Incremental Kernel SVD for Face Recognition with Image Sets[*]

Tat-Jun Chin[†] Konrad Schindler David Suter
Institute for Vision Systems Engineering,
Monash University, Victoria, Australia.

Abstract

Non-linear subspaces derived using kernel methods have been found to be superior compared to linear subspaces in modeling or classification tasks of several visual phenomena. Such kernel methods include Kernel PCA, Kernel DA, Kernel SVD and Kernel QR. Since incremental computation algorithms for these methods do not exist yet, the practicality of these methods on large datasets or online video processing is minimal. We propose an approximate incremental Kernel SVD algorithm for computer vision applications that require estimation of non-linear subspaces, specifically face recognition by matching image sets obtained through long-term observations or video recordings. We extend a well-known linear subspace updating algorithm to the non-linear case by utilizing the kernel trick, and apply a reduced set construction method to produce sparse expressions for the derived subspace basis so as to maintain constant processing speed and memory usage. Experimental results demonstrate the effectiveness of the proposed method.

1. Introduction

Linear subspace methods have been found to be effective for modeling and classification of visual phenomena that are largely linear in nature. Some phenomena, however, are extremely non-linear, and kernel methods, such as Kernel SVD (KSVD), Kernel PCA (KPCA), Kernel QR (KQR) or Kernel Discriminant Analysis (KDA), that produce non-linear subspaces via the kernel trick are found to be superior compared to their linear counterparts. Examples in face recognition include [10, 12, 8].

Here, we focus on the face recognition paradigm that involves matching sets of face images [1, 10, 11]. The faces are usually captured under varying poses and illumination conditions. We concentrate on the case where the image set is very large due to long time observations or video recordings, thus there is a need for incremental updating of the face representation or classifier. We note that linear subspace methods have produced good results [11, 5] at high speeds due to the availability of incremental SVD computation algorithms [3], but have somewhat reached a performance barrier due to the linear assumption of the model (refer to [1]), while non-linear subspaces derived using kernel methods (e.g. KPCA [8], KQR [10]) can potentially deliver higher accuracy, but are impractical for large-scale data or online video processing due to the absence of incremental computation procedures.

In this paper, we propose an approximate incremental KSVD computation algorithm for non-linear subspace updating. A well-known incremental SVD computation procedure [3] is extended to the non-linear (kernel) case. We show how computations of the required elements of the algorithm can be obtained via the kernel trick so that explicit mappings of the input data to the kernel induced feature space can be avoided. Reduced set construction techniques [9] are applied at every iteration to produce sparse representations of the non-linear subspaces so that constant processing speed and memory usage can be achieved. The proposed algorithm is then applied on a face video database to demonstrate its effectiveness for face recognition.

2. Previous Work

The Mutual Subspace Method (MSM) was proposed in [11] for face recognition with temporal image sequences as inputs. A linear subspace is estimated from a sequence to represent a face and distances between subspaces are used to distinguish faces. The Kernel Mutual Subspace (KMS) method [8] was proposed to extend the MSM to accommodate non-linear data by using KPCA. The KMS method outperforms the MSM even though subspaces of lower dimension were used to represent face image sequences, indicating that non-linear subspaces derived using KPCA are more capable in capturing the complex manifold of face images under pose and illumination variations. However, the

[*]The project was part supported by the Australian Research Council Centre for Perceptive and Intelligent Machines in Complex Environments.
[†]Tat-Jun Chin is a recipient of the Endeavour Australia-Asia Award 2004 conferred by the Department of Education, Science and Training of the Australian Government.

KMS method is infeasible for large image sets or online video processing due to KPCA being computable in a batch manner (using all available images) only.

The concept of *kernel principal angles* was introduced in [10] for pattern classification tasks that involve matching vector sets. Essentially the algorithm performs a QR decomposition of vector sets mapped to the kernel induced feature space to obtain linear subspaces (c.f. non-linear in original data space) and finds the principal angles between the subspaces. A good result for face recognition over a small database was reported. Nonetheless, it is unclear how the method can be extended to handle large vector sets or online processing.

An iterative algorithm for performing KPCA, called the Kernel Hebbian Algorithm (KHA), was proposed in [6]. The KHA is based on the Generalized Hebbian Algorithm (GHA) which was introduced as an online algorithm for PCA. The KHA showed comparable results to KPCA and enabled the processing of large databases by iteratively subjecting each image to the GHA neural network over multiple passes. Here, we seek an iterative and incremental computation algorithm for KSVD so that it is unnecessary to consider all available images more than once.

3. The SVD and Face Recognition

We start by creating a data matrix $a = [x_1 \ \cdots \ x_n] \in \mathbb{R}^{m \times n}$ from our image set, with $x_i \in \mathbb{R}^m$ being the i-th input image. Without loss of generality, we will assume that $m > n$. Given the SVD of a, i.e. $a = U\Sigma V^T$, we compute the rank-r *singular value factorization* of a, given by

$$a^r = U^r \Sigma^r (V^r)^T \ , \tag{1}$$

with $r < n$, $U^r = U(:, 1:r)$, $V^r = V(:, 1:r)$ and $\Sigma^r = \Sigma(1:r, 1:r) = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$ by using a Matlab notation.

In face recognition, we use $\mathrm{span}(U^r)$ to represent the face which generated image set a. U^r is an orthonormal basis for an r-dimensional subspace that minimizes

$$\| a - a^r \|_F \ , \tag{2}$$

where $a^r = U^r (U^r)^T a$ is the rank-r approximation of a and $\| \cdot \|_F$ indicates the Frobenius norm. Distances between subspaces are then used to classify face image sets [11, 5, 10].

If the vectors in a are centered prior to performing the SVD, the resulting subspace basis U^r are the principal components of a. Strictly speaking, the SVD is a tool for performing PCA, but some applications might require a raw SVD instead of a PCA. This subtle difference produces vastly different subspaces, and it should be noted that for our purpose here the class-specific subspaces are derived without data centering.

3.1. The Kernel SVD

Non-linear subspaces might be more suitable to describe facial image sets which are known to lie on highly complex manifolds. To this end, we non-linearly map the input data to a higher dimensional space using the function $\phi : \mathbb{R}^m \longrightarrow \mathcal{F}$ and perform the SVD in $\mathcal{F}$. The map ϕ is induced by a kernel function $k(\cdot, \cdot)$ that evaluates the inner product between mapped input data in feature space. If $k(\cdot, \cdot)$ is an appropriately chosen *Mercer* kernel, then $\mathcal{F}$ has the structure of a so-called *Reproducing Kernel Hilbert Space* (RKHS). See [9, 4] for more details.

Using ϕ, we transform a into $A = [\phi(x_1) \ \cdots \ \phi(x_n)]$. Consider $M = A^T A$ and its eigenvalue decomposition $M = Q \Delta Q^T$. Matrix M contains inner products between the columns of A and can be computed via the kernel function without having to evaluate $\phi(x_i)$. The rank-r singular value factorization of A is then

$$A^r = \left[A Q^r (\Delta^r)^{-\frac{1}{2}} \right] \left[(\Delta^r)^{\frac{1}{2}} \right] \left[(Q^r)^T \right] \equiv U^r \Sigma^r (V^r)^T \ , \tag{3}$$

with $Q^r = Q(:, 1:r)$ and $\Delta^r = \Delta(1:r, 1:r)$. This is the *kernel singular value decomposition* (KSVD) [4]. M is positive semi-definite if it is constructed using a Mercer kernel.

Observe that the basis U^r is stored implicitly as linear combinations of mapped input vectors: $U^r = AQ^r (\Delta^r)^{-\frac{1}{2}} := A\alpha$. Given another basis $X^r = B\mu$ obtained via KSVD from mapped input data B, we can compute

$$D = (U^r)^T X^r = \alpha^T A^T B \mu \tag{4}$$

using the kernel function for $A^T B$. By invoking the SVD on D, we obtain $Y^T D Z = \theta$, where $\mathrm{diag}(\theta) = \{\theta_1, \cdots, \theta_r\}$ are the principal angles between $\mathrm{span}(U^r)$ and $\mathrm{span}(X^r)$. Functions on θ can be used to construct distance measures between subspaces (see [10] for more details).

4. Approximate Incremental Kernel SVD

Incremental computation of the SVD is motivated by the infeasibility of performing the SVD on large matrices or when matrices are growing in real-time. A similar need holds for KSVD. Here, we propose an approximate incremental KSVD updating procedure by extending the method for the linear case detailed in [3].

4.1. Incremental SVD Computation

Assume we have made a rank-r approximation of A and obtained the corresponding elements U^r, Σ^r and V^r. For incremental computations, we want to discard A once a singular value factorization is achieved so we should not rely on A to update U^r, Σ^r and V^r when new data arrives. Given new columns $C \in \mathbb{R}^{m\times c}$, we would like to find the SVD of $[A^r\ C]$. We can decompose $[A^r\ C]$ as such:

$$\left[\begin{array}{cc} A^r & C \end{array}\right] = \left[\begin{array}{cc} U^r & J \end{array}\right] \left[\begin{array}{cc} \Sigma^r & L \\ 0_{c\times r} & K \end{array}\right] \left[\begin{array}{cc} V^r & 0_{n\times c} \\ 0_{c\times r} & I_c \end{array}\right]^T , \tag{5}$$

with $0_{a\times b}$ denoting a zero matrix of size $a\times b$ and I_a signifying an identity matrix of size $a\times a$. The other components that make up (5) are defined as $L = (U^r)^T C$, $H = C - U^r L$ and $JK \xleftarrow{\text{QR}} H$. For geometrical interpretations of these components, refer to [3].

Let E, F and G^T be respectively defined as the left, middle and right matrix of the right-hand-side of (5). We can diagonalize F by invoking the SVD, i.e. $F = U'\Sigma'(V')^T$, and substitute it into (5), yielding the updated SVD:

$$\left[\begin{array}{cc} A^r & C \end{array}\right] = U''\Sigma'(V'')^T , \tag{6}$$

with $U'' = EU'$ and $V'' = GV'$. U^r, Σ^r and V^r are then revised as $U^r \longleftarrow U''(:,1:r)$, $\Sigma^r \longleftarrow \Sigma'(1:r,1:r)$ and $V^r \longleftarrow V''(:,1:r)$. For potential numerical issues of the procedure and their solutions, refer to [3]. Note that the update on U^r is dependent only on C and Σ^r. V^r can be discarded for applications that do not make use of it.

4.2. Incremental Kernel SVD Computation

Here, we extend the incremental SVD procedure to the non-linear (kernel) case. The key to update the r-dimensional basis U^r is to store Σ^r and compute J, K and L for new data C. The equivalent procedure in feature space can be performed by making use of the kernel function.

Let U^r be the r-dimensional basis for a linear subspace in feature space obtained from n mapped input images $A = [\phi(x_1)\ \cdots\ \phi(x_n)]$. U^r is computed using the KSVD and thus each basis vector is defined in terms of a linear combination of mapped input vectors, i.e. $U^r = A\alpha$ with the elements of $\alpha \in \mathbb{R}^{n\times r}$ as the coefficients of linear expansions. Trivially, we can compute matrix L as

$$L = \alpha^T A^T C , \tag{7}$$

where $A^T C$ can be evaluated via the kernel function since it contains inner products between columns of A and C.

Defining H is slightly more complicated as it is a function of A as well as C. Nonetheless, we can still express it as linear combinations of mapped input space vectors:

$$H = \left[\begin{array}{cc} A & C \end{array}\right] \left[\begin{array}{c} -\alpha L \\ I_c \end{array}\right] := A'\beta' , \tag{8}$$

with $A' = [A\, C]$ by definition, and β' contains the coefficients of linear combination.

From the previous section, the QR decomposition was applied to obtain an orthonormal basis J for H. Although iterative kernel QR algorithms exist [10], we can derive an equivalent orthonormal basis by performing a KSVD (which is non-iterative) on H and retain all left singular vectors to form J. We can compute the kernel matrix for H as

$$M_H = (\beta')^T (A')^T A'\beta' = (\beta')^T M'\beta' \tag{9}$$

and perform the procedure detailed in Section 3.1. M' can be evaluated using the kernel function on input images from A and C. Hence, M' will be positive semi-definite and this will ensure the positive semi-definiteness of M_H as well.

Let the eigenvalue decomposition of $M_H = Q_H \Delta_H Q_H^T$. From (3), the matrix J and K would then be

$$J = A'\beta' Q_H \Delta_H^{-\frac{1}{2}} := A'\Omega , \quad K = \Delta_H^{\frac{1}{2}} Q_H^T . \tag{10}$$

Matrix F is then constructed as shown in (5) and diagonalized to result in $F = U'\Sigma'(V')^T$. The left singular vectors of the matrix $[A^r\ C]$ are

$$\begin{aligned} U'' &= \left[\begin{array}{cc} A\alpha & A'\Omega \end{array}\right] U' , & (11)\\ &= \left[\begin{array}{cc} A & C \end{array}\right] \Psi , & (12) \end{aligned}$$

with $\Psi = \left[\begin{array}{c} \alpha \\ 0_{c\times r} \end{array}\right] U'(1:r,:) + \Omega U'((r+1):(r+c),:)$. The basis of the r-subspace that approximates the current image set is then updated as $U^r \longleftarrow [A\ C]\,\Psi(:,1:r)$.

4.3. Using Reduced Set Construction Methods

Up to this stage our incremental KSVD computation is still exact, but observe that in this procedure it is unavoidable that the updated basis U^r is defined in terms of linear combinations of old and new mapped image vectors. Thus, although we are spared from computing a batch KSVD at every iteration, we still have to store all seen images. This is detrimental towards speed and memory usage.

To solve this problem, we seek a sparser expression for U^r after every update. Close approximations for the basis vectors can be achieved by constructing a *reduced set expansion* for each vector. Specifically, we apply the *Iterated Pre-Images* approach detailed in [9]: Suppose we have an updated basis U^r. A basis vector will be expanded as

$$u = \sum_{i=1}^{n+c} \psi^i\, \phi(x_i) . \tag{13}$$

We seek a *pre-image* y so that $u = \phi(y)$. Most likely an exact pre-image will not exist (see [9]), and we will have to approximate u with a series of approximate pre-images:

$$u \approx \sum_{i=1}^{b} \eta^i\, \phi(y_i) , \tag{14}$$

where $b < n+c$. The intricacies on how to construct this reduced set (RS) expansion are beyond the scope of this paper. Refer to [9] for a detailed treatment.

Experimental results have shown that such a RS expansion is able to achieve much sparser representations for SVMs while causing no obvious deterioration in performance [9]. The same observation was made for KPCA in [7]. For the pseudo code of the proposed incremental KSVD with RS expansions, see Algorithm 1.

Algorithm 1 The proposed incremental KSVD with RS constructions for non-linear subspace updating.

```
Initialize parameters $R$, $Nini$, $Ninc$ and $Znum$.
Obtain $Nini$ data and perform batch KSVD.
Retain $R$ dimensional basis.
Store data and linear expansion coefficients.
while Image set is not exhausted do
    Obtain $Ninc$ data.
    Update SVD (refer to Section 4.2).
    Update $R$ dimensional basis.
    Update linear expansion coefficients.
    Append new data to storage.
    if (Number of stored data) $>$ ($R \times Znum$) then
        Create $Znum$ pre-images to span each basis vector.
        Overwrite stored data with all pre-images.
        Update linear expansion coefficients.
    end if
end while
```

4.4. Parameter Selection and Drifting

Due to the incremental nature of the algorithm, differences between the iteratively learned subspace from the ground truth subspace is unavoidable. However, it is of interest to keep drifting as minimal as possible. Several input parameters of the algorithm influence the severity of drifting. At this stage, the parameters discussed below are selected experimentally to achieve best results (see Section 5).

In incremental SVD (and KSVD) computations, since a fair sample of the underlying data distribution is unavailable initially, prior knowledge about the subspace on which most of the data lie is reflected through the selection of the subspace dimensionality R. Setting R too low might result in crucial information about the data distribution being projected away during each iteration and causing severe drifting of the learned subspace, whereas if R is too large, excessive computations at each iteration will slow down the system.

In addition, the number of pre-images per basis vector *Znum* for an RS expansion affects drifting as well. Clearly the value $R \times Znum$ must be much less than the total number of data (if known in advance) to realize the benefit of constructing RS expansions. Most of the time however, the larger *Znum* is, the RS expansion will more accurately approximate a feature space vector (see [9] for more details). Hence, setting a *Znum* too low might result in loose approximations of the basis vectors and severe subspace drifting. On the other hand, a high *Znum* will increase the computational load per update and cripple the system speed.

5. Experimental Results

We define acronyms for batch KSVD, the proposed incremental KSVD (without RS constructions) and incremental KSVD with RS constructions respectively as B-KSVD, I-KSVD and I-KSVD-RS. Firstly, a 2D toy problem of 300 data points was created. Refer to Figure 1: x-values have uniform distribution in $[-1, 1]$, whereas y-values are computed from $x^2 + \varepsilon$, where ε is normal noise with standard deviation 0.2. Evaluated visually, the I-KSVD basis is almost identical to the B-KSVD basis. The I-KSVD-RS basis expectedly differs somewhat from the ground truth. Nonetheless, from the largely similar contour patterns, it can be seen that the I-KSVD-RS derived subspace manages to preserve the overall structure of the ground truth subspace obtained from B-KSVD. For all cases, the Gaussian kernel with $\sigma = 10$ was used.

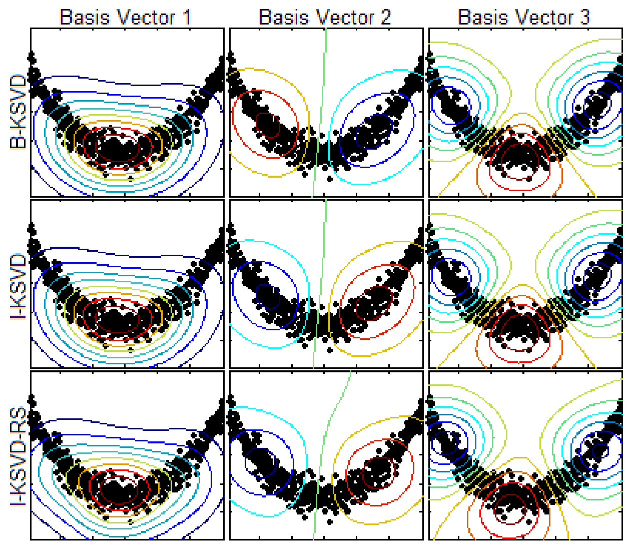

Figure 1. A 2D toy problem. Contour lines show projection values onto basis vectors.

To evaluate drift severity, memory requirements and speed of I-KSVD-RS, the algorithm was run offline on an image sequence of 3600 face images captured at 12 fps over 5 minutes. The recording was done as candidly as possible, and the subject was encouraged to perform arbitrary

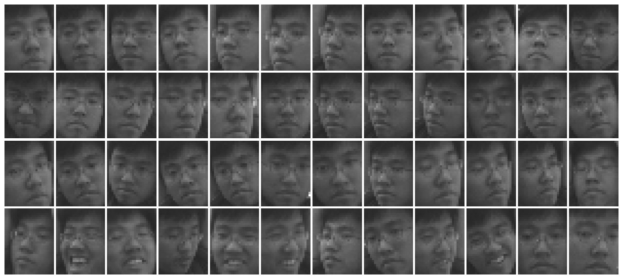

Figure 2. Example frames of the face video sequence used as input to I-KSVD-RS.

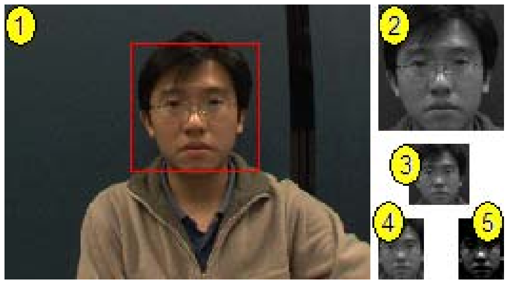

Figure 3. Face image preprocessing. Step 1: Face detector output. Step 2: Color to greyscale conversion. Step 3: Resize to 32×32 pixels. Step 4: Discard first and last 4 columns. Step 5: Adjust overall intensity to zero mean and one unit standard deviation.

movements like fidgeting, swiveling on the chair, translation, head rotation and talking. A face detector implementation of [2] was executed simultaneously at frame rate to crop the face region. Preprocessing described in Figure 3 was done on the face sub-image before raster-scanning it to form a column vector. Finally, the column vectors were concatenated to form an image set. See Figure 2 for example frames (minus Step 5 of preprocessing) of the sequence. Observe the variations in pose, expression, scaling and face localization.

The parameters for the I-KSVD-RS algorithm are $Nini = 20$, $Ninc = 20$, $R = 5$ and $Znum = 20$. As the update progresses, distances to other subspaces were computed. The subspace distance measure used was $\min(\theta_1, \cdots, \theta_5)$, where $\{\theta_1, \cdots, \theta_5\}$ are the principal angles between two subspaces. From Figure 4(a), some amount of drifting from the B-KSVD subspace evidently occurs but does not seem to grow exponentially. In fact, the I-KSVD-RS and B-KSVD subspaces seem to be converging as more samples are seen. Furthermore, the incrementally learned subspace does not "move" towards test subspaces estimated from other faces, indicating that the discrepancy between B-KSVD and I-KSVD-RS subspaces are acceptable for face recognition. From Figure 4(b), I-KSVD-RS manages to update in largely constant durations, whereas for I-KSVD time per update grows exponentially. As expected without RS expansions memory requirements increase linearly for I-KSVD as the sequence progresses whereas I-KSVD-RS maintains constant memory usage. See Figure 4(c). All algorithms were implemented in Matlab on a Pentium 4 2.8Ghz machine with 512MB of memory.

To evaluate the effectiveness of I-KSVD-RS for face verification in comparison to B-KSVD, a face video database of 9 individuals with 3 recording sessions each was created. By designating 1 session for each individual for training and the other 2 as test sets, we have 18 true claimant accesses and 216 impostor attacks. The recordings were done in the manner described previously. Each session is about 15s resulting in 180 vector image sets which are preprocessed as Figure 3 describes. Figure 5 shows samples from the database. The I-KSVD-RS parameters are $Nini = 20$, $Ninc = 20$, $R = 5$ and $Znum = 20$, while for B-KSVD, 5-dimensional subspaces were used. For the kernel methods, the Gaussian kernel with $\sigma = 170$ was used. The MSM method (the linear version) was implemented for comparison. The distance measure used is the same as above. The MSM returned an Equal Error Rate (EER) of 8.18%, while B-KSVD and I-KSVD-RS produced EERs of 7.25% and 7.41%. Refer to Figure 6 for ROC curves. Whilst all 3 methods gave comparable results, the MSM used 25-subspaces compared to 5-subspaces of B-KSVD and I-KSVD-RS. More crucially, by using I-KSVD-RS no obvious deterioration in classification performance compared to B-KSVD was observed.

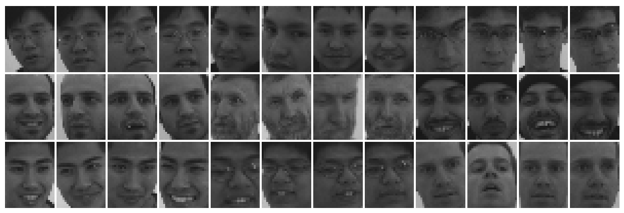

Figure 5. Samples from the face database.

6. Conclusion and Future Work

We proposed an approximate incremental KSVD algorithm for applications in image modeling or pattern classification. The algorithm is an extension of an SVD updating procedure to the non-linear (kernel) case for incremental estimation of non-linear subspaces. Reduced set construction methods are used to compress representations of the non-linear subspaces so that constant speed and memory usage can be achieved during updating. Experiments were done to

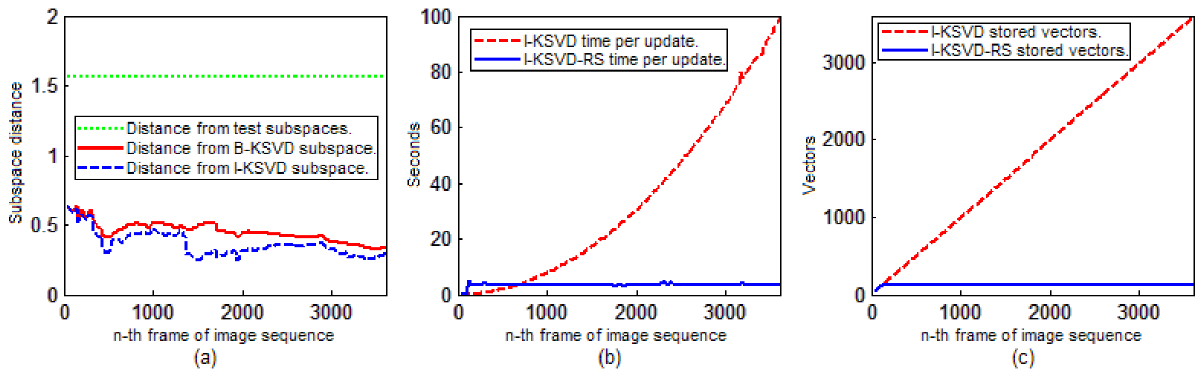

Figure 4. Drift, computation time and memory usage during updates. (a) Distance of the I-KSVD-RS derived subspace from other subspaces. (b) Computation time per update of I-KSVD and I-KSVD-RS. (c) Storage size (number of image vectors) required after each update of I-KSVD and I-KSVD-RS.

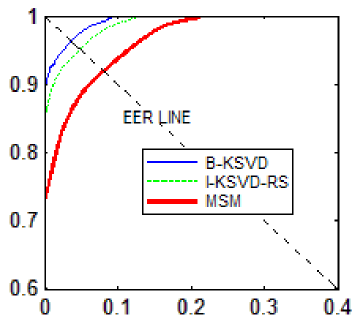

Figure 6. ROC curves for face recognition using 3 different algorithms.

evaluate the performance of the algorithm. In particular, it displayed no obvious deterioration in classification performance for face recognition with image sets. The algorithm is generic and should be beneficial for applications that involve non-linear subspace estimation from large data sets.

Although currently running at an average speed of about 4 frames per second, through proper implementation, substantial speed-up of the algorithm is achievable. This should enable online processing of video sequences and subsequently face recognition from video. Secondly, to realize the advantage of kernel methods for face recognition, efforts should be expended to select or construct kernel functions that can considerably boost the performance of face recognition systems based on kernel methods.

References

[1] O. Arandjelovic, G. Shakhnarovich, J. Fisher, R. Cipolla, and T. Darrell. Face recognition with image sets using manifold density divergence. In *IEEE CVPR*, 2005.

[2] G. Bradski, A. Kaehler, and V. Pisarevsky. Learning-based computer vision with Intel's open source computer vision library. *Intel Technology Journal*, 9(2):119–130, 2005.

[3] M. Brand. Incremental singular value decomposition of uncertain data with missing value. In *ECCV*, 2002.

[4] N. Cristianini and J. Shawe-Taylor. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[5] K. Fukui and O. Yamaguchi. Face recognition using multiviewpoint patterns for robot vision. In *10th International Symposium of Robotics Research*, 2003.

[6] K. I. Kim, M. O. Franz, and B. Schölkopf. Iterative kernel principal component analysis for image modeling. *IEEE PAMI*, 27(9):1351–1366, 2005.

[7] J. Meltzer, M.-H. Yang, R. Gupta, and S. Soatto. Multiple view feature descriptors from image sequences via kernel principal component analysis. In *ECCV*, 2004.

[8] H. Sakano, N. Mukawa, and T. Nakamura. Kernel mutual subspace method and its application for object recognition. *Electronics and Communications in Japan*, 2/88, 2005.

[9] B. Schölkopf and A. Smola. *Learning with kernels*. The MIT press, 2002.

[10] L. Wolf and A. Shashua. Learning over sets using kernel principal angles. *JMLR*, 4:913–931, 2003.

[11] O. Yamaguchi, K. Fukui, and K. Maeda. Face recognition using temporal image sequence. In *IEEE AFGR*, pages 318–323, 1998.

[12] M.-H. Yang, N. Ahuja, and D. Kriegman. Face recognition using kernel eigenfaces. In *IEEE ICIP*, 2000.