Signal Processing 90 (2010) 2456–2467

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

On-line versus off-line accelerated kernel feature analysis: Application to computer-aided detection of polyps in CT colonography

Lahiruka Winter, Yuichi Motai *, Alen Docef

Department of Electrical and Computer Engineering, Virginia Commonwealth University , 601 West Main Street, Richmond, VA 23284-3072, USA

ARTICLE INFO

Article history:
Received 30 November 2008
Received in revised form
29 April 2009
Accepted 3 July 2009
Available online 18 July 2009

Keywords:
On-line learning
Accelerated kernel feature analysis
Extracting features
Computed tomographic colonography

ABSTRACT

A semi-supervised learning method, the on-line accelerated kernel feature analysis (On-line AKFA) is presented. In On-line AKFA, features are extracted while data are being fed to the algorithm in small batches as the algorithm proceeds. The paper compares and contrasts the use of On-line AKFA and Off-line AKFA in CT colonography. On-line AKFA provides the flexibility to allow the feature space to dynamically adjust to changes in the input data with time during the training phase. The computational time, reconstruction accuracy, projection variance, and classification performance of the proposed method are experimentally evaluated for kernel principal component analysis (KPCA), Off-line AKFA, and On-line AKFA. Experimental results demonstrate a significant reduction in computation time for On-line AKFA compared to the other feature extraction methods considered in this paper.

1. Introduction

Computed tomographic (CT) colonography, or virtual colonoscopy, is a promising technique for screening colorectal cancers by means of CT scans of the colon [1]. The prevailing CT technology allows a single image set of the colon to be acquired in 10–20 s, which translates into an easier, more comfortable examination than is available with other screening tests. For CT colonography to be a clinically practical method of screening for colon cancers, the technique must be able to interpret a large number of images in a time-effective fashion, and it must facilitate the detection of polyps—a precursor of colorectal cancers—with high accuracy. However, interpretation of an entire CT colonography examination is time-consuming, and the reader performance for polyp detection varies substantially [2,3,42].

To overcome these difficulties while providing an accurate detection of polyps, computer-aided detection (CAD) schemes have been developed that automatically detect suspicious lesions in CT colonography images [4,5]. CAD for CT colonography provides the locations of the suspicious polyps to radiologists, offering a second opinion that has the potential to improve detection performance [40].

Polyps appear as bulbous, cap-like structures that adhere to the colonic wall and protrude to the lumen, whereas folds appear as elongated, ridge-like structures, and the colonic wall appears as a large, nearly flat structure. Fig. 1 shows images of 48 polyps in our database of CT colonographic images. Most CAD schemes employ a model-based approach for the detection of polyp candidates, in which shape analysis methods that differentiate among these distinct types of shapes play an essential role [5–9]. Nevertheless, there are many naturally occurring normal colonic structures that occasionally imitate such shapes, and therefore the resulting polyp candidates typically include many FP. The reduction of such FP is

* Corresponding author. Tel.: +1 804 828 1281; fax: +1 804 828 0006.
E-mail address: ymotai@vcu.edu (Y. Motai).

0165-1684/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2009.07.004

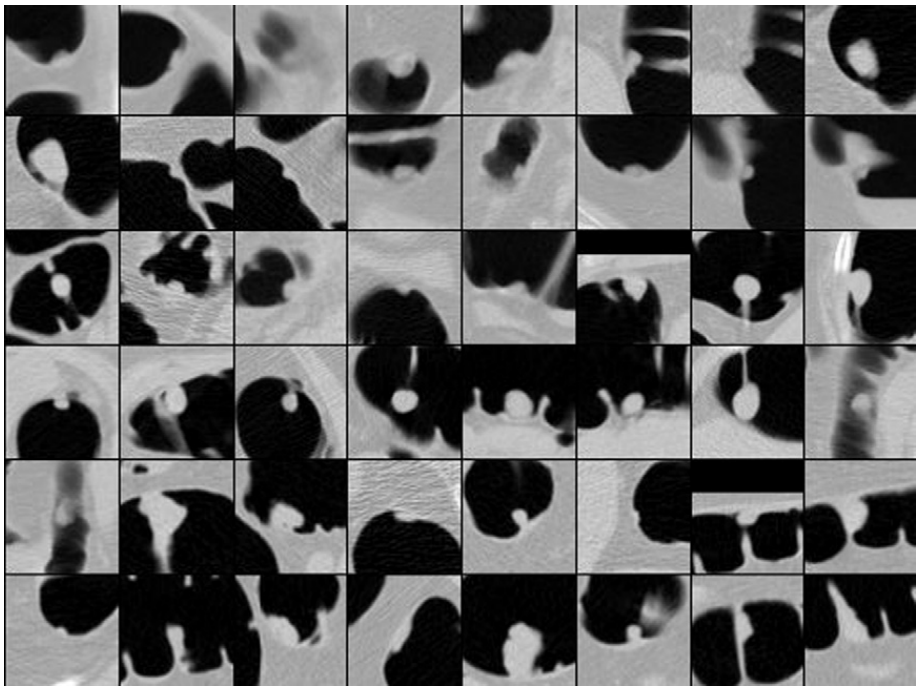

Fig. 1. Representative images of polyps in the DB1 CT Colonography data set.

often performed by first extracting a set of image features from segmented polyp regions, followed by application of a statistical classifier to the feature space for discrimination of FP from actual polyps [8,10–13]. Such CAD schemes tend to show a high sensitivity in the detection of polyps; however, they tend to produce a much larger number of FP than human readers [4].

The goal of this study is to achieve a fast and accurate method for the detection of polyps in CT colonographic images by effectively incorporating semi-supervised on-line learning methods and an appearance-based object recognition approach into a model-based CAD scheme. Traditionally, CAD schemes have been developed for off-line applications, where resources and training requirements are proportional to the number of training instances [41]. Thus, if more training data are collected after the initial tumor model is computed, retraining of the model becomes imperative in order to incorporate the incremental data and preserve the class concept. Hereafter, we will use the term *on-line* learning (as opposed to *batch* learning) to refer to incorporating new data that become available in the already computed model.

The concept of on-line learning has been discussed as an efficient method for non-linear and on-line data analysis [14]. It has been applied to kernel based and non-kernel based feature extraction methods. In [15], Zheng proposed a method, similar to batch learning, where the input data were divided into a few groups of similar size, and KPCA was applied to each group. A set of eigenvectors was obtained for each group and the final set of features was obtained by applying KPCA to a subset of these eigenvectors. In [16], Kivinen, Smola and Williamson introduce on-line learning in a reproducing kernel Hilbert space. The application of the on-line concept to principal component analysis is mostly referred to incremental principal component analysis [16–20] and has shown computational effectiveness in many image processing applications and pattern classification systems. The effective application of on-line learning to non-linear spaces is usually undertaken by using kernel based methods. Kim et al. introduces an incremental non-linear analysis method referred to as on-line and non-linear principal component analysis [21] for classification purposes. The computational effectiveness of the on-line version of the kernel principal component analysis (KPCA) over Off-line KPCA has been reported in [22–24].

The specific contribution of this study is to develop a faster and flexible kernel feature analysis by combining accelerated kernel feature analysis (AKFA), introduced in [25], with on-line learning techniques. The suggested method, in combination with a shape-based polyp detection method, can efficiently differentiate polyps from FP and thus improve detection performance. The key idea behind the proposed method is to allow the feature space to be updated as the training proceeds with more data being fed into the algorithm. The feature space update can be incremental or non-incremental. In incremental update, the feature space is augmented with new features extracted from the new data, with a possible expansion to the feature space if necessary. In non-incremental update, the dimension of the feature space remains constant where the newly computed features may replace some of the existing ones. This flexibility to update the eigenspace allows the capture of significant variations of data with time and allows adaptation of the feature space to the incoming data. AKFA, being a faster feature extraction

method that works in the kernel Hilbert space, provides the base algorithm for On-line AKFA.

AKFA maps the original, raw feature space into a higher dimensional feature space. Such a high-dimensional feature space is expected to have a greater classification power than that of the original feature space, as suggested by the Vapnik–Chervonenkis theory [26]. The on-line learning procedure uses AKFA for each block of input data if the current block of data shows a significant variation of input data with respect to the preceding blocks. The algorithm extracts texture-based features from the polyp candidates generated by a shape-based CAD scheme. The main contribution of this paper lies in the application of on-line learning techniques to feature extraction for classification of texture-based features in order to improve the computation time. The method was tested using real CT colonography data to show that the improved algorithm is faster than the off-line method while achieving a feature space that is comparably similar to the feature space obtained by the off-line learning method.

The remainder of this paper is organized as follows. Section 2 provides a brief review of the kernel principal component analysis and accelerated kernel feature analysis that act as the main feature extraction methods for the proposed on-line learning technique. Section 3 describes the complete on-line AKFA algorithm. Experimental results are discussed in Section 4. Section 5 presents a conclusion to the work discussed in this paper.

2. Background

2.1. Kernel feature extraction

For efficient feature analysis, extraction of the salient features of polyps is essential because of the size and the 3-D nature of the polyp data sets. Moreover, the distribution of the image features of polyps represents a non-linear feature space. These non-linear feature spaces can be handled by finding an operator that maps the non-linear feature space to a linear feature space in which linear feature analysis methods can be applied. Principal component analysis (PCA) [27–30], which is well known as a superior data compression method, has been extended to a non-linear feature space, kernel feature space, and is known as kernel PCA [31–34].

Essential to this approach is the selection of an operator that maps image vectors to the kernel feature space. A non-linear, positive-definite kernel $k : R^d \times R^d \rightarrow R$ of an integral operator, e.g., $k(x,y) = exp\{-\|x-y\|^2\}$, computes the inner product of the transformed vectors $\langle \Phi(x), \Phi(y) \rangle$, where $\Phi : R^d \rightarrow H$ denotes a non-linear embedding (induced by k) into a possibly infinite dimensional Hilbert space H. Given n sample points in the domain $X_n = \{x_i \in R^d | i = 1, \ldots, n\}$, the image $Y_n = \{\Phi(x_i) | i = 1, \ldots, n\}$ of X_n spans a linear subspace of at most $(n-1)$ dimensions. By mapping the sample points into a higher dimensional space, H, the dominant linear correlations in the distribution of the image Y_n may elucidate the important non-linear dependencies in the original data sample X_n. This is beneficial because it permits making PCA non-linear without complicating the original PCA algorithm. The kernel function k is traditionally chosen in the form of a Gaussian function such as the radial basis functions (RBF): $k(x,y) = \exp\{-\|x-y\|^2/2\sigma^2\}$. The RBF kernel turns out to be more applicable for the data used in this paper, compared to other available kernels such as linear, polynomial, Laplace, and sigmoid kernels.

Kernel PCA uses a Mercer kernel [27,30] to perform a linear PCA of this transformed image. Without loss of generality, we assume that the image of the data has been centered so that its scatter matrix H is given by $S = \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T$. Eigenvalues λ_j and eigenvectors e_j are obtained by solving

$$\lambda_j e_j = S e_j = \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T e_j = \sum_{i=1}^{n} \langle e_j, \Phi(x_i) \rangle \Phi(x_i) \tag{1}$$

for $j = 1, \ldots, n$. Since Φ is not known, (1) must be solved indirectly. Our intention is not to find Φ, as working in a higher dimensional space involves unnecessarily complicated computations, which can be avoided by substituting the kernel for computing the inner product of the transformed vectors $\langle \Phi(x), \Phi(y) \rangle$. Let $a_{ji} = 1/\lambda_j \langle e_j, \Phi(x_i) \rangle$ gives

$$e_j = \sum_{i=1}^{n} a_{ji} \Phi(x_i). \tag{2}$$

Multiplying by $\Phi(x_q)^T$ on the left, for $q = 1, \ldots, n$, and substituting $a_{ji} = 1/\lambda_j \langle e_j, \Phi(x_i) \rangle$ yields

$$\lambda_j \langle \Phi(x_q), e_j \rangle = \sum_{i=1}^{n} \langle e_j, \Phi(x_i) \rangle \langle \Phi(x_q), \Phi(x_i) \rangle. \tag{3}$$

Substitution of (2) into (3) produces

$$\lambda_j \left\langle \Phi(x_q), \sum_{i=1}^{n} a_{ji} \Phi(x_i) \right\rangle = \sum_{i=1}^{n} \left(\sum_{k=1}^{n} \langle a_{jk} \Phi(x_k), \Phi(x_i) \rangle \langle \Phi(x_q), \Phi(x_i) \rangle \right), \tag{4}$$

which can be rewritten as, $\lambda_j K a_j = K^2 a_j$, where K is an $n \times n$ Gram matrix, with the element $k_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle$ and $a_j = [a_{j1} a_{j2} \cdots a_{jn}]^T$. The latter is a dual eigenvalue problem equivalent to the problem

$$\lambda_j a_j = K a_j. \tag{5}$$

Using (2),

$$\|e_j\|^2 = \left\langle \sum_{i=1}^{n} a_{ji} \Phi(x_i), \sum_{i=1}^{n} a_{ji} \Phi(x_i) \right\rangle = \langle a_j, K a_j \rangle = \lambda_j \|a_j\|^2,$$

and therefore the normalization of each eigenvector (i.e. $\|e_j\| = 1$) requires $\|a_j\|^2 = 1/\lambda_j$.

In the following, we choose a Gaussian kernel, i.e.,

$$k_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle = \exp\left(-\frac{1}{2\sigma^2} \|x_i - x_j\|^2 \right). \tag{6}$$

If the image of X_n is not centered in the Hilbert space, we need to use the centered Gram matrix deduced by Smola, Mangasarian, and Schölkopf [28]:

$$\hat{K} = K - KT - TK + TKT, \tag{7}$$

where K is the Gram matrix of uncentered data, and

$$T = \begin{bmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \cdots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix}_{n\times n}.$$

Keeping the ℓ eigenvectors associated with the ℓ largest eigenvalues, we can reconstruct data in the mapped space: $\Phi_i' = \sum_{j=1}^{\ell}\langle\Phi_i, e_j\rangle e_j = \sum_{j=1}^{\ell}\beta_{ji}e_j$, where $\beta_{ji} = \langle\Phi_i, \sum_{k=1}^{n}a_{jk}\Phi_k\rangle = \sum_{k=1}^{n}a_{jk}k_{ik}$. Reconstruction of each data using fewer eigenvectors results in a reconstruction error. Thus the reconstruction square error of each data Φ_i, $i=1,\ldots,n$, is $\mathrm{Err}_i = \|\Phi_i - \Phi_i'\|^2 = k_{ii} - \sum_{j=1}^{\ell}\beta_{ji}^2$. The mean square error is $\mathrm{MErr} = 1/n\sum_{i=1}^{n}\mathrm{Err}_i$. Using (5), $\beta_{ji} = \lambda_j a_{ji}$. Therefore, the mean square reconstruction error is $\mathrm{MErr} = 1/n\sum_{i=1}^{n}(k_{ii} - \sum_{j=1}^{\ell}\lambda_j^2 a_{ji}^2)$. Since $\sum_{i=1}^{n}k_{ii} = \sum_{i=1}^{n}\lambda_i$ and $\sum_{i=1}^{n}a_{ji}^2 = \|a_j\|^2 = 1/\lambda_j$, $\mathrm{MErr} = 1/n\sum_{i=\ell+1}^{n}\lambda_i$.

The Kernel PCA algorithm can now be summarized as follows:

Step 1: Calculate the Gram matrix using (6), which contains the inner products between pairs of image vectors.

Step 2: Use (5) to get the coefficient vectors a_j for $j=1,\ldots,n$.

Step 3: The projection of a test point $x\in R^d$ along the j-th eigenvector is

$$\langle e_j, \Phi(x)\rangle = \sum_{i=1}^{n}a_{ji}\langle\Phi(x_i), \Phi(x)\rangle = \sum_{i=1}^{n}a_{ji}k(x, x_i).$$

The above implicitly contains an eigenvalue problem of rank n, so the computational complexity of KPCA is $O(n^3)$. In addition, each resulting eigenvector is represented as a linear combination of n terms; the ℓ features depend on n image vectors of X_n. Thus, all data contained in X_n must be retained, which is computationally cumbersome and unacceptable for on-line learning.

2.2. Sparse kernel feature analysis

Sparse kernel feature analysis (SKFA) [28] can be introduced as a more compact and time efficient method than KPCA, where SKFA improves the computational costs of KPCA, by reducing the time complexity and data retention requirements. SKFA algorithm extracts features one by one in the order of decreasing projection variance. The particular advantage of SKFA is that the ℓ features only depend on ℓ elements of X_n, which is extremely useful for on-line learning. Let $v_i\in H$, for $i=1,\ldots,\ell$ denote the features selected by SKFA. These features are different from the eigenvectors obtained using KPCA. We analyze the scatter matrix of the image data.

Following Eq. (1), with e_j replaced by v_j, we obtain $v_j^T\lambda_j v_j = v_j^T(\sum_{i=1}^{n}\langle v_j, \Phi(x_i)\rangle\Phi(x_i))$, where v_j is the j-th feature with unit length. Thus $\lambda_j = \sum_{i=1}^{n}\langle v_j, \Phi(x_i)\rangle^2$. Therefore, the first feature, corresponding to the maximum eigenvalue, is chosen as the direction with the maximum projected variance:

$$v_1 = \arg\max_{\|v\|^2=1}\frac{1}{n}\sum_{i=1}^{n}|\langle v, \Phi(x_i)\rangle|^2. \tag{8}$$

The global solution of Eq. (8), v_j, needs to satisfy an l_2 normalization constraint, i.e., v_1 should have unit Euclidian length. Changing the l_2 constraint to an l_1 constraint leads to a vertex solution. The l_1 constraint assumed by SKFA is

$$V_{l_1} = \left\{\sum_{j=1}^{n}a_j\Phi_j \middle| \sum_{j=1}^{n}|a_j| \le 1\right\}. \tag{9}$$

The first feature selected by SKFA satisfies (8) which is

$$v_1 = \arg\max_{v\in V_{l_1}}\frac{1}{n}\sum_{i=1}^{n}|\langle v, \Phi(x_i)\rangle|^2.$$

Smola et al. [28] showed that this feature corresponds to an element of the image Y_n, hence

$$v_1 = \arg\max_{\Phi(x_j)\in V_{l_1}}\frac{1}{n}\sum_{i=1}^{n}|\langle\Phi(x_j), \Phi(x_i)\rangle|^2. \tag{10}$$

Subsequent features are obtained iteratively. After $i-1$ features $\{v_t\in H|t=1,\ldots,i-1\}$ have been found, each image $\Phi_j = \Phi(x_j)$ can be projected into the orthogonal subspace to obtain

$$\Phi_j^i = \Phi_j - \sum_{t=1}^{i-1}v_t\frac{\langle\Phi_j, v_t\rangle}{\|v_t\|^2} = \Phi_j - \sum_{t=1}^{i-1}\frac{a_{tj}v_t}{\|v_t\|^2}, \tag{11}$$

where $a_{tj} = \langle\Phi_j, v_t\rangle$. Then Φ_j^i is normalized by the l_1 constraint in (9). The projection variance of the normalized Φ_j^i with all Φ_k, $k=1,\ldots,n$ is then calculated. Finally, we can identify the maximum projection variance and select the corresponding Φ_j^i as the i-th feature, v_i.

Based on the ℓ features extracted by SKFA, each training data in the mapped space can be reconstructed by projecting each training data along the selected feature vectors into the orthogonal subspace,

$$\Phi_i' = \sum_{j=1}^{\ell}\frac{\langle\Phi_i, v_j\rangle v_j}{\|v_j\|^2} = \sum_{j=1}^{\ell}\frac{a_{ji}v_j}{\|v_j\|^2}, \tag{12}$$

where a_{ji}, the projection of i-th training data on j-th feature, is stored after extracting the j-th feature. According to Eq. (11), the feature set $\{v_1,\ldots,v_\ell\}$ only depends upon the set of ℓ image vectors, $\{\Phi_{i\,dx(i)}, i=1,\ldots,\ell\}$, where the i-th element of the index vector $idx(i)$ denotes the subscript of the projected image Φ_j^i that was selected while constructing v_i. Therefore, after training, we only need to retain the ℓ input vectors $\{x_{i\,dx(i)}\in R^d|i=1,\ldots,\ell\}$, where ℓ is the number of features extracted.

SKFA extracts ℓ features, where typically ℓn. As $O(in^2)$ operations are required to extract the i-th feature, the total computational cost for ℓ features is $O(\ell^2n^2)$, which is an improvement over the $O(n^3)$ operations required by KPCA.

2.3. Accelerated kernel feature analysis

AKFA [25] is a method suggested to improve the efficiency and accuracy of sparse kernel feature analysis (SKFA) [28,35] by Mangasarian, Smola, and Schölkopf. SKFA extracts the features one by one in the order of decreasing projection variance and improves the compu-

tational costs of KPCA, associated with both time complexity and data retention requirements. In an attempt to achieve further improvements, AKFA, (i) saves computation time by iteratively updating the Gram matrix, (ii) normalizes the images with the l_2 constraint before the l_1 constraint is applied, and (iii) optionally discards data that falls below a magnitude threshold δ during updates.

First (i), as the Gram matrix gets iteratively updated, instead of extracting features directly from the original mapped space, AKFA extracts the i-th feature based on the i-th updated Gram matrix K^i, where each element is $k^i_{jk} = \langle \Phi^i_j, \Phi^i_k \rangle$.

Since $\Phi^i_j = \Phi^{i-1}_j - v_{i-1}\langle \Phi^{i-1}_j, v_{i-1}\rangle$, $k^i_{jk} = \langle \Phi^{i-1}_j, \Phi^{i-1}_k\rangle - \langle \Phi^{i-1}_j, v_{i-1}\rangle\langle \Phi^{i-1}_k, v_{i-1}\rangle$, when expressed Φ^i_j and k^i_{jk} with respect to the previous iteration, and thereby,

$$k^i_{jk} = k^{i-1}_{jk} - \frac{\langle \Phi^{i-1}_j, \Phi^{i-1}_{i\,dx(i-1)}\rangle\langle \Phi^{i-1}_k, \Phi^{i-1}_{i\,dx(i-1)}\rangle}{\|\Phi^{i-1}_{i\,dx(i-1)}\|^2} = k^{i-1}_{jk} - \frac{k^{i-1}_{j,i\,dx(i-1)}k^{i-1}_{k,i\,dx(i-1)}}{k^{i-1}_{i\,dx(i-1),i\,dx(i-1)}}. \tag{13}$$

By updating the Gram matrix, it is unnecessary to save the projection of each individual data on all previous features. The computational cost for extracting i-th feature becomes $O(n^2)$, instead of $O(in^2)$ as in SKFA or $O(n^3)$ as in KPCA.

The second (ii) improvement is to revise the l_1 constraint. SKFA treats each individual sample data as a possible direction, and computes the projection variances with all data. Since SKFA includes its length in its projection variance calculation, it is biased to select vectors with larger magnitude. We are actually looking for a direction with unit length and when we choose an image vector as a possible direction, we ignore the length and only consider its direction, which improves the accuracy of the features. Therefore, in our AKFA algorithm, we replace the l_1 constraint of SKFA by

$$V^i_{l_1} = \left\{ \sum_{j=1}^{n} a_j \frac{\Phi^i_j}{\|\Phi^i_j\|} \,\middle|\, \sum_{j=1}^{n} |a_j| \le 1 \right\}. \tag{14}$$

The i-th feature is extracted by

$$v_i = \arg\max_{v \in V^i_{l_1}} \frac{1}{n}\sum_{j=1}^{n} |\langle v, \Phi^i_j\rangle|^2. \tag{15}$$

Since v_i is extracted from Φ^i space, the solution is located on one of $\hat{\Phi}^i_j = \Phi^i_j/\|\Phi^i_j\|$ for $j = 1, \ldots, n$. Eq. (15) reduces to

$$v_i = \arg\max_{\hat{\Phi}^i_j} \frac{1}{n}\sum_{t=1}^{n} |\langle \Phi^i_t, \hat{\Phi}^i_j\rangle|^2 = \arg\max_{\hat{\Phi}^i_j} \frac{1}{nk^i_{jj}}\sum_{t=1}^{n} {k^i_{jt}}^2. \tag{16}$$

Each $\hat{\Phi}^i_j$ satisfies the l_1 constraint as $\hat{\Phi}^i_j = \Phi^i_j/\|\Phi^i_j\|$ for $j = 1, \ldots, n$, so the normalization step that appears in SKFA is not required.

Let $\Phi^i_{i\,dx(i)}$ denote the image vector corresponding to the i-th feature. Suppose we have selected $(i-1)$ features with $\mathbf{V}_{(\mathbf{i}-1)} = \Phi_{(\mathbf{i}-1)}\mathbf{C}_{(\mathbf{i}-1)}$, where $\mathbf{V}_{(\mathbf{i}-1)} = [v_1 v_2 \cdots v_{(i-1)}]$, $\Phi_{(\mathbf{i}-1)} = [\Phi_{i\,dx(1)}\Phi_{i\,dx(2)}\cdots\Phi_{i\,dx(i-1)}]$, and $\mathbf{C}_{(\mathbf{i}-1)}$ is the coefficient matrix, which is upper-triangular. Then $\Phi^i_{i\,dx(i)} = \Phi_{i\,dx(i)} - \sum_{t=1}^{i-1}\langle \Phi_{i\,dx(i)}, v_t\rangle v_t$. Let us study the second term:

$$\sum_{t=1}^{i-1}\langle \Phi_{i\,dx(i)}, v_t\rangle v_t = \sum_{t=1}^{i-1} v_t v_t^T \Phi_{i\,dx(i)} = \Phi_{(\mathbf{i}-1)}\mathbf{C}_{\mathbf{i}-1}\mathbf{C}^{\mathbf{T}}_{\mathbf{i}-1}\mathbf{K}_{\mathbf{i}\,\mathbf{dx}(i)},$$

where $\mathbf{K}_{\mathbf{idx(i)}} = [k_{i\,dx(i),\,i\,dx(1)} k_{i\,dx(i),\,i\,dx(2)} \cdots k_{i\,dx(i),\,i\,dx(i-1)}]^T$. Therefore,

$$v_i = (\Phi_{i\,dx(i)} - \Phi_{(\mathbf{i}-1)}\mathbf{C}_{\mathbf{i}-1}\mathbf{C}^{\mathbf{T}}_{\mathbf{i}-1}\mathbf{K}_{\mathbf{i}\,\mathbf{dx}(\mathbf{i})})/\sqrt{k^i_{i\,dx(i),\,i\,dx(i)}}. \tag{17}$$

Let

$$\mathbf{C}_{i,i} = 1/\sqrt{k^i_{i\,dx(i),\,i\,dx(i)}}, \tag{18}$$

and

$$\mathbf{C}_{1:(i-1),i} = -\mathbf{C}_{i,i}\mathbf{C}_{(\mathbf{i}-1)}\mathbf{C}^{\mathbf{T}}_{\mathbf{i}-1}\mathbf{K}_{\mathbf{i}\,\mathbf{dx}(\mathbf{i})}, \tag{19}$$

then $\mathbf{V_i} = \Phi_\mathbf{i}\mathbf{C_i}$, where $\mathbf{V_i} = [\mathbf{V}_{(\mathbf{i}-1)}, v_i]$, and $\Phi_\mathbf{i} = [\Phi_{(\mathbf{i}-1)}, \Phi_{i\,dx(i)}]$,

$$\mathbf{C_i} = \begin{pmatrix} \mathbf{C}_{(\mathbf{i}-1)} & \mathbf{C}_{1:(i-1),i} \\ 0 & \mathbf{C}_{i,i} \end{pmatrix}.$$

The third (iii) improvement is to discard negligible data and thereby eliminate unnecessary computations. In the i-th updated Gram matrix K^i, defined in (13), the diagonal elements k^i_{jj} represent the reconstruction error of the j-th data point with respect to the previous $(i-1)$ features. If k^i_{jj} is relatively small, then the previous $(i-1)$ features contain most of the information that would be acquired from this image vector. One can therefore optionally discard image points that satisfy $k^i_{jj} < \delta$, for some predetermined $\delta > 0$. In the following, we use a *cutoff threshold* that is useful when the data size is very large. It can also be used as a criterion to stop extracting features if one is unsure of the number of features that should be selected.

The complete AKFA algorithm is summarized below:

Step 1: Compute the $n \times n$ Gram matrix $k_{ij} = k(x_i, x_j)$, where n is the number of input vectors. This part requires $O(n^2)$ operations.

Step 2: Let ℓ denote the number of features to be extracted. Initialize the $\ell \times \ell$ coefficient matrix $\mathbf{C}$ to 0, and $i\,dx(\cdot)$ as an empty list which will ultimately store the indices of the selected image vectors. Initialize the threshold value $\delta = 0$ for the reconstruction error. The overall cost is $O(\ell^2)$.

Step 3: For $i = 1$ to ℓ repeat:

1. Using the i-th updated $\mathbf{K}^i$ matrix, extract the i-th feature using (17). If $K^i_{jj} < \delta$, then discard j-th column and j-th row vector without calculating the projection variance. Use $i\,dx(i)$ to store the index. This step requires $O(n^2)$ operations.
2. Update the coefficient matrix by using (18) and (19), which requires $O(i^2)$ operations.
3. Use (13) to obtain $\mathbf{K}^{i+1}$, an updated Gram matrix. Neglect all rows and columns containing diagonal elements less than δ. This step requires $O(n^2)$ operations.

The total computational complexity is increased to $O(\ell n^2)$ when no data are being cut during updating in the AKFA.

3. On-line kernel feature analysis

On-line learning, whether non-incremental or incremental, is very important when obtaining training data is time-consuming, when the statistical properties of data change dynamically or when there is a need for efficient use of limited storage space. In KPCA, since each eigenvector is a linear combination of all training data, we cannot discard any data if we want to use the exact eigenvectors. This is unacceptable for on-line learning. *On-line* training is different from *stochastic* or *batch* training. In on-line training, each pattern is presented once and only once. Thus, additional memory for storing the patterns is not necessary [36]. Therefore, to facilitate on-line training we need a method that requires retention of only a portion of the training data.

One way to address this requirement is to try to find a pre-image $\tilde{x}_j$ in the input space for each eigenvector e_j such that $\Phi(\tilde{x}_j)$ approximates e_j [32,37]. However, as discussed in [38], it is very hard to find pre-images, if they exist, for eigenvectors such that they are unique and orthogonal to each other in the larger dimensional feature space.

In AKFA, ℓ features correspond to ℓ data points in the input space, thus we just need to retain those data points and discard others. In this case, solving a pre-image problem is not necessary. Also, since we only need to retain fewer yet statistically sufficient data points, the feature space resulting from AKFA is computationally manageable and requires less storage. These are encouraging benefits for the iterative use of AKFA in an on-line learning method, the On-line AKFA. When new data are presented to the on-line training procedure, the new data will be represented by their projection on the previous reduced feature space. After some time, when a considerable change in the incoming data causes the accumulated reconstruction error to be larger than a predefined threshold, the existing eigenspace will need to be updated. The update may be incremental or non-incremental, as explained in the introduction.

Since most of the training data have been discarded, we cannot calculate the inner product of all training data with the newly presented data. We assume that the eigenspace we get from the first off-line learning has included dominant information of the training data, and the increasing reconstruction error is caused by the dynamic change of the new incoming data. This means that we only need to extract new features from the new incoming data. Meanwhile, it may not be appropriate to just add new features to the previous eigenspace, as it will make the feature space grow at each step of the on-line training. Instead, we maintain a constant dimension of the feature space by replacing the previous trivial features with the new features. The previous features are considered trivial if the new data have minimum projection along those features. Thus, the proposed On-line AKFA is non-incremental.

It should be noted that the first batch of data, which may need to contain more data than the subsequent batches, is used to extract the dominant features. During the first step we perform off-line training, whereas subsequent batches of new data are used for on-line learning. Therefore, the on-line non-incremental procedure can be summarized into four steps as follows:

Step 1: After a batch of new data is accumulated, they are reconstructed using the feature space in the previous iteration. The vector Φ_i' represents the reconstructed new data, which is calculated indirectly using the kernel trick:

$$\Phi_i' = \sum_{j=1}^{\ell} \langle \Phi_i, v_j \rangle v_j = \Phi_\ell C_\ell C_\ell^T K_i, \tag{20}$$

where $K_i = [k_{i;i\,dx(1)} k_{i;i\,dx(2)} \dots k_{i;i\,dx(1)}]^T$. Then the reconstruction error of new data $\Phi_i, i = n+1, \dots, n+m$, is

$$\mathrm{Err}_i = ||\Phi_i - \Phi_i'||^2 = k_{ii} - K_i^T C_l C_l^T K_i. \tag{21}$$

The vector K_i is obtained from the Gram matrix computed between the new data and the retained data from the previous step. In non-incremental on-line training, the number of retained data from the previous iteration always equals to the number of features extracted in the initial off-line training. The total mean square error (MSE) is $1/m \sum_{i=n+1}^{n+m} \mathrm{Err}_i$. If this error is larger than a set threshold value (η), the eigenspace will be updated. Therefore, if the feature space from the previous step is sufficient to reconstruct the new batch of data with sufficient accuracy, we assume that the previous features accurately represent the dominant features of the new data and that no update for the eigenspace is required. This eliminates the need to perform computations on the new batch of data to extract features, thus saving computational time. Steps 2–4 should be performed if the MSE is greater than the threshold η.

Step 2: Calculate the projections (p_j) of all new data onto each previous feature (v_j):

$$p_j = \sum_{i=n+1}^{n+m} \langle \Phi_i, v_j \rangle^2 = \sum_{i=n+1}^{n+m} (K_i^T C_{1:l,j})^2, \tag{22}$$

where $j = 1, 2, \dots, l$. Then, the vector v_j corresponding to the minimum p_j should be selected as the trivial feature which is going to be replaced with a new feature extracted from the new batch of data. Here, we replace only one of the previous features as we assume that the data are not highly dynamic.

Step 3: A new feature should be extracted from the new batch of data. First, the new Gram matrix ($\hat{K}$) of the difference vectors is calculated:

$$\hat{k}_{ij} = \langle \Phi_i - \Phi_i', \Phi_j - \Phi_j' \rangle = k_{ii} - K_i^T C_l C_l^T K_j, \tag{23}$$

where i, j are from (n+1) to (n+m). This matrix relates to the difference between the newly acquired data and the reconstructed form of these new data. Therefore, we can use the AKFA method to extract the prominent feature from this Gram matrix. The new feature, $\tilde{v} = [\Phi_l \Phi_g] \tilde{C}_{1:(l+1)}$, corresponds to the new data Φ_g. Since each element in $\hat{K}$ has eliminated the projections onto the previous eigen-

space, the new eigenvector must be orthogonal to the previous eigenvectors.

Step 4: As the last step, the coefficient matrix C_l needs to be updated accordingly to reflect the new feature set. From the previous set of features, V_l, the trivial feature (v_t) chosen at Step 2 should be taken out and the new feature ($\tilde{v}$) should be added to the set of features as $\hat{v}_l$. Correspondingly, $\Phi_{i\,dx(t)}$ is taken out from Φ_l to form a new $\hat{\Phi}_{l-1}$ and Φ_g from the new batch of data becomes new $\hat{\Phi}_{i\,dx(l)}$ as it will be retained while rest of the data from the new batch is discarded. Updating V_l and $\hat{\Phi}_l$are actually hypothetical as we work in the kernel space, and use the kernel trick to avoid dealing with the high dimensional space. Therefore, in reality we only need to update the coefficient matrix C_l. The j-th row and column of C_l are taken out to form a new matrix $\hat{C}_{l-1}$. The new matrix $\hat{C}_{1:l,l}$ is calculated using the same procedure as in AKFA to update the co-efficient matrix using Eqs. (13) and (14) above. The updated parameters are used in the next on-line training step.

The proposed On-line AKFA does not require solving an increasing-sized eigenvalue problem. A sufficient amount of training data is used for the initial off-line training and the remaining training data are fed in relatively smaller quantities as the on-line training proceeds. Due to the smaller-sized batches of data that are fed during each training step, the storage requirement of On-line AKFA is less compared to other off-line training methods. At the same time, since the computations are efficiently done for each new batch of data, the computational time for training is greatly reduced. Therefore, the proposed On-line AKFA as a training procedure for larger training data sets shows improvements in computational time, flexibility, and storage efficiency.

4. Experimental analysis

4.1. The CT colonography image data set

The proposed On-line AKFA together with Off-line AKFA and KPCA were evaluated using CT image data sets of colonic polyps comprised true positives (TP) and false positives (FP) detected by our CAD system [5]. We obtained studies of 146 patients who had undergone a colon-cleansing regimen in preparation for same-day optical colonoscopy. Each patient was scanned in both supine and prone positions, resulting in a total of 292 CT studies. Helical single-slice and multi-slice CT scanners (GE HiSpeed CTi, LightSpeed QX/I, and LightSpeed Ultra; GE Medical Systems, Milwaukee, WI) were used, with collimations of 1.25–5.0 mm, reconstruction intervals of 1.0–5.0 mm, X-ray tube currents of 50–260 mA and voltages of 120–140 kVp. In-plane voxel sizes were 0.51–0.94 mm, and the CT image matrix size was 512×512. Out of 146 patients, there were 108 normal cases and 38 abnormal cases with a total of 61 colonoscopy-confirmed polyps larger than 6 mm. Twenty-eight polyps were 6–9 mm and 33 polyps were larger than 10 mm (including seven lesions larger than 30 mm). The CAD scheme processed the supine and prone volumetric data sets generated from a single patient independently to yield polyp candidates. The CAD scheme detected polyp candidates in the 292 CT colonography data sets with a 98% polyp detection sensitivity.

The volumes of interest (VOIs) representing each polyp candidate have been calculated as follows. The CAD scheme provided a segmented region for each candidate. The center of the VOI was placed at the center of mass of the region. The size of the VOI was chosen so that the entire region was covered. Finally, the VOI was resampled to $16 \times 16 \times 16$ voxels. The VOIs so computed comprise the data set DB1, a sample of which is shown in Fig. 1. There were a total of 131 true polyps (some of the larger lesions had multiple detections) and 8008 FP. The same procedure has been carried out using VOIs with dimensions $12 \times 12 \times 12$ voxels to build the data set DB2 that consists of 39 TP and 149 FP. Experiments in Sections 4.2 and 4.3 use both DB1 and DB2 data sets, while DB2 is used in Section 4.4, and DB1 is used in Section 4.5.

4.2. Computational time

We evaluated the computational efficiency of the proposed method by comparing its run time with the other two methods for different data sizes. The algorithms have been implemented in Matlab 7.0.1 (R14) using the statistical pattern recognition toolbox [39] for the Gram matrix calculation and kernel projection. The processor was a 2.5 GHz Intel® Core™2 Duo T9300 with 3 MB of RAM. Run time was determined using the *cputime* command.

4.2.1. Experiment with DB1

First the computational time for DB1 has been analyzed. Computation time results, in seconds, for Off-line AKFA and On-line AKFA are shown in Fig. 2. The initial, off-line, training batch has size 300 and every subsequent batch has 30 new data, adding up to a total of 750 after 16 batches. For On-line AKFA, we measured the computation time after the initial off-line training and at the end of each on-line step. For fair comparison, we ran Off-line AKFA for data sizes equal to the cumulative size used for On-line AKFA (that is, 300, 330, 360…). The parameter σ of the RBF kernel was set to 16, while the dimension of the eigenspace was set to 70 for all the feature extraction algorithms, when using DB1.

For each algorithm, computation time increases with increasing training data size (n), as expected. Off-line AKFA requires the computation of a Gram matrix whose size increases as the data size increases. On-line AKFA computes a large Gram matrix during the initial step and then efficiently updates the feature space in subsequent steps. Therefore, On-line AKFA is significantly faster than Off-line AKFA for the same reduction in eigen-dimension to 70. For example, for $\eta = 0.945$ and $n = 750$, On-line AKFA was about 4.4 times faster than Off-line AKFA. Moreover, when the rate of increase in computational time is considered for both methods, it appears as the computational gain of the proposed method increases for larger data sizes.

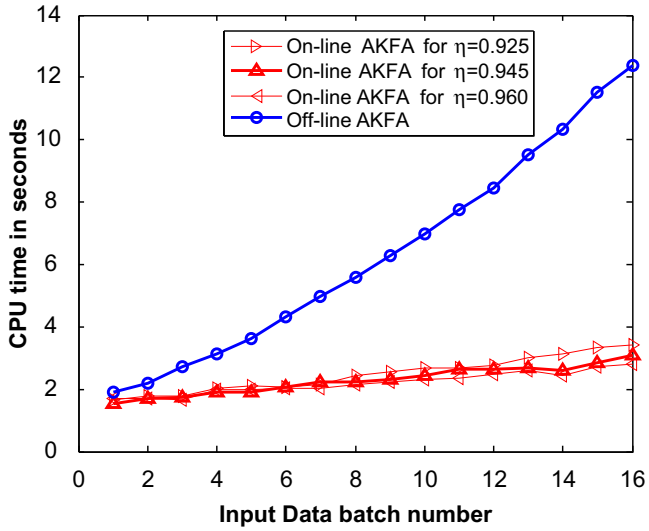

Fig. 2. Computational time for On-line and Off-line AKFA. The initial batch has size 300 and every subsequent batch has size 30.

The computation time for On-line AKFA is plotted for three values of the threshold η. When $\eta = 0.925$, the eigenspace was updated at every on-line learning step as a result of MSE being larger than η. At the other extreme, when $\eta = 0.960$, the eigenspace obtained from the first off-line step was never updated for subsequent batches of data. For $\eta = 0.945$, the eigenspace was updated in some of the on-line learning steps, thus perfectly demonstrating the expected behavior of on-line learning. The change in computational time for different values of η is not significant compared to the reduction we obtain compared to the off-line method. This clearly demonstrates that the application of On-line AKFA for detection of polyps in CT colonography is much faster than the existing off-line training methods; especially when the data size is larger.

4.2.2. Experiment with DB2

The same procedure is used with DB2, except for different data sizes. The initial, off-line, training batch has size 85 and every subsequent batch has nine new data, adding up to a total of 148 after eight batches. The use of DB2 required slight changes to the parameters of kernel space. The dimension of the eigenspace was set to 75 for all three methods. The results for KPCA, Off-line AKFA, and On-line AKFA are presented in Table 1. Due to the smaller size of DB2, the computational savings are less dramatic, but On-line AKFA is still about 21% faster than Off-line AKFA. We also noticed that the decrease in computation time for Off-line AKFA compared to KPCA was relatively small, implying that the use of Off-line AKFA on a smaller training data set does not yield much advantage over KPCA. Note that the CPU time for On-line AKFA is given after 148 batches. The CPU times for KPCA and Off-line AKFA are given after training the same number of data as the On-line AKFA, as feeding data in batches is not used for KPCA and Off-line AKFA. We analyzed the run time for On-line AKFA for three different values of the threshold η. The eigenspace was updated at every on-line learning step

Table 1
CPU time in seconds for DB2.

Feature extraction algorithm	CPU time (s)
KPCA	0.2340
Off-line AKFA	0.2184
On-line AKFA	
$\eta = 0.640$	0.1872
$\eta = 0.750$	0.1716
$\eta = 0.775$	0.1560

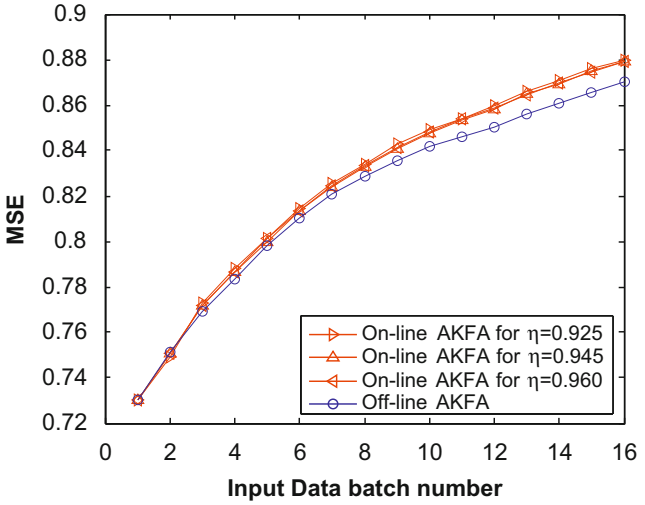

Fig. 3. Mean square reconstruction error for on-line and off-line learning using AKFA. The initial batch has size 300 and every subsequent batch has size 30.

for $\eta = 0.640$, as a result of MSE being larger than η. In contrast, when $\eta = 0.775$ the eigenspace, obtained from the first off-line step, was never updated for subsequent on-line steps. The value $\eta = 0.750$ was chosen to test the situation where the eigenspace updated only in some of the on-line steps.

4.3. Reconstruction error vs. data sizes

4.3.1. Experiment with DB1

The reconstruction error has been evaluated for On-line and Off-line AKFA for increasing training data sizes using DB1. The mean square reconstruction error was calculated using (16) and the dimension of the eigenspace was set to 70. The on-line learning has been tested for three different values of η, as explained in Section 4.2. The results are shown in Fig. 3. The vertical axis indicates mean square error (MSE), which is the mean squared value of error explained by Eq. (21). This reconstruction error increases as the data sizes increase, for both on-line and off-line learning. This could be due to the heterogeneous nature of the data set. The value of the parameter η has little effect on On-line AKFA reconstruction performance. A slight increase (less than 1%) in the reconstruction error of On-line AKFA is observed, as expected, compared to Off-line AKFA. However, the reduction in computation time clearly outweighs this small loss of accuracy.

4.3.2. Experiment with DB2

The same conditions and parameters have been used in this experiment as in Section 4.2. The reconstruction error results for KPCA, Off-line AKFA, and On-line AKFA for this data set are summarized in Table 2. The results show that the reconstruction ability of KPCA is better than that of Off-line AKFA, made evident by the smaller reconstruction error. The optimum value for reconstruction error for On-line AKFA was obtained for $\eta = 0.750$. Compared to the off-line methods, a somewhat higher reconstruction error for On-line AKFA was observed. However, given more training data coupled with the ability to extract more features would have resulted in a more accurate representation of data in the reduced eigenspace, and therefore in comparable results for on-line and off-line learning. Also, we expect the difference in reconstruction error between Off-line AKFA and On-line AKFA to be reduced when the training data set size is increased.

4.4. Evaluation of classification performance for polyp candidates

In order to analyze how feature extraction methods affect classification performance of polyp candidates, we used the *k*-nearest neighborhood classifier on the image vectors in the reduced eigenspace. The data set DB2 was used in the experiments described in this section. We first evaluated the performance of the classifier by applying it to the raw data without any feature extraction (Case 1). Then we applied the classifier to the feature spaces obtained by KPCA (Case 2), Off-line AKFA (Case 3), and On-line AKFA (Case 4).

Case 1 (*Raw data without feature extraction*): The training data and test data were selected according to the arrangement given in Table 3. In this experiment, the *k*-nearest neighbor classifier used $k = 9$ nearest neighbors because, as will be seen later, this works best for classification in the kernel space. We observed classification accuracies between 95.00% and 100.00%. This implies that the *k*-nearest neighbor classifier performs well for this set of polyp candidate data. We performed this experiment to assure that the data are classifiable using the *k*-nearest neighbor classifier. The result was positive. However, performing classification in the original space would be computationally inefficient. The detailed results are presented in Table 4. Only two polyps were incorrectly classified as FN (false negative), while all the FP have been identified correctly as TN (true negative). This yields a classification accuracy of 95.00%.

Table 2
Mean square reconstruction error for DB2.

Feature extraction algorithm	Mean square error (%)
KPCA	5.03
Off-line AKFA	8.20
On-line AKFA	
$\eta = 0.640$	10.90
$\eta = 0.750$	10.68
$\eta = 0.775$	19.11

Table 3
Arrangement of training and test data for classification.

	Proportion from the total data set (%)	Number of vectors	Total
Arrangement 1			
Training set			
TP	80.00	31	148
FP	78.30	117	
Test set			
TP	20	8	40
FP	21.70	32	

The data set DB2 comprised 39 TP data and 149 FP data.

Table 4
Classification results for DB2 without/with feature extraction (considering nine nearest neighbors).

	TP	TN	FN	FP	Classification accuracy (%)
Case 1: Original space	6	32	2	0	(6+32)/40 = 95.00
Case 2: KPCA	6	32	2	0	(6+32)/40 = 95.00
Case 3: Off-line AKFA	5	32	3	0	(5+32)/40 = 92.50
Case 4: On-line AKFA	4	32	4	0	(4+32)/40 = 90.00

Classification accuracy was calculated as (TP+TN)/(TP+TN+FN+FP).

Case 2 (*KPCA*): The *k*-nearest neighborhood classifier was applied on the data after the feature extraction using the KPCA algorithm, which was used to extract a total of 75 features during the training. The data set as described in Arrangement 1 in Table 3 was used, and 1–10 nearest neighbors were considered. The results for classification accuracy against the number of nearest neighbors are given in Table 5. When nine nearest neighbors were considered for classification, the test data in the reduced eigenspace were grouped as given in Table 4, resulting in a classification accuracy of 95.00%.

Case 3 (*Off-line AKFA*): The Off-line AKFA algorithm has been applied on the same training data set given in Table 3 to extract 75 features. Then the test data were classified using *k*-nearest neighborhood classifier considering 1–10 nearest neighbors; the results are summarized in Table 5. The grouping of the test data after the classification (for nine nearest neighbors) was given in Table 4 under Case 3. In this case, one more TP was classified inaccurately as FN compared to the results of Cases 1 and 2. We observe a small decrease in classification accuracy of Off-line AKFA in comparison with KPCA.

Case 4 (*On-line AKFA*): In training the data using On-line AKFA method, we extracted 75 features from the training data set given in Table 3. On-line learning consisted of eight steps where 68 false and 17 true polyp patterns were fed for the first off-line step and each subsequent step consisted of seven false and two true polyp patterns being fed for the on-line training. The initial off-line step in On-line AKFA required sufficient number of training data for better performance in classification. The experiment was performed for 1–10 nearest neighbors and as well as for various values of η. Table 5 presents these classification results for $\eta = 0.640$,

Table 5
Classification accuracy for each feature extraction algorithm against the number k of nearest neighbors.

No. of nearest neighbors	Classification accuracy (%)				
	KPCA	Off-line AKFA	On-line AKFA		
			$\eta = 0.640$	$\eta = 0.750$	$\eta = 0.775$
1	97.5	92.5	90	95	87.5
2	100	90	80	80	80
3	95.5	95	82.5	82.5	80
4	100	92.5	80	80	80
5	100	92.5	82.5	87.5	85
6	97.5	92.5	80	85	85
7	97.5	92.5	85	92.5	82.5
8	95	90	82.5	85	85
9	95	92.5	85	90	95
10	92.5	90	82.5	87.5	92.5

0.750, and 0.775. We noticed that choosing $\eta = 0.750$ leads to optimum classification performance for On-line AKFA. Even though the results for classification accuracy for on-line learning indicates a decrease compared to off-line learning methods, we should note that in this case, an incorrect classification of one pattern gives a decrease of 2.50% in classification accuracy as the data set is smaller. Therefore, with larger data sets the difference in classification accuracy for off-line and on-line learning is expected to be much smaller.

As shown in the four cases above, the classification accuracy was slightly affected by the feature extraction method, the chosen learning technique, and the classifier; there is a certain limit of accuracy that can be achieved by distance based classifiers such as k-NN and k-means. Also, since the On-line AKFA involves feeding of data in batches, the method accounts for relative variations in data in the successive batches. This might have resulted in slightly increased reconstruction error and reduced classification accuracy in On-line AKFA. However, the advantages of the On-line AKFA method over the off-line methods are its computational efficiency and flexible handling of dynamic data.

4.5. Projection variance of training data along selected features

By analyzing the projection variance of training data along each selected feature in the reduced eigenspace, we can explore the suitability of our eigenspace in representing the data. Also, this allows us to identify the most relevant features as the features with the highest projection variance. These features provide good clustering ability, leading to good classification performance. To study this behavior, we calculated the variance of projected data along each new dimension of the new eigenspace created by each feature extraction algorithm. The experimental setup was identical to the one described in Section 4.2 for data set DB1.

The results in Fig. 4 show the projection variance along features selected by the different algorithms. It appears that only a few features have large associated variances and the other dimensions have comparatively very small

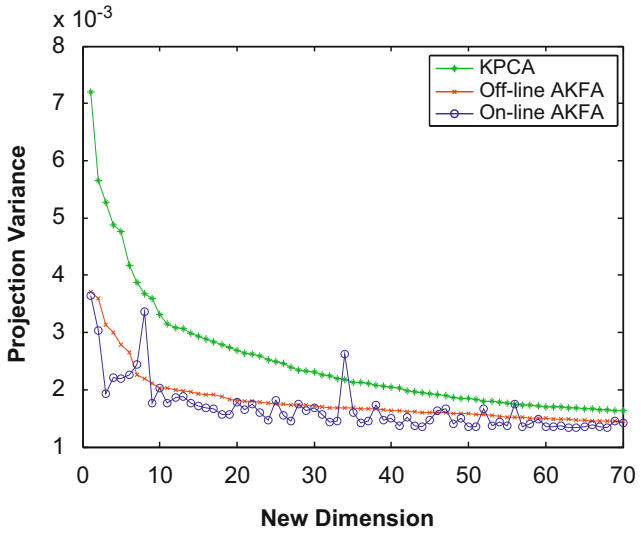

Fig. 4. Projection variance along new dimensions selected by each algorithm.

projection variance. The difference in projection variance, between the successive features, is larger for the first 10 features when compared with that of the features chosen afterwards by the off-line methods. While extracting features using off-line algorithms, we expect a descending order of projection variance along the selected dimensions in the order of selection. The projection variance values obtained for KPCA are almost twice the values obtained for Off-line AKFA. This implies that the kernel space created by the Off-line AKFA spread out the reconstructed patterns in a smaller volume of the kernel space.

The projected variance along the features chosen by the On-line AKFA method depicts a different behavior from the off-line methods. This behavior is expected, since in on-line learning the features chosen in the initial step are replaced by more relevant features that are found in subsequent steps. Fig. 4 displays how the feature space is updated as the on-line learning proceeds. Apart from that, the projection variance values obtained for the On-line AKFA are comparable to the values for Off-line AKFA. This implies that our feature space from on-line learning is

quite similar to that from off-line learning. Using this type of feature analysis, we can expect the On-line AKFA to have comparable clustering performance, thus the classification performance, to that of Off-line AKFA.

In these experiments, the feature space was reduced to 70 dimensions from the 4096-dimensional original space. Results of this section demonstrate that extraction of more features to expand the feature space beyond 70 dimensions would not contribute towards good clustering performance. Therefore, an expansion of feature space is not necessary for this case.

5. Conclusion

This paper proposed a semi-supervised learning method, the On-line accelerated kernel feature analysis for the computer-aided detection of polyps in CT colonography. The method was evaluated in terms of speed and classification performance against KPCA and Off-line AKFA. Although performance depends on the characteristics of a particular data set, On-line AKFA was found, in one typical case, to be about 4.4 times faster than Off-line AKFA, with a very small loss in reconstruction accuracy ($<1\%$). The classification performance decreases somewhat from Off-line to On-line AKFA. However, the use of sufficient training data for the initial off-line step, together with an optimum number of features to be extracted, depending on the characteristics of the data set for a given application, could improve the results for On-line AKFA. The experimental results for projection variance show how the on-line learning method adapts to the changes in data as the data are being fed to the algorithm. Together with the ability to feed data in small quantities, this makes the proposed on-line learning method very well suited for extracting features from data with time-varying statistical properties.

Acknowledgments

The authors would like to acknowledge J. Näppi and H. Yoshida with the Department of Radiology, Massachusetts General Hospital and Harvard Medical School, Boston, MA 02114, USA, who provided CT data sets for this study.

References

[1] S. Winawer, R. Fletcher, D. Rex, J. Bond, R. Burt, J. Ferrucci, T. Ganiats, T. Levin, S. Woolf, D. Johnson, L. Kirk, S. Litin, C. Simmang, Colorectal cancer screening and surveillance: clinical guidelines and rationale—update based on new evidence, Gastroenterology 124 (2003) 544–560.
[2] K.D. Bodily, J.G. Fletcher, T. Engelby, M. Percival, J.A. Christensen, B. Young, A.J. Krych, D.C. Vander Kooi, D. Rodysill, J.L. Fidler, C.D. Johnson, Nonradiologists as second readers for intraluminal findings at CT colonography, Acad. Radiol. 12 (2005) 67–73.
[3] J.G. Fletcher, F. Booya, C.D. Johnson, D. Ahlquist, CT colonography: unraveling the twists and turns, Curr. Opin. Gastroenterol. 21 (2005) 90–98.
[4] H. Yoshida, A.H. Dachman, CAD techniques, challenges, and controversies in computed tomographic colonography, Abdom. Imaging 30 (2005) 26–41.
[5] H. Yoshida, J. Näppi, Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps, IEEE Trans. Med. Imaging 20 (2001) 1261–1274.
[6] R.M. Summers, C.F. Beaulieu, L.M. Pusanik, J.D. Malley, R.B. Jeffrey Jr., D.I. Glazer, S. Napel, Automated polyp detector for CT colonography: feasibility study, Radiology 216 (2000) 284–290.
[7] R.M. Summers, M. Franaszek, M.T. Miller, P.J. Pickhardt, J.R. Choi, W.R. Schindler, Computer-aided detection of polyps on oral contrast—enhanced CT colonography, AJR Am. J. Roentgenol. 184 (2005) 105–108.
[8] G. Kiss, J. Van Cleynenbreugel, M. Thomeer, P. Suetens, G. Marchal, Computer-aided diagnosis in virtual colonography via combination of surface normal and sphere fitting methods, Eur. Radiol. 12 (2002) 77–81.
[9] D.S. Paik, C.F. Beaulieu, G.D. Rubin, B. Acar, R.B. Jeffrey Jr., J. Yee, J. Dey, S. Napel, Surface normal overlap: a computer-aided detection algorithm with application to colonic polyps and lung nodules in helical CT, IEEE Trans. Med. Imaging 23 (2004) 661–675.
[10] A.K. Jerebko, R.M. Summers, J.D. Malley, M. Franaszek, C.D. Johnson, Computer-assisted detection of colonic polyps with CT colonography using neural networks and binary classification trees, Med. Phys. 30 (2003) 52–60.
[11] J. Näppi, H. Frimmel, A.H. Dachman, H. Yoshida, A new high-performance CAD scheme for the detection of polyps in CT colonography, Med. Imaging 2004 Image Process. (2004) 839–848.
[12] A.K. Jerebko, J.D. Malley, M. Franaszek, R.M. Summers, Multiple neural network classification scheme for detection of colonic polyps in CT colonography data sets, Acad. Radiol. 10 (2003) 154–160.
[13] A.K. Jerebko, J.D. Malley, M. Franaszek, R.M. Summers, Support vector machines committee classification method for computer-aided polyp detection in CT colonography, Acad. Radiol. 12 (2005) 479–486.
[14] B.J. Kim, I.K. Kim, K.B. Kim, Feature extraction and classification system for nonlinear and online data, Adv. Knowl. Discovery Data Min. Proc. 3056 (2004) 171–180.
[15] W. Zheng, C. Zou, L. Zhao, An improved algorithm for kernel principal component analysis, Neural Process. Lett. 22 (1) (2005) 49–56.
[16] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 52 (8) (2004) 2165–2176.
[17] S. Ozawa, S. Pang, N. Kasabov, Incremental learning of chunk data for online pattern classification systems, IEEE Trans. Neural Networks 19 (6) (2008) 1061–1074.
[18] H.T. Zhao, P.C. Yuen, J.T. Kwok, A novel incremental principal component analysis and its application for face recognition, IEEE Trans. Syst. Man Cybernet. B—Cybernet. 36 (4) (2006) 873–886.
[19] Y.M. Li, On incremental and robust subspace learning, Pattern Recognition 37 (7) (2004) 1509–1518.
[20] Y. Kim, Incremental principal component analysis for image processing, Opt. Lett. 32 (1) (2007) 32–34.
[21] B.J. Kim, I.K. Kim, Incremental nonlinear PCA for classification, in: Knowledge Discovery in Databases (PKDD 2004), Proceedings, vol. 3202, 2004, pp. 291–300.
[22] B.J. Kim, J.Y. Shim, C.H. Hwang, I.K. Kim, J.H. Song, Incremental feature extraction based on empirical kernel map, Found. Intell. Syst. 2871 (2003) 440–444.
[23] L. Hoegaerts, L. De Lathauwer, I. Goethals, J.A.K. Suykens, J. Vandewalle, B. De Moor, Efficiently updating and tracking the dominant kernel principal components, Neural Networks 20 (2) (2007) 220–229.
[24] T.J. Chin, D. Suter, Incremental kernel principal component analysis, IEEE Trans. Image Process. 16 (6) (2007) 1662–1674.
[25] X. Jiang, Y. Motai, R. Snapp, X. Zhu, Accelerated kernel feature analysis, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, pp. 109–116.
[26] V.N. Vapnik, The Nature of Statistical Learning Theory, second ed., Springer, New York, 2000.
[27] R. Courant, D. Hilbert, Methods of Mathematical Physics, vol. 1, 1966, pp. 138–140.
[28] O.L. Mangasarian, A.J. Smola, B. Schölkopf, Sparse kernel feature analysis, Technical Report 99–04, University of Wisconsin, 1999.
[29] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: Proceedings of 17th International Conference on Machine Learning, 2000.
[30] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[31] J. Meltzer, S. Soatto, M.H. Yang, R. Gupta, Multiple view feature descriptors from image sequences via kernel principal component analysis, in: Computer Vision—ECCV 2004, 2004, pp. 215–227.
[32] S. Mika, B. Schölkopf, A. Smola, K. Müller, M. Scholz, G. Rätsch, Kernel PCA and de-noising in feature spaces, in: M.J. Kearns, S.A.

Solla, D.A. Cohn (Eds.), NIPS, MIT Press, Cambridge, MA, 1998, pp. 536–542.
[33] B. Schölkopf, S. Mika, C. Burges, P. Knirsch, K. Müller, G. Rätsch, A. Smola, Input space vs. feature space in kernel-based methods, IEEE Trans. Neural Networks 10 (5) (1999) 1000–1017.
[34] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.
[35] M.E. Tipping, Sparse kernel principal component analysis, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), NIPS, MIT Press, Cambridge, MA, 2000, pp. 633–639.
[36] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley, New York, 2001.
[37] J.T. Kwok, I.W. Tsang, The pre-image problem in kernel methods, in: T. Fawcett, N. Mishra (Eds.), ICML, AAAI Press, 2003, pp. 408–415.
[38] P. Sarma, L.J. Durlofsky, K. Aziz, Kernel principal component analysis for efficient, differentiable parameterization of multipoint geostatistics, Math. Geosci. 40 (1) (2008) 3–32.
[39] J. Franc, V. Hlavac, Statistical pattern recognition toolbox for Matlab, 〈http://cmp.felk.cvut.cz/~xfrancv/stprtool/〉, 2004.
[40] J. Li, A. Huang, J. Yao, J.M. Liu, R.L. Van Uitert, N. Petrick, R.M. Summers, Optimizing computer-aided colonic polyp detection for CT colonography by evolving the Pareto front, Med. Phys. 36 (1) (2009) 201–212.
[41] C. Robinson, S. Halligan, S.A. Taylor, S. Mallett, D.G. Altman, CT colonography: a systematic review of standard of reporting for studies of computer-aided detection, Radiology 246 (2) (2008) 426–433.
[42] D. Bielen, G. Kiss, Computer-aided detection for CT colonography: update 2007, Abdom. Imaging 32 (5) (2007) 571–581.