International Journal of Innovative
Computing, Information and Control
Volume **7**, Number **10**, October **2011**

 ISSN 1349-4198
pp. 6019–6040

A FAST INCREMENTAL MULTILINEAR PRINCIPAL COMPONENT ANALYSIS ALGORITHM

Jin Wang[1], Armando Barreto[1], Naphtali Rishe[2], Jean Andrian[1] and Malek Adjouadi[1]

[1]Department of Electrical and Computer Engineering
[2]School of Computing and Information Science
Florida International University
11200 S.W. 8th Street, Miami, Florida, U.S.
adjouadi@fiu.edu

Received April 2010; revised September 2010

Abstract. *This study establishes the mathematical foundation for a fast incremental multilinear method which combines the traditional sequential Karhunen-Loeve (SKL) algorithm with the newly developed incremental modified fast Principal Component Analysis algorithm (IMFPCA). In accordance with the characteristics of the data structure, the proposed algorithm achieves both computational efficiency and high accuracy for incremental subspace updating. Moreover, the theoretical foundation is analyzed in detail as to the competing aspects of IMFPCA and SKL with respect to the different data unfolding schemes. Besides the general experiments designed to test the performance of the proposed algorithm, incremental face recognition system was developed as a real-world application for the proposed algorithm.*
Keywords: Multilinear principal component analysis, Fast principal component analysis, Incremental subspace learning, Sequential Karhunen-Loeve algorithm, Mean update

1. **Introduction.** The so-called appearance-based techniques, such as the Principal Component Analysis (PCA) and the Linear Discriminant Analysis (LDA), have been extensively used in the literature with a wide range of applications in fields such as computer vision, pattern classification, signal/image processing, among others. However, their computational complexity and their batch mode computational frameworks still impose practical constraints in applications that demand concurrently faster execution speed and higher accuracy in the results. A variation on the singular value decomposition (R-SVD) [1] provides a faster approach for obtaining a specific subspace of a given data structure. Based on R-SVD, A. Levy and M. Lindenbaum developed the sequential Karhunen-Loeve (SKL) algorithm [2], which is characterized by a faster execution speed and higher suitability for dealing with image sequences. Many other applications were consequently reported by utilizing the SKL algorithm. For instance, D. Ross et al. [3] proposed a visual tracking system based on an incremental subspace method with sample mean update. Also, Zhao et al. [4] developed a novel incremental PCA with specific application to face recognition. Moreover, T. Jun et al. [5] developed the incremental subspace method for kernel PCA, and applied it to offline and online face recognition as well as visual tracking. In study [6], L. Hoegaertsa et al. proposed a method which is similar to the research concept in [4], but extended it into the kernel space and included both updating and downdating procedure for tracking purposes. Another kind of fast principal component extraction method called Principal Component Orthogonal Projection Approximation and Subspace Tracking (PC-OPAST) was introduced in [7] by S. Bartelmaos and K. Abed-Meraim to

be applied for incremental learning as well. The PC-OPAST method alleviates the computational burden for estimating the principal eigenvectors of the covariance matrix using Givens rotations.

With the use of tensors in multilinear algebra being firmly established, great efforts have been devoted to their potential for dimensionality reduction. Fewer features, based on multilinear PCA, obtained to achieve the same accuracy, were described in [8]. In [9], the alternative least squares method was used to find a desired tensor with minimum cost to the original tensor. This method is applied to the multidimensional data directly. A new framework of multilinear PCA for dimensionality reduction and feature extraction was provided in [10] with an application to gait recognition. The iterative local optimization procedure was applied to find projection matrices. Moreover, there are some studies on the incremental learning of tensors. A visual tracking system proposed by X. Li et al. [11] was based on an incremental tensor subspace learning method, and the subspace update deployed the SKL algorithm. In [12], multilinear analysis and wavelets were combined for the analysis of time evolving data. Alternating minimization was adopted for unfolding modes without including time dimension as a compression step, and then discrete wavelet transform was adopted on the results of the compression step. Moreover, in [13], the subspace update is based on the characteristic of chunk data input. Unlike most articles for incremental learning, which keep a static number of eigenvectors, the approach in [14] was based on the reconstruction error, in which the number of eigenvectors used can change with each incremental update.

In all of those studies, the challenge remains in finding the appropriate balance between computational efficiency and high accuracy in estimating the eigenvectors. To come to terms with this challenge, this study proposes a modified fast PCA algorithm embedding an incremental multilinear method. Based on the characteristics of multilinear method, a new incremental subspace update method is described. Practical implementations of this new incremental procedure on different kinds of targets in image sequences are chosen to prove the validity of the incremental multilinear PCA system. The rest of the paper is structured in the following way: Section 2 introduces the theoretical framework of the modified fast principal component analysis; Section 3 describes the concept of tensor incremental learning, and evaluates the computational complexity of the proposed algorithm; Section 4 provides the experimental results comparing fast PCA versus the modified fast PCA, and the proposed incremental algorithm versus the traditional SKL-based incremental algorithm for tensor objects; in addition, a real-world application of face recognition is tested by using the proposed incremental algorithm to assess its practical merit.

2. **Modified Fast Principal Component Analysis.**

PCA is used for approximating a set of vectors that will establish the lowest dimension subspace that can still contain most of the relevant information. The theoretical merit and practical impact of the PCA have been studied thoroughly over the years, especially with regard to its computational complexity, and many of the improvements that have been accomplished, as reported in the Cyclic Jacobi's method [15], power method [16] and modified Cyclic Jacobi's method [17]. Moreover, the fast PCA [15] is a computationally efficient technique that is designed for finding the leading eigenvectors. With covariance matrices of 2000×2000 in size, the fast PCA achieved a better computational efficiency requiring only 2.28s for its execution, while the eigenvalue decomposition PCA required 153.26s, as reported in [18]. Besides dealing with these heavy computational requirements, estimating the eigenvectors from using these different methods is also important in order not to compromise on the desired high accuracy in the results. For these two contending but essential objectives of accuracy

and computational efficiency, the fast PCA algorithm is described, with emphasis placed on the modifications that were made as a necessity for enhancing its key execution steps.

2.1. **Fast principal component analysis.** The fast PCA is obtained by minimizing the mean square error between the original vectors and their reconstructed versions from a dimensionally-reduced principal component transform, while no or minimal concessions are made on accuracy.

Suppose that $x \in \mathcal{R}^{m\times 1}$ represents a vector with a mean $\mu = E[x]$, the reduced dimensional feature vector is then denoted as $y \in \mathcal{R}^{h\times 1}$. With the reconstructed vector of x being $\hat{x} \in \mathcal{R}^{m\times 1}$, the mean square error can be presented as:

$$MSE = E\left[||x - \hat{x}||^2\right] \tag{1}$$

where $||\bullet||$ denotes the norm value and the function $E[\bullet]$ is defined as the expectation operation.

In the PCA transform, the reduced eigenbasis is supposed to be $U \in \mathcal{R}^{m\times h}$, and then the reduced dimensional vector can be computed as $y = U^T(x - \mu)$, with zero empirical mean. With the reduced dimensional vector y, the reconstructed $\hat{x}$ of x can be computed as $\hat{x} = Uy + \mu = UU^T(x - \mu) + \mu$. Therefore, Equation (1) can be rewritten as follows:

$$MSE = E\left[||(I - UU)^T(x - \mu)||^2\right] \tag{2}$$

The scalar function $||(I - UU)^T(x - \mu)||^2$, which determines the norm of a vector, can thus be simplified as follows:

$$\begin{aligned} ||(I - UU^T)(x - \mu)||^2 &= ((I - UU^T)(x - \mu))^T(I - UU^T)(x - \mu) \\ &= (x - \mu)^T(I - UU^T)^T(I - UU^T)(x - \mu) \\ &= (x - \mu)^T(I - 2UU^T + UU^TUU^T)(x - \mu) \end{aligned} \tag{3}$$

Since the eigenvectors are orthomormal, the simplification $UU^TUU^T = UIU^T = UU^T$ can be used, which further simplifies Equation (3) to the following squared norm:

$$||(I - UU^T)(x - \mu)||^2 = (x - \mu)^T(I - UU^T)(x - \mu) \tag{4}$$

Then, finding the derivative of MSE [16] as:

$$\frac{\partial}{\partial U}E\left[(x - \mu)^T(I - UU^T)(x - \mu)\right] = -2E\left[(x - \mu)(x - \mu)^TU\right] \tag{5}$$

The fixed-point algorithm [18] and Gram-Schmidt orthonormalization process [17] can then be used to estimate the leading eigenvectors.

2.2. **Modified fast principal component analysis.** The eigenbasis $U \in \mathcal{R}^{m\times h}$ of a matrix $D \in \mathcal{R}^{m\times n}$ should evidently satisfy the standard relation $\lambda U = DU$, where λ represents the corresponding eigenvalues. The eigenbasis $U = [e_1, e_2, \ldots, e_h]$ should be composed of eigenvectors with the largest eigenvalues, and should be ordered as e_1, e_2, …, e_h column-wise in accordance to their respective eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_h$. Under usual circumstances, eigenvectors are computed in a descending order of their corresponding eigenvalues by the fast PCA. Unfortunately, the algorithm tends to be numerically unstable if the initial vector is generated randomly [20,21]. Due to this instability, the eigenvectors are not set in descending order. The intuitive approach is to repeat the iteration with various initial vectors. In this case, the additional step is to check if the new eigenvalue is smaller than the previous one. If so, proceed computing the next eigenvector; otherwise go back to the previous step, and recalculate that eigenvalue and respective eigenvector using a different initial vector. The pseudo-code implementation of the modified fast PCA (MFPCA) is provided as below in Table 1. The processing

sub step as shown in bold and italic within Step 3 indicates the modification that was introduced in the MFPCA.

TABLE 1. Pseudocode implementation of MFPCA algorithm

Input: The scatter matrix φ of a centered matrix $\bar{D}$, the number of eigenvectors k and the error tolerance ε.
Output: The eigenvectors of $\bar{D}$.
Algorithm:
Step 1 (Parameter Initialize): Set $i = 1$.
Step 2 (Vector Initialize): Initialize eigenvector $U(i)$ randomly and define $U_{int} = U(i)$.
Step 3 (Optimization Computation):
– Calculate $U(i) = \varphi * U(i)$.
– Implement the Gram-Schmidt process.
$U(i) = U(i) - \sum_{j=1}^{i-1} proj_{U(j)}U(i)$,
where $proj_u v = \dfrac{\langle u, v\rangle}{\langle u, u\rangle} u$ and $\langle u, v\rangle$ denotes the inner product of vector u and v.
In this case, $\langle u, u\rangle = 1$.
– Calculate the norm $\lambda(i)$ of $U(i)$, $\lambda(i) = norm(U(i))$.
–∗ ***If*** $i > 1$ ***and*** $\lambda(i) < \lambda(i-1)$, ***set*** $i = i - 1$ ***and go to Step 2***, ***else go to Step 1***.
– If $|U(i)U_{int}| < \varepsilon$ is not satisfied, go to Step 3.
– Set $i = i + 1$ and go to Step 2 until $i = k$.

Considering this computational aspect of the modified fast PCA, its application as an incremental algorithm can now be explored by incrementally updating the leading eigenvectors with the inclusion of new data. The merit of incremental algorithms is that with new data received, the previously obtained leading eigenvectors together with the newly acquired data will generate a new set of leading eigenvectors for the whole dataset, minimizing to a great extend the computational burden required of regenerating all the leading eigenvectors using the whole dataset all over again.

3. **Incremental Algorithms for Tensor Objects.** The conventional method used for incremental PCA is the SKL algorithm. In fact, most articles referenced earlier make use of the SKL algorithm mainly for its computational efficiency. For most image-as-vector systems, SKL is indeed very efficient. However, if the so-called "image-as-vector" systems are re-arranged as tensor objects with different data structures after different modes of unfolding, sometimes the covariance matrix itself provides new means for seeking additional computational benefits. In our study, the modified fast PCA and SKL algorithms are utilized for different unfolding modes in order to achieve better computational efficiency. The following sections introduce basic multilinear algebra, propose the new unfolding method, explain the incremental procedure and evaluate the computational complexity as it pertains to the incremental algorithm.

3.1. **Basic multilinear algebra and related terminology.** A high-order tensor is denoted as $\mathcal{A} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_N}$, where I_n with $n = 1, \ldots, N$ represents the size of the n^{th} dimension of the tensor. The mode-n product of a tensor $\mathcal{A}$ by a matrix $U \in \mathcal{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n U$ is determined by the tensor entries $(\mathcal{A} \times_n U)_{i_1,\ldots,i_{n-1},j_n,i_{n+1},\ldots,i_N} = \sum_{i_n} a_{i_1 \ldots i_N} u_{j_n i_n}$, where i_n denotes the mode-n of $\mathcal{A}$. The scalar product of two tensors $\mathcal{A}, \mathcal{B} \in \mathcal{R}^{I_1 \times I_2 \times \ldots \times I_N}$

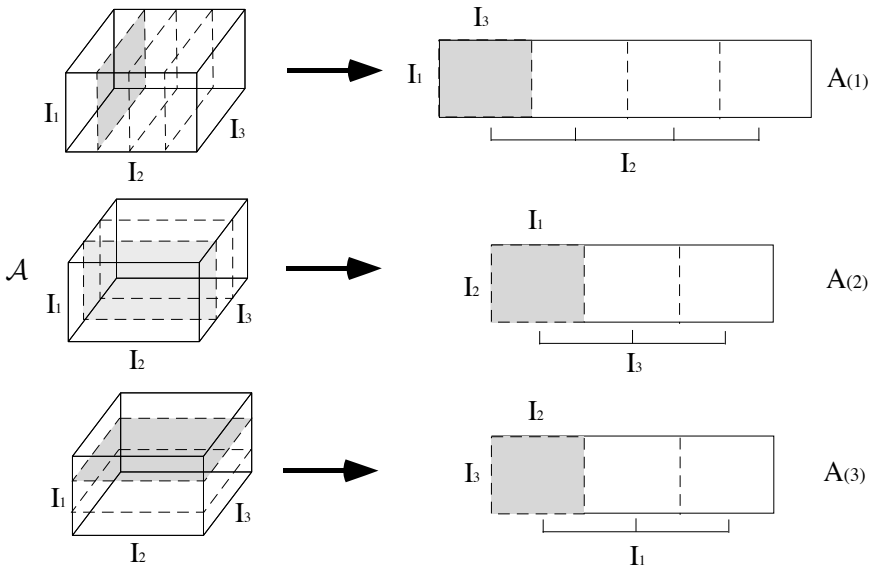


FIGURE 1. Graphic description of the unfolding process for different modes. From the top to the bottom, they are mode-1 unfolding, mode-2 unfolding and mode-3 unfolding, respectively.

is defined as $\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1} \sum_{i_2} \dots \sum_{i_N} a_{i_1 i_2 \dots i_N} b_{i_1 i_2 \dots i_N}$. And the Frobenius norm of a tensor $\mathcal{A} \in \mathcal{R}^{I_1 \times I_2 \dots \times I_N}$ is defined as $||\mathcal{A}|| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. Any tensor can be expressed as the product $\mathcal{A} = \mathcal{G} \times_1 U_{A_{(1)}} \times_2 U_{A_{(2)}} \dots \times_N U_{A_{(N)}}$, where $\mathcal{G} \in \mathcal{R}^{J_1 \times J_2 \dots \times J_N}$ is the core tensor, defined as $\mathcal{G} = \mathcal{A} \times_1 U_{A_{(1)}} \times_2 U_{A_{(2)}} \dots \times_N U_{A_{(N)}}$, with $U_{A_{(i)}} = (u_{A_{(i)}1}, u_{A_{(i)}2}, \dots, u_{A_{(i)}k_{A_{(i)}}})$ being a unitary matrix. A mode-n unfolding gives a matrix $A_{(n)} \in \mathcal{R}^{I_n \times I_1 I_2 \dots I_{n-1} I_{n+1} \dots I_N}$. The different unfolding mechanisms for a 3rd-order tensor are illustrated Figure 1.

Traditional unfolding methods are called backward cyclic and forward cyclic. How to implement incremental learning based on the traditional unfolding method was addressed in detail in [22]. In order to do the incremental learning, an extra step of matrix computation is required. However, a new unfolding method is utilized in this study. Taking the backward cyclic unfolding for example, the elements in $A_{(n)}$ can be defined as $(A_{(n)})_{(index)} = a_{i_1 i_2 \dots i_N}$, where

$$index = \left[i_n, \sum_{p_1=n+1}^{N} (i_{p_1} - 1) \left(\prod_{p_2=p_1+1}^{N} I_{p_2} \right) \left(\prod_{p_2=1}^{n-1} I_{p_2} \right) + \sum_{p_1=1}^{n-1} (i_{p_1} - 1) \left(\prod_{p_2=p_1+1}^{n-1} I_{p_2} \right) \right] \quad (6)$$

with I_{p_2} being the length of the dimension p_2.

The elements obtained by the new unfolding method are defined with a different index,

$$index = \left[i_n, \sum_{p_2=N}^{n+1} (i_{p_2} - 1) \left(\prod_{p_1=p_2}^{n+1} I_{p_1} \right) \left(\prod_{p_1=n-1}^{1} I_{p_1} \right) + \sum_{p_2=n-1}^{1} (i_{p_2} - 1) \left(\prod_{p_1=p_2-1}^{1} I_{p_1} \right) \right] \quad (7)$$

Figure 2 shows the difference graphically among the new proposed unfolding method for mode-1 and mode-2, the backward cyclic method as well as the forward cyclic method for a 3rd-order tensor.

The new method keeps the newly added data at the end of the matrix, which can be directly used in the incremental algorithm, instead of requiring other matrix computations. The problem in the structuring of the unfolding between mode-1 and mode-2 as seen in Figure 2 is now resolved by the proposed algorithm, where the structuring of the unfolding

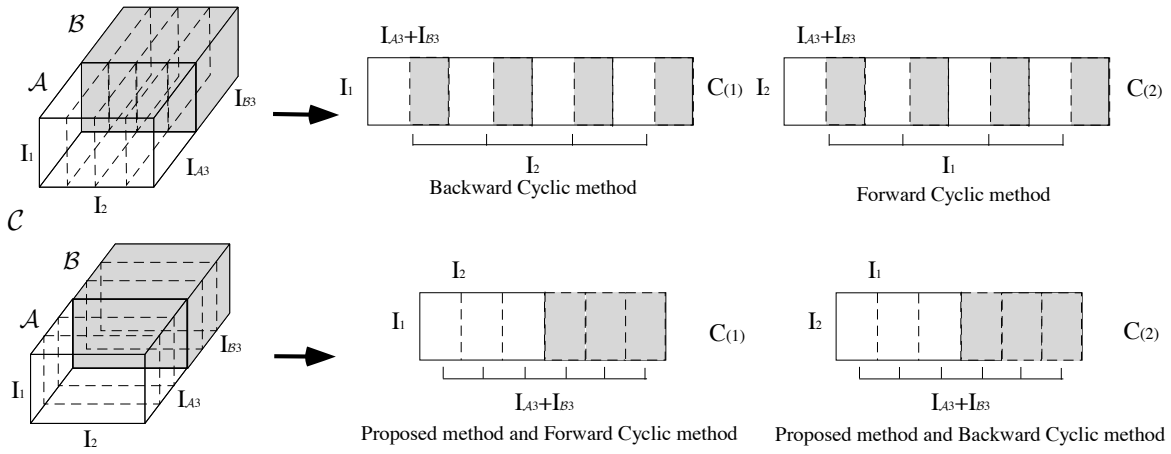


FIGURE 2. Graphic description of the mode-1 and mode-2 unfolding process for different methods. The grey parts are newly added data. Different unfolding methods provides different positions of the newly added data.

is unified in that the new data is placed at the end of the matrix for both mode-1 and mode-2.

Therefore, if $\mathcal{A}$ is the old tensor and $\mathcal{B}$ is the new tensor, their unfolding can be expressed as $C_{(n)} = [A_{(n)}, B_{(n)}]$ for $n = 1, 2, \ldots, N-1$ and $C_{(N)} = \begin{bmatrix} A_{(N)} \\ B_{(N)} \end{bmatrix}$ for the last mode ($n = N$). Due to the multilinear property, the transpose of the data after unfolding is utilized for mode-N.

3.2. **Incremental procedure.** Both incremental processes of SKL and MFPCA algorithms include mean update and total number of samples update, and these updates are defined as follows:

Mean update is given by $\mathcal{M}^{\mathcal{C}} = \frac{I_N^{\mathcal{A}}\mathcal{M}^{\mathcal{A}} + I_N^{\mathcal{B}}\mathcal{M}^{\mathcal{B}}}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}}$, where $\mathcal{M}^{\mathcal{A}}$ and $\mathcal{M}^{\mathcal{B}}$ represent the mean tensors for $\mathcal{A}$ and $\mathcal{B}$, $I_N^{\mathcal{A}}$ and $I_N^{\mathcal{B}}$ define the number of tensors in the old and new tensor sequences, respectively. The number of samples thus becomes $I_N^{\mathcal{C}} = I_N^{\mathcal{A}} + I_N^{\mathcal{B}}$.

Moreover, when new samples are taken into account, the mean value changes, affecting as a consequence the old centered data in the sequence. Such a change should be taken into consideration. Mean value update was first provided in [8], and was then extended in [3], which not only provided explanation for mean update, but also included the concept of "forgetting factor". The so-called forgetting factor gives more weight to recent data over old data. In this study, these aforementioned concepts are extended to a tensor object. To facilitate the understanding of the mathematical foundation of the two incremental procedures, the mean update is described in the ***Proposition*** and the forgetting factor is defined in the ***Corollary*** below. The proof for ***Proposition*** and ***Corollary*** are provided in Sections A of the Appendix.

Proposition 3.1. *Let $\mathcal{M}^{\mathcal{A}}$ be the mean tensor of $\mathcal{A}$, with $\bar{\mathcal{A}}$ being tensor $\mathcal{A}$ after centering. Let $U_{\bar{A}_{(n)}}(i)$ and $\lambda_{\bar{A}_{(n)}}(i)$, $i_n = 1, \ldots, k_{\bar{A}_{(n)}}$ be the largest $k_{\bar{A}_{(n)}}$ eigenvectors and eigenvalues of old unfolding data $\bar{A}_{(n)}$, respectively. A new tensor sequence is denoted as $\mathcal{B}$, with $\bar{\mathcal{B}}$ being tensor $\mathcal{B}$ after centering. Suppose further that $\mathcal{M}^{\mathcal{B}}$ is the mean tensor of $\mathcal{B}$. Then for the incremental modified fast PCA (IMFPCA) algorithm, the mode-n covariance matrix for the whole sequence with mean update can be expressed as:*

$$\varphi_{\bar{C}_{(n)}} = \sum_{i=1}^{k^{A_{(n)}}} \lambda_{\bar{A}_{(n)}}(i) U_{\bar{A}_{(n)}}(i)^T + \bar{B}_{(n)} \bar{B}_{(n)}^T + \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}} \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right) \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right)^T$$

As for the SKL algorithm, the mode-n unfolding of $\bar{\mathcal{B}}$ *with mean update is generated by* $\hat{B}_{(n)} = \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}}} \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right)\right]$ *and the matrix* R *is generated as:*

$$\begin{bmatrix} diag(\lambda_{\bar{A}_{(n)}}) & U_{\bar{A}_{(n)}}^T \hat{B}_{(n)} \\ 0 & \tilde{E}\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}\right) \end{bmatrix},$$

where $\tilde{E} = orth\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}\right)$ *with the orth function being used to orthonormalize the column-wise vectors in the resulting matrix [13].*

Corollary 3.1. *For the same definitions provided in the aforementioned the* ***Proposition*** *with the inclusion of the forgetting factor* f*, the covariance matrix can now be generated as follows:*

$$\begin{aligned} \varphi_{\bar{C}_{(n)}} = f^2 & \sum_{i=1}^{k^{A_{(n)}}} \lambda_{\bar{A}_{(n)}}(i) U_{\bar{A}_{(n)}}(i)^T + \bar{B}_{(n)} \bar{B}_{(n)}^T \\ & + \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}} \left(f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}}\right)}{\left(I_N^{\mathcal{A}} + I_N^{\mathcal{B}}\right)^2} \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right) \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right)^T \end{aligned}$$

for the IMFPCA algorithm.

As for the SKL algorithm, the mode-n unfolding of $\bar{\mathcal{B}}$ *with mean update is generated by* $\hat{B}_{(n)} = \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}}} (M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})\right]$ *while matrix* R *is generated as:*

$$\begin{bmatrix} fdiag(\lambda_{\bar{A}_{(n)}}) & U_{\bar{A}_{(n)}}^T \hat{B}_{(n)} \\ 0 & \tilde{E}\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}\right) \end{bmatrix}$$

where $\tilde{E} = orth\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}\right)$.

Note that the proposition is a special case of the corollary, which considers the forgetting factor as 1. The proposition described earlier applies to Steps 1 through 3 for the IMFPCA algorithm described in Table 2 and to only Step 6 for the SKL algorithm described in Table 3.

In order to achieve better efficiency overall, the IMFPCA and SKL algorithms are applied in accordance to their computational requirements, which means that for mode-1 up to mode-$(N-1)$, IMFPCA is applied; while for the specific mode-N, the SKL algorithm is used instead. Detail analyses of these computational requirements are explored next.

3.3. **Computational complexity.**

3.3.1. *Computational complexity of MFPCA.* In the iterative procedure of the pseudocode of the MFPCA algorithm given in Table 1, the major processing steps and their respective computational requirements are as shown in Table 4.

TABLE 2. Pseudocode implementation of the IMFPCA algorithm

Input: Centered mode-n unfolding matrix $\bar{B}_{(n)}$ of tensor $\mathcal{B}$, mean tensor $\mathcal{M}^{\mathcal{B}}$ for mode-1, ..., mode-$(N-1)$ of $\mathcal{B}$, the last mode $I_N^{\mathcal{B}}$ of tensor $\mathcal{B}$, the last mode $I_N^{\mathcal{A}}$ of previous tensor $\mathcal{A}$, mean tensor $\mathcal{M}^{\mathcal{A}}$ for mode-1, ..., mode-$(N-1)$ of $\mathcal{A}$, eigenvectors $U_{\bar{A}_{(n)}}$ and eigenvalues $\lambda_{\bar{A}_{(n)}}$, number of eigenvectors $k_{\bar{C}_{(n)}}$.

Output: The eigenvectors $U_{\bar{C}_{(n)}}$ of $\bar{C}_{(n)}$ and the eigenvalues $\lambda_{\bar{C}_{(n)}}$, where $\mathcal{C}$ is represented as $\mathcal{C} = [\mathcal{A}, \mathcal{B}]$, $\mathcal{C} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_{N-1} \times (I_N^{\mathcal{A}} + I_N^{\mathcal{B}})}$.

Algorithm:

Step 1: Find the approximate covariance matrix for old data $\varphi_{\bar{A}_{(n)}} = \sum_{i=1}^{k_{A_{(n)}}} \lambda_{\bar{A}_{(n)}}(i) U_{\bar{A}_{(n)}}(i)^T$.

Step 2: Obtain the covariance matrix for the new data $\varphi_{\bar{B}_{(n)}} = \bar{B}_{(n)} \bar{B}_{(n)}^T$.

Step 3: Compute the covariance matrix for whole data with mean update $\varphi_{\bar{C}_{(n)}} = f^2 \varphi_{\bar{A}_{(n)}} + \varphi_{\bar{B}_{(n)}} + \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}} (f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}} (M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})^T$.

Step 4: MFPCA algorithm (in Table 1).

TABLE 3. Pseudocode implementation of the SKL algorithm

Input: Centered mode-n unfolding matrix $\bar{B}_{(n)}$ of tensor $\mathcal{B}$, mean tensor $\mathcal{M}^{\mathcal{B}}$ for mode-1, ..., mode-$(N-1)$ of $\mathcal{B}$, the last mode $I_N^{\mathcal{B}}$ of tensor $\mathcal{B}$, the last mode $I_N^{\mathcal{A}}$ of previous tensor $\mathcal{A}$, mean tensor $\mathcal{M}^{\mathcal{A}}$ for mode-1, ..., mode-$(N-1)$ of $\mathcal{A}$, eigenvectors $U_{\bar{A}_{(n)}}$ and eigenvalues $\lambda_{\bar{A}_{(n)}}$, number of eigenvectors $k_{\bar{C}_{(n)}}$.

Output: The eigenvectors $U_{\bar{C}_{(n)}}$ of $\bar{C}_{(n)}$ and the eigenvalues $\lambda_{\bar{C}_{(n)}}$, where $\mathcal{C}$ is represented as $\mathcal{C} = [\mathcal{A}, \mathcal{B}]$, $\mathcal{C} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_{N-1} \times (I_N^{\mathcal{A}} + I_N^{\mathcal{B}})}$.

Algorithm:

Step 1: Generate data $\hat{B}_{(n)} = \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}}} (M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}) \right]$.

Step 2: Calculate $D = U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}$.

Step 3: Projection of new data $E = \hat{B}_{(n)} - U_{\bar{A}_{(n)}} D$.

Step 4: Calculate the orthogonal basis by QR decomposition $\tilde{E} = qr(E)$, where $\tilde{E}$ is an orthogonal matrix.

Step 5: Construct matrix $R = \begin{bmatrix} fdiag(\lambda_{\bar{A}_{(n)}}) & U_{\bar{A}_{(n)}} \\ 0 & \tilde{E}\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}\right) \end{bmatrix}$.

Step 6: Calculate $SVD(R) = \tilde{U}\tilde{V}\tilde{Q}^T$.

Step 7: Calculate $U = \left[U_{\bar{A}_{(n)}}, \tilde{D}^{(n)} \right] \tilde{U}$.

Step 8: Define $\lambda_{\bar{C}_{(n)}}(i) = \tilde{V}(i,i)$ and $U_{\bar{C}_{(n)}}(i) = U(i)$, where $i = 1, \ldots, k_{\bar{C}_{(n)}}$.

In these operations, i takes on the values from 1 to k. Therefore, with the given relation $1^2 + 2^2 + 3^2 + \ldots + k^2 = k(k+1)(2k+1)/6$, the order of complexity in the number of operations can thus be approximated for all L iterations as $O(I_n^2 L) + O(I_n k^3 L)$.

3.3.2. *Computational complexity of the IMFPCA.* The major processing steps with their respective computational complexities are given in Table 5.

These results followed the same reasoning used for finding the order of complexity in the number of operations for the MFPCA.

TABLE 4. Computational complexity of the MFPCA for a single iteration

Processing steps for the i^{th} eigenvector	Computational Complexity
Calculate $U(i) = \varphi * U(i)$	$O(I_n^2)$
Gram-Schmidt process	$O(I_n i^2)$
Calculate the norm $\lambda(i)$ of $U(i)$	$O(I_n^2)$

TABLE 5. Computational complexity for steps in IMFPCA

Major processing steps in IMFPCA	Computational Complexity
Approximate covariance matrix for $\bar{A}_{(n)}$	$O\left(I_n^2 k_{\bar{A}_{(n)}}\right)$
Covariance matrix for $\bar{B}_{(n)}$	$O(I_n I_1 I_2 \dots I_{N-1} I_N^{\mathcal{B}})$
Covariance matrix for $\bar{C}_{(n)}$	$O(I_n I_1 I_2 \dots I_{N-1})$
Computational steps in MFPCA	$O(I_n^2 L) + O\left(I_n k_{\bar{C}_{(n)}}^3 L\right)$

3.3.3. *Computational complexity of the SKL.* Similarly, the major processing steps of SKL with their computational complexities are as shown in Table 6.

TABLE 6. Computational complexity for steps in SKL

Major processing steps in SKL	Computational Complexity
Calculate $D = U_{\bar{A}_{(n)}}^T \hat{B}_{(n)}$	$O(k_{\bar{A}_{(n)}} I_1 I_2 \dots I_{N-1} I_N^{\mathcal{B}})$
Projection of new data $E = \hat{B}_{(n)} - U_{\bar{A}_{(n)}} D$	$O(k_{\bar{A}_{(n)}} I_1 I_2 \dots I_{N-1} I_N^{\mathcal{B}})$
Calculate the orthogonal basis by QR decomposition $\tilde{E} = qr(E)$	$O(I_n I_1 I_2 \dots I_{N-1} I_N^{\mathcal{B}})$
Suppose $S = \min(I_n, I_1 I_2 \dots I_{n-1} I_{n+1} \dots I_N)$, caculate $SVD(R) = \tilde{U}\tilde{V}\tilde{Q}^T$	$O(I_n I_1 I_2 \dots I_{N-1 I_N^{\mathcal{B}}}) + O((k_{\bar{A}_{(n)}} + S)^3)$

From Table 5 and Table 6, we can observe that the total number of operations for the IMFPCA algorithm can be approximated by $O(I_n^2(k_{\bar{A}_{(n)}} + L)) + O\left(I_n I_1 I_2 \dots I_N I_N^{\mathcal{B}}\right)$, while the total number of operations for SKL can be approximated by $O((k_{\bar{A}_{(n)}} + S)^3) + O((k_{\bar{A}_{(n)}} + I_n) I_1 I_2 \dots I_N I_N^{\mathcal{B}})$. Since the number $k_{\bar{A}_{(n)}}$ is usually less than 15, and L is empirically observed to be less than 10, the two computational complexities can be simplied into $O(I_n^2) + O(I_n I_1 I_2 \dots I_N I_N^{\mathcal{B}})$ and $O(S^3) + O(I_n I_1 I_2 \dots I_N I_N^{\mathcal{B}})$. Therefore, for mode-1 up to mode-$(N-1)$, it is obvious that IMFPCA outperforms SKL (in this case, $S = I_n$).

It is important to note that for the last mode ($n = N$), the dimensional parameters need to be changed to estimate the computational complexity. Suppose the parameters for the dimensions are $I_1^{old}, I_2^{old}, \dots, I_{N-1}^{old}, I_N^{old}$, then the new parameters that are used for evaluating the computational complexity are defined as $I_N^{new} = I_1^{old} I_2^{old} \dots I_{N-1}^{old}$, $I_1^{new}, I_2^{new}, \dots, I_{N-2}^{new} = 1$ and $I_{N-1}^{new} = I_N^{old}$. The proof for this required change is provided in Section A.2 of the Appendix. The computational complexity for the IMFPCA is thus far more simplified than that of the SKL algorithm (in this case, $S \ll I_N$).

For incremental learning, these theoretical results support the use of IMFPCA for the mode-1 up to mode-$(N-1)$ and the use of SKL solely for mode-N. Following these

theoretical findings, empirical results are provided in the next section to verify the effectiveness of IMFPCA in a series of tests and the efficiency of the proposed incremental method for face recognition as a real-world application.

4. **Experimental Results.** Several experiments were designed to assess the merit of the modified fast PCA (MFPCA) with its efficient incremental procedure. The key aspects of the algorithm investigated include:
1) Performance evaluation for finding leading eigenvectors of the MFPCA algorithm in comparison to FPCA for face recognition application;
2) Accuracy obtained from the incremental learning procedure, contrasting the results of both IMFPCA and SKL methods;
3) Processing time required for the incremental subspace update under different unfolding modes between IMFPCA and SKL;
4) Effectiveness of the proposed incremental method for online face recognition application.

In the experiments used for computational performance and accuracy evaluations provided in Sections IV.A and IV.B, the AT&T database of faces [23] and the MNIST database of handwritten digits [24] are used. The AT&T database of faces contains images with dimensions 112×92 and 360 images were included; MNIST handwritten digit database contains images with dimensions 28×28, and 1000 images were considered. All the experiments were evaluated using MATLAB on a Windows Vista-based PC with Intel Core 2 1.60GHz and 2G RAM.

4.1. **Performance of the modified fast PCA (MFPCA).** To test the modified method, images in the AT&T database are normalized into different dimensions 30×30, 40×40 and 50×50, respectively. Then, all images are reshaped into a vector, and matrices are generated. After centering the matrix and finding the covariance matrix, covariance matrices of dimensions 900×900, 1600×1600 and 2500×2500 are obtained. The comparative results, in terms of both accuracy in estimating the eigenvectors and computational requirements between the Eigenvalue Decomposition (EVD), FPCA and MFPCA, are given.

Figure 3 and Figure 4 show that MFPCA provides better similarity than the fast PCA under different parameter settings as function of error tolerance and the number of dimensions used. However, the defect of the fast PCA, which can be seen in the eighth eigenvectors for all tests, cannot be overcome by the modified PCA. Although the eigenvalues are in descending order, the local minimum still shows up and gives a low similarity to EVD eigenvector. However, since the obtained eigenvector satisfied the iterative requirement, the eigenvectors after the eight one still can have a good similarity with the EVD eigenvectors. Moreover, under the same error tolerance, with the increase of the dimension, the accuracy increases, too. As can be seen in Table 7, FPCA and MFPCA have comparable processing time, both of which are significantly faster than EVD. The entry in Table 7, with the image size of 30×30 and error tolerance of $\varepsilon = 10^{-5}$, is the only case where the MFPCA is faster than FPCA. The reason that less iterations were required to achieve the error tolerance requirement is a purely coincidental case based on the 100 random trials considered.

A face recognition application [25] was conducted to evaluate the performance of the modified fast PCA (MFPCA) method in search of leading eigenvectors. For the face images selected from the AT&T database, nine randomly picked images of one subject are used as feature bases, and the one image remaining is then used for testing. The number of the eigenvectors is set to be nine, with the consideration that the eigenface

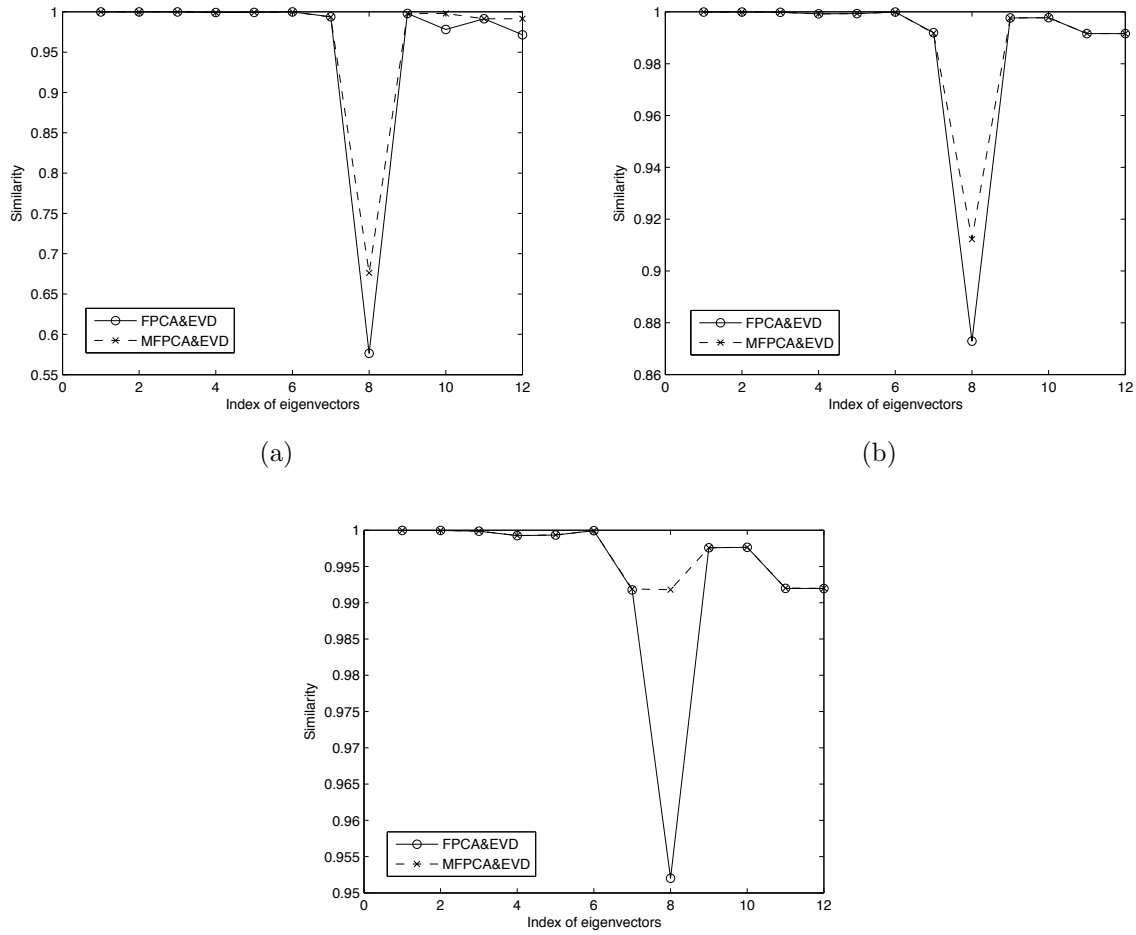


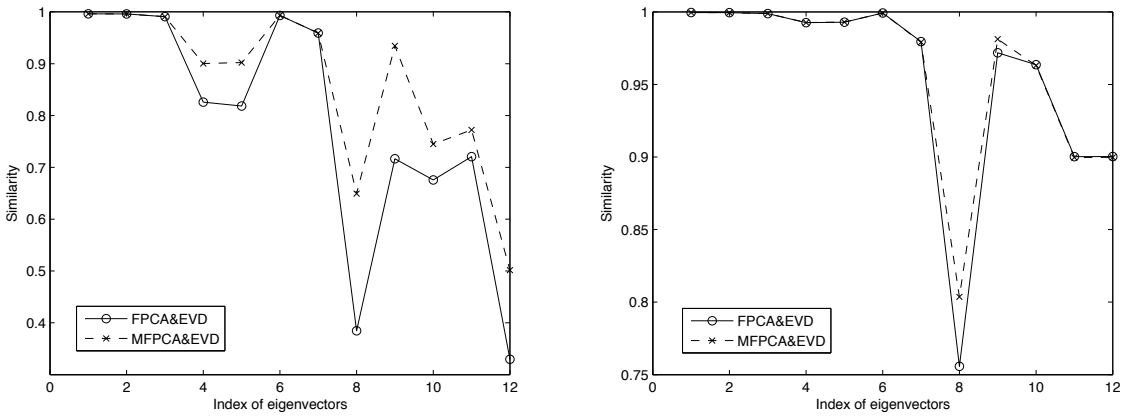
FIGURE 3. Similarity comparison between eigenvectors from FPCA, MFPCA and EVD methods with dimension changes for the same error tolerance $\varepsilon = 10^{-5}$, and those subfigures are: (a) 30×30; (b) 40×40 and (c) 50×50. The curve is from the average over 100 trials.

TABLE 7. Processing time comparison

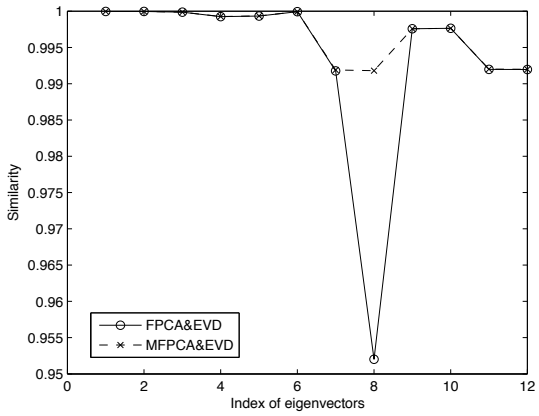
Image Size	EVD	FPCA $\varepsilon = 10^{-3}$	MFPCA $\varepsilon = 10^{-3}$	FPCA $\varepsilon = 10^{-4}$	MFPCA $\varepsilon = 10^{-4}$	FPCA $\varepsilon = 10^{-5}$	MFPCA $\varepsilon = 10^{-5}$
30×30	6.4860s	0.6221s	0.7758s	1.1759s	1.2575s	1.8219s	1.7588s
40×40	39.6555s	1.8825s	2.4675s	3.4587s	3.6353s	5.2412s	5.2497s
50×50	165.1443s	4.9189s	6.5651s	7.6038s	7.9852s	13.0582s	13.2501s

algorithm can achieve the highest recognition accuracy with nine eigenvectors retained. The results shown in Table 8 indicate that the minimum accuracy for the MFPCA is always higher than the minimum accuracy for the FPCA, while the maximum being either higher or equal to that of the FPCA. The average recognition rate of MFPCA was thus better than FPCA.

4.2. **Accuracy for the incremental procedure.** There are two sets of experiments conducted in this section. First, the similarity between the first four eigenvectors from SKL and IMFPCA, which is obtained by dot product, is ascertained. Second, the subspace



(a) (b)



(c)

FIGURE 4. Similarity comparison between eigenvectors from FPCA, MFPCA and EVD methods with error tolerance changes for the dimension 50×50, and those subfigures are: (a) $\varepsilon = 10^{-3}$, (b) $\varepsilon = 10^{-4}$ and (c) $\varepsilon = 10^{-5}$. The curve is from the average over 100 trials.

TABLE 8. Comparison of recognition rates (%)

Methods	EVD	FPCA $\varepsilon = 10^{-5}$	MFPCA $\varepsilon = 10^{-5}$	FPCA $\varepsilon = 10^{-4}$	MFPCA $\varepsilon = 10^{-4}$	FPCA $\varepsilon = 10^{-3}$	MFPCA $\varepsilon = 10^{-3}$
Recognition Rate	95.00	94.70 [90, 95]	95.00 [95, 95]	93.90 [90, 95]	94.88 [92.5, 95]	93.33 [85, 95]	93.90 [90, 97.5]

Note: Z is the mean and [X, Y] is the the lowest and highest recognition rate for 100 trials.

distance among SKL, IMFPCA and batch mode PCA is determined. The forgetting factor is set to be one in this test. The MNIST dataset, due to its large size, allows for more iterations to be tested. Moreover, the image size (28×28) of MNIST dataset is found to be more computationally suitable, since the dimension of the covariance matrix for mode-3 in the AT&T dataset (112×92) were considered unjustifiably large for the same tests that were considered.

The parameters for the dataset are set as follows: (a) PCA: $k = 4$, (b) SKL: $k = 8$ and (c) IMFPCA: $k = 8$. Only the first four eigenvectors are used in the test, the remaining four eigenvectors for the incremental algorithms are used to minimize the error. The batch

number for MNIST is 30. A different error tolerance of $\varepsilon = 10^{-7}$ is chosen to ensure that the incremental algorithm used will provide more accurate results.

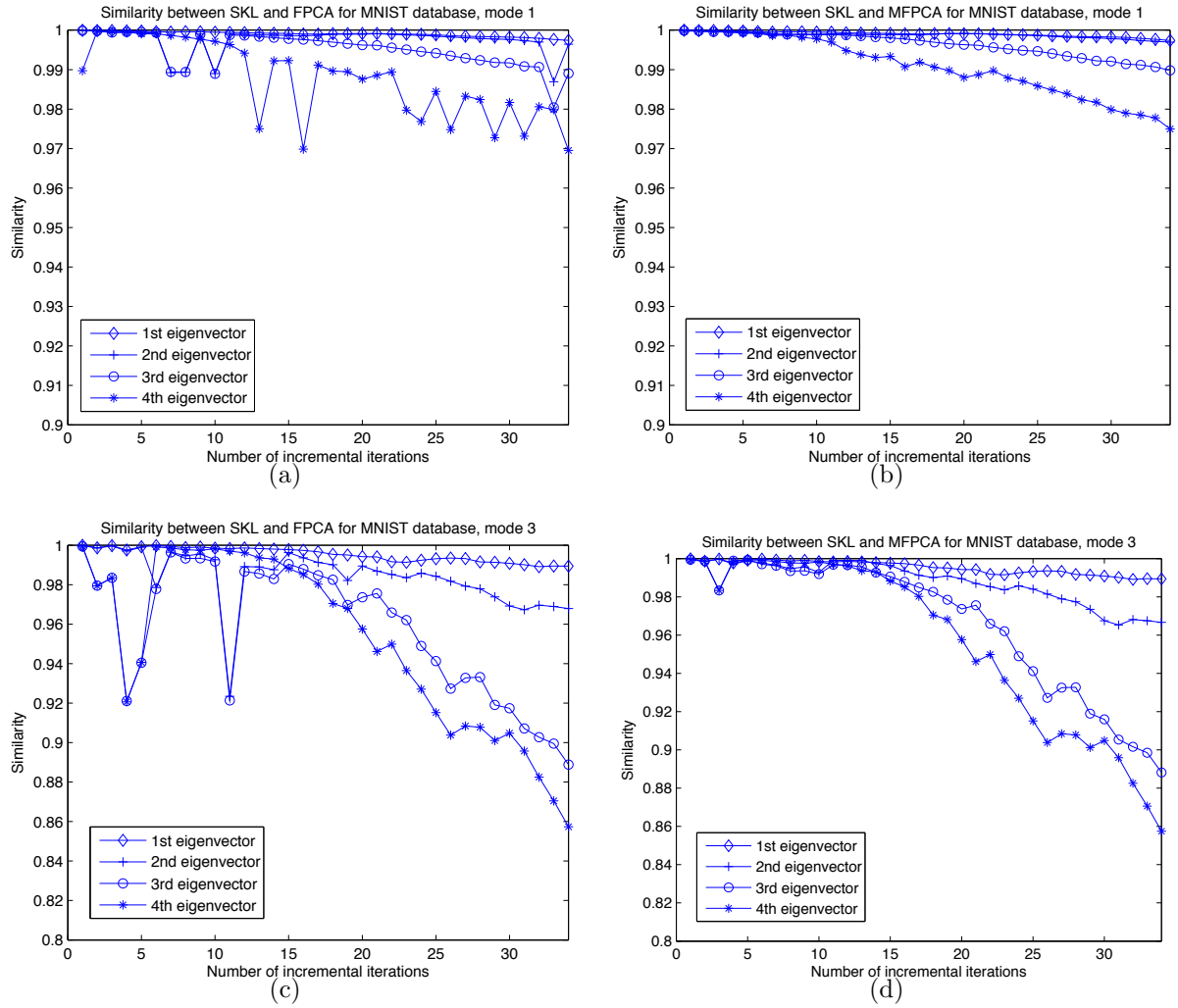


FIGURE 5. Similarity between eigenvectors from SKL and IMPCA and from SKL and IMFPCA using the MNIST database with mode-1 unfolding, and mode-3 unfolding, respectively. The x-axis is used for the number of updates, and the y-axis is used for the similarity measure. For better visualization, different scales were adopted for different tests; however, the same test with different algorithms is visualized with the same scale.

Since mode-2 is similar to mode-1, only mode-1 results were shown for simplicity sake. From the similarity results in Figure 5, it is obvious that IMFPCA performs better than IFPCA, and with smoother curves or transitions. Some similarities of the FPCA algorithm, as described in Section 4.1, drop to a low value where a local minimum happens; however, it satisfied the criteria of minimizing the mean square error, which can still allow for the eigenvectors estimated after the local minimum to achieve a higher similarity. Moreover, although the final similarity values are close among the different algorithms, the IMFPCA is the more consistent in estimating the eigenvectors over many trials, as can be clearly seen in Figure 5. This outcome is significant and will yield better accuracy in the real-world applications, such as incremental face recognition provided in this study.

It is also essential to carry out experiments involving the use of IMFPCA with different error tolerance settings, as evidenced in the results provided below.

The error accumulated along the incremental procedure brings up the discrepancy between the incremental results and the true results (from batch methods). This defect is unavoidable for incremental procedures. A distance measure based on the principal angles in [26] between subspaces was used here to examine the nature of this discrepancy. The distance is expressed by $d(U_0, U_1) = \sqrt{\sum_{i=1}^{k} \theta_i^2}$ where U_0 and U_1 are k-dimensional subspaces and $\theta_1, \theta_2, \ldots, \theta_k$ are the principal angles between them. Given $\theta_i \in \left[0, \frac{\pi}{2}\right]$, the distance satisfies $d \in \left[0, \frac{\sqrt{k}\pi}{2}\right]$. If the two subspaces are identical, then $d \longrightarrow 0$. It can be seen in Figure 6 that the subspace distance converges to the true result gradually.

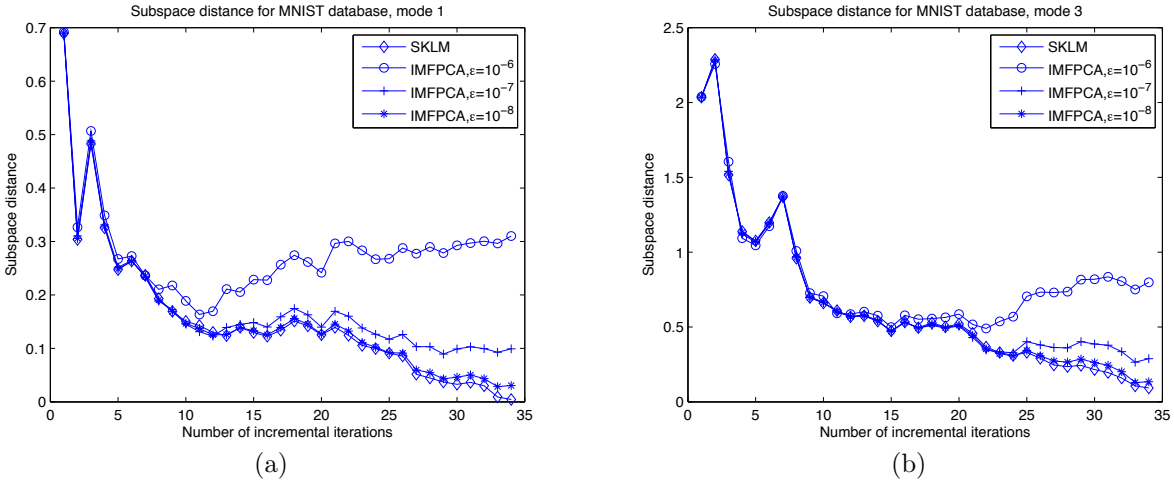


FIGURE 6. Subspace distance examination on MNIST database. (a) Mode-1: SKL, IMFPCA with $\varepsilon = 10^{-6}$, IMFPCA with $\varepsilon = 10^{-7}$ and IMFPCA with $\varepsilon = 10^{-8}$. (b) Mode-3: SKL, IMFPCA with $\varepsilon = 10^{-6}$, IMFPCA with $\varepsilon = 10^{-6}$ and IMFPCA with $\varepsilon = 10^{-6}$. And the curve is the average results from 100 trials.

In Figure 6, the SKL algorithm is proven to perform better in this case than IMFPCA in all tests. With error tolerance of 10^{-6} for both mode-1 and mode-3, IMFPCA had the distance increased after ten iterations, which means the error tolerance cannot be adopted. However, if the error tolerance is set at a lower scale, such as 10^{-7} and 10^{-8}, then the distance can be considered as a stable discrepancy from the true result (which is the result from eigen decomposition based on the whole dataset). The lower error tolerance leads to better convergence. It is worth noticing that the subspace distance for mode-1 is smaller than mode-3 (take error tolerance 10^{-7}, for example, where the final subspace distance for mode-1 was 0.0991 and for mode-3 was 0.2888), which does not conform to the conclusion made from the experiment described in Section 4.1, "under same error tolerance, with the increase of the dimension, the accuracy increases, too". The reason is that in the incremental procedure, the dimension for mode-1 is 28, while the dimension for mode-3 is 784, and both of them used the first eight eigenvectors. Obviously, for the larger dimension mode, the error was higher. From this test, it can be concluded that both SKL and IMFPCA methods can provide acceptable discrepancy distance in the incremental procedure with proper parameter settings.

4.3. **Processing time for the incremental procedure.** The processing times of SKL and IMFPCA for the incremental procedure are compared next. It can be observed that for mode-1 unfolding, IMFPCA outperforms SKL, while for mode-3 unfolding, it is the SKL algorithm that outperforms IMFPCA.

TABLE 9. The total incremental learning processing time for different modes unfolding corresponding to Figure 6

figure and ε value		$(a)10^{-6}$	$(a)10^{-7}$	$(a)10^{-8}$	$(b)10^{-6}$	$(b)10^{-7}$	$(b)10^{-8}$
Processing time	IMFPCA	0.3340s	0.3927s	0.4607s	14.1518s	18.3362s	23.7908s
	SKL	11.1481s			0.4938s		

In Table 9, it shows that for mode-1, IMFPCA performs faster while for mode-3, the SKL is faster, regardless of the error tolerance. Those results validate the discussion on the computational complexity in Section 3.3 and it shows the merit of the proposed algorithm.

4.4. **Real-world application for the online face recognition.** The face image dataset used in this experiment was collected from the face recognition database of University of Essex [27], the Georgia Tech face database [28] and the AT&T the database of faces [23]. The dataset considered in this experiment is composed of 240 subjects with 10 images each. All the images are preprocessed to include only the face and normalized into 168×118. For some subjects, the images were taken at different times, with different lighting conditions, different facial expressions (open/closed eyes, smiling/not smiling, etc.) and different facial details (glasses/no glasses, etc.). Figure 7 provides sample images from the database as illustrative examples.

FIGURE 7. Sample images from the dataset used for the incremental face recognition experiment

Different from the traditional face recognition system with a certain training set, the training set of the online face recognition increases as more faces are introduced. As one application of the proposed algorithm, a simple system is designed with an arbitrary threshold and without the verification step. To make the system more robust, usually adaptive threshold and verification are suggested. The flow chart for the incremental face recognition is shown in Figure 8.

The so-called L-2 norm distance is used for recognition. In the test, a threshold is given to evaluate if this image is already in the accumulated data. The projection of the q^{th} image on mode-n best $k_{\bar{A}_{(n)}}$ eigenvectors $U_{\bar{A}_{(n)}}$ is given as follows: $Y_{q(n)} = \left(A_q - \bar{A}\right) \times_n U_{\bar{A}_{(n)}}^T$ for $n < N$. For storage and computational convenience, the projection of each image under each mode is rearranged into a vector $y_{q(n)} \in \mathcal{R}^{l \times 1}$. To classify the test image

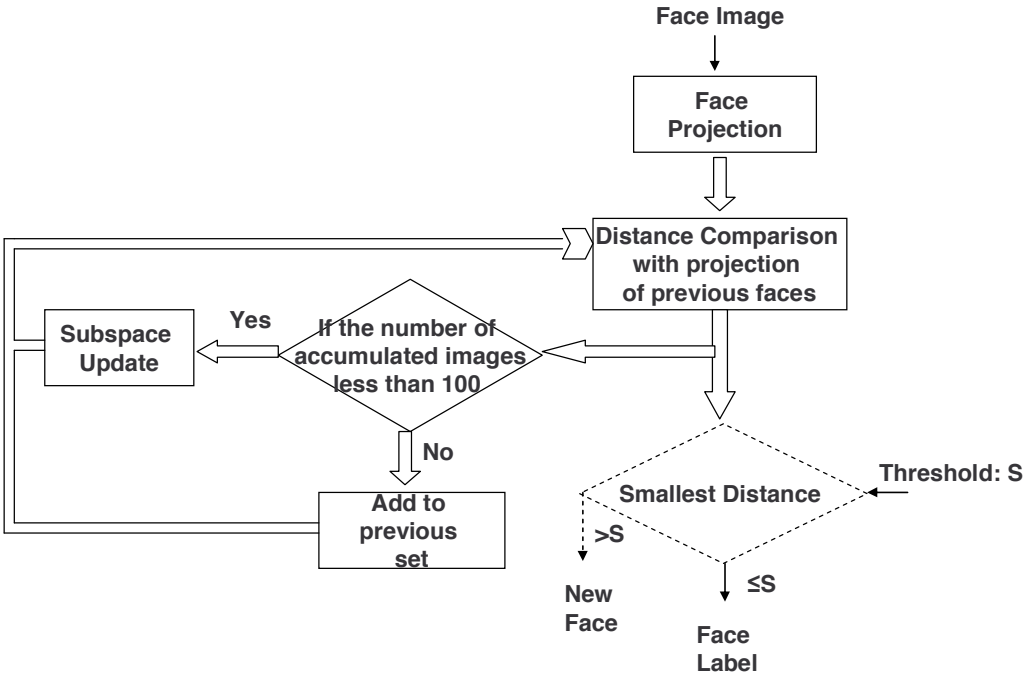


FIGURE 8. Flow chart for the incremental face recognition procedure

with the projection $y_{t(n)}$, the norm between projections of the two images is determined as $d(q) = \sum_{n=1}^{N} ||y_{t(n)} - y_{q(n)}||$, where $y_{q(n)}$ represents the projection of mode-n unfolding of the q^{th} image in in the accumulated set. The index of $\min(d)$ gives the preliminary recognition result in the accumulated data for the testing image. Then it is assumed that when the distance is less than the threshold, the face is associated to a person in the accumulated data. Otherwise, this person will be considered as a new subject who is not yet included in the accumulated images.

Since there are too many combinations among k_1, k_2 and k_3, in our case, we just select $k_1 = 2$, $k_2 = 3$, $k_3 = 15$ randomly among many other combinations that can be selected. It provides an example of the incremental face recognition experiments, where the batch number is set at 100. This batch number is used to indicate that after every 100 images, the eigenvectors can be updated incrementally (a different number could also be used, keeping in mind that the larger this number is, the less iterations there would be and the less meaningful would be the comparison between the two methods).

The threshold [29-32] is chosen based on the following steps. The first 200 images are used as the basis for the incremental learning with their identification. First, compute the eigenvectors of the first 100 images. Second, perform the recognition process on the other 100 images using their previous images as the training pool. Third, record the distance of the 100 images from the recognition process. Fourth, determine the threshold that will be used for deciding whether the new image is in the previous image data. Since we have two classes, the optimal approach to select a threshold in bimodal histogram is used as detailed in Section A.3 of the Appendix.

With the conditions set above, the threshold is determined as 2.26×10^7. And the incremental accuracy is determined by $\frac{N_c}{N}$, where N_c is the number of images that are recognized correctly and N is the accumulated number of images that were used for incremental recognition.

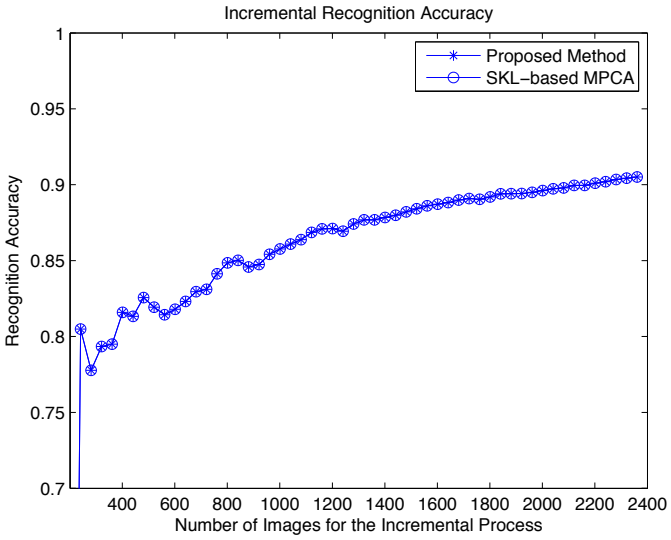


FIGURE 9. Incremental face recognition accuracy

From the recognition accuracy results given in Figure 9, the two methods provide identical performance in terms of recognition rate. However, when comparing the processing time, the proposed subspace update method has faster speed of ***85.3325s*** than the SKL-based incremental multilinear PCA of ***285.9654s***.

Table 10 below provides evidence that the selection of the values of k_1, k_2 and k_3 can be randomly made and the results will always show that the proposed method is faster for the same accuracy than the SKL-based method.

TABLE 10. Processing time and accuracy for different parameter settings

	Proposed/SKL	$k_1 = 2$, $k_2 = 2$	$k_1 = 2$, $k_2 = 4$	$k_1 = 4$, $k_2 = 2$	$k_1 = 4$, $k_2 = 4$
$k_3 = 10$	Time	86.3s/286.0s	84.8s/289.5s	86.2s/289.6s	87.2s/298.4s
	Accuracy	88.1%/88.1%	89.7%/89.7%	89.3%/89.3%	90.2%/90.2%
$k_3 = 20$	Time	86.6s/288.7s	87.5s/291.3s	88.2s/294.5s	87.0s/292.0s
	Accuracy	89.2%/89.2%	90.1%/90.1%	89.8%/89.8%	90.6%/90.6%

It should be noted that the proposed method focuses on improving the computational efficiency for the incremental multilinear PCA. There are already many algorithms provided that can improve the recognition accuracy such as LDA, ICA, Kernal PCA and other PCA related algorithms [33-35]. For those cases, when the incremental multilinear process is necessary, the proposed method can also be adopted to reduce the computational complexity without loss of accuracy.

5. **Conclusion.** This study introduced a new integrated incremental multilinear method that combines an incremental modified fast PCA (IMFPCA) and the sequential Karhunen-Loeve (SKL) that was tested for incremental face recognition. The incremental accuracy and processing time for IMFPCA and SKL were thoroughly examined. Mathematical derivations prove the advantage of fast processing speed when combining these two methods without compromising accuracy. The key aspect of this research is in establishing an effective method for estimating leading eigenvectors in an accurate and computationally efficient way as a critical first step. The results were supported by experimental evaluations involving different tests on the AT&T face database and MNIST database. The proposed method was tested for incremental face recognition using all three face databases

(the AT&T database, the database from University of Essex and the face database from Georgia Institute of Technology). The results were obtained with a much faster speed than traditional SKL while achieving the same recognition accuracy rate in the incremental process. These improvements in results show that accurate estimation of the leading eigenvectors as a critical first step is a meaningful contribution that led to the development of this new method for incremental learning.

Acknowledgment. The authors appreciate the support provided by the National Science Foundation under grants HRD-0833093, CNS 0959985 and CNS 1042341. The authors are also thankful to the FIU Graduate School for the dissertation year fellowship and the scholarship from the Cultural and Educational Scholarship Foundation of Chinese Women's Club of Greater Miami both provided to Ms. Jin Wang, and to all the colleagues who helped in providing us with the AT&T database of faces, the face recognition data from the University of Essex, the Georgia Tech face database and the MNIST database.

REFERENCES

[1] G. H. Golub and C. Reinsch, Singular value decomposition and least squares solutions, *Numer. Math.*, vol.14, no.5, pp.403-420, 1970.
[2] A. Levy and M. Lindenbaum, Sequential Karhunen-Loeve basis extraction and its application to images, *IEEE Trans. Image Process.*, vol.9, no.8, 2002.
[3] D. Ross, J. Lim, R. Lin and M. Yang, Sequential Karhunen-Loeve basis extraction and its application to images, *Int. J. of Comput. Vision*, 2007.
[4] H. Zhao, P. C. Yuen and J. T. Kwok, A novel incremental principal component analysis and its application for face recognition, *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol.36, no.4, 2006.
[5] T. Chin and D. Suter, Incremental kernel principal component analysis, *IEEE Trans. Image Process.*, vol.16, no.6, 2006.
[6] L. Hoegaertsa, L. de Lathauwerb, I. Goethalsa, J. A. K. Suykensa, J. Vandewallea and B. de Moora, Efficiently updating and tracking the dominant kernel principal components, *Neural Netw.*, vol.20, no.2, pp.220-229, 2007.
[7] S. Bartelmaos and K. Abed-Meraim, Fast principal component extraction using givens rotations, *IEEE Signal Processing Letters*, vol.15, pp.369-372, 2008.
[8] J. Wang, A. Barreto, L. Wang, Y. Chen, N. Rishe, J. Andrian and M. Adjouadi, Multilinear principal component analysis for face recognition with fewer features, *Neurocomputing*, vol.73, no.10-12, pp.1550-1555, 2010.
[9] H. Wang and N. Ahuja, A tensor approximation approach to dimensionality reduction, *Int. J. of Comput. Vision*, vol.76, no.3, pp.217-229, 2008.
[10] H. Lu, K. N. Plataniotis and A. N. Venetsanopoulos, MPCA: Multilinear principal component analysis for tensor objects, *IEEE Trans. Neural Netw.* vol.19, no.1, pp.18-39, 2008.
[11] X. Li, W. Hu, Z. Zhang, X. Zhang and G. Luo, Robust visual tracking based on incremental tensor subspace learning, *IEEE the 11th Int. Conf. on Comput. Vision*, 2007.
[12] J. Sun, C. E. Tsourakakis, E. Hoke, C. Faloutsos and T. Eliassi-Rad, Two heads better than one: Pattern discovery in time-evolving multi-aspect data, *Data Mining and Knowledge Discovery*, vol.17, no.1, pp.111-128, 2008.
[13] S. Ozawa, S. Pang and N. Kasabov, Incremental learning of chunk data for online pattern classification systems, *IEEE Trans. Neural Netw.*, vol.19, no.6, pp.1061-1074, 2008.
[14] P. Hall, D. Marshall and R. Martin, Adding and subtracting eigenspaces with eigenvale decomposition and singular value decomposition, *Image and Vision Computing*, vol.20, no.13-14, pp.1009-1016, 2002.
[15] G. H. Golub and C. F. van Loan, *Matrix Computation*, Johns Hopkins University Press, 1996.
[16] K. M. Reddy and T. J. Herron, Computing the eigen decomposition of a symmetric matrix in fixed-point algorithms, *IEEE Bangalore Section, The 10th Annual Symposium on Multimedia Communications and Signal Processing*, pp.23-24, 2001.
[17] R. J. Schilling and S. L. Harris, *Applied Numerical Methods for Engineers Using Matlab and C*, Brooks/Cole Publishing Company, 2000.

[18] A. Sharma and K. K. Paliwal, Fast principal component analysis using fixed-point algorithm, *Pattern Recognition Letters*, vol.28, pp.1151-1155, 2007.
[19] A. Hyväinen and E. Oja, A fast fixed-point algorithm for independent component analysis, *Neural Comput.*, vol.9, no.7, pp.1483-1492, 1997.
[20] G. H. Bakir, B. Scholkopf and J. Weston, On the pre-image problem in kernel methods, *Kernel Methods in Bioengineering, Signal and Image Processing*, pp.284-302, 2007.
[21] V. Berinde, *Iterative Approximation of Fixed Points*, Springer, 2007.
[22] L. de Lathauwer, B. de Moor and J. Vandewalle, A multilinear singular value decomposition, *SIAM J. Matrix and Application*, vol.12, no.4, pp.1253-1278, 2000.
[23] AT&T, *The Database of Faces*, http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[24] *MNIST Database of Handwritten Digits*, http://yann.lecun.com/exdb/mnist.
[25] M. Turk and A. Pentland, Eigenfaces for recognition, *J. Cognitive Neuroscience*, vol.3, no.1, pp.71-86, 1991.
[26] A. V. Knyazev and M. E. Argentati, Principal angles between subspaces in an a-based scalar product: Algorithms and perturbation estimates, *SIAM J. Sci. Comput.*, vol.23, no.6, pp.2009-2041, 2002.
[27] *Face Recognition Data*, University of Essex, UK, http://cswww.essex.ac.uk/mv/allfaces/index.html.
[28] *Georgia Tech Face Database*, http://www.anefian.com/face reco.htm.
[29] M. Adjouadi, C. Reyes, P. Vidal and A. Barreto, An analytical approach to signal reconstruction using gaussian approximations applied to randomly generated and flow cytometric data, *IEEE Transactions on Signal Processing*, vol.48, no.10, pp.2839-2849, 2000.
[30] F. Candocia and M. Adjouadi, A similarity measure for stereo feature matching, *IEEE Transactions on Image Processing*, vol.6, no.10, pp.1460-1464, 1997.
[31] M. Adjouadi and F. Candocia, A stereo matching paradigm-based on the walsh transformation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.16, no.12, pp.1212-1218, 1994.
[32] M. Adjouadi, A man-machine vision interface for sensing the environment, *Journal of Rehabilitation Research and Development*, vol.29, no.2, pp.57-76, 1992.
[33] S. D. Lin, J. Lin and C. Chiang, Combining scale invaraint feature transform with principal component analysis in face recognition, *ICIC Express Letters*, vol.3, no.4(A), pp.927-932, 2009.
[34] C. Chang and H. Hse, Application of principal component analysis to radial-basis function committee machine for face recognition, *International Journal of Innovative Computing, Information and Control*, vol.5, no.11(B), pp.4145-4154, 2009.
[35] J. Li, J. Pan and Z. Lu, Kernel optimization-based discriminant analysis for face recognition, *Neural Computing and Applications*, vol.18, no.6, pp.603-612, 2009.

Appendix A.

A.1. Proof of the proposition and the corollary. Since the proposition is considered as a special case of the corollary when the forgetting factor is 1, the proof for the corollary will be analyzed first.

This proof is to show the theoretical derivation of equation

$$\begin{aligned}\varphi_{\bar{\mathcal{C}}_{(n)}} =& f^2 \sum_{i=1}^{k_{A_{(n)}}} \lambda_{\bar{A}_{(n)}}(i) U_{\bar{A}_{(n)}}(:,i) U_{\bar{A}_{(n)}}(:,i)^T + \bar{B}_{(n)} \bar{B}_{(n)}^T \\ &+ \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}} (I_N^{\mathcal{A}} + f^2 I_N^{\mathcal{B}})}{\left(I_N^{\mathcal{A}} + I_N^{\mathcal{B}}\right)^2} \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right) \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right)^T\end{aligned}$$

for IMFPCA and equation

$$\hat{B}_{(n)} = \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}} (I_N^{\mathcal{A}} + f^2 I_N^{\mathcal{B}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2}} (M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}) \right]$$

and

$$R^{(n)} = \begin{bmatrix} f diag(\lambda_{\bar{A}_{(n)}}) & U_{\bar{A}_{(n)}} \\ 0 & \tilde{B}_{(n)} \left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)T}} \hat{B}_{(n)} \right) \end{bmatrix},$$

where $\tilde{B}_{(n)} = orth\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}} U_{\bar{A}_{(n)}}\right)$ for SKL with the forgetting factor f included.

The mean tensor for a tensor sequence $\mathcal{A} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_N^{\mathcal{A}}}$ is defined as $\mathcal{M}^{\mathcal{A}} = \frac{1}{I_N^{\mathcal{A}}} \sum_{i=1}^{I_N^{\mathcal{A}}} \mathcal{A}(:, \ldots, i)$ where $\mathcal{M}^{\mathcal{A}} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_{N-1}}$. Similarly, for the new tensor sequence $\mathcal{B} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_N^{\mathcal{B}}}$, the mean computed in the same fashion as for $\mathcal{A}$ is $\mathcal{M}^{\mathcal{B}} \in \mathcal{R}^{I_1 \times I_2 \ldots \times I_{N-1}}$.

The centered tensor sequences are given by $\bar{\mathcal{A}}(:, \ldots, i) = \mathcal{A}(:, \ldots, i) - \mathcal{M}^{\mathcal{A}}$ and $\bar{\mathcal{B}}(:, \ldots, i) = \mathcal{B}(:, \ldots, i) - \mathcal{M}^{\mathcal{B}}$, where $i = 1, \ldots, I_N^{\mathcal{A}}$ or $I_N^{\mathcal{B}}$. Then a new tensor sequence $\mathcal{C}$ is generated as $\mathcal{C} = [\mathcal{A}, \mathcal{B}]$ to yield $\mathcal{C} \in \mathcal{R}^{I_1 \times \ldots \times (I_N^{\mathcal{A}} + I_N^{\mathcal{B}})}$.

Suppose further that the parameters provided are eigenvectors $U_{\bar{A}_{(n)}} \in \mathcal{R}^{I_n \times k_{A_{(n)}}}$, eigenvalues $\lambda_{\bar{A}_{(n)}} \in \mathcal{R}^{k_{A_{(n)}} \times 1}$ for the mode-n unfolding matrix $\bar{A}_{(n)}$, mean tensor $\mathcal{M}^{\mathcal{A}}$ of tensor $\mathcal{A}$ and new tensor sequence $\mathcal{B}$ with mean $\mathcal{M}^{\mathcal{B}}$. Let $\mathcal{M}^{\mathcal{C}}$ be the mean tensor of $\mathcal{C}$, and $\mathcal{X}(i) = \mathcal{X}(:, \ldots, i)$ be one tensor object in the tensor sequence. We can compute the covariance matrix for mode-n unfolding matrix $\bar{C}_{(n)}$ with the forgetting factor f as follows:

$$
\begin{aligned}
\varphi_{\bar{\mathcal{C}}_{(n)}} =& \sum_{i=1}^{I_N^{\mathcal{A}}} (f(C(i)_{(n)} - M_{(n)}^{\mathcal{C}}))(f(C(i)_{(n)} - M_{(n)}^{\mathcal{C}}))^T \\
&+ \sum_{j=I_N^{\mathcal{A}}+1}^{I_N^{\mathcal{A}}+I_N^{\mathcal{B}}} \left(C(j)_{(n)} - M_{(n)}^{\mathcal{C}}\right)\left(C(j)_{(n)} - M_{(n)}^{\mathcal{C}}\right)^T \\
=& f^2 \sum_{i=1}^{I_N^{\mathcal{A}}} \left(A(i)_{(n)} - M_{(n)}^{\mathcal{A}} + M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{C}}\right)\left(A(i)_{(n)} - M_{(n)}^{\mathcal{A}} + M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{C}}\right)^T \\
&+ \sum_{j=1}^{I_N^{\mathcal{B}}} \left(B(i)_{(n)} - M_{(n)}^{\mathcal{B}} + M_{(n)}^{\mathcal{B}} - M_{(n)}^{\mathcal{C}}\right)\left(B(i)_{(n)} - M_{(n)}^{\mathcal{B}} + M_{(n)}^{\mathcal{B}} - M_{(n)}^{\mathcal{C}}\right)^T \\
=& f^2 \varphi_{\bar{A}_{(n)}} + \varphi_{\bar{\mathcal{B}}_{(n)}} + f^2 I_N^{\mathcal{A}} \left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{C}}\right)\left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{C}}\right)^T \\
&+ I_N^{\mathcal{B}} \left(M_{(n)}^{\mathcal{B}} - M_{(n)}^{\mathcal{C}}\right)\left(M_{(n)}^{\mathcal{B}} - M_{(n)}^{\mathcal{C}}\right)^T
\end{aligned}
$$

Since the relation among $\mathcal{M}^{\mathcal{A}}$, $\mathcal{M}^{\mathcal{B}}$ and $\mathcal{M}^{\mathcal{C}}$ is given by $\mathcal{M}^{\mathcal{C}} = \frac{1}{I_N^{\mathcal{A}} + I_N^{\mathcal{B}}}(I_N^{\mathcal{A}} \mathcal{M}^{\mathcal{A}} + I_N^{\mathcal{B}} \mathcal{M}^{\mathcal{B}})$, we can simplify the formula above to yield

$$
\varphi_{\bar{\mathcal{C}}_{(n)}} = f^2 \varphi_{\bar{\mathcal{A}}_{(n)}} + \varphi_{\bar{\mathcal{B}}_{(n)}} + \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}} (f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2} (M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})^T
$$

Therefore, for IMFPCA algorithm, the mode-n covariance matrix for the sequence with mean update is given by

$$
\begin{aligned}
\varphi_{\bar{\mathcal{C}}_{(n)}} =& f^2 \sum_{i=1}^{k_{A_{(n)}}} \lambda_{\bar{A}_{(n)}}(i) U_{\bar{A}_{(n)}}(:, i) U_{\bar{A}_{(n)}}(:, i)^T + \bar{B}_{(n)} \bar{B}_{(n)}^T \\
&+ \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}} (f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2} (M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})^T,
\end{aligned}
$$

which completes the proof.

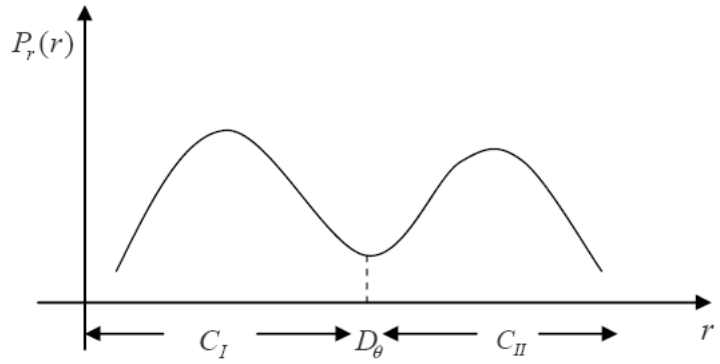


FIGURE 10. Bimodal histogram example

For the SKL algorithm as s second part of the *Proposition*, we have

$$\begin{aligned}\varphi_{\bar{C}_{(n)}} &= \varphi_{\bar{A}_{(n)}} + \bar{B}_{(n)}\bar{B}_{(n)}^T + \frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}(f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2}(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})^T \\ &= \varphi_{\bar{A}_{(n)}} + \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}(f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2}}(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})\right] \\ &\quad \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}(f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2}}(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}})\right]^T\end{aligned}$$

And recall that this equation can also be expressed as $\varphi_{\bar{C}_{(n)}} = \varphi_{\bar{A}_{(n)}} + \hat{B}_{(n)}\hat{B}_{(n)}^T$. Therefore, the mode-$n$ unfolding of $\mathcal{B}$ with mean update is generated simply by using $\hat{B}_{(n)}$ as derived below

$$\hat{B}_{(n)} = \left[\bar{B}_{(n)}, \sqrt{\frac{I_N^{\mathcal{A}} I_N^{\mathcal{B}}\,(f^2 I_N^{\mathcal{B}} + I_N^{\mathcal{A}})}{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}})^2}}\,\left(M_{(n)}^{\mathcal{A}} - M_{(n)}^{\mathcal{B}}\right)\right]$$

Moreover, take the forgetting factor in $f^2\varphi_{\bar{A}_{(n)}}$ into consideration, the matrix $R^{(n)}$ is generated as

$$\begin{bmatrix} f diag\left(\lambda_{\bar{A}_{(n)}}\right) & U_{\bar{A}_{(n)}} \\ 0 & \tilde{B}_{(n)}\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}}U_{\bar{A}_{(n)T}}\hat{B}_{(n)}\right) \end{bmatrix},$$

where $\tilde{B}_{(n)} = orth\left(\hat{B}_{(n)} - U_{\bar{A}_{(n)}}U_{\bar{A}_{(n)}}\right)$.

In the case, where $f = 1$, then the equations for the Proposition can be obtained.

A.2. Proof of the statement in Section 3.3. Using the same definitions provided in the *Proposition*, for the last mode unfolding, we have $A_{(N)} \in \mathcal{R}^{I_N^{\mathcal{A}} \times I_1 \ldots I_{N-1}}$, $B_{(N)} \in \mathcal{R}^{I_N^{\mathcal{B}} \times I_1 \ldots I_{N-1}}$ and $C_{(N)} \in \mathcal{R}^{(I_N^{\mathcal{A}} + I_N^{\mathcal{B}}) \times I_1 \ldots I_{N-1}}$. Since the procedure applies to their transpose, it gives $A_{(N)}^T \in \mathcal{R}^{I_1 \ldots I_{N-1} \times I_N^{\mathcal{A}}}$, $B_{(N)}^T \in \mathcal{R}^{I_1 \ldots I_{N-1} \times I_N^{\mathcal{B}}}$ and $C_{(N)}^T \in \mathcal{R}^{I_1 \ldots I_{N-1} \times (I_N^{\mathcal{A}} + I_N^{\mathcal{B}})}$. In order to follow the Pseudocode and be consistent with the computational complexity definition, we set $I_N^{new} = I_1^{old} I_2^{old} \ldots I_{N-1}^{old}$, then let $I_1^{new}, I_2^{new}, \ldots, I_{N-2}^{new} = 1$ and ${I_{N-1}^{\mathcal{A}}}^{new} = I_N^{\mathcal{A}}$, ${I_{N-1}^{\mathcal{B}}}^{new} = I_N^{\mathcal{B}}$ to obtain $A_{(N)}^T \in \mathcal{R}^{I_N^{new} \times I_1^{new} I_2^{new} \ldots {I_{N-1}^{\mathcal{A}}}^{new}}$, $B_{(N)}^T \in \mathcal{R}^{I_N^{new} \times I_1^{new} I_2^{new} \ldots {I_{N-1}^{\mathcal{B}}}^{new}}$ and $C_{(N)}^T \in \mathcal{R}^{I_N^{new} \times I_1^{new} I_2^{new} \ldots ({I_{N-1}^{\mathcal{A}}}^{new} + {I_{N-1}^{\mathcal{B}}}^{new})}$.

A.3. Threshold by bimodal histogram. Consider the bimodal histogram example in Figure 10, we have $P_I = P_r(C_I) = \sum_{i=1}^{\theta} P_i$ and $P_{II} = P_r(C_{II}) = \sum_{i=\theta+1}^{N} d_i$, where θ is considered as the index of threshold and D_θ the threshold that separates the two classes, P_I is the probability density of distances smaller than the threshold D_θ and P_{II} is the probability density of distances bigger than the threshold D_θ, and Nd is the number of the distances in the recognition process.

The mean of the distances of all the images is $\mu = \sum_{i=1}^{Nd} D_i P_i$. And the mean distance of correctly recognized images and the mean distance of the wrongly recognized images are defined respectively by $\mu_I = \frac{1}{P_I} \sum_{i=1}^{\theta} D_i P_i$ and $\mu_{II} = \frac{1}{P_{II}} \sum_{i=\theta+1}^{Nd} D_i P_i$.

The threshold is optimal when the interclass variance given by $V(D_\theta) = P_I(\mu - \mu_I)^2 + P_{II}(\mu - \mu_{II})^2$ is maximized.

In the example, the parameter chosen are $k_1 = 2, k_2 = 3$ and $k_3 = 15$ as mentioned in Section 4.4 to obtain the histogram shown in Figure 11, which gives the threshold as 2.26×10^7.

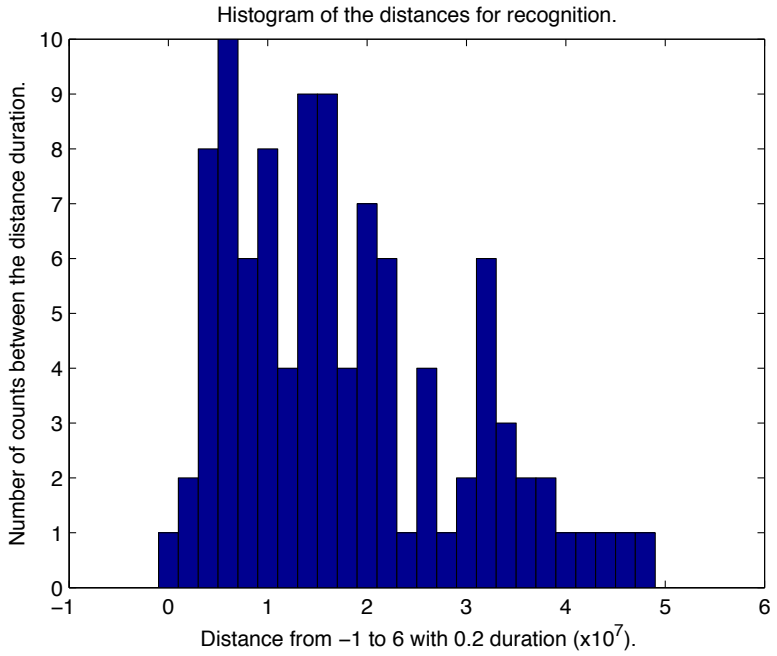


FIGURE 11. Histogram in our experiment to determine the threshold