{85}$$

Equation (84) can be transformed to a quadratic equation as

$$A_1\left(u^l_{i_n,n}\right)^2+(B_2-A_2)u^l_{i_n,n}-B_1^{(t-1)}\tilde{u}^l_{i_n,n}=0 \tag{86}$$

and the solution that can guarantee a nonnegative decomposition is

$$u^l_{i_n,n}=\frac{A_2-B_2+\sqrt{(B_2-A_2)^2+4A_1B_1{}^{(t-1)}\tilde{u}^l_{i_n,n}}}{2A_1}. \tag{87}$$

REFERENCES

[1] M. Turk and A. P. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.
[2] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.
[3] A. J. Bell and T. J. Sejnowski, "The independent components of natural scenes are edge filters," *Vis. Res.*, vol. 37, pp. 3327–3338, 1997.
[4] A. Hyvarinen, J. Karhunen, and E. Oja*, Independent Component Analysis*. New York: Wiley Interscience, 2001.
[5] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Neural Inf. Process. Syst.*, pp. 556–562, 2000.
[6] P. O. Hoyer, "Modeling receptive fields with non-negative sparse coding," *Neurocomputing*, vol. 52–54, pp. 547–552, 2003.
[7] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J.f Mach. Learn. Res.*, vol. 5, pp. 1457–1469, 2004.
[8] S. E. Palmer, "Hierarchical structure in perceptual representation," *Cogn. Psychol.*, no. 9, pp. 441–474, 1977.
[9] E. Wachsmuth, M. W. Oram, and D. I. Perrett, "Recognition of objects and their component parts: Responses of single units in the temporal cortex of the macaque," *Cereb. Cortex*, vol. 4, no. 5, pp. 509–522, 1994.
[10] S. Ullman*, High-Level Vision: Object Recognition and Visual Cognition*. Cambridge, MA: MIT Press, 1996.
[11] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychol. Rev.*, vol. 94, pp. 115–147, 1987.
[12] N. K. Logothetis and D. L. Sheinberg, "Visual object recognition," *Annu. Rev. Neurosci.*, vol. 19, pp. 577–621, 1996.
[13] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.
[14] S. Z. Li, X. W. Hou, and H. J. Zhang, "Learning spatially localized, parts-based representation," in *Proc. Comput. Vis. Pattern Recognit.*, Kauai, HI, Dec. 8–14, 2001, pp. 207–212.
[15] I. Buciu and I. Pitas, "Application of non-negative and local non negative matrix factorization to facial expression recognition," in *Proc. Int. Conf. Pattern Recognit.*, Cambridge, U.K., Aug. 23–26, 2004, pp. 288–291.
[16] I. Buciu and I. Pitas, "A new sparse image representation algorithm applied to facial expression recognition," presented at the Mach. Learn. Signal Process. Conf., Sao Lus, Brazil, Sep. 29–Oct. 1 2004.
[17] Y. Wang, Y. Jia, C. Hu, and M. Turk, "Non-negative matrix factorization framework for face recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 4, pp. 1–17, 2005.
[18] A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui, "Nonsmooth nonnegative matrix factorization (nsNMF)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 403–415, Mar. 2006.
[19] S. Sra and I. S. Dhillon, "Nonnegative matrix approximation: Algorithms and applications," Dept. Comput. Sci., Univ. Texas at Austin, Austin, TX, Tech. Rep. TR-06-27, 2006.
[20] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas, "Exploiting discriminant information in nonnegative matrix factorization with application tom frontal face verification," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 683–695, May 2006.
[21] M. Kirby and L. Sirovich, "Application of the karhunen-loeve procedure for the characterization of human faces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 103–108, Jan. 1990.
[22] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2004, vol. 17.
[23] T. Hazan, S. Polak, and A. Shashua, "Sparse image coding using a 3d non-negative tensor factorization," in *Proc. 10th IEEE Int. Conf. Comput. Vis.*, Oct. 17–21, 2005, vol. 1, pp. 50–57.
[24] M. P. Friedlander and K. Hatz, "Computing nonnegative tensor factorizations," Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada, Tech. Rep. TR-2006-21, Oct. 2006.

[25] A. Shashua and T. Hazan, “Non-negative tensor factorization with applications to statistics and computer vision,” in *Proc. Int. Conf. Mach. Learn.*, Aug. 2005, pp. 792–799.

[26] J. B. Kruskal, “Three way arrays: Rank and uniqueness of trilinear decomposition, with application to arithmetic complexity and statistics,” *Linear Algebra Appl.*, vol. 18, pp. 95–138, 1977.

[27] N. D. Sidiropoulos and R. Bro, “On the uniqueness of multilinear decomposition of n-way arrays,” *J. Chemometrics*, vol. 37, pp. 229–239, 2000.

[28] Y. Wang, Y. Jiar, C. Hu, and M. Turk, “Fisher non-negative matrix factorization for learning local features,” in *Proc. Asian Conf. Comput. Vis.*, 2004, pp. 27–30.

[29] I. Buciu and I. Pitas, “DNMF modeling of neural receptive fields involved in human facial expression perception,” *J. Vis. Commun. Image Represent.*, vol. 17, no. 5, pp. 958–969, Oct. 2006.

[30] J. Yang, S. Yang, Y. Fu, X. Li, and T. Huang, “Nonnegative graph embedding,” in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[31] T. Zhang, B. Fang, Y. Y. Tang, G. He, and J. Wen, “Topology preserving nonnegative matrix factorization for face recognition,” *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 574–584, Apr. 2008.

[32] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H.-J. Zhang, “Multilinear discriminant analysis for face recognition,” *IEEE Trans. Image Process.*, vol. 16, no. 1, pp. 212–220, Jan. 2007.

[33] D. Tao, X. Li, X. Wu, and S. J. Maybank, “General tensor discriminant analysis and Gabor features for gait recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1700–1715, Oct. 2007.

[34] L. Finesso and C. Spreij, “Nonnegative matrix factorization and i-divergence alternative minimization,” *Linear Algebra Appl.*, vol. 416, no. 2–3, pp. 270–286, 2006.

[35] C.-J. Lin, “On the convergence of multiplicative update for nonnegative matrix factorization,” *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1589–1596, Nov. 2007.

[36] C.-J. Lin, “Projected gradients for nonnegative matrix factorization,” *Neural Comput.*, no. 19, pp. 2756–2779, 2007.

[37] D. D. Lee and H. S. Seung, “Unsupervised learning by convex and conic coding,” *Neural Inf. Process. Syst.*, pp. 515–521, 1996.

[38] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[39] J. Duchene and S. Leclercq, “An optimal transformation for discriminant and principal component analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, no. 6, pp. 978–983, Jun. 1988.

[40] D. Cai, X. He, J. Han, and H.-J. Zhang, “Orthogonal Laplacianfaces for face recognition,” *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3608–3614, Nov. 2006.

[41] J. Yang, A. F. Frangi, J. Yang, D. Zhang, and Z. Jin, “KPCA plus LDA: A complete kernel Fisher discriminant framework for feature extraction and recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 2, pp. 230–244, Feb. 2005.

[42] I. Kotsia, S. Zafeiriou, and I. Pitas, “Texture and shape information fusion for facial expression and facial action unit recognition,” *Pattern Recognit.*, vol. 41, no. 3, pp. 833–851, 2008.

[43] K. Messer, J. Matas, J. V. Kittler, J. Luettin, and G. Maitre, “XM2VTSDB: The extended M2VTS database,” in *Proc. Int. Conf. Audio- Video-Based Person Authenticat.*, 1999, pp. 72–77.

[44] T. Kanade, J. Cohn, and Y. Tian, “Comprehensive database for facial expression analysis,” in *Proc. IEEE Int. Conf. Face Gesture Recognit.*, Mar. 2000, pp. 46–53.

[45] M. Pantic and L. J. M. Rothkrantz, “Expert system for automatic analysis of facial expressions,” *Image Vis. Comput.*, vol. 18, no. 11, pp. 881–905, Aug. 2000.

[46] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski, “Classifying facial actions,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 974–989, Oct. 1999.

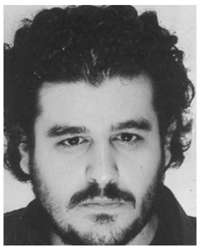
[47] I. Kotsia, N. Nikolaidis, and I. Pitas, “Fusion of geometrical and texture information for facial expression recognition,” in *Proc. Int. Conf. Image Process.*, Atlanta, GA, Oct. 2006, pp. 2649–2652.

[48] J. Devore and R. Peck*, Statistics: The Exploration and Analysis of Data*, 3rd ed. Pacific Grove, CA: Brooks Cole, 1997.

[49] E. Benetos and C. Kotropoulos, “A tensor-based approach for automatic music genre classification,” in *Proc. 16th Eur. Signal Process. Conf.*, Lausanne, Switzerland, Aug. 2008, pp. 1–4.

[50] E. Benetos, M. Kotti, and C. Kotropoulos, “Large scale musical instrument identification,” in *Proc. 4th Sound Music Comput. Conf.*, Jul. 2007.

[51] B. Gökberk, H. Dutagaci, A. Ulas, L. Akarun, and B. Sankur, “Representation plurality and fusion for 3-D face recognition,” *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 38, no. 1, pp. 155–173, Feb. 2008.



Stefanos Zafeiriou was born in Thessaloniki, Greece, in 1981. He received the B.Sc. (with highest honors) and Ph.D. degrees in informatics from Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2003 and 2007, respectively.

He has coauthored over than 30 journal and conference publications. During 2007–2008, he was a Senior Researcher at the Department of Informatics, Aristotle University of Thessaloniki. Currently, he is a Senior Researcher at the Department of Electrical and Electronic Engineering, Imperial College London, U.K. His current research interests lie in the areas of signal and image processing, computational intelligence, pattern recognition, machine learning, computer vision and detection, and estimation theory.

Dr. Zafeiriou received various scholarships and awards during his undergraduate, doctoral, and postdoctoral studies.