World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:11, 2009

Adaptive Kernel Principal Analysis for Online Feature Extraction

Mingtao Ding, Zheng Tian, and Haixia Xu

***Abstract*—**The batch nature limits the standard kernel principal component analysis (KPCA) methods in numerous applications, especially for dynamic or large-scale data. In this paper, an efficient adaptive approach is presented for online extraction of the kernel principal components (KPC). The contribution of this paper may be divided into two parts. First, kernel covariance matrix is correctly updated to adapt to the changing characteristics of data. Second, KPC are recursively formulated to overcome the batch nature of standard KPCA.This formulation is derived from the recursive eigen-decomposition of kernel covariance matrix and indicates the KPC variation caused by the new data. The proposed method not only alleviates sub-optimality of the KPCA method for non-stationary data, but also maintains constant update speed and memory usage as the data-size increases. Experiments for simulation data and real applications demonstrate that our approach yields improvements in terms of both computational speed and approximation accuracy.

***Keywords*—**adaptive method, kernel principal component analysis, online extraction, recursive algorithm.

I. Introduction

KERNEL principal component analysis (KPCA), which is a nonlinear extension of principal component analysis (PCA) [1], has gained significant attention as a learning machine [2] in pattern recognition [3-6], statistical analysis [7, 8] and image processing [9, 10]. The core idea of KPCA is to first map the input space into a feature space using a nonlinear mapping and then compute the principal components in the feature space. As a result, the extracted kernel principal component (KPC) of the mapped data is nonlinear with regards to the original input space.

The main challenge in KPCA for online feature extraction is its batch nature. On one hand, the batch nature hinders KPCA in terms of computation and memory demands as the data-size increases. For KPCA, the kernel Gram matrix has to be wholly available before eigendecomposition can be carried out. The arrival of a new data will require the addition of a new row and column for kernel Gram matrix, and eigendecomposition has to be constantly reevaluated for the ever-growing matrix. In addition, all data must be saved to represent the KPC. This translates into high costs for storage resources and computational load during run-time application of large datasets. On the other hand, the batch KPCA, which relies on a fixed model, cannot be employed for non-stationary data whose input distributions could be temporally changed. According to [11], the non-stationary property in input space will reappear in feature space when the nonlinear mapping is smooth and continuous. However, KPCA spends equivalent modeling power on the mapped data to extract the KPC. This is not desirable for tracking the changing characteristics of the input data.

In the last decade, several approaches have been proposed to deal with the batch nature of KPCA, and could be mostly grouped into one of the following three approaches: iterative KPCA based on Hebbian updates algorithms[10, 12], incremental KPCA[13, 14] and the greedy KPCA[15, 16]. The first kernelizes the generalized Hebbian algorithm which is an iterative self-organizing procedure for linear PCA. Whereas this method can potentially lower the time complexity of computing KPCA, it is unclear how to continue the iterations when a new data added to the training set and its convergence is comparatively slow. The second performs incremental singular value decomposition (SVD) in the kernel induced feature space and iteratively constructs reduced-set expansions to maintain constant update speed and memory usage. The third filters samples the original training set for a lesser but representative subset of vectors which span approximately the same subspace as the subspace in the kernel induced feature space spanned by the training set. However, in last two approaches, constructing of reduced-set expansion and prior filtering of the training data could be computationally expensive by themselves. Furthermore, these methods cannot be directly used to adapt the KPC to process non-stationary data.

This paper proposes an adaptive KPCA (AKPCA) method with rapid and accurate computation for online KPC extraction. Rather than directly operating on mapped kernel data or kernel Gram matrix, the proposed method commence with the kernel covariance matrix, from which the AKPCA arises. The basis of the solution lies in decomposing the new data into a component orthogonal and a component parallel to the previous KPC, and adaptively adjusting KPC based on the rank-1 update [17] of

This work was supported in part by the National Nature Science Foundation of China under Grant No. BS60972150, the Astronautics Basal Science Foundation of China under Grant No. 03I53059 and Science and Technology Innovation Foundation of Northwestern Polytechnical University under Grant No. 2007KJ01033.

Mingtao Ding* is with the Department of Science, Northwestern Polytechnical University, Xi'an, China (telephone: +86-13720603654, e-mail: dingmingtao@tom.com).

Zheng Tian is with the Department of Science and Department of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China and National Key Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China (e-mail: zhtian@nwpu.edu.cn).

Haixia Xu is with the Department of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an, China (e-mail: xuhaixia_xhx@163.com).

kernel covariance matrix. The contribution made in this paper is twofold: (1) recursively formalizing KPC to adapt to the changing characteristics of non-stationary data, (2) reducing the computational complexity and memory usage of KPCA to $O(N)$.

The rest of the paper is organized as follows. Section II presents a brief description of the KPCA method. Section III elucidates the proposed AKPCA approach, including the update of kernel covariance matrix, the recursive decomposition of kernel covariance matrix, and recursive KPC formulation. Section IV details the experiments and analysis aiming at assessing the performance of the proposed method. Section V draws some concluding remarks.

II. Kernel Principal Analysis

Given a data matrix $\mathbf{x}=(x_1,x_2,\cdots,x_n)\in\mathbb{R}^{m\times n}$, with $x_i\in\mathbb{R}^m$ being the data vector at time $i\geq 1$, KPCA nonlinearly maps $\mathbf{x}$ into a higher dimensional space F, and subsequently performs linear PCA in F. Assuming that the mapped data is centered in feature space (This assume will be removed later), its covariance matrix is given by

$$\mathbf{C}_n=\frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^T, \tag{1}$$

where ϕ is the nonlinear mapping function. The map ϕ is induced by a kernel function $k(\cdot,\cdot)$ that allows us to evaluate inner products in F: $k(x_i,x_j)=\phi(x_i)\bullet\phi(x_j)$, *for* $i,j=1,\cdots,n$. Given that the explicit mapping function ϕ is unknown, eigendecomposition can't be performed on $\mathbf{C}_n$ to compute the KPC. KPCA circumvents the KPC by a dual eigendecomposition problem for kernel Gram matrix $K_n\mathbf{a}^k=n\lambda_k\mathbf{a}^k$, in which $\mathbf{a}^k=(a_1^k,a_2^k,\cdots,a_n^k)^T$ is the normalized eigenvector[1]. Then, the r most significant KPC in feature space take the form of

$$\mathbf{V}_n=\left[\mathbf{v}^1,\mathbf{v}^2,\cdots,\mathbf{v}^r\right]=\left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n)\right]\mathbf{A}_n, \tag{2}$$

where $\mathbf{A}_n$ is the matrix with the columns of $\mathbf{a}^k$ associated with the k-th largest eigenvalues, $k=1,2,\cdots r$. Let $z\in\mathbb{R}^m$ be a test data, with an image $\phi(z)$, then its k-th principal component corresponding to ϕ is

$$\mathbf{p}(k)=\mathbf{v}^k\bullet\phi(z)=\sum_{i=1}^{n}a_i^k k(x_i,z), \tag{3}$$

where $\mathbf{v}^k$ is the k-th columns of $\mathbf{V}_n$.

III. Adaptive Kernel Principal Analysis

In this section, the batch nature and non-adaptability of KPCA is solved by a recursive KPC formulation, i.e., adjusting the KPC based on the effect of new data on the previous KPC. The updating in detail for one step of the iterative procedure is described and it compares favourablely to standard KPCA.

A. Exponential Forgetting Window for the Update of Kernel Covariance matrix

Assume that data $(x_1,x_2,\cdots,x_n)$ have been given with their KPCA result so far and the new data x_{n+1} is given for current processing. To estimate the kernel covariance matrix in a manner that de-emphasizes past observations, the exponential forgetting window for the update of kernel covariance matrix is used

$$\mathbf{C}_{n+1}=\varepsilon\mathbf{C}_n+(1-\varepsilon)\phi(x_{n+1})\phi(x_{n+1})^T, \tag{4}$$

where ε is the exponential forgetting factor, which is often set smaller than $\frac{n}{n+1}$ to de-emphasize the contribution of earlier observations.

With the previous KPCA result $\mathbf{C}_n\doteq\mathbf{V}_n\mathbf{\Lambda}_n\mathbf{V}_n^T$, $\phi(x_{n+1})$ can be decomposed into a component $\phi^{\mathbf{V}_n}(x_{n+1})$ parallel and component $\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})$ orthogonal to $\mathbf{V}_n$

$$\phi(x_{n+1})=\phi^{\mathbf{V}_n}(x_{n+1})+\phi^{\mathbf{V}_n^{\perp}}(x_{n+1}), \tag{5}$$

Following (3),

$$\phi^{\mathbf{V}_n}(x_{n+1})=\mathbf{V}_n\mathbf{p}_{x_{n+1}} \tag{6}$$

and

$$\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})=\left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]\begin{bmatrix}-\mathbf{p}_{x_{n+1}}\\1\end{bmatrix}, \tag{7}$$

where $\mathbf{p}_{x_{n+1}}$ reads entrywise $\mathbf{p}_{x_{n+1}}(k)=\mathbf{v}^k\bullet\phi(x_{n+1})$.

For $\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})$, the norm

$$\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|^2=k(x_{n+1},x_{n+1})-\mathbf{p}_{x_{n+1}}{}^T\mathbf{p}_{x_{n+1}}. \tag{8}$$

When $\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|$ is small enough to be modeled as noise, one can say $\phi(x_{n+1})$ is linearly dependent on $\phi(x_1),\phi(x_2),\cdots,\phi(x_n)$. The current KPC can be obtained by rotating the previous KPC in the kernel subspace to most faithfully preserve the covariance structure (Subsection III.B). Otherwise, $\phi(x_{n+1})$ is linearly independent on $\phi(x_1),\phi(x_2),\cdots,\phi(x_n)$, and new KPC should be added into consideration (Subsection III.C).

B. Rotate the KPC to most faithfully preserves covariance structure

If $\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|$ is smaller than a given threshold τ, then $\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})$ can be viewed as noise, and has no effect on the original KPC. In such case, the variation caused by x_{n+1} mainly depend on $\phi^{\mathbf{V}_n}(x_{n+1})$. Thus, (4) can be rewritten as the following recursion:

$$\mathbf{C}'_{n+1}=\varepsilon\mathbf{C}_n+(1-\varepsilon)\phi^{\mathbf{V}_n}(x_{n+1})\phi^{\mathbf{V}_n}(x_{n+1})^T. \tag{9}$$

[1] $\mathbf{a}^k$ should be scaled to ensure $\mathbf{a}^k\bullet\mathbf{a}^k=\frac{1}{n\lambda_k}$.

Substituting the previous KPCA result and (6) into (9), the kernel covariance matrix is rewritten by

$$\mathbf{C}'_{n+1} = \mathbf{V}_n \left(\varepsilon\mathbf{\Lambda}_n + (1-\varepsilon)\mathbf{p}_{x_{n+1}}\mathbf{p}_{x_{n+1}}{}^T\right)\mathbf{V}_n{}^T . \quad (10)$$

Denoting the eigendecomposition of the matrix $\varepsilon\mathbf{\Lambda}_n + (1-\varepsilon)\mathbf{p}_{x_{n+1}}\mathbf{p}_{x_{n+1}}{}^T$ as $\mathbf{UDU}^T$, where $\mathbf{U}$ is orthonormal and $\mathbf{D}$ is diagonal, (10) becomes

$$\mathbf{C}'_{n+1} = \mathbf{V}_n\mathbf{UDU}^T\mathbf{V}_n{}^T . \quad (11)$$

Consequently, the eigensystem variation of kernel covariance matrix caused by $\frac{1}{n+1}\phi^{\mathbf{V}_n}(x_{n+1})\phi^{\mathbf{V}_n}(x_{n+1})^T$ turns out to be

$$\mathbf{A}_{n+1} = \mathbf{A}_n\mathbf{U}$$
$$\mathbf{V}_{n+1} = \mathbf{V}_n\mathbf{U} = \left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n)\right]\mathbf{A}'_{n+1},$$
$$\text{and } \mathbf{\Lambda}_{n+1} = \mathbf{D} . \quad (12)$$

In (12), $\mathbf{U}$ represents directional variation of KPC caused by $\phi^{\mathbf{V}_n}(x_{n+1})$, and $\mathbf{D}$ represents component ration of the updated KPC.

C. Add new KPC to fit the orthogonal component

If $\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|$ is larger than the threshold τ, new KPC should be added into consideration. The first key observation is that previous KPC $\mathbf{V}_n$ can be rewritten as

$$\mathbf{V}_n = \left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]\begin{bmatrix}\mathbf{A}_n\\ \mathbf{0}\end{bmatrix}, \quad (13)$$

where $\mathbf{0}$ is a $1\times n$ vector of zeros. The second key observation is that $\phi^{\mathbf{V}_n}(x_{n+1})$ and $\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})$ can be rewritten as

$$\phi^{\mathbf{V}_n}(x_{n+1}) = \left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]\begin{bmatrix}\mathbf{A}_n\\ \mathbf{0}\end{bmatrix}\mathbf{p}_{x_{n+1}},$$

$$\phi^{\mathbf{V}_n^{\perp}}(x_{n+1}) = \left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]\begin{bmatrix}-\mathbf{A}_n\mathbf{p}_{x_{n+1}}\\ 1\end{bmatrix}. \quad (14)$$

Substituting (5), (13) and (14) into (4), the current kernel covariance matrix is given by

$$\mathbf{C}_{n+1} = \left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]\begin{bmatrix}\mathbf{A}_n & -\mathbf{A}_n\mathbf{p}_{x_{n+1}}\\ \mathbf{0} & 1\end{bmatrix}$$
$$\cdot\begin{bmatrix}\varepsilon\mathbf{\Lambda}_n + (1-\varepsilon)^2\mathbf{p}_{x_{n+1}}\mathbf{p}_{x_{n+1}}{}^T & (1-\varepsilon)^2\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|\mathbf{p}_{x_{n+1}}\\ (1-\varepsilon)^2\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|\mathbf{p}_{x_{n+1}}{}^T & (1-\varepsilon)^2\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|^2\end{bmatrix}$$
$$\cdot\begin{bmatrix}\mathbf{A}_n & -\mathbf{A}_n\mathbf{p}_{x_{n+1}}\\ \mathbf{0} & 1\end{bmatrix}^T\left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]^T . \quad (15)$$

Similar with (11), eigendecomposition of the matrix $\begin{bmatrix}\varepsilon\mathbf{\Lambda}_n + (1-\varepsilon)^2\mathbf{p}_{x_{n+1}}\mathbf{p}_{x_{n+1}}{}^T & (1-\varepsilon)^2\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|\mathbf{p}_{x_{n+1}}\\ (1-\varepsilon)^2\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|\mathbf{p}_{x_{n+1}}{}^T & (1-\varepsilon)^2\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\|^2\end{bmatrix}$ is denoted as $\mathbf{WEW}^T$, where $\mathbf{W}$ is orthonormal and $\mathbf{E}$ is diagonal.

Consequently, the eigensystem variation of kernel covariance matrix caused by $\frac{1}{n+1}\phi^{\mathbf{V}_n}(x_{n+1})\phi^{\mathbf{V}_n}(x_{n+1})^T$ turns out to be

$$\mathbf{A}_{n+1} = \begin{bmatrix}\mathbf{A}_n & -\mathbf{A}_n\mathbf{p}_{x_{n+1}}\\ \mathbf{0} & 1\end{bmatrix}\mathbf{W}$$
$$\mathbf{V}_{n+1} = \left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})\right]\mathbf{A}_{n+1},$$
$$\text{and } \mathbf{\Lambda}'_{n+1} = \mathbf{E} . \quad (16)$$

For $\left\|\phi^{\mathbf{V}_n^{\perp}}(x_{n+1})\right\| > \tau$, different from the results in (10), the spanning vectors of the current KPC are extended from $\left[\phi(x_1),\phi(x_2),\cdots,\phi(x_n)\right]$ to $\left[\phi(x_1),\cdots,\phi(x_n),\phi(x_{n+1})\right]$.

D. Recursive KPC formulation with mean update

For the sake of simplicity, it was assumed that all the mapped data are zero-mean. However, the assumption is often invalid and the sample mean of the training data maybe changes over time as new data arrive. One shall now drop this assumption.

Denoting the current means of mapped data as $\mu_{n+1} = (1-\varepsilon)\mu_n + \varepsilon\phi(x_{n+1})$, (4) can be rewritten as $C_{n+1} = \sum_{i=1}^{n+1}\varepsilon(1-\varepsilon)^{n+1-i}C_n\left(\phi(x_i)-\mu_{n+1}\right)\left(\phi(x_i)-\mu_{n+1}\right)^T$. By the scatter matrix update[19], it can be easily achieved that kernel covariance matrices have following recursive formula

$$C_{n+1} = (1-\varepsilon)C_n + \varepsilon(1-\varepsilon)\left(\phi(x_{n+1})-\mu_n\right)\left(\phi(x_{n+1})-\mu_n\right)^T . \quad (17)$$

The recursive KPC formulation of (17) can be computed following the above approach. The procedure will not be repeated here for obtaining the updated KPC because it is completely analogous with (5)~(16), only the mapped data $\phi(x_1),\phi(x_2),\cdots,\phi(x_n),\phi(x_{n+1})$ should be modified as

$$\phi(x_1)-\mu_n,\phi(x_2)-\mu_n,\cdots,\phi(x_n)-\mu_n,\sqrt{\frac{n}{n+1}}\left(\phi(x_{n+1})-\mu_n\right)$$

respectively. The computational issues in this modified procedure can be seen in Appendix A of [1].

E. AKPCA Algorithm

As a summary, the AKPCA algorithm can be detailed in steps as follows:

AKPCA Algorithm

Step 1. Obtain n data and perform batch KPCA.
Step 2. For $i = n+1, n+2, \cdots$
Step 3. Obtain x_i.
Step 4. Decompose $\phi(x_i)-\mu_{i-1}$ into a component parallel and a component orthogonal to previous KPC.
Step 5. If $\left\|\left(\phi(x_i)-\mu_{i-1}\right)^{\mathbf{V}^{\perp}}\right\| \le \tau$, go to **Step 6**; **otherwise**, skip to **Step 7.**
Step 6. Update the KPC by rotation (Subsection III.B) based on (17). Skip to **Step 8**.
Step 7. Update the KPC by adding a new kernel component. (Subsection III.C) based on (17).

Step 8. end if
Step 9. Update the implicit mean of mapped data.
Step 10. end for

By examining the algorithm in Table 1, it can see that the time complexity of each update is dependent on the mapped data which are used to span the KPC. The number of the data increases very slowly in the update procedure, and can be viewed as constant. So the proposed algorithm has linear time complexity. This is confirmed by the experiments in Section IV.

IV. Experiments

Several experiments were conducted to examine three properties of the proposed AKPCA algorithm: (1) the accuracy of in approximating batch KPCA; (2) empirical time complexity; (3) the effectiveness of the proposed method in visual tracking which essentially deals with non-stationary data. For the following, 'IKPCA' is defined as acronym of incremental KPCA method in[13].

The first experiment, which is carried on the 2000 simulation data in a non-stationary environment, serves to test the effectiveness of AKPCA in approximation accuracy. The two-dimensional data was generated in the following way: x -values have uniform distribution in $[-1,1]$, y -values are generated from $y_i = 0.002ix_i^2 + \xi$, where ξ is normal noise with standard deviation 0.2. The first 2000 vectors are plotted in Fig. 1.

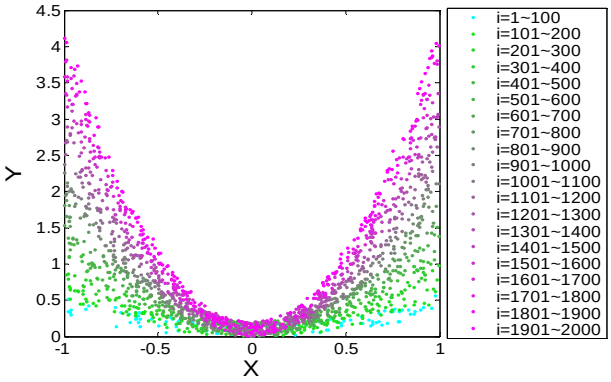

Fig. 1 Non-stationary simulation data.

The AKPCA and IKPCA are used to find the first r KPC of the data. The IKPCA parameters are $n = 40, r = 4, \tau = 0.12$, $\varepsilon = 0.9$, and $p = 40, r = 4, l = 1$ for IKPCA. The Gaussian kernel with $\sigma = 1$ is used. Since the characteristics of data varied as the time, some adding KPC were involved in the AKPCA processing. To assess the quality of the solutions of AKPCA and IKPCA, an alternative measure is adapted, i.e., the

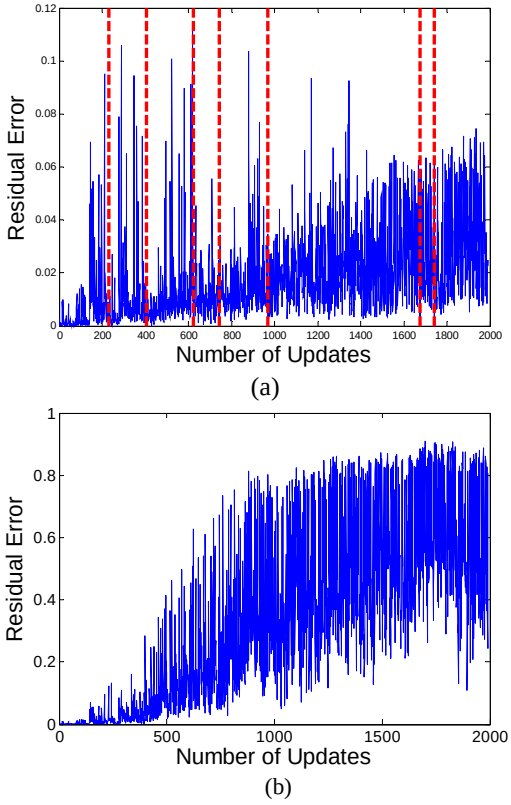

Fig. 2. The residual error of AKPCA (a) and IKPCA(b). In the AKPCA processing, seven new kernel components at time 230, 406, 623, 745, 970, 1674, 1741 are added into KPC (red dashed lines).

residual error of training data points in the RKHS[18]. Residual error measures distances between the data points and their projections onto the KPC. Fig. 2 shows the results. It can be seen that AKPCA overcomes the danger of drift, i.e., error accumulates across the update procedure causing the divergency between the AKPCA results and the ground truth results to grow indefinitely large.

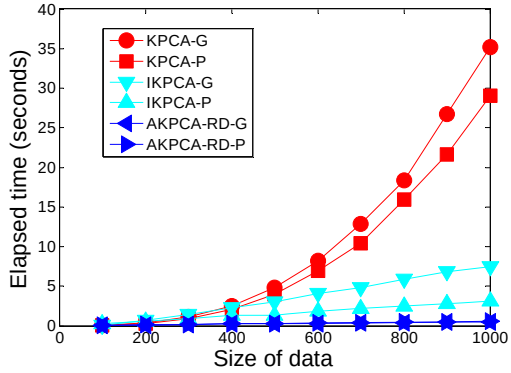

Fig. 3. Empirical time complexity of RKPCA with toy data

The second experiment serves to examine the empirical time complexity of AKPCA. The toy data [1]was created in various sizes and processed by AKPCA with $n = 20, r = 6, \tau = \infty$ and $\varepsilon(i) = \frac{i-1}{i}$. The Gaussian kernel ('AKPCA-G') with $\sigma = 1$ and third-degree polynomial kernel ('AKPCA-P') are used.

The processing time for each instance was recorded. The elapsed times of standard KPCA ('KPCA-G' for Gaussian kernel, 'KPCA-P' for Polynomial kernel) and IKPCA with $p = 10, r = 6, l = 1$ were also recorded for comparison. Fig. 5 (a) shows the results which confirm that processing time for AKPCA scales linearly with the problem size and KPCA scales with complexity $O(N^3)$. All curves in Fig. 3 were obtained by averaging results from 100 repetitions

Finally, we investigate the feasibility of AKPCA in non-stationary visual tracking. We advance the adaptive appearance model [19] by using AKPCA to train the model. Given an AKPCA model which encodes the recent appearance of the object of interest up to the previous frame, candidate object of the current frame would potentially enclose the KPC in this model. The object with the closest matching appearance is determined to be the solution. Multi-resolution face kernel [20] was used for the specific objects in the task of tracking. Fig. 4 shows AKPCA tracking results of four video sequences[2]. The AKPCA parameter are $n = 40, r = 8, \varepsilon = 0.9$ and $\tau = 0.4$ for the first 'fish' sequence, $\tau = 0.5$ for the second 'sylv' sequence, $\tau = 0.8$ for the third 'car4' sequence, $\tau = 1.4$ for the forth 'davidin'. As a qualitative benchmark, we also ran IKPCA on the same sequences, with $p = 10, r = 8, l = 5$. As can be seen in Fig. 4, our method provides comparable performance to the IKPCA tracker. Conclusion and Future Work

KPCA is a powerful method to extract the nonlinear feature of data. Although the adaptive algorithm for extracting KPC is important, it has not been reported so far. Our algorithm is devoted to filling this gap. In this paper, we proposed an AKPCA method for the online KPC extraction. This is achieved by formatting the recursive description of the eigensystem in kernel induced space and updating of the kernel covariance matrix to de-emphasize the past observations. The recursive description enables the proposed method to maintain constant update speed and memory usage in the learning process. The update of the kernel covariance matrix gives AKPCA the flexibility of accurately tracking the KPC. We show the accuracy and good computation properties of the proposed approach through several experiments.

Several possible improvements to the proposed method are of interest for future research. For instance, in this paper we update the KPC once for every new data. However, for a trade-off between computational efficiency and tracking ability, how often shall we update the KPC, and how to do it? Second, we wish to investigate the unsupervised threshold for the augmentation of new KPC.

V. Conclusion and Future Work

KPCA is a powerful method to extract the nonlinear feature of data. Although the adaptive algorithm for extracting KPC is important, it has not been reported so far. Our algorithm is devoted to filling this gap. In this paper, we proposed an AKPCA method for the online KPC extraction. This is achieved by formatting the recursive description of the eigensystem in kernel induced space and updating of the kernel covariance matrix to de-emphasize the past observations. The recursive description enables the proposed method to maintain constant update speed and memory usage in the learning process. The update of the kernel covariance matrix gives AKPCA the flexibility of accurately tracking the KPC. We show the accuracy and good computation properties of the proposed approach through several experiments.

Several possible improvements to the proposed method are of interest for future research. For instance, in this paper we update the KPC once for every new data. However, for a trade-off between computational efficiency and tracking ability, how often shall we update the KPC, and how to do it? Second, we wish to investigate the unsupervised threshold for the augmentation of new KPC.

[2]Available at http://www.cs.toronto.edu/~dross/ivt/.

Acknowledgment

The authors would like to thank Ph.D. David Ross for the supply of video sequences and partial source code. The authors also wish to thank the anonymous reviewers for their perceptive comments.

References

[1] B. Scholkopf, A. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299-1319, 1998.

[2] K. R. Muller, S. Mika, G. Ratsch *et al.*, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 181-201, 2001.

[3] J. Yang, Z. Jin, J. Y. Yang, D. Zhang, and A. F. Frangi , "Essence of kernel Fisher discriminant: KPCA plus LDA," *Pattern Recognition*, vol. 37, no. 10, pp. 2097-2100, 2004.

[4] H. Sahbi, "Kernel PCA for similarity invariant shape recognition," *Neurocomputing*, vol. 70, pp. 3034-3045, 2007.

[5] K. I. Kim, K. Jung, and H. J. Kim, "Face recognition using kernel principal component analysis," *IEEE Signal Processing Letters*, vol. 9, no. 2, pp. 40-42, 2002.

[6] X. L. Yu, X. G. Wang, and B. Y. Liu, "Supervised kernel neighborhood preserving projections for radar target recognition," *Signal Processing*, vol. 88, no. 9, pp. 2335-2339, 2008.

[7] G. Blanchard, O. Bousquet, and L. Zwald, "Statistical properties of kernel principal component analysis," *Machine Learning*, vol. 66, no. 2-3, pp. 259-294, 2007.

[8] X. Liu, U. Kruger, T. Littler, L. Xie, S.-Q. Wang, "Moving window kernel PCA for adaptive monitoring of nonlinear processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 96, no. 2, pp. 132-143, 2009.

[9] W.-J. Ma, H. Chen, and H.-Y. Tian, "Image registration based on piecewise linear group parameters," *Journal of Computer Applications*, pp. 976-8, 2007.

[10] K. I. Kim, M. O. Franz, and B. Scholkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans.Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1351-1366, 2005.

[11] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Tran. Neural Networks*, vol. 13, no. 3, pp. 780-784, 2002.

[12] S. Gunter, N. N. Schraudolph, and S. V. N. Vishwanathan, "Fast iterative kernel principal component analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1893-1918, 2007.

[13] T.-J. Chin, and D. Suter, "Incremental kernel principal component analysis," *IEEE Trans. Image Processing*, vol. 16, no. 6, pp. 1662-74, 2007.

[14] C. Tat-jun, and D. Suter, "Incremental Kernel PCA for Efficient Non-linear Feature Extraction," *British Machine Vision Conference*, Edinburgh, 4-7 September 2006, (III:939).

[15] V. Franc, and V. Hlavac, "Greedy algorithm for a training set reduction in the kernel methods," *Computer Analysis of Images and Patterns, Proceedings*, vol. 2756, pp. 426-433, 2003.
[16] V. Franc, "Optimization Algorithms for Kernel Methods," Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, Prague, 2005.
[17] K. B. Yu, "Recuresive updating the eigenvalue deccomposition of a covariance-matrix," *IEEE Trans. Signal Processing*, vol. 39, no. 5, pp. 1136-1145, 1991.
[18] N. Critianini, and J. Shawe-Taylor, *Kernel Method for pattern Analysis*, New York: Cambridge Univ. Press, 2004.
[19] D. A. Ross, J. Lim, R. S. Lin, and M-H, Yang , "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 125-141, 2008.
[20] M. Pavlou, and N. M. Allinson, "An unsupervised multiresolution face-kernel recognition model," *The Irish Machine Vision and Image Processing Conference*, Aug, 2005.

Fig. 4. A comparison of AKPCA (indicated with a green solid box) and IKPCA (depicted by a red dashed box) in visual tracking.