Improving KPCA Online Extraction by Orthonormalization in the Feature Space

João B. O. Souza Filho and Paulo S. R. Diniz, *Fellow, IEEE*

***Abstract*— Recently, some online kernel principal component analysis (KPCA) techniques based on the generalized Hebbian algorithm (GHA) were proposed for use in large data sets, defining kernel components using concise dictionaries automatically extracted from data. This brief proposes two new online KPCA extraction algorithms, exploiting orthogonalized versions of the GHA rule. In both the cases, the orthogonalization of kernel components is achieved by the inclusion of some low complexity additional steps to the kernel Hebbian algorithm, thus not substantially affecting the computational cost of the algorithm. Results show improved convergence speed and accuracy of components extracted by the proposed methods, as compared with the state-of-the-art online KPCA extraction algorithms.**

***Index Terms*— Kernel methods, machine learning and generalized Hebbian algorithm (GHA), online kernel algorithms, principal component analysis (PCA).**

I. Introduction

Among the several kernel methods developed in the last decades, a remarkable technique is the Kernel PCA (KPCA), a powerful nonlinear generalization of the well-known PCA technique [3]. The KPCA is a useful tool for several applications, notably for image denoising [2] and novelty detection [4].

The KPCA was originally presented as a Gram matrix eigendecomposition problem [5], thus solvable by standard computational linear algebra techniques [6]. However, the adoption of this method on large-scale data sets, such as those involved in image processing applications, may result in high dimensional Gram matrices, turning the extraction process computationally infeasible.

Motivated by this fact, several iterative extraction algorithms were proposed [2], [7], [8]. One example is the kernel Hebbian algorithm (KHA) [2], which consists in a kernelization of the generalized Hebbian algorithm (GHA) [1], a widely used online PCA extraction algorithm. Although KHA does not involve Gram matrix estimation, it requires that all extraction data sets have to be known in advance.

Online KPCA extraction was only effectively achieved by some recently proposed algorithms [9]–[11], exploiting techniques, such as the approximated linear dependence (ALD) criterion [12] and coherence [13], to define concise dictionaries used in the expansion of kernel components. The online KHA (OKHA) [9] and subset KHA (SubKHA) [10] are useful online extensions of the KHA [2], essentially differing in how the dictionaries are built and updated. Moreover, as shown by Tanaka *et al.* [11], the extraction of kernel components using a subset of training samples can be stated as a generalized eigenvalue problem [11]. In his work, Tanaka also proposed a recursive least-square algorithm for KPCA extraction based on the kernelization of the iterative weighted rule technique [14].

The adoption of additional steps for the normalization or orthonormalization of component estimates has been shown to be effective for some principal subspace extraction methods, such as Oja [15], [16] and PAST [17], as well as for kernel subspace component learning [18], improving the convergence behavior of such algorithms. In the case of GHA, the lack of orthogonality between component estimates often compromises its accuracy and convergence speed.

Inspired by these ideas, this brief proposes two new KPCA extraction algorithms. In both, we have included an additional orthogonalization or orthonormalization step implicitly applied to component estimates in the feature space after each KHA iteration. The proposed methods exploit the classical Gram–Schmidt algorithm [6] and possess low computational cost. Results show a significant improvement in both the convergence speed and accuracy of the extracted components. The supplementary material [19] includes the derivations of some paper equations.

Manuscript received December 23, 2015; revised October 28, 2016; accepted January 23, 2017.

The authors are with the Electrical Engineering Program, COPPE/POLI/Federal University of Rio de Janeiro, Rio de Janeiro 21941-972, Brazil (e-mail: jbfilho@poli.ufrj.br; diniz@smt.ufrj.br).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2017.2660441

II. Orthogonal GHA

A well-known rule for online principal component extraction is the GHA. Assuming the extraction of p components from a sequence of random vectors denoted by $\boldsymbol{x}(k)$, this algorithm can be described by the following set of equations:

$$y_j(k) = \boldsymbol{w}_j^T(k-1)\boldsymbol{x}(k) \tag{1}$$

$$\boldsymbol{w}_j(k) = \boldsymbol{w}_j(k-1) + \eta y_j(k)\boldsymbol{e}_j(k), \quad 1 \le j \le p \tag{2}$$

with

$$\boldsymbol{e}_j(k) = \boldsymbol{x}(k) - y_j(k)\boldsymbol{w}_j(k-1) - \boldsymbol{d}_j(k) \tag{3}$$

$$\boldsymbol{d}_j(k) = \begin{cases} 0, & j = 1 \\ \sum_{i=1}^{j-1} y_i(k)\boldsymbol{w}_i(k-1), & 2 \le j \le p \end{cases} \tag{4}$$

where the parameter η is the learning rate factor, which directly affects the convergence behavior of this algorithm.

Another relevant issue in GHA convergence is the orthonormality between component estimates. Even when assuming that two arbitrary component estimates $\boldsymbol{w}_i(k-1)$ and $\boldsymbol{w}_j(k-1)$ are orthonormal at the iteration $(k-1)$, the inner product between them after a GHA update is given by [19, Sec. I]

$$\boldsymbol{w}_i^T(k)\boldsymbol{w}_j(k) \approx \eta y_i(k) y_j(k), \quad 1 \le i \ne j \le p. \tag{5}$$

However, the convergence speed and accuracy of GHA may be improved if we periodically orthogonalize component estimates using the Gram–Schmidt orthogonalization (GSO) [20], [21], or the householder transformation [22], or employing the givens rotations [23]. The classical GSO (cGSO) is an attractive technique for this task, since it can be easily extended to work in the feature space, and has sufficient accuracy for the scope of applications commonly solved by the GHA. Besides, as we will show, the cGSO procedure can be incorporated to GHA equations with a mild increase in computational cost.

For this, initially consider the Amari rule [24] for the extraction of the first principal component as follows:

$$y_1(k) = \bar{\boldsymbol{w}}_1^T(k-1)\boldsymbol{x}(k) \tag{6}$$

$$\boldsymbol{w}_1(k) = \bar{\boldsymbol{w}}_1(k-1) + \eta y_1(k)\boldsymbol{x}(k) \tag{7}$$

$$\bar{\boldsymbol{w}}_1(k) = \frac{\boldsymbol{w}_1(k)}{||\boldsymbol{w}_1(k)||_2}. \tag{8}$$

As shown by Oja [25], if we assume the value of η small, the normalization step can be incorporated into (7), resulting in the following equations:

$$y_1(k) = \boldsymbol{w}_1^T(k-1)\boldsymbol{x}(k) \tag{9}$$

$$\boldsymbol{w}_1(k) = \boldsymbol{w}_1(k-1) + \eta y_1(k)[\boldsymbol{x}(k) - y_1(k)\boldsymbol{w}_1(k-1)]. \tag{10}$$

Note that a simple extension of Oja algorithm for the extraction of p $(1 \leq j \leq p)$ components is given by

$$y_j(k) = \bar{\boldsymbol{w}}_j^T(k-1)\boldsymbol{x}(k) \tag{11}$$

$$\boldsymbol{w}_j(k) = \bar{\boldsymbol{w}}_j(k-1) + \eta y_j(k)\boldsymbol{f}_j(k) \tag{12}$$

$$\tilde{\boldsymbol{w}}_j(k) = \boldsymbol{w}_j(k) - \boldsymbol{g}_j(k) \tag{13}$$

$$\bar{\boldsymbol{w}}_j(k) = \frac{\tilde{\boldsymbol{w}}_j(k)}{||\tilde{\boldsymbol{w}}_j(k)||_2} \tag{14}$$

where

$$\boldsymbol{f}_j(k) = \boldsymbol{x}(k) - y_j(k)\bar{\boldsymbol{w}}_j(k-1) \tag{15}$$

$$\boldsymbol{g}_j(k) = \begin{cases} 0, & j = 1 \\ \sum_{i=1}^{j-1} c_{ji}(k)\bar{\boldsymbol{w}}_i(k), & 2 \leq j \leq p \end{cases} \tag{16}$$

and

$$c_{ji}(k) = \boldsymbol{w}_j^T(k)\bar{\boldsymbol{w}}_i(k). \tag{17}$$

This extension simply employs multiple Oja rules coupled by a straightforward cGSO orthonormalization procedure applied to component estimates, according to (13), (14), (16), and (17).

Considering (12) and (13), $j > i$, as well as assuming component estimates orthonormal at iteration $(k-1)$, an approximated value for the projection coefficient defined by (17) is given by [19, Sec. II]

$$c_{ji}(k) \approx 2\eta y_i(k) y_j(k). \tag{18}$$

Assuming $\bar{\boldsymbol{w}}_i(k) = \bar{\boldsymbol{w}}_i(k-1) + \boldsymbol{\Delta}_{\bar{\boldsymbol{w}}_i}(k)$, where the vector $\boldsymbol{\Delta}_{\bar{\boldsymbol{w}}_i}(k)$ represents an increment to the component estimate $\bar{\boldsymbol{w}}_i(k-1)$ proportional to η, and considering (4) and (18), (13) can be rewritten as

$$\begin{aligned} \tilde{\boldsymbol{w}}_j(k) &\approx \boldsymbol{w}_j(k) - 2\eta y_j(k)\sum_{i=1}^{j-1} y_i(k)[\bar{\boldsymbol{w}}_i(k-1) + \boldsymbol{\Delta}_{\bar{\boldsymbol{w}}i}(k)] \\ &\approx \boldsymbol{w}_j(k) - 2\eta y_j(k)\hat{\boldsymbol{d}}_j(k) + O(\eta^2) \\ &\approx \hat{\boldsymbol{w}}_j(k) \end{aligned} \tag{19}$$

with

$$\hat{\boldsymbol{w}}_j(k) = \boldsymbol{w}_j(k) - 2\eta y_j(k)\hat{\boldsymbol{d}}_j(k) \tag{20}$$

$$\hat{\boldsymbol{d}}_j(k) = \begin{cases} 0, & j = 1 \\ \sum_{i=1}^{j-1} \bar{\boldsymbol{w}}_i(k-1)y_i(k), & 2 \leq j \leq p. \end{cases} \tag{21}$$

Replacing (12) in (20), it results in the following equation:

$$\begin{aligned} \hat{\boldsymbol{w}}_j(k) &= \bar{\boldsymbol{w}}_j(k-1) + \eta y_j(k)\boldsymbol{f}_j(k) - 2\eta y_j(k)\hat{\boldsymbol{d}}_j(k) \\ &= \bar{\boldsymbol{w}}_j(k-1) + \eta y_j(k)[\boldsymbol{x}(k) - y_j(k)\bar{\boldsymbol{w}}_j(k-1) - \hat{\boldsymbol{d}}_j(k)] \\ &\quad - \eta y_j(k)\hat{\boldsymbol{d}}_j(k) \\ &= \bar{\boldsymbol{w}}_j(k-1) + \eta y_j(k)\hat{\boldsymbol{e}}_j(k) - \eta y_j(k)\hat{\boldsymbol{d}}_j(k). \end{aligned} \tag{22}$$

Thus, the proposed algorithm, here named as orthonormal GHA (ON-GHA), has the following equations:

$$y_j(k) = \bar{\boldsymbol{w}}_j^T(k-1)\boldsymbol{x}(k) \tag{23}$$

$$\hat{\boldsymbol{e}}_j(k) = \boldsymbol{x}(k) - y_j(k)\bar{\boldsymbol{w}}_j(k-1) - \hat{\boldsymbol{d}}_j(k) \tag{24}$$

$$\hat{\boldsymbol{w}}_j(k) = \bar{\boldsymbol{w}}_j(k-1) + \eta y_j(k)\hat{\boldsymbol{e}}_j(k) - \eta y_j(k)\hat{\boldsymbol{d}}_j(k) \tag{25}$$

$$\bar{\boldsymbol{w}}_j(k) = \frac{\hat{\boldsymbol{w}}_j(k)}{||\hat{\boldsymbol{w}}_j(k)||_2} \tag{26}$$

where the vector $\hat{\boldsymbol{d}}_j(k)$ is given by (21). Note that the proposed ON-GHA has quite similar equations to GHA. Indeed, they essentially differ by the term $-\eta y_j(k)\hat{\boldsymbol{d}}_j(k)$ in (25) and by the normalization step imposed by (26). If the normalization step is ignored, the ON-GHA equations become similar to the stochastic gradient ascent (SGA) algorithm proposed by Oja [26]. The orthonormality between component estimates from the ON-GHA corresponds to [19, Sec. III]

$$\bar{\boldsymbol{w}}_j^T(k)\bar{\boldsymbol{w}}_i(k) \approx 0, \quad 1 \leq i \neq j \leq p \tag{27}$$

as expected.

III. Online Orthonormal Kernel Hebbian Algorithm

To kernelize the ON-GHA, we will consider a concise and constructive dictionary $\boldsymbol{D}_m$ with m elements, composed of some dynamically selected input samples as follows:

$$\boldsymbol{D}_m = [\,\boldsymbol{x}(a) \quad \ldots \quad \boldsymbol{x}(q)\,], \quad a \neq q. \tag{28}$$

The mapping of dictionary members into the feature space can be expressed as

$$\boldsymbol{\Psi}_m = [\,\boldsymbol{\Phi}(\boldsymbol{x}(a)) \quad \ldots \quad \boldsymbol{\Phi}(\boldsymbol{x}(q))\,] \tag{29}$$

where the arbitrary function $\boldsymbol{\Phi}(\boldsymbol{x}(a))$ denotes the mapping of an input sample $\boldsymbol{x}(a)$ into this space. As in the OKHA algorithm, we will explore the ALD criterion to evaluate dictionary augmentation at each algorithm iteration. For this, initially consider the following approximate mapping for $\boldsymbol{\Phi}(\boldsymbol{x}(k))$:

$$\boldsymbol{\Phi}(\boldsymbol{x}(k)) \approx \boldsymbol{\Psi}_m\boldsymbol{\beta}_{k,m}. \tag{30}$$

This mapping corresponds to a linear combination of the columns of $\boldsymbol{\Psi}_m$ according to the components of the vector $\boldsymbol{\beta}_{k,m}$. Thus, an optimal value for $\boldsymbol{\beta}_{k,m}$ is given by [12]

$$\boldsymbol{\beta}_{k,m} = \boldsymbol{K}_m^{-1}\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)} \tag{31}$$

where the matrix $\boldsymbol{K}_m$ and the vector $\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)}$ are defined as follows:

$$\boldsymbol{K}_m = \begin{bmatrix} k(\boldsymbol{x}(a),\boldsymbol{x}(a)) & \cdots & k(\boldsymbol{x}(a),\boldsymbol{x}(q)) \\ & \ddots & \\ k(\boldsymbol{x}(q),\boldsymbol{x}(a)) & \cdots & k(\boldsymbol{x}(q),\boldsymbol{x}(q)) \end{bmatrix} \tag{32}$$

$$\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)} = [\,k(\boldsymbol{x}(k),\boldsymbol{x}(a)); \quad \cdots \quad k(\boldsymbol{x}(k),\boldsymbol{x}(q))\,]. \tag{33}$$

The square error produced in the approximative mapping from (30) is given by

$$\epsilon_k^2 = ||\boldsymbol{\Phi}(\boldsymbol{x}(k)) - \boldsymbol{\Psi}_m\boldsymbol{\beta}_{k,m}||_2^2 \tag{34}$$

Thus, if this error is greater than a threshold, denoted as the constant ν in [9], the dictionary should be updated. Using (31), (34) can be rewritten in the following form [9]:

$$\epsilon_k^2 = k(\boldsymbol{x}(k),\boldsymbol{x}(k)) - \boldsymbol{\beta}_{k,m}^T\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)}. \tag{35}$$

Now, assume that kernel principal component estimates can be expressed as a linear combination of dictionary members mapped into the feature space as follows $(1 \leq j \leq p)$:

$$\hat{\boldsymbol{w}}_j(k) = \boldsymbol{\Psi}_m\hat{\boldsymbol{\alpha}}_j(k) \tag{36}$$

$$\bar{\boldsymbol{w}}_j(k) = \boldsymbol{\Psi}_m\bar{\boldsymbol{\alpha}}_j(k). \tag{37}$$

Using (30), (31), and (37), (23) can be rewritten considering the ON-GHA operating in the feature space as follows[1]:

$$\begin{aligned} y_j^{fs}(k) &= \tilde{\boldsymbol{w}}_j^T(k-1)\boldsymbol{\Phi}(\boldsymbol{x}(k)) \\ &= \tilde{\boldsymbol{\alpha}}_j^T(k-1)\boldsymbol{\Psi}_m^T\boldsymbol{\Psi}_m\boldsymbol{\beta}_{k,m} \\ &= \tilde{\boldsymbol{\alpha}}_j^T(k-1)\boldsymbol{K}_m\boldsymbol{K}_m^{-1}\boldsymbol{\kappa}_{\boldsymbol{x}(k)}^m \\ &= \tilde{\boldsymbol{\alpha}}_j^T(k-1)\boldsymbol{\kappa}_{\boldsymbol{x}(k)}^m. \end{aligned} \tag{38}$$

From (21), we have

$$\hat{\boldsymbol{d}}_j^{fs}(k) = \begin{cases} 0, & j = 1 \\ \boldsymbol{\Psi}_m\hat{\boldsymbol{h}}_j(k), & 2 \le j \le p \end{cases} \tag{39}$$

where

$$\hat{\boldsymbol{h}}_j(k) = \sum_{i=1}^{j-1}\tilde{\boldsymbol{\alpha}}_i(k-1)y_i^{fs}(k). \tag{40}$$

Considering (24), it follows that:

$$\begin{aligned} \hat{\boldsymbol{e}}_j^{fs}(k) &= \boldsymbol{\Psi}_m\left[\boldsymbol{\beta}_{k,m} - y_j^{fs}(k)\tilde{\boldsymbol{\alpha}}_j(k-1) - \hat{\boldsymbol{h}}_j(k)\right] \\ &= \boldsymbol{\Psi}_m\hat{\boldsymbol{r}}_j(k). \end{aligned} \tag{41}$$

Similarly, using (36), (37), (40), and (41), (25) can be expressed as

$$\begin{aligned} \boldsymbol{\Psi}_m\hat{\boldsymbol{\alpha}}_j(k) &= \boldsymbol{\Psi}_m\tilde{\boldsymbol{\alpha}}_j(k-1) + \eta y_j^{fs}(k)\boldsymbol{\Psi}_m[\hat{\boldsymbol{r}}_j(k) - \hat{\boldsymbol{h}}_j(k)] \\ \hat{\boldsymbol{\alpha}}_j(k) &= \tilde{\boldsymbol{\alpha}}_j(k-1) + \eta y_j^{fs}(k)[\hat{\boldsymbol{r}}_j(k) - \hat{\boldsymbol{h}}_j(k)]. \end{aligned} \tag{42}$$

Finally, (26) can be rewritten as follows:

$$\begin{aligned} \boldsymbol{\Psi}_m\tilde{\boldsymbol{\alpha}}_j(k) &= \frac{\boldsymbol{\Psi}_m\hat{\boldsymbol{\alpha}}_j(k)}{\sqrt{\hat{\boldsymbol{\alpha}}_j^T(k)\boldsymbol{\Psi}_m^T\boldsymbol{\Psi}_m\hat{\boldsymbol{\alpha}}_j(k)}} \\ \tilde{\boldsymbol{\alpha}}_j(k) &= \frac{\hat{\boldsymbol{\alpha}}_j(k)}{\sqrt{\hat{\boldsymbol{\alpha}}_j^T(k)\boldsymbol{K}_m\hat{\boldsymbol{\alpha}}_j(k)}} = \frac{\hat{\boldsymbol{\alpha}}_j(k)}{\sqrt{\hat{\boldsymbol{\alpha}}_j^T(k)\hat{\boldsymbol{p}}_j(k)}} \end{aligned} \tag{43}$$

where

$$\hat{\boldsymbol{p}}_j(k) = \boldsymbol{K}_m\hat{\boldsymbol{\alpha}}_j(k). \tag{44}$$

Thus, considering matrices $\hat{\boldsymbol{A}}$ and $\tilde{\boldsymbol{A}}$ with columns defined by the vectors $\hat{\boldsymbol{\alpha}}_j$ and $\tilde{\boldsymbol{\alpha}}_j$ $(1 \le j \le p)$, respectively, the online orthonormal KHA (ON-KHA) can be written in the following matrix form:

$$\boldsymbol{y}(k) = \tilde{\boldsymbol{A}}^T(k-1)\boldsymbol{\kappa}_{\boldsymbol{x}(k)}^m \tag{45}$$

$$\hat{\boldsymbol{A}}(k) = \tilde{\boldsymbol{A}}(k-1) + \eta[\boldsymbol{\beta}_{k,m}\boldsymbol{y}^T(k) - \tilde{\boldsymbol{A}}(k-1)\boldsymbol{M}(k)] \tag{46}$$

$$\hat{\boldsymbol{P}}(k) = [\,\hat{\boldsymbol{p}}_1(k) \quad \cdots \quad \hat{\boldsymbol{p}}_p(k)\,] = \boldsymbol{K}_m\hat{\boldsymbol{A}}(k) \tag{47}$$

$$\boldsymbol{n}(k) = \left[\,1/\sqrt{\hat{\boldsymbol{\alpha}}_1^T(k)\hat{\boldsymbol{p}}_1(k)}; \quad \cdots \quad 1/\sqrt{\hat{\boldsymbol{\alpha}}_p^T(k)\hat{\boldsymbol{p}}_p(k)}\,\right] \tag{48}$$

$$\tilde{\boldsymbol{A}}(k) = \hat{\boldsymbol{A}}(k)\mathrm{diag}[\boldsymbol{n}(k)] \tag{49}$$

with

$$\boldsymbol{M}(k) = 2\mathrm{ut}[\boldsymbol{y}(k)\boldsymbol{y}^T(k)] - \mathrm{diag}[\boldsymbol{y}(k)\boldsymbol{y}^T(k)] \tag{50}$$

where the operator $\mathrm{ut}[\boldsymbol{A}]$ returns the upper triangular part of the matrix $\boldsymbol{A}$, while $\mathrm{diag}[\boldsymbol{n}]$ produces a diagonal matrix, whose elements are defined by the components of the vector $\boldsymbol{n}$.

A less computational complex version of ON-KHA may be derived if the normalization involved in cGSO is ignored. In this case, we have the online orthogonal KHA (OO-KHA), which may be considered as a kernelized version of the SGA [26] algorithm. The OO-KHA equations are given by

$$\boldsymbol{y}(k) = \boldsymbol{A}^T(k-1)\boldsymbol{\kappa}_{\boldsymbol{x}(k)}^m \tag{51}$$

$$\boldsymbol{A}(k) = \boldsymbol{A}(k-1) + \eta[\boldsymbol{\beta}_{k,m}\boldsymbol{y}^T(k) - \boldsymbol{A}(k-1)\boldsymbol{M}(k)] \tag{52}$$

where the matrix $\boldsymbol{M}(k)$ is given by (50). Note that the iterative equations of KHA, OKHA, and SubKHA may be written in the same form of (52), but employing a matrix $\boldsymbol{M}$ given by

$$\boldsymbol{M}(k) = \mathrm{ut}[\boldsymbol{y}(k)\boldsymbol{y}^T(k)]. \tag{53}$$

Thus, comparing (50) and (53), we conclude that the proposed orthogonalization scheme does not significantly increase the computational efforts involved in KHA.

A. Dictionary Augmentation

In case of member inclusion, the new dictionary mapped into the feature space will be given by

$$\boldsymbol{\Psi}_{m+1} = [\,\boldsymbol{\Psi}_m \quad \boldsymbol{\Phi}(\boldsymbol{x})\,]. \tag{54}$$

Thus, the mapping of the current input sample according to (30) will have a value of $\boldsymbol{\beta}_{k,m+1}$ corresponding to

$$\boldsymbol{\beta}_{k,m+1} = \begin{bmatrix} \boldsymbol{o}_{m\times 1} \\ 1 \end{bmatrix} \tag{55}$$

where all m components of the column vector $\boldsymbol{o}_{m\times 1}$ are zero.

Assume that component estimates before and after member inclusion are given by $\boldsymbol{W}^m$ and $\boldsymbol{W}^{m+1}$, respectively. Since dictionary member inclusion should not modify these estimates, we have

$$\boldsymbol{W}^{m+1} = \boldsymbol{W}^m \tag{56}$$

$$\boldsymbol{\Psi}_{m+1}\boldsymbol{A}^{m+1} = \boldsymbol{\Psi}_m\boldsymbol{A}^m \tag{57}$$

$$\boldsymbol{A}^{m+1} = \begin{bmatrix} \boldsymbol{A}^m \\ \boldsymbol{o}_{1\times p} \end{bmatrix} \tag{58}$$

where $\boldsymbol{A}^m$ and $\boldsymbol{A}^{m+1}$ will correspond to the matrix $\tilde{\boldsymbol{A}}$ (ON-KHA) or $\boldsymbol{A}$ (OO-KHA) before and after member inclusion, according to the algorithm considered, respectively.

Finally, the inclusion of a dictionary member modifies the matrices $\boldsymbol{K}_m^{-1}$ and $\boldsymbol{K}_m$ used in (31) and (47). Low computational complex formulas to calculate these matrices are the following [9]:

$$\boldsymbol{K}_{m+1}^{-1} = \begin{bmatrix} \boldsymbol{K}_m^{-1} & \boldsymbol{o}_{p\times 1} \\ \boldsymbol{o}_{1\times p} & 0 \end{bmatrix} + \frac{1}{\epsilon_k^2}\begin{bmatrix} -\boldsymbol{\beta}_{k,m} \\ 1 \end{bmatrix}\begin{bmatrix} -\boldsymbol{\beta}_{k,m}^T & 1 \end{bmatrix} \tag{59}$$

$$\boldsymbol{K}_{m+1} = \begin{bmatrix} \boldsymbol{K}_m & \boldsymbol{\kappa}_{\boldsymbol{x}(k)}^m \\ \boldsymbol{\kappa}_{\boldsymbol{x}(k)}^{m\,T} & k(\boldsymbol{x}(k),\boldsymbol{x}(k)) \end{bmatrix} \tag{60}$$

where the vector $\boldsymbol{\beta}_{k,m}$ and the variable ϵ_k^2 are given by (31) and (35), respectively.

B. Computational Cost

The ON-KHA algorithm is summarized in Table I. The algorithm is similar to OKHA in the phase of dictionary evaluation, as both follows the ALD criterion. This evaluation is conducted in Steps 4–7 and has a computational cost per iteration of $O(m^2)$. If the dictionary is updated, the matrix $\boldsymbol{K}_m^{-1}$ is incrementally produced in Step 10 by a simple $O((m+1)^2)$ procedure. The update of kernel component estimates, which occurs in Steps 16–19, is an $O((m/2)p^2)$ procedure, similar to the OKHA and SubKHA algorithms. The normalization phase adds $O(m^2p)$ operations, mainly due to Step 20. The learning rate follows a heuristically motivated exponential decay formula stated in Step 17. Clearly, the overall ON-KHA computational cost is $O(mp(m+(p/2)+(m/p)))$. For the OO-KHA, the steps 20 to 22 are eliminated, resulting in additional $O((p(p-1))/2)$ operations when compared with OKHA and SubKHA.

[1]Note that the orthonormalization of the component estimates of ON-GHA will occur in the feature space.

TABLE I
ON-KHA ALGORITHM

Define a subspace distance threshold limit ν, an initial learning rate value $\eta(0)$, and a learning decay factor γ_η.
Initialize: $m = 1$, $\boldsymbol{K}_1 = k(\boldsymbol{x}(1), \boldsymbol{x}(1))$, $\boldsymbol{K}_1^{-1} = 1/k(\boldsymbol{x}(1), \boldsymbol{x}(1))$, $\boldsymbol{D}_1 = [\boldsymbol{x}(1)]$ and $\bar{\boldsymbol{A}}(1) = [a_1 \cdots a_p]$, where $a_1 \cdots a_p$ are small random values.
for all $k \geq 2$ **do**
  $\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)} = \left[k(\boldsymbol{d}_1, \boldsymbol{x}(k)) \ldots k(\boldsymbol{d}_m, \boldsymbol{x}(k))\right]$, where $\boldsymbol{d}_i$ is the i-th column of $\boldsymbol{D}_m$.
  $\boldsymbol{\beta}_{k,m} = \boldsymbol{K}_m^{-1}\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)}$
  $\epsilon_k^2 = k(\boldsymbol{x}(k), \boldsymbol{x}(k)) - \boldsymbol{\beta}_{k,m}^T \boldsymbol{\kappa}^m_{\boldsymbol{x}(k)}$
  if $\epsilon_k^2 \geq \nu$ **then**
    $m = m + 1$
    $\boldsymbol{\kappa}_m = \begin{bmatrix} \boldsymbol{K}_{m-1} & \boldsymbol{\kappa}^{m-1}_{\boldsymbol{x}(k)} \\ \boldsymbol{\kappa}^{m-1\,T}_{\boldsymbol{x}(k)} & k(\boldsymbol{x}(k), \boldsymbol{x}(k)) \end{bmatrix}$
    $\boldsymbol{\kappa}_m^{-1} = \begin{bmatrix} \boldsymbol{K}_{m-1}^{-1} & \boldsymbol{0}_{(m-1)\mathrm{x}1} \\ \boldsymbol{0}_{1\mathrm{x}(m-1)} & 0 \end{bmatrix} + \frac{1}{\epsilon_k^2}\begin{bmatrix} -\boldsymbol{\beta}_{k,m-1} \\ 1 \end{bmatrix}\begin{bmatrix} -\boldsymbol{\beta}_{k,m-1}^T & 1 \end{bmatrix}$
    $\boldsymbol{\beta}_k = \begin{bmatrix} \boldsymbol{0}_{1\mathrm{x}(m-1)}; & 1 \end{bmatrix}$
    $\boldsymbol{D}_m = \begin{bmatrix} \boldsymbol{D}_{m-1} & \boldsymbol{x}(k) \end{bmatrix}$
    $\bar{\boldsymbol{A}}(k-1) = \begin{bmatrix} \bar{\boldsymbol{A}}(k-1) \\ \boldsymbol{0}_{1\mathrm{x}p} \end{bmatrix}$
    $\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)} = \begin{bmatrix} \boldsymbol{\kappa}^{m-1}_{\boldsymbol{x}(k)} \\ k(\boldsymbol{x}(k), \boldsymbol{x}(k)) \end{bmatrix}$
  end if
  $\boldsymbol{y}(k) = \bar{\boldsymbol{A}}(k-1)\boldsymbol{\kappa}^m_{\boldsymbol{x}(k)}$
  $\eta(k) = \gamma_\eta \eta(k-1)$
  $\boldsymbol{M}(k) = 2\mathrm{ut}[\boldsymbol{y}(k)\boldsymbol{y}^T(k)] - \mathrm{diag}[\boldsymbol{y}(k)\boldsymbol{y}^T(k)]$
  $\hat{\boldsymbol{A}}(k) = \bar{\boldsymbol{A}}(k-1) + \eta(k)\left(\boldsymbol{\beta}_{k,m}^T \boldsymbol{y}(k) - \bar{\boldsymbol{A}}(k-1)\boldsymbol{M}(k)\right)$
  $\hat{\boldsymbol{P}}(k) = \begin{bmatrix} \hat{\boldsymbol{p}}_1(k) & \cdots & \hat{\boldsymbol{p}}_p(k) \end{bmatrix} = \boldsymbol{K}_m \hat{\boldsymbol{A}}(k)$
  $\boldsymbol{n}(k) = \left[1/\sqrt{\hat{\boldsymbol{\alpha}}_1^T(k)\hat{\boldsymbol{p}}_1(k)}; \quad \cdots \quad 1/\sqrt{\hat{\boldsymbol{\alpha}}_p^T(k)\hat{\boldsymbol{p}}_p(k)}\right]$
  $\bar{\boldsymbol{A}}(k) = \hat{\boldsymbol{A}}(k)\mathrm{diag}[\boldsymbol{n}(k)]$
end for

IV. KPCA Quality Extraction Assessment

To evaluate the convergence speed and accuracy of the proposed algorithms, we considered the average value of the cosine of the angle between components produced by a reference extraction ($\boldsymbol{w}_{i,\mathrm{ref}}$) and by the algorithms upon evaluation ($\boldsymbol{w}_{i,alg}$) as follows:

$$\mathrm{AC} = \frac{1}{p}\sum_{i=1}^{p} \frac{\boldsymbol{w}_{i,alg}^T \boldsymbol{w}_{i,\mathrm{ref}}}{||\boldsymbol{w}_{i,alg}||_2 ||\boldsymbol{w}_{i,\mathrm{ref}}||_2}. \tag{61}$$

Here, the reference components were produced by standard KPCA. Clearly, this index attains a maximum value of 1 when both components only differ in their norms. This value, here denoted as the average cosine (AC) index, can be expressed in the following form [19, Sec. IV]:

$$\mathrm{AC} = \frac{1}{p}\sum_{i=1}^{p} \frac{\boldsymbol{\alpha}_{i,alg}^T \left[\, \boldsymbol{K}_m \quad \boldsymbol{K}^{\mathrm{ref}}_{\mathrm{dic}\times\bar{\mathrm{dic}}} \,\right]\boldsymbol{\alpha}_{i,\mathrm{ref}}}{\sqrt{\boldsymbol{\alpha}_{i,alg}^T \boldsymbol{K}_m \boldsymbol{\alpha}_{i,alg}}\sqrt{\boldsymbol{\alpha}_{i,\mathrm{ref}}^T \boldsymbol{K}^{\mathrm{ref}} \boldsymbol{\alpha}_{i,\mathrm{ref}}}} \tag{62}$$

where the matrix $\boldsymbol{K}_m$ is defined by (32), and the Gram matrix employed in the reference extraction is denoted by $\boldsymbol{K}^{\mathrm{ref}}$. The matrix $\boldsymbol{K}^{\mathrm{ref}}_{\mathrm{dic}\times\bar{\mathrm{dic}}}$ is obtained by the following decomposition of the matrix $\boldsymbol{K}^{\mathrm{ref}}$:

$$\boldsymbol{K}^{\mathrm{ref}} = \begin{bmatrix} \boldsymbol{K}_m & \boldsymbol{K}^{\mathrm{ref}}_{\mathrm{dic}\times\bar{\mathrm{dic}}} \\ \boldsymbol{K}^{\mathrm{ref}}_{\bar{\mathrm{dic}}\times\mathrm{dic}} & \boldsymbol{K}^{\mathrm{ref}}_{\bar{\mathrm{dic}}\times\bar{\mathrm{dic}}} \end{bmatrix}. \tag{63}$$

V. Results

The proposed algorithms were experimentally evaluated using the USPS database of handwritten digits [27]. This set is composed of grayscale handwritten digit images of size 16×16, which were coded using integer numbers in the range of 0–255. As we reproduced one experiment conducted in [28], our evaluation data set involved only the first 100 images related to the digits 1 to 3 from USPS, and we considered the extraction of 16 kernel components using a radial basis function kernel with $\sigma = 8$.

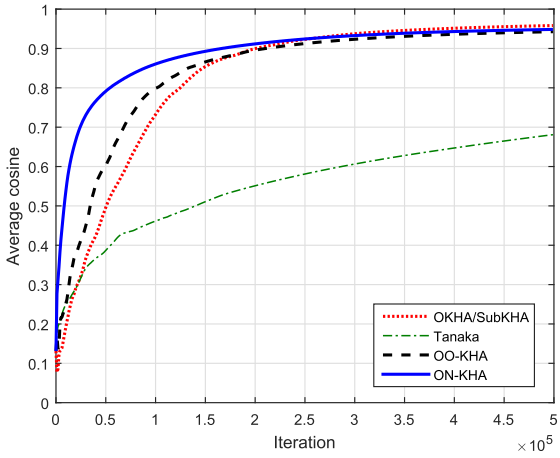

Fig. 1. Mean curve of the AC index obtained by each method considering the full sample dictionary (see text).

The accuracy and convergence speed of the proposed ON-KHA and OO-KHA were experimentally compared with SubKHA, OKHA, and Tanaka algorithms using the AC index as the figure of merit. Two evaluation scenarios were considered for kernel component expansion: using all data set members (full sample dictionary), and based on a small subset of them (small sample dictionary). In the last case, the parameters of each method were tuned to result in about 49 members (16.3% of the total).

We produced 25 simulations to each method, considering that the elements of the matrix $\boldsymbol{A}$ follow a random zero-mean Gaussian distribution with variance equal to 0.01. To define the inputs of the algorithms, each one of the 300 selected images was converted into a column vector having 256 dimensions, and was normalized to be in 0 to 1 range. These vectors were randomly presented to the algorithms. The initial learning rate value (η) and the learning rate decay factor (γ_η) were chosen as 0.05 and 0.999995, respectively. Simulations were conducted for 500 000 iterations.

All algorithms, except the SubKHA and the Tanaka, employed a value of ν equal to 10^{-3} and 0.25 for full and small sample dictionaries, respectively. Similarly, for the SubKHA, we considered the maximum number of dictionary members given by 300 and 49, and the constant α was chosen equal to 1. For the Tanaka algorithm, we utilized: decreasing weighting factors starting from 0.9 with a regular step of 0.05; a factor β equal to 0.9999; the coherence threshold equal to 1 and 0.742 (full and small sample dictionaries, respectively); and the matrix $\boldsymbol{Q}$ with random elements following a zero-mean Gaussian distribution whose variance equals 0.1.

The mean curves of the AC index produced by each method in case of full and small dictionaries are shown in Figs. 1 and 2. In the case of full dictionary, the OKHA and SubKHA curves are superimposed, as expected. The Tanaka algorithm was the less accurate and slowest converging method in this comparison. The proposed OO-KHA performed better than the OKHA and SubKHA, reaching the performance of the ON-KHA in about 200 000 iterations. Assuming an AC of 0.8, the OKHA and SubKHA took about 125 000 iterations to reach this value, whereas the ON-KHA and OO-KHA achieved it in about 55 000 and 100 000 iterations, respectively. Thus, the proposed methods were ≈2.27 (ON-KHA) and ≈1.25 (OO-KHA) times faster in this experiment.

Concerning the small sample case, a similar behavior to full-dictionary simulations was observed, as expected. However, the use of

TABLE II

MEAN AND STANDARD DEVIATION VALUES FOR THE VALUE OF AC CONSIDERING DIFFERENT ITERATIONS, ALGORITHMS, AND DICTIONARIES SIZES (SEE TEXT)

	Full sample dictionary						
Iteration	**OKHA**	**Tanaka**	**SubKHA**	**OO-KHA**	**ON-KHA**	$\chi^2(4)$	p **value**
50,000	0.4887 ± 0.0572	0.3841 ± 0.0719	0.4887 ± 0.0572	0.5974 ± 0.0513	$\mathbf{0.7891 \pm 0.0537}$	101.7	$p < 0.001$
150,000	$\mathbf{0.8525 \pm 0.0579}$	0.5097 ± 0.0594	$\mathbf{0.8525 \pm 0.0579}$	$\mathbf{0.8650 \pm 0.0777}$	$\mathbf{0.8923 \pm 0.0566}$	63.17	$p < 0.001$
500,000	$\mathbf{0.9582 \pm 0.0306}$	0.6809 ± 0.0869	$\mathbf{0.9582 \pm 0.0306}$	$\mathbf{0.9426 \pm 0.0571}$	$\mathbf{0.9484 \pm 0.0432}$	59.24	$p < 0.001$
	Small sample dictionary						
50,000	$\mathbf{0.4220 \pm 0.041}$	0.3294 ± 0.0765	0.2637 ± 0.0185	0.4905 ± 0.0486	$\mathbf{0.6327 \pm 0.0499}$	106.71	$p < 0.001$
150,000	$\mathbf{0.6320 \pm 0.0644}$	0.4249 ± 0.0727	0.3820 ± 0.0446	$\mathbf{0.6718 \pm 0.0582}$	$\mathbf{0.6800 \pm 0.0465}$	91.63	$p < 0.001$
500,000	$\mathbf{0.6917 \pm 0.0531}$	0.5518 ± 0.0830	0.4229 ± 0.0439	$\mathbf{0.7021 \pm 0.0510}$	$\mathbf{0.6965 \pm 0.0499}$	83.46	$p < 0.001$

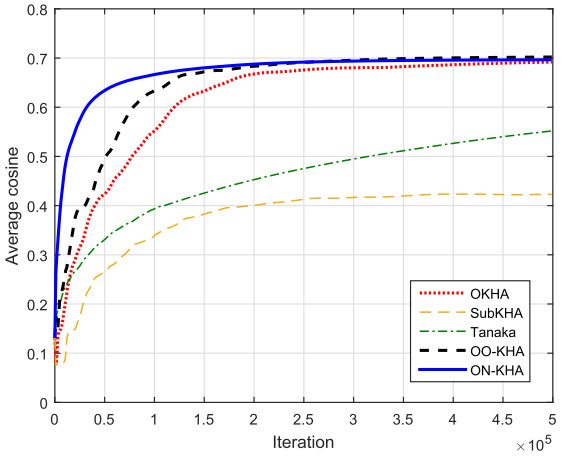

Fig. 2. Mean curve of the AC index obtained by each method considering the small sample dictionary (see text).

only ≈16% of data set samples to expand kernel components resulted in lower values of the AC index in all algorithms, as expected. Another remarkable result was the slow convergence observed for the SubKHA, caused by a frequent exchange of dictionary members. In this case, the SubKHA performed worse than the Tanaka algorithm. Again, the proposed ON-KHA and OO-KHA converged faster than the remaining algorithms in about 100 000 and 150 000 iterations, respectively.

We used the nonparametric test of Kruskal–Wallis (KW) [29] to statistically evaluate the values of the AC index attained by each method at the following iterations: 50 000, 150 000, and 500 000, considering both small and full sample dictionaries. Post-hoc analysis was performed using the Tuckey's HSD [29] procedure. We considered the confidence level equal to 0.10. Table II summarizes the results, where boldface letters indicate statistically similar values. Since the values of p reported in Table II are lower than the confidence level for all simulations, we should refuse the hypothesis that the methods performed similarly in both evaluation scenarios. At iteration 50 000, according to the post-hoc tests, the proposed ON-KHA was the best performing method, achieving the highest value for the AC index for both dictionary sizes, and a value of p lower than 0.0747 in the worst case comparison, i.e., the values of p attained in all pairwise comparisons satisfy $p < 0.0747$. At iteration 150 000, the ON-KHA still showed a higher value of the AC index than the proposed OO-KHA, but both performed similar to most of the alternative methods, concerning small and full-dictionary sizes ($p < 0.4707$). However, the ON-KHA outperformed Tanaka algorithm ($p < 0.001$) in full-dictionary simulations, and the Tanaka ($p < 0.001$) and the SubKHA ($p < 0.001$) in small-dictionary experiments. At iteration 500 000, the proposed methods and OKHA practically showed the same value for the AC index for both dictionary sizes ($p > 0.9932$).

VI. CONCLUSION

In this brief, we have proposed two new online KPCA extraction algorithms, exploiting an implicit orthogonalization (OO-KHA) or orthonormalization (ON-KHA) of kernel component estimates in the feature space. Both methods involve simple equations, which eases their embedded implementation and possesses a similar computational cost to the state-of-the-art online KPCA algorithms. Simulations have shown impressive gains in the accuracy of the components extracted and in the convergence speed.

REFERENCES

[1] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Netw.*, vol. 2, no. 6, pp. 459–473, 1989.
[2] K. I. Kim, M. Franz, and B. Scholkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1351–1366, Sep. 2005.
[3] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York, NY, USA: Springer, 2002.
[4] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognit.*, vol. 40, no. 3, pp. 863–874, 2007.
[5] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.
[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.
[7] T.-J. Chin and D. Suter, "Incremental kernel principal component analysis," *IEEE Trans. Image Process.*, vol. 16, no. 6, pp. 1662–1674, Jun. 2007.
[8] W. Liao, A. Pizurica, W. Philips, and Y. Pi, "A fast iterative kernel PCA feature extraction for hyperspectral images," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2010, pp. 1317–1320.
[9] P. Honeine, "Online kernel principal component analysis: A reduced-order model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1814–1826, Sep. 2012.
[10] Y. Washizawa, "Adaptive subset kernel principal component analysis for time-varying patterns," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 12, pp. 1961–1973, Dec. 2012.
[11] T. Tanaka, Y. Washizawa, and A. Kuh, "Adaptive kernel principal components tracking," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1905–1908.
[12] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.
[13] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.
[14] J. Yang, Y. Zhao, and H. Xi, "Weighted rule based adaptive algorithm for simultaneously extracting generalized eigenvectors," *IEEE Trans. Neural Netw.*, vol. 22, no. 5, pp. 800–806, May 2011.
[15] K. Abed-Meraim, S. Attallah, A. Chkeif, and Y. Hua, "Orthogonal Oja algorithm," *IEEE Signal Process. Lett.*, vol. 7, no. 5, pp. 116–119, May 2000.

[16] S. Attallah and K. Abed-Meraim, "Fast algorithms for subspace tracking," *IEEE Signal Process. Lett.*, vol. 8, no. 7, pp. 203–206, Jul. 2001.
[17] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal PAST algorithm," *IEEE Signal Process. Lett.*, vol. 7, no. 3, pp. 60–62, Mar. 2000.
[18] T. Tanaka, "Dictionary-based online kernel principal subspace analysis with double orthogonality preservation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 4045–4049.
[19] *Supplementary Material for the paper Improving KPCA Online Extraction by Orthonormalization in the Feature Space*. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2017.2660441
[20] A. Dax, "A modified Gram–Schmidt algorithm with iterative orthogonalization and column pivoting," *Linear Algebra Appl.*, vol. 310, nos. 1–3, pp. 25–42, May 2000.
[21] W. Hoffmann, "Iterative algorithms for Gram–Schmidt orthogonalization," *Computing*, vol. 41, no. 4, pp. 335–348, 1989.
[22] Å. Björck, "Numerics of Gram–Schmidt orthogonalization," *Linear Algebra Appl.*, vols. 197–198, pp. 297–316, Jan./Feb. 1994.
[23] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, "Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization," *Math. Comput.*, vol. 30, no. 136, pp. 772–795, Oct. 1976.
[24] A. Cichocki and S.-I. Amari, *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. Hoboken, NJ, USA: Wiley, 2002.
[25] E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Biol.*, vol. 15, no. 3, pp. 267–273, 1982.
[26] E. Oja, "Principal components, minor components, and linear neural networks," *Neural Netw.*, vol. 5, no. 6, pp. 927–935, Nov./Dec. 1992.
[27] *USPS Handwritten Digits*, accessed on Dec. 12, 2014. [Online]. Available: http://www.cs.nyu.edu/~roweis/data.html
[28] S. Günter, N. N. Schraudolph, and S. V. N. Vishwanathan, "Fast iterative kernel principal component analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1893–1918, Aug. 2007.
[29] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 4th ed. London, U.K.: Chapman & Hall, 2007.