Nonlinear Non-Negative Component Analysis Algorithms

Stefanos Zafeiriou, *Member, IEEE*, and Maria Petrou, *Senior Member, IEEE*

***Abstract*—In this, paper general solutions for nonlinear non-negative component analysis for data representation and recognition are proposed. Motivated by a combination of the non-negative matrix factorization (NMF) algorithm and kernel theory, which has lead to a recently proposed NMF algorithm in a polynomial feature space, we propose a general framework where one can build a nonlinear non-negative component analysis method using kernels, the so-called projected gradient kernel non-negative matrix factorization (PGKNMF). In the proposed approach, arbitrary positive definite kernels can be adopted while at the same time it is ensured that the limit point of the procedure is a stationary point of the optimization problem. Moreover, we propose fixed point algorithms for the special case of Gaussian radial basis function (RBF) kernels. We demonstrate the power of the proposed methods in face and facial expression recognition applications.**

***Index Terms*—Face recognition, facial expression recognition, kernel methods, non-negative matrix factorization, subspace techniques.**

I. Introduction

In the computer vision and pattern recognition fields, one of the most popular ways to represent an object is by expressing it as a linear combination of basis objects. The basis objects are in many cases used to extract features and/or find a low dimensionality object representation to be subsequently used for recognition. The basis can also be used for compressing the object representation. One of the most popular methods to find a basis is the *principal component analysis* (PCA) [1]. PCA has attracted a lot of attention in computer vision and especially in recognition after its application to facial image representation [2]. More specifically, its use for face detection and recognition problems was of great interest [3]. Another very popular method, that works on the statistical independence of the basis objects or the weights of the representation, is the independent component analysis (ICA) [4]. ICA has been widely used for the problem of face recognition [5], [6]. In this paper, we deal with the problem of object representation using images and we are particularly interested in face and facial expression recognition problems.

Manuscript received March 16, 2009; revised October 23, 2009. First published December 18, 2009; current version published March 17, 2010. This work was supported by the EPSRC project EP/E028659/1 Face Recognition using Photometric Stereo. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Miles N. Wernick.

The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, South Kensington Campus, London SW7 2AZ U.K. (e-mail: s.zafeiriou@imperial.ac.uk; maria.petrou@imperial.ac.uk).

Digital Object Identifier 10.1109/TIP.2009.2038816

In [7] and [8], a decomposition of objects using a linear basis was proposed by considering non-negativity constraints for both the basis and the weights of the linear combination, the so-called non-negative matrix factorization (NMF). NMF, like PCA, represents an 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 an image only by additions of weighted basis images. The non-negativity constraint arises in many real image processing applications, since the pixels in a grayscale image have non-negative intensities. As stated in [8], an object (represented as an image) is more naturally coded into its parts by using only additions of the different bases. Moreover, the non-negativity constraints correspond better to the intuitive notion of combining facial parts to create a complete facial image. Apart from that, in [8], NMF has been also motivated by biological indications, like the fact that the firing rates in visual perception neurons are non-negative. 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 [8]–[15].

Both NMF and PCA are linear models; thus, they may fail to model efficiently the nonlinearities that are present in most real life applications. Nonlinear component analysis is a research topic that has been greatly developed in the past decade [17]–[19]. This is mainly attributed to the great success of combining Support Vector Machines (SVMs) with kernels [20]. Since then, kernels have been widely used for finding nonlinear counterparts of PCA, the so-called Kernel Principal Component Analysis (KPCA) [17] and for discovering nonlinear high order dependencies of data, the so-called Kernel Independent Component Analysis (KICA) [18]. Recently, a nonlinear counterpart of NMF has been proposed, the so-called Polynomial non-negative Matrix Factorization (PNMF) [19]. The PNMF has been partly motivated by biological issues like yielding a model compatible with the neurophysiology paradigms (non-negativity constraints and nonlinear image decomposition [21]–[23]) and has been used to discover higher-order correlations between image pixels that may lead to more powerful latent features. For more details on the motivation of PNMF the interested reader is referred to [19].

In PNMF, the original images are initially projected into some polynomial feature spaces of arbitrary dimensionality using polynomial functions. The problem is formulated as follows: find a set of non-negative weights and non-negative basis vectors, such that the nonlinearly mapped training vectors can be written as linear combinations of the nonlinear mapped non-negative basis vectors. These basis vectors have high

resemblance with the so-called pre-images of the kernel based methods [24]–[26]. For more details regarding the notion of pre-images, the interested reader is referred to [24]–[26]. The PNMF has followed a similar to the original NMF approach for solving the optimization problem [7]. That is, a proper auxiliary function was defined and minimized in order to define proper multiplicative updating rules that guaranteed the nonincreasing evolution of the cost function. The PNMF method proposed in [19] uses as nonlinear functions only polynomial kernels. Moreover, the cost function, even though it was defined as a nonlinear function of the images using arbitrary degree polynomial kernels, was approximated by a quadratic function. Under these two simplifications, in [19], multiplicative updating rules were proposed for solving the defined optimization problem. These updating rules only guaranteed the nonincreasing evolution of the cost function while it was not proven that the limit point was a stationary point of the optimization procedure. Stationarity is an important property of a limit point since every local minimum has to be stationary. A lot of research has been recently conducted in order to define methods that ensure the stationarity of NMF, since the original NMF algorithm [7] has been criticised for not possessing such a property [27]–[30].

Finally, we should comment that in [31] the original NMF method was applied to the kernel matrix of the original data. This is different from the approach proposed in [19] and the approach followed in this paper. In our case, the problem is formulated as follows: find a set of non-negative weights and non-negative basis vectors, such that the non-negative training vectors (under the nonlinear mapping) can be written as a linear combination of the learned non-negative nonlinearly mapped basis vectors. On the other hand, in [31] the aim was to find a non-negative decomposition of the kernel matrix which was just the application of NMF to a non-negative matrix of inner products.

In this paper, we propose a general method for nonlinear non-negative component analysis using arbitrary positive definite kernels (Mercer's kernels [24]), in order to remedy the above mentioned limitations of PNMF [19] and the NMF of kernel matrices [31]. Moreover, we present a method for nonlinear non-negative component analysis using Gaussian RBF kernels. Summarizing, the contributions of the proposed approach are the following.

- A general method for nonlinear non-negative matrix factorization (NNMF) in which arbitrary kernels can be used (contrary to [19] where only polynomial kernels have been used).
- In [19] the cost function was always approximated using quadratic terms, which is unsatisfactory since the actual nonlinear cost function was not truly minimized. In the proposed method we consider the problem in the general case, without having to approximate the cost function by a quadratic function.
- The method is based on projected gradients which guarantee that the limit point will be a stationary point of the optimization procedure, unlike the method in [19], which was based on optimizing an auxiliary function, which did not guarantee that the limit point would be stationary.
- By exploiting certain properties of the Gaussian RBF kernels, we propose simple fixed point NNMF methods. Moreover, we reformulate the optimization problem and propose a robust convex kernel non-negative matrix factorization method.
- The proposed method, in contrast to [31], not only decomposes the kernel matrix but also identifies the projected vectors (pre-images).

The rest of this paper is organized as follows. The problem of nonlinear non-negative matrix factorization with polynomial kernels is briefly outlined in Section II. The proposed solution, using projected gradients is presented in Section III. In Section IV, a fixed point algorithm is proposed for the special case of Gaussian RBF kernels. Experimental results of the proposed method in face and facial expression recognition applications are described in Section V. Finally, conclusions are drawn in Section VI. A short version of this paper can be found in [32].

II. Nonlinear Non-Negative Matrix Factorization in a Polynomial Feature Space

In this paper we consider the problem of representing facial images in a nonlinear way. Every facial image is scanned row-wise to form an image vector $\mathbf{x}_i \in \Re_+^F$. Let us assume that we have a database of M images in total. The problem of PNMF, in [19], was formulated as follows. Let $\phi : \Re_+^F \rightarrow \mathcal{H}$ be a mapping that projects image $\mathbf{x}_i$ to a Hilbert space $\mathcal{H}$ of arbitrary dimensionality. Our aim is to find a set of K vectors $\mathbf{z}_j \in \Re_+^F$ and a set of weights $h_{ji} \geq 0$ such as

$$\phi(\mathbf{x}_i) \approx \sum_{j=1}^{K} h_{ji}\phi(\mathbf{z}_j) \tag{1}$$

or more generally

$$\mathbf{X}^{\Phi} \approx \mathbf{Z}^{\Phi}\mathbf{H} \tag{2}$$

where $\mathbf{X}^{\Phi} \equiv [\phi(\mathbf{x}_1)\dots\phi(\mathbf{x}_M)]$, $\mathbf{Z}^{\Phi} \equiv [\phi(\mathbf{z}_1)\dots\phi(\mathbf{z}_P)]$ and $[\mathbf{H}]_{ji} \equiv h_{ji}$ with $\mathbf{H} \in \Re_+^{P\times M}$. Vectors $\mathbf{z}_j$ are the so-called pre-images [24]–[26] of the approximation. The dot product in $\mathcal{H}$ is written by means of kernels as $k(\mathbf{x}_i, \mathbf{x}_j) \equiv \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. In this paper, we apply our method using only positive definite, continuous and symmetric kernels. The use of indefinite kernels [33]–[35] for performing nonlinear non-negative component is a very interesting topic for further research.

In order to find the preimage matrix $\mathbf{Z} \equiv [\mathbf{z}_1 \dots \mathbf{z}_K]$ and the weights matrix $\mathbf{H}$, the least squares error is used for measuring the error of the approximation

$$D_{\phi}(\mathbf{X}^{\Phi}, \mathbf{Z}^{\Phi}\mathbf{H}) = \sum_{i=1}^{M} ||\phi(\mathbf{x}_i) - \sum_{j} h_{ji}\phi(\mathbf{z}_j)||^2. \tag{3}$$

The optimization problem is as follows:

$$\min_{z_{ik}\geq 0,\ h_{kj}\geq 0} D_{\phi}(\mathbf{X}^{\Phi}, \mathbf{Z}^{\Phi}\mathbf{H}) \tag{4}$$

where $i = 1, \dots, F$, $k = 1, \dots, K$ and $j = 1, \dots, M$.

In order to provide further motivation for the approach, we may express problem (4) as follows. Given a database of images $\mathbf{X} \in \Re_+^{F\times M}$ and a nonlinear mapping ϕ we want to find a matrix $\mathbf{Z}$ of preimages $\mathbf{z}_j$ of the same domain as $\mathbf{x}_i$ (i.e., if $\mathbf{x}_i$ are non-negative grayscale images then we want the preimages $\mathbf{z}_j$ to be non-negative images, as well). After the projection, under mapping ϕ, we want the weights h_{ji} of the linear combination to be non-negative. Moreover, let us consider the kernels K-means

clustering algorithms [36], [37]. K-means uses K prototypes, the centroids of clusters. These centroids are then used in order to characterize the data. Let that our dataset $\mathbf{X}^{\Phi}$ be partitioned in K clusters with cluster centroids being the columns of $\mathbf{Z}^{\Phi}$. Matrix $\mathbf{H}$ then contains the cluster indicators. That is, in case that $\phi(\mathbf{x}_i)$ belongs to p-cluster, then $h_{pj}=1$ else $h_{pj}=0$. Let us consider the K-means kernel clustering objective function on the columns of matrix $\mathbf{X}^{\Phi}$

$$\begin{aligned} D_{K,\phi} &\triangleq \sum_{p=1}^{K}\sum_{\phi(\mathbf{x}_i)\in\mathcal{C}_p} \|\phi(\mathbf{x}_i)-\phi(\mathbf{z}_p)\|^2 \\ &= \sum_{i=1}^{n}\sum_{p=1}^{K} h_{pi}\|\phi(\mathbf{x}_i)-\phi(\mathbf{z}_p)\|^2 \\ &= \|\mathbf{X}^{\Phi}-\mathbf{Z}^{\Phi}\mathbf{H}\|^2 \end{aligned} \tag{5}$$

where $\mathcal{C}_p$ is the pth cluster. The proof of the derivation of (5) is given in Appendix I. From the above, if we relax the elements of $\mathbf{H}$ to take values not only in the set $\{0,1\}$ but in $\Re^{+}$, we can deduce that the proposed nonlinear non-negative component analysis of the optimization problem (4) is a relaxed kernel K-means optimization problem, and the bases $\mathbf{z}_j$ are the pre-images of the centroids of the clusters.

In [19], in order to solve the constrained optimization problem (4) the authors used auxiliary functions for both $\mathbf{H}$ and $\mathbf{Z}$. Before proceeding in describing how the algorithm in [19] was formulated, we should define the following matrices:

$$\begin{aligned} [\mathbf{K}_{x,x}]_{ij} &\equiv \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) = k(\mathbf{x}_i,\mathbf{x}_j) \\ [\mathbf{K}_{z,z}]_{ij} &\equiv \langle\phi(\mathbf{z}_i),\phi(\mathbf{z}_j)\rangle = \phi(\mathbf{z}_i)^T\phi(\mathbf{z}_j) = k(\mathbf{z}_i,\mathbf{z}_j) \\ [\mathbf{K}_{z,x}]_{ij} &\equiv \langle\phi(\mathbf{z}_i),\phi(\mathbf{x}_j)\rangle = \phi(\mathbf{z}_i)^T\phi(\mathbf{x}_j) = k(\mathbf{z}_i,\mathbf{x}_j) \\ \mathbf{K}_{x,z} &\equiv \mathbf{K}_{z,x}^T. \end{aligned} \tag{6}$$

In [19], the kernel considered was the polynomial kernel

$$k(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j)^d \tag{7}$$

where d was the degree of the polynomial. Assuming that vectors $\mathbf{x}_i$ are linearly independent, then matrix $\mathbf{K}_{x,x}$ (which has the properties of a Gram matrix [24]) is positive definite. The same holds for the Gram matrix $\mathbf{K}_{z,z}$, in the case that $\mathbf{z}_j$ are linearly independent. In all cases, both $\mathbf{K}_{x,x}$ and $\mathbf{K}_{z,z}$ are at least positive semidefinite matrices.

In order to calculate the solution of the optimization problem, Buciu *et al.* [19] defined auxiliary functions and derived in that way a nice set of updating rules. In an iterative scheme, let us denote by t the iteration step and by $y^{(t)}$ the current estimate of the solution of the problem. A function G is an auxiliary function for $f(\mathbf{y}):\Re^d\rightarrow\Re$ if $G(\mathbf{y}^{(t)},\mathbf{y}^{(t-1)})\geq f(\mathbf{y}^{(t)})$ for any $\mathbf{y}^{(t-1)}\in\Re^d$ and $G(\mathbf{y}^{(t)},\mathbf{y}^{(t)})=f(\mathbf{y}^{(t)})$. Let us define $f(\mathbf{h}_i)$ ($\mathbf{h}_i$ is the ith column of matrix $\mathbf{H}$) as $f(\mathbf{h}_i)\equiv D_{\phi}(\phi(\mathbf{x}_i),\mathbf{Z}^{\Phi}\mathbf{h}_i)$ keeping matrix $\mathbf{Z}$ constant (if $\mathbf{Z}$ is kept constant then $\mathbf{Z}^{\Phi}$ is constant, as well). The auxiliary function for $f(\mathbf{h}_i)$ is defined as

$$\begin{aligned} G(\mathbf{h}_i^{(t)},\mathbf{h}_i^{(t-1)}) &= f(\mathbf{h}_i^{(t-1)}) + (\mathbf{h}_i^{(t)}-\mathbf{h}_i^{(t-1)})^T \\ &\times\nabla f(\mathbf{h}_i^{(t)}) + \frac{1}{2}(\mathbf{h}_i^{(t)}-\mathbf{h}_i^{(t-1)})^T\mathbf{L}(\mathbf{h}_i^{(t)}-\mathbf{h}_i^{(t-1)}) \end{aligned} \tag{8}$$

where $\mathbf{L}$ is a diagonal matrix with $\mathbf{L}_{kk}=[\mathbf{K}_{z,z}\mathbf{h}_i]_k/h_{ki}$. As we can see, the cost function is quadratic in terms of $\mathbf{h}_i$, thus near $\mathbf{h}_i^{(t-1)}$ it can be written as

$$\begin{aligned} f(\mathbf{h}_i^{(t)}) &= f(\mathbf{h}_i^{(t-1)}) + (\mathbf{h}_i^{(t)}-\mathbf{h}_i^{(t-1)})^T\nabla f(\mathbf{h}_i^{(t)}) \\ &+\frac{1}{2}(\mathbf{h}_i^{(t)}-\mathbf{h}_i^{(t-1)})^T\nabla^2 f(\mathbf{h}_i^{(t)})(\mathbf{h}_i^{(t)}-\mathbf{h}_i^{(t-1)}). \end{aligned} \tag{9}$$

Using the above expansion it can be proven that G is an auxiliary function of f [19].

When keeping all but $\mathbf{z}_i$ fixed, then $f(\mathbf{z}_i)$ is defined as $f(\mathbf{z}_i)=D_{\phi}(\mathbf{X}^{\Phi},\mathbf{Z}^{\Phi}\mathbf{H})$. Although the expansion of f in (9) is valid, since f is quadratic in terms of $\mathbf{h}_i$, it is not valid in case we want to expand $f(\mathbf{z}_i^{(t)})$ in a similar way around $\mathbf{z}_i^{(t-1)}$. This is due to the fact that, apart from the case of linear kernel $k(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$, the cost function for arbitrary degree polynomial kernels is not quadratic. Nevertheless, in [19], the $f(\mathbf{z}_i^{(t)})$ was expanded near $\mathbf{z}_i^{(t-1)}$ only in quadratic terms (i.e., keeping only the first three terms of the Taylor expansion)

$$\begin{aligned} f(\mathbf{z}_i^{(t)}) &\approx f(\mathbf{z}_i^{(t-1)}) + (\mathbf{z}_i^{(t)}-\mathbf{z}_i^{(t-1)})^T\nabla f(\mathbf{z}_i^{(t)}) \\ &+\frac{1}{2}(\mathbf{z}_i^{(t)}-\mathbf{z}_i^{(t-1)})^T\nabla^2 f(\mathbf{z}_i^{(t)})(\mathbf{z}_i^{(t)}-\mathbf{z}_i^{(t-1)}) \end{aligned} \tag{10}$$

and an auxiliary function $G(\mathbf{z}_i^{(t)},\mathbf{z}_i^{(t-1)})$ similar to (8) was defined. In summary, in [19]:

- cost function $D_{\phi}(\mathbf{X}^{\Phi},\mathbf{Z}^{\Phi}\mathbf{H})$ was defined using only polynomial kernels;
- cost function $D_{\phi}(\mathbf{X}^{\Phi},\mathbf{Z}^{\Phi}\mathbf{H})$ was further approximated using up to quadratic terms;
- the auxiliary function $G(\mathbf{z}_i^{(t)},\mathbf{z}_i^{(t-1)})$ used was defined only for polynomial kernels and only after the quadratic approximation of the cost function.

In [19], by letting $\partial G(\mathbf{z}_i^{(t)},\mathbf{z}_i^{(t-1)})/\partial z_{ik}^{(t)}=0$ and $\partial G(\mathbf{h}_i^{(t)},\mathbf{h}_i^{(t-1)})/\partial h_{kj}^{(t)}=0$ the following multiplicative updating rules were proposed for minimizing the cost function (3)

$$\begin{aligned} \mathbf{H}^{(t)} &\leftarrow \mathbf{H}^{(t-1)}\odot\frac{\mathbf{K}_{x,z}^{(t-1)}}{(\mathbf{K}_{z,z}^{(t-1)}\mathbf{H}^{(t-1)})} \\ \tilde{\mathbf{Z}}^{(t)} &\leftarrow \mathbf{Z}^{(t-1)}\odot\frac{(\mathbf{X}\acute{\mathbf{K}}_{x,z}^{(t-1)})}{(\mathbf{Z}^{(t-1)}\boldsymbol{\Omega}\acute{\mathbf{K}}_{z,z}^{(t-1)})} \\ \mathbf{Z}^{(t)} &\leftarrow \frac{\tilde{\mathbf{Z}}^{(t)}}{\mathbf{S}} \end{aligned} \tag{11}$$

where matrix $\boldsymbol{\Omega}$ is a diagonal matrix such that $[\boldsymbol{\Omega}]_{jj}=\sum_{k=1}^{M}h_{kj}$ and $\mathbf{S}$ is a normalization matrix such that the columns of $\mathbf{Z}^{(t)}$ sum up to one. Matrices $\acute{\mathbf{K}}_{x,z}$ contain parts of the first derivatives with respect to z_{ik} of the polynomial kernels (see Appendix I) and are defined as $[\acute{\mathbf{K}}_{x,z}]_{ij}\equiv d(\mathbf{x}_i^T\mathbf{z}_j)^{d-1}$ and $[\acute{\mathbf{K}}_{z,z}]_{ij}\equiv d(\mathbf{z}_i^T\mathbf{z}_j)^{d-1}$. Operator $\odot$ is used for denoting element-wise matrix multiplication while/denotes element-wise matrix division. The main calculation in terms of complexity is that of $\mathbf{X}\acute{\mathbf{K}}_{x,z}^{(t-1)}$ which requires $O(FMPd)$ calculations. Thus, the complexity of the multiplicative updating rules is $O(rFMPd)$, where r is the maximum allowed number of iterations.

As we have already mentioned, the updating rules (11) hold only for polynomial kernels. Moreover, in [19], the cost function

(3) does not increase, under the above updating rules, only if $0 \leq z_{ik} \leq 1$ (since nonincrease requires $(\mathbf{x}^T\mathbf{z})^{d-2} \geq (\mathbf{z}^T\mathbf{z})^{d-2}$, which is true for $x \in [0, 255]$ and $z \in [0, 1]$). Hence, if $z_{ik} > 1$ the above updating rules are not valid for minimizing the cost. In NMF [7], a similar normalization procedure was considered for the basis (i.e., the elements of every basis function had to sum up to one), but this was used in order to limit the solution for the NMF algorithm and it was not a requirement for algorithmic convergence, contrary to [19], in which the normalization of $\mathbf{Z}$ was a requirement for convergence.

Summarizing, the only thing that is guaranteed by the above optimization procedure is that the cost D_ϕ, when approximated by a quadratic function in terms of $\mathbf{H}$ or in terms of $\mathbf{Z}$ (with $0 \leq z_{ik} \leq 1$), is nonincreasing. The procedure can only be used for polynomial kernels. Moreover, it is not proven that the limit point is stationary nor that it is a local minimum. In the following, we propose novel procedures, where a wide variety of kernels can be used in the decomposition, without requiring the cost function (3) to be quadratic. Moreover, we guarantee that the limit point of the procedure is a stationary point of the optimization problem.

Before describing the proposed algorithms, we briefly describe the difference between the Nonlinear non-negative Component Analysis proposed in this paper and the non-negative Matrix Factorization on Kernels proposed in [31].

In our approach, we consider the problem of approximating $\mathbf{X}^\Phi$ using a matrix $\mathbf{Z}^\Phi$ and a matrix of weights $\mathbf{H}$, with $\mathbf{x}_i$ and $\mathbf{z}_j$ being in the same domain (i.e., $\Re_+^F$) [this problem is formally expressed by (2)]. On the other hand, in [31], the authors considered an easier and more restricted problem, namely the problem of finding a non-negative decomposition of matrix $\mathbf{X}^{\Phi^T}\mathbf{X}^\Phi = \mathbf{K}_{x,x}$. That is, they set up the problem of approximating $\mathbf{K}_{x,x} \in \Re_+^{M \times M}$

$$\mathbf{K}_{x,x} \approx \mathbf{GW} \tag{12}$$

with two non-negative matrices $\mathbf{G} \in \Re_+^{M \times P}$ and $\mathbf{W} \in \Re_+^{P \times M}$ using the NMF algorithm of [29], [30]. As it can be seen, the above problem is just the application of NMF to matrix $\mathbf{K}_{x,x}$.

In the proposed approach, we not only find an approximation of the kernel matrix $\mathbf{K}_{x,x}$, but we also identify the pre-images $\mathbf{Z}^\Phi$, as well. The latter cannot be achieved by following the approach in [31].

III. Projected Gradient Methods For Nonlinear Non-Negative Matrix Factorization

Using the notion of kernels, metric (3), that quantifies the approximation of the vectors in $\mathbf{X}^\Phi$ as a linear combination of the basis in $\mathbf{Z}^\Phi$, can be expanded as (13), shown at the bottom of the page.

The minimization of (13), subject to non-negative constraints for the weights matrix $\mathbf{H}$ and the basis matrix $\mathbf{Z}$, yields the nonlinear non-negative decomposition. This optimization problem will be solved using projected gradients in order to guarantee that the limit point is stationary and that the non-negativity constraints of $\mathbf{z}_i$ and $\mathbf{h}_j$ are met. In order to find the limit point, two functions are defined

$$\begin{aligned} f_{\mathbf{Z}}(\mathbf{H}) &\equiv D_\phi(\mathbf{X}^\Phi, \mathbf{Z}^\Phi\mathbf{H}) \text{ and} \\ f_{\mathbf{H}}(\mathbf{Z}) &\equiv D_\phi(\mathbf{X}^\Phi, \mathbf{Z}^\Phi\mathbf{H}) \end{aligned} \tag{14}$$

by keeping $\mathbf{Z}$ and $\mathbf{H}$ fixed, respectively.

The projected gradient method, used in this paper, successively optimizes two subproblems [30]

$$\begin{aligned} &\min_{\mathbf{Z}} f_{\mathbf{H}}(\mathbf{Z}) \\ &\text{subject to } z_{ik} \geq 0 \end{aligned} \tag{15}$$

and

$$\begin{aligned} &\min_{\mathbf{H}} f_{\mathbf{Z}}(\mathbf{H}) \\ &\text{subject to } h_{kj} \geq 0. \end{aligned} \tag{16}$$

The first partial derivative with respect to h_{ab} is

$$\begin{aligned} \frac{\partial f_{\mathbf{Z}}}{\partial h_{ab}} &= [(\mathbf{Z}^{\Phi^T}\mathbf{Z}^\Phi)\mathbf{H} - \mathbf{Z}^{\Phi^T}\mathbf{X}^\Phi]_{ab} \\ &= [\mathbf{K}_{z,z}\mathbf{H} - \mathbf{K}_{z,x}]_{ab}. \end{aligned} \tag{17}$$

$$\begin{aligned} D_\phi(\mathbf{X}^\Phi, \mathbf{Z}^\Phi\mathbf{H}) &= \sum_{i=1}^{M} \|\phi(\mathbf{x}_i) - \sum_{j=1}^{P} h_{ji}\phi(\mathbf{z}_j)\|^2 \\ &= \sum_{i=1}^{M} \left(\phi(\mathbf{x}_i) - \sum_{j=1}^{P} h_{ji}\phi(\mathbf{z}_j)\right)^T \left(\phi(\mathbf{x}_i) - \sum_{j=1}^{P} h_{ji}\phi(\mathbf{z}_j)\right) \\ &= \sum_{i=1}^{M} \Bigg(\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_i) - \sum_{j=1}^{P} h_{ji}\phi(\mathbf{z}_j)^T\phi(\mathbf{x}_i) \\ &\qquad - \sum_{j=1}^{P} h_{ji}\phi(\mathbf{x}_i)^T\phi(\mathbf{z}_j) + \sum_{j=1}^{P}\sum_{l=1}^{P} h_{ji}h_{li}\phi(\mathbf{z}_l)^T\phi(\mathbf{z}_j)\Bigg) \\ &= \sum_{i=1}^{M} \left(k(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{j=1}^{P} h_{ji}k(\mathbf{z}_j, \mathbf{x}_i) + \sum_{j=1}^{P}\sum_{l=1}^{P} h_{ji}h_{li}k(\mathbf{z}_l, \mathbf{z}_j)\right) \end{aligned} \tag{13}$$

For the first partial derivative with respect to z_{ab} we have (18), shown at the bottom of the page.

The projected gradient KNMF method is an iterative method that comprises two main phases. These two phases are iteratively repeated until the ending condition is met or the number of iterations exceeds a given number. In the first phase, an iterative procedure is followed for the optimization of (15), while in the second phase, a similar procedure is followed for the optimization of (16). At the beginning, the basis matrix $\mathbf{Z}^{(1)}$ and the weight matrix $\mathbf{H}^{(1)}$ are initialized randomly, in such a way that their entries are non-negative.

A. Solving Subproblem (15)

Consider the subproblem of optimizing with respect to $\mathbf{Z}$, while keeping matrix $\mathbf{H}$ constant. The optimization is an iterative procedure that is repeated until $\mathbf{Z}^{(t)}$ becomes a stationary point of (15). At every iteration, a proper step size a_t is required to updating matrix $\mathbf{Z}^{(t)}$. When a proper updating is found, the stationarity condition is checked for and, if met, the procedure stops.

1) Updating Matrix $\mathbf{Z}$*:* For a number of iterations $t = 1, 2, \ldots$ the following updatings are performed [30]:

$$\mathbf{Z}^{(t+1)} = P\left[\mathbf{Z}^{(t)} - a_t \nabla f_{\mathbf{H}}(\mathbf{Z}^{(t)})\right] \tag{19}$$

where $a_t = \beta^{g_t}$ and g_t is the first non-negative integer such that

$$f_{\mathbf{H}}(\mathbf{Z}^{(t+1)}) - f_{\mathbf{H}}(\mathbf{Z}^{(t)}) \leq \sigma \left\langle \nabla f_{\mathbf{H}}(\mathbf{Z}^{(t)}), \mathbf{Z}^{(t+1)} - \mathbf{Z}^{(t)} \right\rangle. \tag{20}$$

The projection rule $P[.] = \max[., 0]$ refers to the elements of the matrix and guarantees that the updating will not contain any negative entries. Operator $\langle ., . \rangle$ is the inner product between matrices, defined as

$$\langle \mathbf{A}, \mathbf{B} \rangle = \sum_i \sum_j a_{ij} b_{ij} \tag{21}$$

where $[\mathbf{A}]_{ij} = a_{ij}$ and $[\mathbf{B}]_{ij} = b_{ij}$. Condition (20) ensures the sufficient decrease of the $f_{\mathbf{H}}(\mathbf{Z})$ function values per iteration.

The search for a proper value for a_t is the most time consuming procedure, thus, as few iteration steps as possible are desired. Several procedures have been proposed for the selection and updating of the a_t values [38], [39]. Algorithm 4 in [30] has been used in our experiments. The values of parameters β and σ are chosen to be equal to 0.1 and 0.01 ($0 < \beta < 1$, $0 < \sigma < 1$), respectively. These values are typical values used in other projected gradient methods as in [30]. The choice of σ has been thoroughly studied in [30], [38], and [39]. During experiments it was observed that a smaller value of β reduces more aggressively the step size, but it may also result in a step size that is too small. The search for a_t is repeated until point $\mathbf{Z}^{(t)}$ becomes a stationary point.

2) Checking for Stationarity: In this step it is checked whether or not at the limit point the first order derivatives are close to zero (stationarity condition). A commonly used condition to check the stationarity of a point is the following [38]:

$$\|\nabla^P f_{\mathbf{H}}(\mathbf{Z}^{(t)})\|_F \leq \epsilon_{\mathbf{Z}} \|\nabla f_{\mathbf{H}}(\mathbf{Z}^{(1)})\|_F \tag{22}$$

where $\nabla^P f_{\mathbf{H}}(\mathbf{Z})$ is the projected gradient for the constrained optimization problem defined as

$$[\nabla^P f_{\mathbf{H}}(\mathbf{Z})]_{ik} = \begin{cases} [\nabla f_{\mathbf{H}}(\mathbf{Z})]_{ik}, & \text{if } z_{ik} > 0 \\ \min(0, [\nabla f_{\mathbf{H}}(\mathbf{Z})]_{ik}), & z_{ik} = 0. \end{cases} \tag{23}$$

and $0 < \epsilon_{\mathbf{Z}} < 1$ is the predefined stopping tolerance. A very low $\epsilon_{\mathbf{Z}}$ (i.e., $\epsilon_{\mathbf{Z}} \approx 0$) leads to a termination after a large number of iterations. On the other hand, a tolerance close to 1 will result in a premature iteration termination.

B. Solving Subproblem (16)

A similar procedure should be followed in order to find a stationary point for subproblem (16) while keeping fixed matrix $\mathbf{Z}$ and optimizing with respect to $\mathbf{H}$. A value for a_t is iteratively sought and the weight matrix is updatingd according to

$$\mathbf{H}^{(t+1)} = P\left[\mathbf{H}^{(t)} - a_t \nabla f_{\mathbf{Z}}(\mathbf{H}^{(t)})\right] \tag{24}$$

until the value of function $f_{\mathbf{Z}}(\mathbf{H})$ is sufficiently decreased, i.e.,

$$f_{\mathbf{Z}}(\mathbf{H}^{(t+1)}) - f_{\mathbf{Z}}(\mathbf{H}^{(t)}) \leq \sigma \left\langle \nabla f_{\mathbf{Z}}(\mathbf{H}^{(t)}), \mathbf{H}^{(t+1)} - \mathbf{H}^{(t)} \right\rangle. \tag{25}$$

As we shall see also in Section IV, since subproblem (16) is always quadratic in terms of $\mathbf{H}$, the left hand side of inequality (25) can be expanded for any vector $\mathbf{b}$ as

$$f_{\mathbf{Z}}(\mathbf{h}_i^{(t)} + \mathbf{b}) = f_{\mathbf{Z}}(\mathbf{h}_i^{(t)}) + \mathbf{b}^T \nabla f_{\mathbf{Z}}(\mathbf{h}_i^{(t)}) + \frac{1}{2} \mathbf{b}^T \nabla^2 f_{\mathbf{Z}}(\mathbf{h}_i^{(t)}) \mathbf{b} \tag{26}$$

where $\nabla^2 f_{\mathbf{Z}}(\mathbf{h}_i^{(t)})$ is given by

$$\nabla^2 f_{\mathbf{Z}}(\mathbf{h}_i) = \mathbf{Z}^{\Phi^T} \mathbf{Z}^{\Phi} = \mathbf{K}_{z,z}. \tag{27}$$

Using expansion (26), criterion (25) can be substituted by the following:

$$(1 - \sigma) \left\langle \nabla f_{\mathbf{Z}}(\mathbf{H}^{(t)}), \mathbf{H}^{(t+1)} - \mathbf{H}^{(t)} \right\rangle + \frac{1}{2} \left\langle \mathbf{H}^{(t+1)} - \mathbf{H}^{(t)}, \mathbf{K}_{z,z}(\mathbf{H}^{(t+1)} - \mathbf{H}^{(t)}) \right\rangle \leq 0 \tag{28}$$

which is less computationally expensive than (25).

$$\frac{\partial f_{\mathbf{Z}}}{\partial z_{ab}} = 2 \sum_{i=1}^{M} \left[-h_{bi} \frac{\partial k(\mathbf{z_b}, \mathbf{x}_i)}{\partial z_{ab}} + \frac{1}{2} \left[\sum_{l=1}^{P} h_{bi} h_{li} \frac{\partial k(\mathbf{z_l}, \mathbf{z_b})}{\partial z_{ab}} + \sum_{l=1, l \neq b}^{P} h_{bi} h_{li} \frac{\partial k(\mathbf{z}_l, \mathbf{z}_b)}{\partial z_{ab}} \right] \right] \tag{18}$$

This procedure is repeated until the limit point $\mathbf{H}^{(t)}$ is stationary. The stationarity is checked using a similar criterion to (22), i.e.,

$$\|\nabla^P f_{\mathbf{Z}}(\mathbf{H}^{(t)})\|_F \leq \epsilon_{\mathbf{H}} \|\nabla f_{\mathbf{Z}}(\mathbf{H}^{(1)})\|_F \tag{29}$$

where $\epsilon_{\mathbf{H}}$ is the predefined stopping tolerance for this subproblem.

C. Convergence Criterion

The procedure followed for the minimization of the two subproblems, in Sections Sections III-A and III-B, is iteratively followed until the global convergence criterion is met

$$\|\nabla f(\mathbf{H}^{(t)})\|_F + \|\nabla f(\mathbf{Z}^{(t)})\|_F \leq \epsilon \left(\|\nabla f(\mathbf{H}^{(1)})\|_F + \|\nabla f(\mathbf{Z}^{(1)})\|_F \right) \tag{30}$$

which checks the stationarity of the solution pair $(\mathbf{H}^{(t)}, \mathbf{Z}^{(t)})$. The computational complexity of the PGKNMF approach is discussed in Appendix IV.

D. Feature Extraction

Matrix $\mathbf{Z}$ may be subsequently used for extracting features as follows. Explicit computation of $\phi(\mathbf{z})$ (which may have infinite dimensions for example for Gaussian RBF kernels) is not needed due to the fact that all calculations are performed using the so-called kernel trick.

Let $\mathbf{y}$ be a vector such that $\mathbf{y} \in \Re_+^F$. Then, the projected vector $\tilde{\mathbf{y}} \in \Re^P$ is calculated as follows:

$$\tilde{\mathbf{y}} = {\mathbf{Z}^{\Phi}}^{\dagger}(\phi(\mathbf{y}) - \mathbf{m}^{\Phi}) \tag{31}$$

where $\mathbf{m}^{\Phi} = 1/M \sum_i \phi(\mathbf{x}_i)$, the ${\mathbf{Z}^{\Phi}}^{\dagger}$ is the pseudo-inverse of $\mathbf{Z}^{\Phi}$ and is calculated as

$${\mathbf{Z}^{\Phi}}^{\dagger} = ({\mathbf{Z}^{\Phi}}^T \mathbf{Z}^{\Phi})^{-1} {\mathbf{Z}^{\phi}}^T = \mathbf{K}_{z,z}^{-1} {\mathbf{Z}^{\phi}}^T. \tag{32}$$

The inverse $\mathbf{K}_{z,z}^{-1}$ can be, in most cases, calculated, since usually $P \ll M$; thus, $\mathbf{K}_{z,z}$ is of full rank.

Now, using (32), feature extraction (31) may be reformulated as

$$\tilde{\mathbf{y}} = \mathbf{K}_{z,z}^{-1} {\mathbf{Z}^{\Phi}}^T (\phi(\mathbf{y}) - \mathbf{m}^{\Phi}) = \mathbf{K}_{z,z}^{-1} \mathbf{g}(\mathbf{y}) \tag{33}$$

where $\mathbf{g}(\mathbf{y}) \equiv [k(\mathbf{z}_1, \mathbf{y}) - 1/M \sum_i k(\mathbf{x}_i, \mathbf{z}_1), \ldots, k(\mathbf{z}_P, \mathbf{y}) - 1/M \sum_i k(\mathbf{x}_i, \mathbf{z}_P)]^T$. In [19], $\mathbf{Z}$ was directly used for feature extraction as $\tilde{\mathbf{y}} = \mathbf{Z}^{\dagger}\mathbf{y}$. This procedure leads to only linear features.

IV. Nonlinear non-negative Component Analysis Approach for Gaussian RBF Kernels

In this section, we consider alternative approaches for nonlinear non-negative component analysis based on Gaussian RBF kernels.

A. A Different Approach With Gaussian RBF Kernels

In this section, we consider the problem of nonlinear non-negative component analysis using Gaussian RBF kernels and we propose an alternative fixed point algorithm for the minimization of the cost function. In kernel methods, the Gaussian RBF kernel is given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma} \tag{34}$$

where σ is the spread of the Gaussian function.

Let us consider the problem as follows. First, we consider the kernel expansion

$$\mathbf{g}_i = \sum_j h_{ji} \phi(\mathbf{z}_j). \tag{35}$$

Then, we seek to approximate it by $\acute{\mathbf{g}}_i = \beta_i \phi(\mathbf{x}_i)$. We allow $\beta_i \neq 1$, which is reasonable, since the length of $\mathbf{g}_i$ is not crucial for building decision functions [24]. The problem is to minimize the following:

$$f(\mathbf{Z}, \mathbf{H}) = \sum_i \|\mathbf{g}_i - \acute{\mathbf{g}}_i\|^2 = \sum_i \left\| \sum_j h_{ji} \phi(\mathbf{z}_j) - \beta_i \phi(\mathbf{x}_i) \right\|^2. \tag{36}$$

As in the algorithm in the previous section, we consider solving the two partial minimization problems

$$\min_{h_{kj} \geq 0} f(\mathbf{h}_j) \tag{37}$$

and

$$\min_{z_{ik} \geq 0} f(\mathbf{z}_k) \tag{38}$$

in an iterative manner, where $f(\mathbf{h}_j)$ and $f(\mathbf{z}_k)$ are equal to $f(\mathbf{Z}, \mathbf{H})$ by keeping all but $\mathbf{h}_j$ and $\mathbf{z}_k$ constant, respectively. For the minimization problem (37) we consider the actual optimization problem (36), while for the minimization problem (38), we consider a transformed version of the problem, as explained next.

In the tth iteration for the solution of subproblem (37), we follow the procedure of the auxiliary function. That is, we identify a solution via the definition and the optimization of a proper auxiliary function. In our case, we choose an auxiliary as the one in (8), but now we choose as $\mathbf{L}$ the diagonal matrix with $\mathbf{L}_{kk} = [\mathbf{K}_{z,z}\mathbf{h}_i]_k / \beta_i^2 h_{ki}$ (the proof is straightforward by using the results in [19]). The updating rules are the following:

$$h_{ji}^{(t)} = h_{ji}^{(t-1)} \frac{\beta_i [\mathbf{K}_{x,z}^{(t-1)}]_{ji}}{[\mathbf{K}_{z,z}^{(t-1)} \mathbf{h}_i^{(t-1)}]_j} \tag{39}$$

or in matrix notation

$$\mathbf{H}^{(t)} \leftarrow \mathbf{H}^{(t-1)} \odot \mathbf{C} \odot \frac{\mathbf{K}_{x,z}^{(t-1)}}{\mathbf{K}_{z,z}^{(t-1)} \mathbf{H}^{(t-1)}} \tag{40}$$

with $[\mathbf{K}_{x,z}]_{ij} = e^{-\|\mathbf{x}_i - \mathbf{z}_j\|^2/\sigma}$, $[\mathbf{K}_{z,z}]_{ij} = e^{-\|\mathbf{z}_i - \mathbf{z}_j\|^2/\sigma}$ and $\mathbf{C}_{ji} = \beta_i$.

We use a similar reasoning as the one followed in [24] for finding the preimages of kernel algorithms, in order to specify

the updating rules (38). That is, as in [24], we minimize the orthogonal projection of $\phi(\mathbf{x}_i)$ onto $\sum_j h_{ji}\phi(\mathbf{z}_j)$ [(41), shown at the bottom of the page], which is equivalent to

$$\max_{\mathbf{z}_j} d_{RBF}(\mathbf{Z}) = \sum_{i=1}^{N} \frac{\left\langle \phi(\mathbf{x}_i), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle^2}{\left\langle \sum_j h_{ji}\phi(\mathbf{z}_j), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle}. \quad (42)$$

In order to further simplify the optimization procedure, we only optimize the numerator

$$\max_{\mathbf{z}_j,\beta_i} \sum_{i=1}^{N} \left\langle \phi(\mathbf{x}_i), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle^2. \quad (43)$$

Alternative, since all terms are non-negative, we may minimize

$$\max_{\mathbf{z}_j,\beta_i} \sum_{i=1}^{N} \left\langle \phi(\mathbf{x}_i), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle. \quad (44)$$

By using fixed point iteration algorithms like in [24] (i.e., setting $\partial d(\mathbf{Z})/\partial z_{ik} = 0$), the updating rules for z_{ik} for (43) may be derived as

$$z_{ik}^{(t)} = \frac{\sum_j x_{ij} h_{kj} k(\mathbf{z}_k^{(t-1)}, \mathbf{x}_j) \left\langle \phi(\mathbf{x}_j), \sum_m h_{mj}\phi(\mathbf{z}_m) \right\rangle}{\sum_j h_{kj} k(\mathbf{z}_k^{(t-1)}, \mathbf{x}_j) \left\langle \phi(\mathbf{x}_j), \sum_m h_{mj}\phi(\mathbf{z}_m) \right\rangle}. \quad (45)$$

Alternatively for (44)

$$z_{ik}^{(t)} = \frac{\sum_j x_{ij} h_{kj} k(\mathbf{z}_k^{(t-1)}, \mathbf{x}_j)}{\sum_j h_{kj} k(\mathbf{z}_k^{(t-1)}, \mathbf{x}_j)}. \quad (46)$$

In compact matrix notation, the updating rules for (45) may be written as

$$\mathbf{Z}^{(t)} \leftarrow \frac{\mathbf{X}((\mathbf{H}^{(t)} \odot \mathbf{K}_{z,x}^{(t-1)})\mathrm{diag}(\mathbf{K}_{x,z}^{(t-1)}\mathbf{H}^{(t)}))}{\mathbf{B}^{(t-1)}} \quad (47)$$

where $\mathbf{B}^{(t-1)}$ has in all its rows the diagonal elements of matrix $\mathbf{H}^{(t)}\mathrm{diag}(\mathbf{K}_{x,z}^{(t-1)}\mathbf{H}^{(t)})\mathbf{K}_{x,z}^{(t-1)}$. After the iterations in (45), the optimal $\beta_i^{(t)}$ is given by

$$\beta_i^{(t)} = \left\langle \phi(\mathbf{x}_i), \sum_j h_{ji}^{(t)} \phi(\mathbf{z}_j^{(t)}) \right\rangle = \sum_j h_{ji}^{(t)} k(\mathbf{x}_i, \mathbf{z}_j^{(t)}). \quad (48)$$

The algorithm proposed in this Section is referred to as KNMF-RBF in the rest of the paper.

B. Convex Nonlinear Non-Negative Component Analysis

In this section, instead of finding both pre-images $\mathbf{z}_j$ and $\mathbf{H}$ simultaneously, we follow a different strategy. That is, we follow similar lines as the convex non-negative matrix factorization method proposed in [40]. In K-means and kernel K-means algorithms the centroids, for reasons of interpretability, are often considered to be in the space defined by the columns of $\mathbf{X}^\Phi$. Hence, the centroids $\phi(\mathbf{z}_j)$ can be written as

$$\begin{aligned} \phi(\mathbf{z}_j) &= m_{1j}\phi(\mathbf{x}_1) + \cdots + m_{Nj}\phi(\mathbf{x}_N) \\ &= \sum_{i=1}^{N} m_{ij}\phi(\mathbf{x}_i) \Leftrightarrow \\ \mathbf{Z}^\Phi &= \mathbf{X}^\Phi \mathbf{M}. \end{aligned} \quad (49)$$

In case that we additionally incorporate a non-negativity constraint for the weights m_{ij}, the centroids $\phi(\mathbf{z}_j)$ can be interpreted as convex weighted sums of certain data points $\phi(\mathbf{x}_i)$. Moreover, expressing the centroids $\phi(\mathbf{z}_j)$ as in (49) is a convenient way to find the preimages $\mathbf{z}_j$ for Gaussian RBF and other kernels, by using simple fixed point algorithms [24].

Using (49), approximation (2) is reformulated as

$$\begin{aligned} \mathbf{X}^\Phi &\approx \mathbf{X}^\Phi \mathbf{M}\mathbf{H} \Rightarrow \\ \mathbf{K}_{x,x} &\approx \mathbf{K}_{x,x}\mathbf{M}\mathbf{H} \end{aligned} \quad (50)$$

with $m_{ik} \geq 0$ and $h_{ki} \geq 0$. The corresponding optimization problem is given by

$$\min_{m_{ik}\geq 0, h_{ki}\geq 0} D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{M}\mathbf{H}) = \|\mathbf{X}^\Phi - \mathbf{X}^\Phi\mathbf{M}\mathbf{H}\|^2. \quad (51)$$

$$\min_{\mathbf{z}_j} \sum_{i=1}^{N} \left\| \frac{\left\langle \phi(\mathbf{x}_i), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle}{\left\langle \sum_j h_{ji}\phi(\mathbf{z}_j), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle} \left(\sum_j h_{ji}\phi(\mathbf{z}_j) \right) - \phi(\mathbf{x}_i) \right\|^2 =$$

$$\min_{\mathbf{z}_j} \sum_{i=1}^{N} \left(\|\phi(\mathbf{x}_j)\|^2 - \frac{\left\langle \phi(\mathbf{x}_i), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle^2}{\left\langle \sum_j h_{ji}\phi(\mathbf{z}_j), \sum_j h_{ji}\phi(\mathbf{z}_j) \right\rangle} \right) \quad (41)$$

The above cost function can be written as

$$D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})=\sum_{i=1}^{M}\|\phi(\mathbf{x}_i)-\sum_{j=1}^{P}h_{ji}\phi(\mathbf{z}_j)\|^2$$
$$=\sum_{i=1}^{M}(k(\mathbf{x}_i,\mathbf{x}_i)-2\sum_{j=1}^{P}h_{ji}\sum_{k=1}^{N}m_{kj}k(\mathbf{x}_k,\mathbf{x}_i)$$
$$+\sum_{j=1}^{P}\sum_{l=1}^{P}h_{ji}h_{li}\sum_{k=1}^{N}\sum_{n=1}^{N}m_{kl}m_{nj}k(\mathbf{x}_k,\mathbf{x}_n)). \quad (52)$$

It can be proven (see Appendix V) that the following updating rules guarantee the nonincreasing behavior of (52):

$$\mathbf{M}^{(t)}=\mathbf{M}^{(t-1)}\odot\left(\frac{\mathbf{K}_{x,x}\mathbf{H}^{(t-1)^T}}{\mathbf{K}_{x,x}\mathbf{M}^{(t-1)}\mathbf{H}^{(t-1)}\mathbf{H}^{(t-1)^T}}\right)^p \quad (53)$$

and

$$\mathbf{H}^{(t)}=\mathbf{H}^{(t-1)}\odot\left(\frac{\mathbf{M}^{(t)^T}\mathbf{K}_{x,x}}{\mathbf{M}^{(t)^T}\mathbf{K}_{x,x}\mathbf{M}^{(t)}\mathbf{H}^{(t-1)}}\right)^p \quad (54)$$

where $p=1$ or $p=1/2$. Updating rules for the $p=1/2$ were independently proposed in [40].

There are two subtle issues that deserve further consideration:

- the denominators in (53) and (54) might be zero;
- if $h_{ki}^{(t)}=0$ and $\partial D_\Phi/\partial h_{ki}<0$ then according to the updating rules (54), $h_{ki}^{(t+1)}$ does not change. Hence, the proof of convergence for fixed point methods applied to unconstrained function minimization cannot be repeated.

The above observations hold for w_{ik}, as well. Another alternative for defining updating rules that guarantee the convergence to stationary limit points and deal with the above issues, is to follow similar lines to those in [29]. This leads to the following updating rules:

$$\mathbf{M}^{(t,k)}=\mathbf{M}^{(t-1)}$$
$$-\bar{\mathbf{M}}^{(t-1)}\odot\left(\frac{\nabla_{\mathbf{M}^{(t-1)}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH}^{(t-1)})}{\mathbf{K}_{x,x}\mathbf{M}^{(t-1)(t)}\mathbf{H}^{(t-1)}\mathbf{H}^{(t-1)^T}+\mathbf{\Delta}_1}\right) \quad (55)$$

and

$$\mathbf{H}^{(t,k)}=\mathbf{H}^{(t-1)}$$
$$-\bar{\mathbf{H}}^{(t-1)}\odot\left(\frac{\nabla_{\mathbf{H}^{(t-1)}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{M}^{(t,k)}\mathbf{H})}{\mathbf{M}^{(t,k)^T}\mathbf{K}_{x,x}\mathbf{M}^{(t,k)}\mathbf{H}^{(t-1)}+\mathbf{\Delta}_2}\right). \quad (56)$$

Matrices $\mathbf{M}^{(t,k)}$ and $\mathbf{H}^{(t,k)}$ must be normalized to produce $\mathbf{M}^{(t+1)}$ and $\mathbf{H}^{(t+1)}$, respectively, so that the elements of each column of $\mathbf{M}^{(t+1)}$ sum up to one. The gradients $\nabla_{\mathbf{M}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})$ and $\nabla_{\mathbf{H}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})$ are given by

$$\nabla_{\mathbf{M}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})=\mathbf{K}_{x,x}\mathbf{MHH}^T-\mathbf{K}_{x,x}\mathbf{H}$$
$$\nabla_{\mathbf{H}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})=\mathbf{M}^T\mathbf{K}_{x,x}\mathbf{MH}-\mathbf{M}^T\mathbf{K}_{x,x} \quad (57)$$

and

$$\bar{m}_{ik}\triangleq\begin{cases}m_{ik}, & \text{if } [\nabla_{\mathbf{M}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})]_{ik}\geq 0\\ \max(m_{ik},\sigma), & \text{if } [\nabla_{\mathbf{H}}D_\phi(\mathbf{X}^\Phi,\mathbf{X}^\Phi\mathbf{MH})]_{ik}<0.\end{cases} \quad (58)$$

Similar is the definition of $\bar{h}_{kj}$. Parameter σ takes a small value (e.g., 10^{-6}) and $\mathbf{\Delta}_1,\mathbf{\Delta}_2$ are matrices with elements equal to a small constant δ (e.g., 10^{-6}).

TABLE I
SIMULATION RESULTS AND TIMES FOR RANDOM MATRICES $\mathbf{X}$, $\mathbf{Z}$ AND $\mathbf{H}$ AND THE PGKNMF ALGORITHM

ϵ	Starting Cost	Ending Cost	iterations	time(secs)
0.001	10.2×10^4	2.7×10^3	25	1
0.0001	10.2×10^4	1.5×10^3	69	2
0.00001	10.2×10^4	9×10^2	85	4

(a) (b)

Fig. 1. (a) PNMF preimages; (b) PGKNMF preimages.

After the convergence of the above sequence, the pre-images $\mathbf{z}_j$ can now be found by solving the following optimization problem:

$$\min_{\mathbf{z}_j}\left\|\phi(\mathbf{z}_j)-\beta_j\sum_{i=1}^{N}m_{i,j}\phi(\mathbf{x}_i)\right\|^2 \quad \text{subject to} \quad z_{kj}\geq 0. \quad (59)$$

The above optimization problem can be solved using many algorithms [24]. One of them is the projected gradient algorithm in Section III-A. For the special case of Gaussian-RBF functions, we can use the following updating rules for obtaining $\mathbf{z}_j$, with $j=1,\ldots,P$

$$\mathbf{z}_j^{(t+1)}=\frac{\sum_{i=1}^{N}m_{ij}k(\mathbf{z}_j^{(t)},\mathbf{x}_i)\mathbf{x}_i}{\sum_{i=1}^{N}m_{ij}k(\mathbf{z}_j^{(t)},\mathbf{x}_i)}. \quad (60)$$

V. EXPERIMENTAL RESULTS

A. Experiments Using Simulated Data

We experimented with simulated data in order to confirm that the proposed method can indeed reduce the defined cost function. For that, we produced matrices $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{H}$ from $|N(0,1)|$ for an initial $\mathbf{X}\in\Re_+^{25\times25}$, $\mathbf{Z}\in\Re_+^{25\times5}$, $\mathbf{H}\in\Re_+^{5\times25}$. We present experiments with $\epsilon=0.001, 0.0001$ and 0.00001 (ϵ is the predefined constant that controls the convergence of the projected gradient algorithm (30)) and for polynomial degree $d=2$. The time needed for execution, the starting cost and the ending costs can be found in Table I. As it can be seen, the proposed PKGNMF minimizes satisfactorily the cost function in a reasonable time.

Moreover, we compared PGKNMF, in terms of time and capability to reach a minimum, with PNMF. For that, we include an experiment using the Cohn-Kanade (CK) database (for the description of the CK database see Section V-C). We used from CK 25 randomly chosen samples, of resolution 40×30, and we formed matrices $\mathbf{X}\in\Re_+^{1200\times25}$, $\mathbf{Z}\in\Re_+^{1200\times9}$ and $\mathbf{H}\times\Re_+^{9\times25}$. The simulation results, using polynomial kernel of degree equal

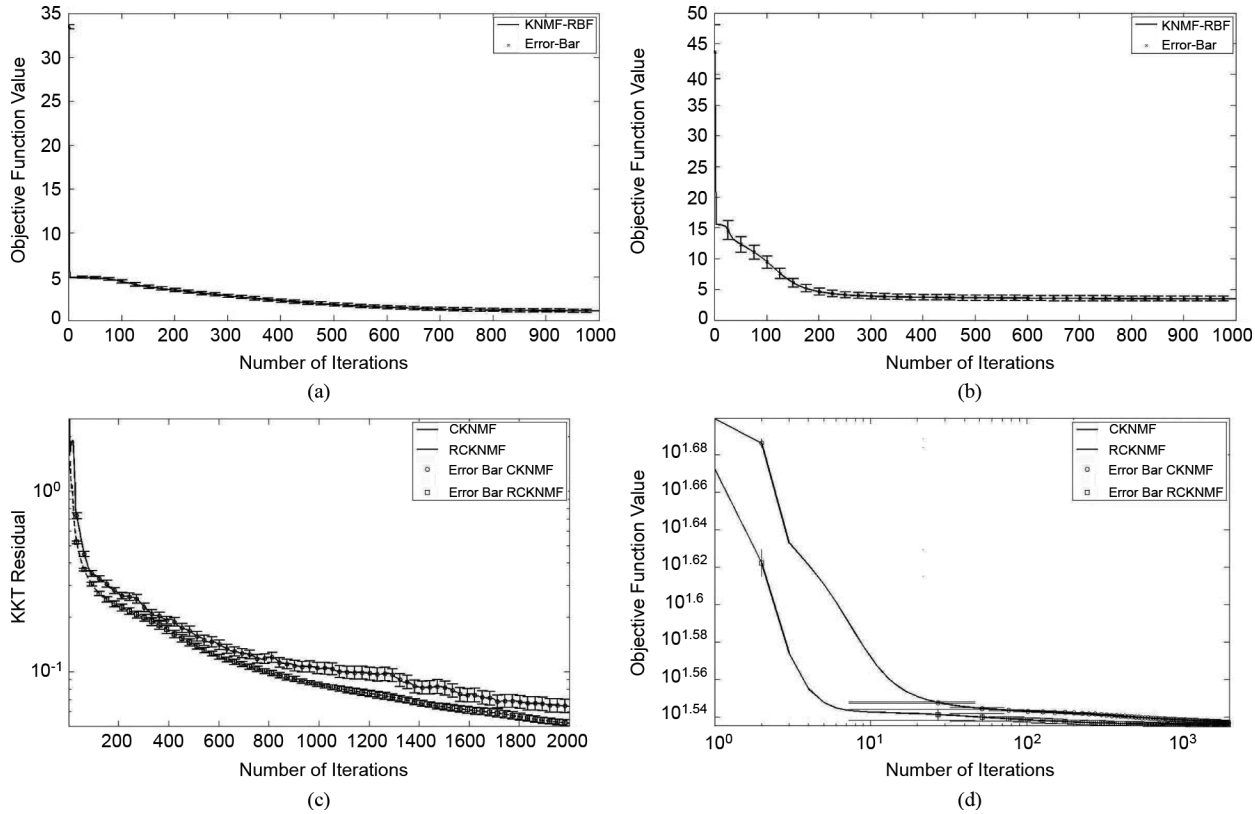


Fig. 2. (a) Mean approximation error and variance (error bars) for 100 random restarts for 100 images from the CK database; (b) mean approximation error and variance (error bars) for 100 random restarts for 400 images from the XM2VTS; (c) mean KKT residual norm plotted versus 2000 iterations; (d) mean objective function value versus 2000 iterations.

TABLE II
SIMULATION RESULTS AND TIMES FOR IMAGE DECOMPOSITION, USING 25 IMAGES FROM THE CK DATABASE

Algorithm	ϵ	Starting Cost	Ending Cost	iterations	time(min)
PGKNMF	0.0001	2.0819×10^6	8×10^3	85	3
PNMF	-	2.0819×10^6	1.1×10^5	100,000	10

to 2, can be found in Table II. From this Table it can be seen that the reached minimum by PGKNMF is at least one order of magnitude smaller than the one reached with PNMF. Moreover, PNMF had to be executed for 100,000 iterations in order to reach that minimum, taking up 10 mins. The calculated preimages (i.e., columns of matrix $\mathbf{Z}$) of PGKNMF and PNMF can be found in Fig. 1. As it can be seen, the preimages of the proposed method resemble more human faces than the preimages of PNMF, which are more like distorted versions of faces.

We also provide experimental results that show the ability of the method described in Section IV to decrease the objective function value (cost) in every iteration using the updating rules (40), (47), and (48). In Fig. 2(a), the mean approximation error (3) is plotted versus the number of iterations after 100 random restarts for 100 images from the CK database. In Fig. 2(b), the mean approximation error is plotted for 100 random restarts using now 400 images from the XM2VTS database (the XM2VTS database is described in Section V-F) (for both experiments we used $\sigma = 10^3$). As it can be seen, the objective function value is monotonically decreasing.

Furthermore, we conducted experiments in order to compare the ability of the updating rules (53), (54) with $p = 1/2$ (abbreviated as CKNMF) and (55), (56) (abbreviated as RCKNMF) to reach a stationary point. Let $D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH})$ in (52) be the cost function that measures the quality of the approximation. From the Karush-Kuhn-Tucker (KKT) conditions [41], m_{ik} and h_{ki} correspond to a stationary point of the minimization of (52) iff [see (61), shown at the bottom of the page].

The convergence to a stationary point of the minimization of (52) may be tested by checking the KKT conditions for optimization problem (51). The KKT conditions in case of matrix $\mathbf{M}$ can be rewritten as

$$\min(\mathbf{M}, \nabla D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH})) = 0 \tag{62}$$

which state that both $\mathbf{M}$ and $\nabla D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH}))$ should be component-wise non-negative and at least one (the smallest of

$$\begin{array}{cc} m_{ik} \geq 0, & h_{kj} \geq 0 \\ \frac{\partial}{\partial m_{ik}} D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH}) \geq 0, & \frac{\partial}{\partial h_{kj}} D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH}) \geq 0 \\ m_{ik}\frac{\partial}{\partial m_{ik}} D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH}) = 0, & h_{kj}\frac{\partial}{\partial h_{kj}} D_\phi(\mathbf{X}^\Phi, \mathbf{X}^\Phi\mathbf{MH}) = 0 \end{array} \tag{61}$$

TABLE III
BEST ACCURACY (%)/NUMBER OF BASIS IMAGES, FOR THE 13 SUBJECTS OF COHN-KANADE

Classifier	NMF	LNMF	PNMF	ICA	PCA	KICA	KPCA	PGKNMF	KNMF-RBF	RCKNMF
CSM	77.4/36	81.4/25	81.8/16	71.4/25	72.9/16	74.3/36	74.9/29	82.2/26	**82.5**/48	81.7/46
SVM	78.6/100	81.4/81	83.9/100	80/64	81.4/100	82.9/25	82.9/32	84.2/120	**84.2**/47	82.9/111

the two) is allowed to be zero (and similarly for matrix $\mathbf{H}$). Obviously, the KKT residual norm, defined as the L_1 norm of the left hand side of (62)[27], must tend to zero. Consequently, by monitoring its size (i.e., the number of nonzero elements), we may test whether the proposed algorithm converges to a stationary point.

We compute and plot the ensemble averaged objective function and the ensemble averaged KKT residual norm over 100 runs of the updating rules, that is randomly initialized, for 2000 iterations [27]. Let us call the just mentioned figures of merit "mean objective function" and "mean KKT residual norm", for brevity. The mean KKT residual norms of the proposed updating rules (55), (56) and the updating rules in (53), (54) are plotted in Fig. 2 c and 2-D for 50 randomly selected samples of XM2VTS database (K was selected as 15 and $\sigma = 10^2$), respectively. As it can be seen in both figures, the mean KKT residual norms and the objective function value of the proposed updating rules are smaller than that of the updatings rules (55), (56). This demonstrates that the solution of the proposed framework (RCKNMF) is closer to the stationary point of the minimization of the objective function (52), than the solution obtained by the updating rules of CKNMF.

B. Testing Methods and Classification Procedure

In this section we demonstrate the performance of the proposed method to face and facial expression recognition applications. For comparison, NMF, [8], LNMF [9], ICA [6], [42], and PCA [3] have also been implemented. Moreover, we also consider the nonlinear alternative of NMF, the PNMF in [19] and the nonlinear variants of PCA and ICA, namely KPCA [17] and KICA [18], respectively.

For NMF and LNMF, each facial vector $\mathbf{x}$, after subtracting the mean vector of the training set, is projected to a lower dimensionality space using the pseudoinverse basis image matrix, resulting in a feature vector $\mathbf{f} = \mathbf{Z}^{\dagger}\mathbf{x}$. For the ICA approach, which has been used for comparison purposes, we used the architecture described in [42] and [6] that yields the features to be used for classification. The ICA decomposition coefficients of each image form essentially a row of matrix $\mathbf{F}_{tr} = \mathbf{X}_{tr}\mathbf{P}_p\mathbf{A}^{-1}$. Here, suffix tr corresponds to the training set while te corresponds to the testing set, $\mathbf{P}_p$ is the projection matrix resulting from the PCA procedure applied prior to ICA and $\mathbf{A}$ is the unmixing matrix found by the ICA algorithm. The number of independent components is controlled by the first p eigenvectors. PCA alone was also applied at the experimental data. The same strategies were adopted for KICA and KPCA using polynomial and Gaussian RBF kernels. For PNMF, we used only polynomial kernels and for the proposed PGKNMF we considered both polynomial and Gaussian RBF kernels. For these methods, features were extracted as described in Section III-D.

Let us denote by $\tilde{l}(\mathbf{f}_{te})$ the label of the input pattern $\mathbf{f}_{te}$ and by $l(\mathbf{f}_{te})$ the correct label of the same pattern. Then the accuracy of the classifier is defined as the percentage of the correctly classified test images, i.e., the percentage of images for which $\tilde{l}(\mathbf{f}_{te}) = l(\mathbf{f}_{te})$. Three classifiers were employed for classifying the features extracted by the tested algorithms. The first classifier is a nearest neighbor classifier based on the cosine similarity measure (CSM). This approach uses as similarity measure the cosine of the angle between a test feature vector and a prototype one, i.e one derived from the training phase. More specifically, $\tilde{l} = l(\mathbf{f}_{k,tr})$ where $k = \arg\max_{i=1,\ldots,n_{tr}}\{d_i\}$ and $d_i = \mathbf{f}_{te}^T\mathbf{f}_{i,tr}/\|\mathbf{f}_{te}\|\|\mathbf{f}_{tr}\|$. The second classifier is a two layer neural network based on Gaussian-RBFs (RBFNN). Finally, the third classifier is based on SVMs [20] with different kernels (linear, polynomial, and Gaussian-RBF). The sequential minimal optimization technique [43] was used to train the SVMs. Since classical SVM theory was intended to solve a two class classification problem, we chose the decision directed acyclic graph (DDAG) learning architecture proposed in [44] which has been adopted for multiclass classification. In the following, we report the results for SVMs and Gaussian-RBF classifiers only in the case that they achieved a better classification rate than the simple CSM classifier.

C. Facial Expression Recognition Experiments With the Cohn-Kanade (CK) Database

The first database used for the facial expression recognition experiments was created using the CK database [45]. This database is annotated with Facial Action Units (FAUs). These combinations of FAUs were translated into facial expressions according to [46], in order to define the corresponding ground truth for the facial expressions.

We used two different experimental setups for facial expression recognition in the CK database. For the first setup, we used the same samples as in [19]. That is, in [19], the authors have used only 13 posers, the ones that display all the facial expressions, i.e., anger, disgust, fear, happiness, sadness, and surprise. These thirteen persons were chosen to create the image database that was used in the facial expression recognition experiments. Each subject from the CK database forms an expression over time starting from the neutral pose and ending with a very intense expression, thus having several video frames with different expression intensities. However, the number of these intermediate video frames is not the same for the various posers. In [19], the authors have selected three poses with low (close to neutral), medium and high (close to maximum) facial expressions intensity and used them to form the database utilized in their experiments. For the first experimental setup we used the same data and the same experimental protocol [19].

The images were aligned manually using the eyes' coordinates and were downsampled to 40×30 pixels. To form the training set, $n_{tr} = 164$ images were chosen for training the system, i.e., learn matrices $\mathbf{Z}$ and $\mathbf{H}$, and the remaining $n_{te} = 70$ were used for testing. For this setup, a matrix $\mathbf{X} \in \Re^{1200\times 164}$ was formed in the training phase. The whole procedure was repeated four times and the mean expression recognition rate was calculated. The best results are summarized in Table III. As it can be seen, the PGKNMF method repeatedly produces the best results.

TABLE IV
BEST ACCURACY (%)/VARIANCE (%), FOR THE DIFFERENCE IMAGES OF COHN-KANADE AND THE STATISTICAL ROBUST PROTOCOL

Classifier	NMF	LNMF	PNMF	ICA	PCA	KICA	KPCA	PGKNMF	KNMF-RBF	RCKNMF
CSM	67.4/2.2	75.6/1.9	77.8/2.3	61.2/2.6	49.9/2.6	61.6/2.7	62.7/2.9	**81.1**/2.3	77.1/2.5	79.8/2.4
RBFNN	67.9/2.6	76.4/2.5	78.2/2.1	62.1/2.2	53.5/2.3	63.4/2.6	67.5/2.6	**82.4**/2.4	78.2/2.3	80.5/2.5
SVM	69.3/2.5	78.5/2.6	80.3/2.6	65.7/2.5	59.2/2.7	65.4/2.6	69.4/2.3	**83.5**/2.0	80.5/2.1	81.4/2.2

TABLE V
MEAN RECOGNITION ACCURACY (%)/VARIANCE (%), FOR SUBJECTS OF THE JAFFE DATABASE USING THE ROBUST PROTOCOL

Classifier	NMF	LNMF	PNMF	ICA	PCA	KICA	KPCA	PGKNMF	KNMF-RBMF	RCKNMF
CSM	58.7/2.0	58.9/2.3	61.5/2.4	57.4/2.2	58.3/2.3	59.4/2.2	57.2/2.2	**63.5**/2.6	60.5/2.6	61.5/2.4
SVM	65.3/2.3	64.5/2.4	66.7/2.1	65.7/2.2	65.7/2.2	65.7/1.9	64.6/1.9	**69.3**/2.4	66.3/2.4	67.9/2.3

TABLE VI
MEAN RECOGNITION ACCURACY (%)/VARIANCE (%), FOR THE DIFFERENCE IMAGES OF THE JAFFE DATABASE AND THE ROBUST EXPERIMENTAL PROTOCOL

Classifier	NMF	LNMF	PNMF	ICA	PCA	KICA	KPCA	PGKNMF	KNMF-RBF	RCKNMF
CSM	77.5/2.1	85.5/1.8	86.7/1.8	85.5/1.9	83.4/1.7	84.3/1.7	87.7/1.8	**88.6**/1.7	86.5/1.9	88.6/1.9

Apart from the above experimental setup, where we used the whole CK database, we also used the difference images. 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 facial expression image, were calculated. Each difference image was initially normalized, resulting in an image built only from positive values and afterwards scanned row-wise to form a vector $\mathbf{x} \in \Re_{+}^{1200}$. The difference images were used instead of the original facial expression images, due to the fact that in the difference images, the facial parts in motion are emphasized [47].

For the difference images we used the following testing protocol [48], [49]. In the experimental procedure, ten 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 was used as the test set, while the remaining sets formed the training set (i.e., 80% for training and 20% for testing). In order to learn the parameters of all methods (i.e, the kernel in kernel methods, the dimensionality for all the tested methods and the parameters of the SVM and Gaussian-RBF classifiers) the training set was further randomly divided into training (80% of the images) and validation (20% of the images) sets. This inner loop was repeated ten times and the parameters, which gave the best mean recognition rate for the validation sets, were adopted for the test set. After the classification procedure was performed, the samples forming the test set were incorporated into the current training set while a new set of samples was extracted to form the new test set. The remaining samples created the new training set and the inner loop was applied in order to learn the parameters. The whole procedure was repeated ten times. The average classification accuracy was the mean value of the percentages of the correctly classified facial expressions in the test sets. The mean recognition results and the variance of this setup are summarized in Table IV. As it can be seen, the proposed methods produced the best facial expression recognition rates.

D. Facial Expression Recognition Experiments With the Jaffe Database

The second database used for experiments contains 213 images of Japanese female facial expressions (JAFFE) [50]. Ten subjects produced three or four examples of each of the six facial expressions plus a neutral pose, thus producing a total of 213 images of facial expressions. Image registration was performed in the same way as for the CK database. The same experimental protocol as in the case of the CK database was applied in the Jaffe database, as well. That is, the dataset was randomly split into training (80%) and testing (20%). In every cycle, in order to learn the parameters, the training set was further split in training and validation subsets. The results for this database are summarized in Table V. As it can be seen, the proposed method achieved the best recognition accuracy on this dataset.

We also experimented using Jaffe difference images. As it can be seen from Table VI, the recognition rate for all the feature extraction methods is dramatically increased by using difference images. The proposed method also produced the best results for this case, too.

E. Face Recognition Experiments With the Yale Database

The Yale face database [51] contains 65 grayscale images of 15 individuals. There are 11 images per subject, one per different facial expression or configuration: center-light, with glasses, happy, left-light, without glasses, normal, right-light, sad, sleepy, surprised and winking. For computational reasons the image size was reduced to 42×31 pixels after manual facial image alignment.

For this database, the first 7 images of each subject were used to form the training set. The remaining four samples were used as test images. The training set was further randomly split into four and three samples in order to implement the inner circle for learning the parameters for each tested method and classifier (as implemented in the CK and JAFFE database experiments). The best results of all the tested methods and classifiers can be found in Table VII. As it can be seen, for the simple cosine classifier, the proposed method had the best performance, while when using SVMs the PNMF showed the best performance.

F. Face Verification Experiments Using the XM2VTS Database

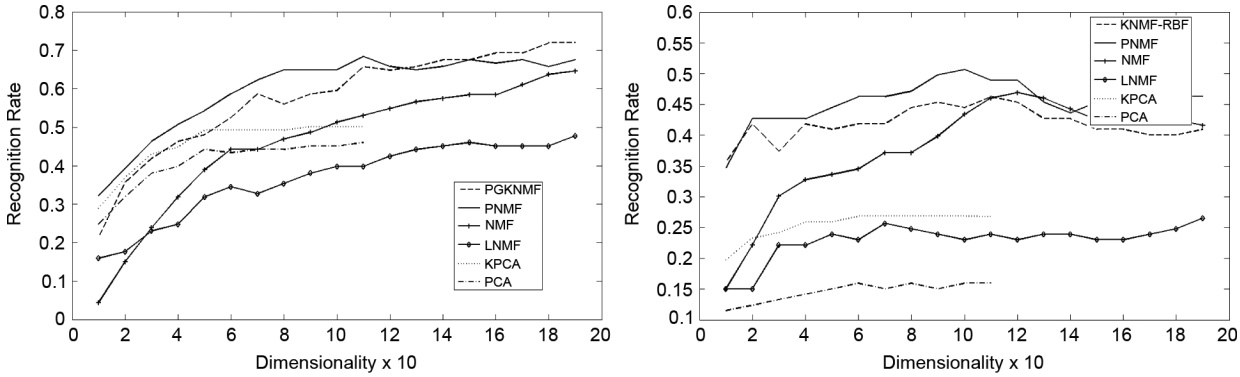
The experiments conducted with the XM2VTS database used the protocol described in [52]. The images were aligned semi-automatically according to the eyes' position of each facial image using the eye coordinates. The facial images were down-scaled to a resolution of 64×64 pixels. Histogram equalization was used for the normalization of the facial image luminance.

TABLE VII
BEST ACCURACY (%)/VARIANCE (%), FOR THE YALE DATABASE

Classifier	NMF	LNMF	PNMF	ICA	PCA	KICA	KPCA	PGKNMF	KNMF-RBF	RCKNMF
CSM	89.2/1.5	88.8/1.6	90.3/1.7	87.6/1.9	87.6/1.6	90.1/1.7	89.2/2.0	**91.5**/1.5	89.7/1.7	89.7/1.7
RBFNN	92.6/2.0	92.6/2.0	93/2.4	91.3/1.7	90.3/1.8	93.4/1.8	93.4/1.8	**93.6**/1.7	92.9/1.8	93.6/2.0
SVM	94.0/1.2	94.0/1.2	**94.8**/1.1	93.8/1.1	93.1/1	94.5/1	94.8/1.0	94.7/1.2	93.9/1.2	94.0/1.2

TABLE VIII
BEST EER (%) FOR THE XM2VTS DATABASE

	NMF	LNMF	PNMF	ICA	PCA	KICA	KPCA	PGKNMF	KNMF-RBF	RCKNMF
EER%	8.5	8.2	5.4	4.1	4.3	3.5	3.4	3.4	**2.9**	3.2



(a) (b)

Fig. 3. (a) Recognition rate of the grayscale albedo image; (b) Recognition rate of the depth images.

The XM2VTS database contains 295 subjects, four recording sessions, and two shots (repetitions) per recording session. It provides two strict experimental setups, namely, Configuration I and Configuration II [52]. Each configuration is divided into three different sets: the training set, the validation (in [52] the set was named 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 both the original and the revised manuscript the evaluation set was used in order to learn the parameters of the various methods. That is, the feature dimensionality, kernel parameter and the thresholds were learned using the evaluation set and then the ones which lead to the best EER for every tested method were applied afterwards to the test set. For both configurations, the training set had 200 clients, 25 evaluation impostors and 70 test impostors. The two configurations differed in the distribution of client training and client evaluation data. For additional details concerning the XM2VTS database an interested reader is referred to [52].

The procedure followed in the experiments was the one also used in [16] and [53]. For comparison reasons the same methodology, using Configuration I of the XM2VTS database, was used. The performance of the algorithms is quoted for the equal error rate (EER) which is the scalar figure of merit that is often used to judge the performance of a verification algorithm. An interested reader is referred to [16], [53], and [52] for more details concerning the XM2VTS protocol and the experimental procedure followed. The best EER achieved for the XM2VTS database can be found in Table VIII.

G. Face Recognition Using Photometric Stereo

In this section, we describe experiments of face recognition using photometric stereo. We collected a database of faces by setting a device for proper capture of four images under four different lighting directions.

The four intensity images were processed using a standard photometric stereo method [54]–[56]. This resulted in a dense field of surface normals, which we then integrated to form height maps using the well-known Frankot and Chellappa method [57]. The albedo and the depth images were manually aligned according to the eye coordinates and were scaled to resolution 90×100.

The device was installed in the General Dynamics. Staff and visitors were kindly asked to use it. After a period of more than six months more than 250 persons used it. For 113 persons we collected images that were taken with more than a week's interval. For the majority of them (about 90) we collected samples with more than one month interval. For the experiments presented here we have a very challenging experimental procedure using only one grayscale albedo image for training and one grayscale albedo image for testing. Moreover, one depth image is used for training and one for testing.

As we have already mentioned, most of the training and testing images were captured with more than one month's interval and most of the training and testing images display a different facial expression.

The recognition rate versus the dimensionality for the grayscale albedo is plotted in Fig. 3(a), while the recognition rate for the depth images is plotted in Fig. 3(b). As it can be seen, the proposed approach achieved the best recognition rates, as well.

VI. CONCLUSION

In this paper, we proposed a method for nonlinear non-negative matrix factorization using projected gradients. Unlike other

methods for this purpose, the proposed method allows the use of a wide variety of kernels, in addition to the polynomial kernels considered in [19]. Moreover, usage of projected gradient procedure [30] guarantees that the limit point of the algorithm is a stationary point of the optimization procedure. For the special case of Gaussian RBF kernels, fixed point algorithms were proposed for non-negative nonlinear component analysis. In the particular setup proposed in this paper, we applied only Gaussian RBF kernels in order to calculate the basis. The method can be extended to other types of kernel as long as these kernels are positive definite. This constitutes a subject of further research on the topic. Moreover, further research on the topic includes the adaptation of indefinite kernels [33]–[35]. The experimental results have shown that the proposed methods can be successfully used for feature extraction and recognition and can lead to better classification rates when compared with well-known and widely used nonlinear feature extraction techniques (like KPCA and KICA).

APPENDIX A
DERIVATION OF (5)

Since, $h_{pi} = 1$ only if $\phi(\mathbf{x}_i)$ belongs to $\mathcal{C}_p$, else $h_{pi} = 0$, we have

$$\begin{aligned}||\mathbf{X}^{\Phi} - \mathbf{Z}^{\Phi}\mathbf{H}||^2 &= \sum_{i=1}^{M} ||\phi(\mathbf{x}_i) - \sum_{p=1}^{K} h_{pi}\phi(\mathbf{z}_j)||^2 \\ &= \sum_{i=1}^{M} \left|\left| \sum_{p=1}^{K} h_{pi}(\phi(\mathbf{x}_i) - \phi(\mathbf{z}_p)) \right|\right|^2 \\ &= \sum_{p=1}^{K}\sum_{i=1}^{M} h_{pi}||\phi(\mathbf{x}_i) - \phi(\mathbf{z}_p)||^2 \\ &= \sum_{p=1}^{K} \sum_{\phi(\mathbf{x}_i)\in\mathcal{C}_p} ||\phi(\mathbf{x}_i) - \phi(\mathbf{z}_p)||^2. \quad (63)\end{aligned}$$

APPENDIX B
CALCULATION OF $\nabla f_{\mathbf{H}}(\mathbf{Z})$ FOR POLYNOMIAL KERNELS

For the polynomial kernel the partial derivatives in terms of z_{ab} are (for the case $j \neq b$)

$$\begin{aligned}\frac{\partial k(\mathbf{z}_j, \mathbf{z}_b)}{\partial z_{ab}} &= \frac{\partial \left(\sum_{m=1}^{N} z_{mj} z_{mb} \right)^d}{\partial z_{ab}} \\ &= d z_{aj} \left(\sum_{m=1}^{N} z_{mj} z_{mb} \right)^{d-1} \\ &= d z_{aj} (\mathbf{z}_j^T \mathbf{z}_b)^{d-1}. \quad (64)\end{aligned}$$

Accordingly, the first partial derivative $\partial f_{\mathbf{H}}(\mathbf{Z})/\partial z_{ab}$ is extracted from (64) as

$$\begin{aligned}\frac{\partial f_{\mathbf{H}}(\mathbf{Z})}{\partial z_{ab}} = &- \sum_{i=1}^{M} h_{bi} x_{ai} d(\mathbf{x}_i^T \mathbf{z}_b)^{d-1} \\ &+ \sum_{i=1}^{M}\sum_{l=1}^{P} h_{bi} h_{li} d(\mathbf{z}_l^T \mathbf{z}_b)^{d-1} z_{al}. \quad (65)\end{aligned}$$

In matrix form, $\nabla f_{\mathbf{H}}(\mathbf{Z})$ may be written as

$$\nabla f_{\mathbf{H}}(\mathbf{Z}) = -\mathbf{X}(\mathbf{H} \odot \acute{\mathbf{K}}_{z,x})^T + \mathbf{Z}(\mathbf{H}\mathbf{H}^T \odot \acute{\mathbf{K}}_{z,z}) \quad (66)$$

where $\odot$ represents the elementwise multiplication between matrices of the same size. The matrices $\acute{\mathbf{K}}_{z,x}$ and $\acute{\mathbf{K}}(z,z)$ are defined as

$$[\acute{\mathbf{K}}_{z,x}]_{ij} \triangleq d(\mathbf{z}_i^T \mathbf{x}_j)^{d-1}, \; [\acute{\mathbf{K}}_{z,z}]_{ij} \triangleq d(\mathbf{z}_i^T \mathbf{z}_j)^{d-1}. \quad (67)$$

APPENDIX C
CALCULATION OF $\nabla f_{\mathbf{H}}(\mathbf{Z})$ FOR GAUSSIAN RBF KERNEL

The Gaussian Radial Basis Kernel is defined as $k(\mathbf{x}, \mathbf{y}) \triangleq e^{-\gamma||\mathbf{x}-\mathbf{y}||^2}$, with γ being related to the spread of the Gaussian function. For the polynomial kernel the partial derivatives in terms of z_{ab} are (for the case $j \neq b$)

$$\begin{aligned}\frac{\partial k(\mathbf{z}_j, \mathbf{z}_b)}{\partial z_{ab}} &= \frac{\partial e^{-\gamma||\mathbf{z}_j - \mathbf{z}_b||^2}}{\partial z_{ab}} \\ &= 2\gamma(-z_{ab} + z_{aj}) e^{\gamma||\mathbf{z}_j - \mathbf{z}_b||^2} \\ &= 2\gamma(-z_{ab} + z_{aj}) k(\mathbf{z}_j, \mathbf{z}_b). \quad (68)\end{aligned}$$

Accordingly, the first partial derivative $\partial f_{\mathbf{H}}(\mathbf{Z})/\partial z_{ab}$ is extracted using (68) as

$$\begin{aligned}\frac{\partial f_{\mathbf{H}}(\mathbf{Z})}{\partial z_{ab}} = &- \sum_{i=1}^{M} h_{bi} 2\gamma(-z_{ab} + x_{ai}) k(\mathbf{x}_i, \mathbf{z}_b) \\ &+ \sum_{i=1}^{M}\sum_{l=1}^{P} h_{bi} h_{li} 2\gamma(-z_{ab} + z_{al}) k(\mathbf{z}_l, \mathbf{z}_b). \quad (69)\end{aligned}$$

In matrix form, $\nabla f_{\mathbf{H}}(\mathbf{Z})$ may be written as

$$\begin{aligned}\nabla f_{\mathbf{H}}(\mathbf{Z}) = &-\mathbf{X}(\mathbf{H} \odot 2\gamma \mathbf{K}_{x,z})^T + \mathbf{Z} \odot (\mathbf{1}_F^T \mathrm{diag}(2\gamma \mathbf{H}\mathbf{K}_{x,z})) \\ &+ \mathbf{Z}(2\gamma \mathbf{H}\mathbf{H}^T \odot \mathbf{K}_{z,z}) - \mathbf{Z} \odot (\mathbf{1}^T \mathrm{diag}(2\gamma \mathbf{H}\mathbf{H}^T \mathbf{K}_{z,z})) \quad (70)\end{aligned}$$

where $\mathbf{1}_F$ is an F-dimensional vector of ones.

APPENDIX D
COMPUTATIONAL COMPLEXITY OF THE PGKNMF

The computational complexity of the NMF using multiplicative updatings is $r \times O(FMP)$[7], where r is the number of iterations of the algorithm. For the NMF using the alternative projected gradient approach in [30] the complexity is $r \times (O(FMP) + r_1 \times (O(tFP^2) + O(tMP^2))$ where r is the number of iterations of the whole procedure, r_1 is the number of iterations of the two partial minimization problems (i.e., minimizing $\mathbf{H}$ when keeping $\mathbf{Z}$ constant and vice versa) and t is the mean number of iterations for finding a proper α_t. A direct projected gradient approach for NMF has a computational complexity of $r \times O(tFMP)$ [30]. As we have already mentioned, the complexity of the PNMF [19] is $r \times O(FMPd)$ for d-degree polynomial kernels.

We shall try now to calculate the complexity of the proposed projected gradient approach. Problem (4) may be expressed as

$$\begin{aligned}&\min_{\mathbf{H}} f_{\mathbf{Z}}(\mathbf{H}) = ||\mathbf{X}^{\Phi} - \mathbf{Z}^{\Phi}\mathbf{H}||_F^2 \\ &\text{subject to } h_{kj} \geq 0, \quad \forall k, j \quad (71)\end{aligned}$$

keeping $\mathbf{Z}^{\Phi}$ and $\mathbf{X}^{\Phi}$ constant. Thus, as in [30], we may write the cost $f_{\mathbf{Z}}(\mathbf{H})$ by representing $\mathbf{H}$ as a vector (i.e., $\mathrm{vec}(\mathbf{H})$) as

$$\begin{aligned} f_{\mathbf{Z}}(\mathbf{H}) &= ||\mathbf{X}^{\Phi} - \mathbf{Z}^{\Phi}\mathbf{H}||_F^2 \\ &= \mathrm{vec}(\mathbf{H})^T \begin{bmatrix} \mathbf{K}_{z,z} & & \\ & \ddots & \\ & & \mathbf{K}_{z,z} \end{bmatrix} \mathrm{vec}(\mathbf{H}) \\ &\quad + O(\mathbf{H}) \end{aligned} \tag{72}$$

where $O(\mathbf{H})$ are terms linear in the elements of $\mathbf{H}$.

Thus, the above problem has been reformulated to a quadratic optimization problem with linear constraints. The Hessian matrix (i.e., the second derivative of $f_{\mathbf{Z}}(\mathbf{H})$) is block diagonal and moreover each block $\mathbf{K}_{z,z}$ is a $P \times P$ positive semi-definite matrix when we deal with positive kernels. As $P \ll M$, the Hessian matrix (which is a Gram matrix as well [24]) tends to be well conditioned, which is a good property for optimization. Thus, for this problem, gradient-based methods may converge fast enough.

For this subproblem, when Algorithm 4 of [30] is used to solve (71), we should calculate $\nabla f_{\mathbf{Z}}(\mathbf{H}) = \mathbf{K}_{z,z}\mathbf{H} - \mathbf{K}_{z,x}$. The constant Gram matrix $\mathbf{K}_{z,z}$ can be computed in $O(FdP^2)$ time for polynomial kernels, while it takes $O(Ft_eP^2)$ time for Gaussian RBF kernels (t_e is the time needed to calculate the exponential e^x). The calculation of matrix $\mathbf{K}_{z,x}$ requires $O(FdMP)$ in case of polynomial kernels and $O(Ft_eMP)$ in case of Gaussian RBF kernels. Thus, the cost per iteration depends on product $\mathbf{K}_{z,z}\mathbf{H}$ which is an $O(MP^2)$ operation. Hence, the cost for obtaining the gradients is

$$O(FMPd) + r_1 \times O(MP^2) \tag{73}$$

where r_1 is the number of subiterations.

The most time consuming part is to find an α_t such that

$$\mathbf{H}^{(t)} = P[\mathbf{H}^{(t-1)} - \alpha_t \nabla f_{\mathbf{Z}}(\mathbf{H}^{(t-1)})]. \tag{74}$$

In order to calculate the proper $\mathbf{H}^{(t)}$, using (74), we should calculate $f_{\mathbf{Z}}(\mathbf{H}^{(t)})$ at every iteration. This needs $O(FMPd)$ iterations (or $O(FMPt_e)$ for Gaussian RBF kernels). However, since the cost is quadratic in $\mathbf{H}$, we may substitute the checking of inequality (25) with checking inequality (28), which requires the calculation of $\mathbf{K}_{z,z}(\mathbf{H}^{(t)} - \mathbf{H}^{(t-1)})$, which takes $O(P^2M)$. Thus, the cost of $O(tFMPd)$ of checking (25) is reduced to $O(tP^2M)$. This way, the complexity of using the algorithm to solve (16) is

$$O(FMPd) + r_1 \times O(tP^2M). \tag{75}$$

The second problem

$$\begin{aligned} &\min_{\mathbf{Z}} f_{\mathbf{H}}(\mathbf{Z}) = ||\mathbf{X}^{\Phi} - \mathbf{Z}^{\Phi}\mathbf{H}||_F^2 \\ &\text{subject to } \mathbf{z}_{ik} \geq 0, \quad \forall i,k \end{aligned} \tag{76}$$

is nonlinear with more than quadratic dependency on $\mathbf{Z}$.[1] Thus, for this problem, we cannot use the trick of substituting the checking of condition (20) with an equivalent one of smaller computational cost.

As shown in Appendix II, $\nabla f_{\mathbf{H}}(\mathbf{Z})$ can be written as

$$\nabla f_{\mathbf{H}}(\mathbf{Z}) = -\mathbf{X}(\mathbf{H} \odot \acute{\mathbf{K}}_{z,x})^T + \mathbf{Z}(\mathbf{H}\mathbf{H}^T \odot \acute{\mathbf{K}}_{z,z}). \tag{77}$$

The most expensive procedure is the calculation of $\mathbf{X}(\mathbf{H} \odot \acute{\mathbf{K}}_{z,x})^T$, which needs $O(FMPd)$ calculations. As for the case of Gaussian RBF kernels, we have (see Appendix III)

$$\begin{aligned} &\nabla f_{\mathbf{H}}(\mathbf{Z}) = -\mathbf{X}(\mathbf{H} \odot \gamma\mathbf{K}_{x,z})^T + \mathbf{Z} \odot (\mathbf{1}_F^T \mathrm{diag}(2\gamma\mathbf{H}\mathbf{K}_{x,z})) \\ &+\mathbf{Z} \odot (2\gamma\mathbf{H}\mathbf{H}^T \odot \mathbf{K}_{z,z}) - \mathbf{Z} \odot (\mathbf{1}_F^T \mathrm{diag}(2\gamma\mathbf{H}\mathbf{H}^T\mathbf{K}_{z,z})) \end{aligned} \tag{78}$$

where $\mathbf{1}_F$ is an F-dimensional vector of ones. The calculation of $f_{\mathbf{H}}(\mathbf{Z})$ needs $O(MFPd)$ and $O(MFPt_e)$, for a polynomial of degree d and Gaussian RBF kernels, respectively.

Thus, the overall cost of solving (4) with polynomial kernels is

$$r \times (O(MFPd) + r_1 \times O(tP^2M) + r_2 \times O(tMFPd)) \tag{79}$$

where r is total number of iterations of the algorithm, r_1 and r_2 are the number of subiterations for solving subproblems (71) and (76), respectively, and t is the mean number of iterations for finding a proper α_t. In the same manner, the complexity, when using Gaussian RBF kernels, is

$$r \times (O(MFPt_e) + r_1 \times O(tP^2M) + r_2 \times O(tMFPt_e)). \tag{80}$$

We can apply directly Algorithm 4 of [30] in order to solve (4). That is, from a solution pair $(\mathbf{Z}^{(t-1)}, \mathbf{H}^{(t-1)})$ we may directly obtain an updating $(\mathbf{Z}^{(t)}, \mathbf{H}^{(t)})$ as (81), shown at the bottom of the page. Now $f(\mathbf{Z}^{(t-1)}, \mathbf{H}^{(t-1)})$ is not quadratic; thus, we need to calculate $f(\mathbf{Z}^{(t)}, \mathbf{H}^{(t)})$ at every iteration. This takes $O(FMPd)$ or $O(FMPt_e)$ for polynomial and Gaussian RBF kernels, respectively. The total computational cost is now

$$r \times O(tFMPd), \quad r \times O(tFMPt_e) \tag{82}$$

for polynomial degree d kernels and Gaussian RBF kernels, respectively.

Summarizing, we expect that the proposed algorithm has smaller computational cost than the direct application of projected gradients (81), since in the proposed procedure we take advantage of the fact that subproblem (71) is quadratic in terms of $\mathbf{H}$. Moreover, since in [19] the cost has been approximated by a quadratic function with respect to $\mathbf{Z}$, we anticipate that the proposed method will lead to a better minimum. In the experimental results section, we shall use the abbreviation PGKNMF (Projected Gradient Kernel non-negative Matrix Factorization) for referring to the proposed method.

[1] Except the case of the linear kernel, i.e., $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$ which corresponds to the projected gradients NMF [30]

$$(\mathbf{Z}^{(t)}, \mathbf{H}^{(t)}) = P[(\mathbf{Z}^{(t-1)}, \mathbf{H}^{(t-1)}) - \alpha(\nabla_{\mathbf{Z}^{(t-1)}} f(\mathbf{Z}, \mathbf{H}), \nabla_{\mathbf{H}^{(t-1)}} f(\mathbf{Z}, \mathbf{H}))] \tag{81}$$

APPENDIX E
AUXILIARY FUNCTIONS

Let us define the following auxiliary functions [see (83) and (84), shown at the bottom of the page].

It is straightforward to show that $G_1(\mathbf{M},\mathbf{M}) = ||\mathbf{X}^{\Phi} - \mathbf{X}^{\Phi}\mathbf{M}\mathbf{H}||_F^2$ and $G_2(\mathbf{M},\mathbf{M}) = ||\mathbf{X}^{\Phi} - \mathbf{X}^{\Phi}\mathbf{M}\mathbf{H}||_F^2$. By following similar lines to [8], [16] we have to prove that

$$\begin{aligned}
&\sum_{i=1}^{M}\sum_{j=1}^{P}\sum_{l=1}^{P} h_{ji}h_{li}\sum_{k=1}^{M}\sum_{n=1}^{M} m_{kl}m_{nj}k(\mathbf{x}_k,\mathbf{x}_n)\\
&\quad\le \sum_{i=1}^{M}\sum_{k=1}^{P}[\mathbf{K}_{x,x}\mathbf{M}^{(t-1)}\mathbf{H}\mathbf{H}^T]_{ik}\frac{\mathbf{M}_{ik}^{(t)2}}{\mathbf{M}_{ik}^{(t-1)}}\\
&\qquad\Leftrightarrow \sum_{i=1}^{N}[\mathbf{M}^T\mathbf{K}_{x,x}\mathbf{M}^{(t)}\mathbf{H}\mathbf{H}^T]_{i,i}\\
&\quad\le \sum_{i=1}^{M}\sum_{k=1}^{P}[\mathbf{K}_{x,x}\mathbf{M}^{(t-1)}\mathbf{H}\mathbf{H}^T]_{ik}\frac{\mathbf{M}_{ik}^{(t)2}}{\mathbf{M}_{ik}^{(t-1)}} \qquad (85)
\end{aligned}$$

and that

$$\begin{aligned}
&\sum_{i=1}^{M}\sum_{j=1}^{P} h_{ji}\sum_{k=1}^{N} m_{kj}k(\mathbf{x}_k,\mathbf{x}_i)\\
&\ge \sum_{i=1}^{M}\sum_{k=1}^{P} 2[\mathbf{K}_{x,x}\mathbf{H}^T]_{ik}m_{ik}^{(t-1)}\left(1+\ln\frac{m_{ik}^{(t)}}{m_{ik}^{(t)}}\right)\\
&\quad\times\sum_{i=1}^{M} 2[\mathbf{H}\mathbf{K}_{x,x}\mathbf{M}^{(t)}]_{ii}\\
&\ge \sum_{i=1}^{M}\sum_{k=1}^{P} 2[\mathbf{K}_{x,x}\mathbf{H}^T]_{ik}\mathbf{M}_{ik}^{(t-1)}\left(1+\ln\frac{\mathbf{M}_{ik}^{(t)}}{\mathbf{M}_{ik}^{(t-1)}}\right). \qquad (86)
\end{aligned}$$

By assuming that $\mathbf{M}_{ik}^{(t)} = \mathbf{M}_{ik}^{(t-1)}q_{ik}$, (85) becomes (87), shown at the bottom of the page. Relation (86) holds from the fact that

$$\frac{\mathbf{M}_{jk}^{(t)}}{\mathbf{M}_{jk}^{(t-1)}} \ge 1+\ln\frac{\mathbf{M}_{jk}^{(t)}}{\mathbf{M}_{jk}^{(t-1)}} \qquad (88)$$

since $\mathbf{M}_{jk}^{(t)} \ge 0$ and $\mathbf{M}_{jk}^{(t-1)} \ge 0$. The updating rules (53) and (54) can be found by setting $\nabla_{\mathbf{M}^{(t)}}G_1(\mathbf{M}^{(t)},\mathbf{M}^{(t-1)}) = \mathbf{0}$ or $\nabla_{\mathbf{M}^{(t)}}G_2(\mathbf{M}^{(t)},\mathbf{M}^{(t-1)}) = \mathbf{0}$.

Finally, for deriving the updating rules (56) we should define for every column $\mathbf{h}_j$ of matrix $\mathbf{H}$ the following auxiliary function [see (89), shown at the top of the following page], where $\mathcal{I}$ is the set of all variables, i.e., see (90), shown at the top of the following page. $\mathbf{D}_{\mathcal{I}\mathcal{I}}$ is the submatrix of the diagonal matrix

$$\mathbf{D}_{ll} \triangleq \begin{cases} \frac{(\mathbf{M}^T\mathbf{K}_{x,x}\mathbf{M}\bar{\mathbf{h}}_j)_l+\delta}{h_{lj}}, & \text{if } l\in\mathcal{I}\\ 0, & \text{if } l\neq\mathcal{I}. \end{cases} \qquad (91)$$

and $\mathbf{a}_{\mathcal{I}}$ is the subvector of vector $\mathbf{a}$ with indices in $\mathcal{I}$.

$$\begin{aligned}
G_1(\mathbf{M}^{(t)},\mathbf{M}^{(t-1)}) = &-\sum_{i=1}^{M}\sum_{k=1}^{P} 2[\mathbf{K}_{x,x}\mathbf{H}^T]_{ik}\tilde{m}_{ik}^{(t)}\left(1+\log\frac{m_{ik}^{(t)}}{m_{ik}^{(t-1)}}\right)\\
&+\sum_{i=1}^{M}\sum_{k=1}^{P}[\mathbf{K}_{x,x}\mathbf{M}^{(t-1)}\mathbf{H}\mathbf{H}^T]_{ik}\frac{m_{ik}^{(t)2}}{m_{ik}^{(t-1)}}+\sum_{i=1}^{M}[\mathbf{K}_{x,x}]_{ii} \qquad (83)
\end{aligned}$$

$$\begin{aligned}
G_2(\mathbf{M}^{(t)},\mathbf{M}^{(t-1)}) = &-\sum_{i=1}^{M}\sum_{k=1}^{P} 2[\mathbf{K}_{x,x}\mathbf{H}^T]_{ik}m_{ik}^{(t)}\\
&+\sum_{i=1}^{M}\sum_{k=1}^{P}[\mathbf{K}_{x,x}\mathbf{M}^{(t-1)}\mathbf{H}\mathbf{H}^T]_{ik}\frac{m_{ik}^{(t)2}}{m_{ik}^{(t-1)}}+\sum_{i=1}^{M}[\mathbf{K}_{x,x}]_{ii} \qquad (84)
\end{aligned}$$

$$\begin{aligned}
&\sum_{i}\sum_{j}\sum_{k}\sum_{l}[\mathbf{K}_{x,x}]_{ij}\mathbf{M}_{jk}^{(t-1)}[\mathbf{K}_{x,x}\mathbf{H}^T]_{kl}\mathbf{M}_{il}^{(t-1)}(q_{ik}^2-q_{ik}q_{jl})\\
&\quad=\frac{1}{2}\sum_{i}\sum_{j}\sum_{k}\sum_{l}[\mathbf{K}_{x,x}]_{ij}\mathbf{M}_{jk}^{(t-1)}[\mathbf{K}_{x,x}\mathbf{H}^T]_{kl}\mathbf{M}_{il}^{(t-1)}(q_{ik}-q_{jl})^2\ge 0 \qquad (87)
\end{aligned}$$

$$G_3(\mathbf{h}_j^{(t)}, \mathbf{h}_j^{(t-1)}) = D_\phi(\mathbf{h}_j^{(t-1)}) + (\mathbf{h}^{(t)} - \mathbf{h}^{(t-1)})_I^T \nabla D_\phi(\mathbf{h}_j^{(t-1)})_{\mathcal{I}} + (\mathbf{h}^{(t)} - \mathbf{h}^{(t-1)})_{\mathcal{I}}^T \mathbf{D}_{\mathcal{I}\mathcal{I}} (\mathbf{h}^{(t)} - \mathbf{h}^{(t-1)})_{\mathcal{I}} \quad (89)$$

$$\begin{aligned} \mathcal{I} &\triangleq \{l | h_{kj} > 0, [\nabla D_\phi(\mathbf{h}_j)]_k \neq 0 \text{ or } h_{kj} = 0, [\nabla D_\phi(\mathbf{h}_j)]_k < 0\} \\ &= \{l | \bar{h}_{kj}^l > 0, [\nabla D_\phi(\mathbf{h}_j)]_k \neq 0\} \end{aligned} \quad (90)$$

REFERENCES

[1] K. Fukunaga, *Statistical Pattern Recognition*. San Diego, CA: Academic, 1990.

[2] 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.

[3] M. Turk and A. P. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[4] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.

[5] L. Chengjun and H. Wechsler, "Independent component analysis of gabor features for face recognition," *IEEE Trans. Neural Netw.*, vol. 14, no. 7, pp. 919–928, Jul. 2003.

[6] M. S. Bartlett, J. R. Movellan, and T. J. Sejnowski, "Face recognition by independent component analysis," *IEEE Trans. Neural Netw.*, vol. 13, pp. 1450–1464, 2002.

[7] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. NIPS*, 2000, pp. 556–562.

[8] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[9] S. Z. Li, X. W. Hou, and H. J. Zhang, "Learning spatially localized, parts-based representation," in *Proc. CVPR*, Kauai, HI, Dec. 8–14, 2001, pp. 207–212.

[10] I. Buciu and I. Pitas, "Application of non-negative and local non negative matrix factorization to facial expression recognition," in *Proc. ICPR*, Cambridge, U.K., Aug. 23–26, 2004, pp. 288–291.

[11] I. Buciu and I. Pitas, "A new sparse image representation algorithm applied to facial expression recognition," presented at the MLSP, Sao Lus, Brazil, Oct. 1, 2004.

[12] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, 2004.

[13] 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.

[14] A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui, "Nonsmooth non-negative matrix factorization (nsNMF)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 403–415, Mar. 2006.

[15] S. Sra and I. S. Dhillon, non-negative Matrix Approximation: Algorithms and Applications, Dept. Comput. Sci., Univ. Texas, Austin, 2006, TR-06-27.

[16] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas, "Exploiting discriminant information in non-negative matrix factorization with application to frontal face verification," *IEEE Trans. Neural Netw.*, vol. 17, pp. 683–695, 2006.

[17] A. Scholkopf, B. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, pp. 1299–1319, 1998.

[18] F. R. Bach and M. J. Jordan, "Kernel independent component analysis," *J. Mach. Learn. Res.*, vol. 3, pp. 1–48, 2002.

[19] I. Buciu, N. Nikolaidis, and I. Pitas, "Non-negative matrix factorization in polynomial feature space," *IEEE Trans. Neural Netw.*, vol. 19, pp. 1090–1100, 2007.

[20] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[21] E. P. Simoncelli, "Vision and the statistics of the visual enviroment," *Curr. Opin. Neurobiol.*, vol. 13, pp. 144–149, 2003.

[22] J. Touryan, G. Felsen, and Y. Dan, "Spatial structure of complex cell receptive fields measured with natural images," *Neuron*, vol. 45, pp. 781–791, 2005.

[23] J. Rapela, J. M. Mendel, and N. M. Grzywacz, "Estimation nonlinear receptive fields from natural images," *J. Vis.*, vol. 6, no. 4, pp. 441–474, 2006.

[24] B. Scholkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[25] B. Scholkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. J. Smola, "Input space vs. feature space in kernel-based methods," *IEEE Trans. Neural Netw.*, vol. 10, pp. 1000–1017, 1999.

[26] J. T.-Y. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," *IEEE Trans. Neural Netw.*, vol. 15, pp. 1517–1525, 2004.

[27] E. Gonzalez and Y. Zhang, Accelerating the Lee-Seung Algorithm for non-negative Matrix Factorization, Rice Univ., Houston, TX, 2005, Tr-05-02.

[28] L. Finesso and C. Spreij, "non-negative matrix factorization and i-divergence alternative minimization," *Linear Algebra Appl.*, vol. 416, no. 2–3, pp. 270–286, 2006.

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

[30] C. J. Lin, "Projected gradients for non-negative matrix factorization," *Neural Comput.*, vol. 19, no. 10, pp. 2756–2779, Oct. 2007.

[31] D. Zhang, Z.-H. Zhou, and S. Chen, "Non-negative matrix factorization on kernels," presented at the 9th Pacific Rim Int. Conf. Artificial Intelligence, 2006, LANI 4099.

[32] S. Zafeiriou and M. Petrou, "Nonlinear non-negative component analysis," presented at the CVPR2009, Miami Beach, FL, Jun. 20–25, 2009.

[33] E. Pekalska, P. Paclik, and R. P. W. Duin, "A generalized kernel approach to dissimilarity-based classification," *J. Mach. Learn. Res.*, vol. 2, pp. 175–211, 2002.

[34] B. Haasdonk, "Feature space interpretation of SVMs with indefinite kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 482–492, 2005.

[35] E. Pekalska and B. Haasdonk, "Kernel discriminant analysis for positive definite and indefinite kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1017–1032, Jun. 2009.

[36] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel k-means: Spectral clustering and normalized cuts," in *Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2004, pp. 551–556.

[37] C. Ding, X. He, and H. D. Simon, "On the equivalence of non-negative matrix factorization and spectral clustering," in *Proc. SIAM Data Mining Conf.*, 2005, pp. 606–610.

[38] C.-J. Lin and J. J. More, "Newton's method for large-scale bound constrained problems," *SIAM J. Optim.*, vol. 9, pp. 1100–1127, 1999.

[39] P. H. Calamai and J. J. More, "Projected gradient methods for linearly constrained problems," *Math. Programm.*, vol. 39, pp. 93–116, 1987.

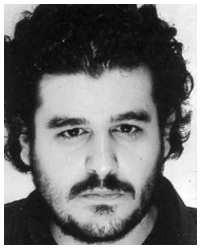
[40] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-non-negative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, unpublished.

[41] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[42] M. S. Barlett, H. M. Lades, and T. K. Sejnowski, "Independent component representations for face recognition," in *Proc. SPIE Conf. Human Vision and Electronic Imageing III*, 1998, vol. 3299, pp. 528–539.

[43] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Adv. Kernel Meth.—Support Vector Learn.*, vol. 12, pp. 185–208, 1999.

[44] J. C. Platt, N. Cristianini, and J. S. Taylor, "Large margin dags for mutliclass classification," *Adv. Neural Inf. Proces. Syst.*, vol. 12, pp. 547–553, 2000.
[45] T. Kanade, J. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in *Proc. IEEE Int. Conf. Face and Gesture Recognit.*, Mar. 2000, pp. 46–53.
[46] 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.
[47] 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.
[48] P. Jonathan, W. J. Krzanowski, and W. V. McCarthy, "On the use of cross-validation to assess performance in multivariate prediction," *Statist. Comput.*, vol. 10, no. 3, 2000.
[49] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *J. Roy. Statist. Soc. B*, pp. 111–147, 1974.
[50] M. J. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proc. 3rd IEEE Int. Conf. Automatic Face and Gesture Recognition*, 1998, pp. 200–205.
[51] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.
[52] K. Messer, J. Matas, J. V. Kittler, J. Luettin, and G. Maitre, "XM2VTSDB: The extended M2VTS database," in *Proc. AVBPA*, Washington, DC, Mar. 22–23, 1999, pp. 72–77.
[53] S. Srisuk, M. Petrou, W. Kurutach, and A. Kadyrov, "A face authentication system using the trace transform," *Pattern Anal. Appl.*, vol. 8, pp. 50–61, 2005.
[54] S. Barsky and M. Petrou, "The 4-source photometric stereo technique for three-dimensional surfaces in the presence of highlights and shadows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, pp. 1239–1252, 2003.
[55] S. Barsky and M. Petrou, "Design issues for a colour photometric stereo system," *J. Math. Imag. Vis.*, vol. 24, pp. 143–162, 2006.
[56] V. Argyriou and M. Petrou, "Photometric stereo: An overview," *Adv. Imag. Electron. Phys.*, vol. 156, pp. 1–55, 2008.
[57] R. T. Frankot and R. Chellappa, "A method for enforcing integrability in shape from shading algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, pp. 439–451, 1988.



Stefanos Zafeiriou (M'08) was born in Thessaloniki, Greece, in 1981. He received the B.Sc. degree in informatics with highest honors in 2003 and the Ph.D. degree in informatics in 2007, both from the Aristotle University of Thessaloniki.

Currently, he is a Research Associate at the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., and a postdoctoral scholar of the State Scholarship Foundation of Greece. From 2003 to 2007, he was a research and teaching assistant in the Department of Informatics, Aristotle University of Thessaloniki. During 2007–2008, he was a senior researcher in the same department. During 2009, he served as a Researcher at the University of Houston, Houston, TX. He has coauthored over than 18 journal and 19 conference papers and contributed in one book in his areas of expertise. His current research interests lie in the areas of signal and image processing, pattern recognition, machine learning, computer vision, computational intelligence, and detection and estimation theory.

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

Maria Petrou (SM'06) studied physics at the Aristotle University of Thessaloniki, Greece, and applied mathematics at Cambridge University, Cambridge, U.K. She received the Ph.D. degree from the Institute of Astronomy, Cambridge, and the D.Sc. degree from Cambridge in 2009.

She is currently a Professor of signal processing at Imperial College, London, U.K., and Director of the Informatics and Telematics Institute, CERTH, Greece. She has published more than 350 scientific papers on astronomy, remote sensing, computer vision, machine learning, color analysis, industrial inspection, and medical signal and image processing. She has coauthored two books: *Image Processing: The Fundamentals* (Wiley, 1999) and *Image Processing: Dealing with Texture* (Wiley, 2006). She has also coedited the book *Next Generation Artificial Vision Systems: Reverse Engineering the Human Visual System*.

Dr. Petrou is a Fellow of the Royal Academy of Engineering, a Fellow of the City and Guilds Institute, a Fellow of IET, a Fellow of IAPR, a Fellow of the Institute of Physics, and a Distinguished Fellow of the British Machine Vision Association.