PERGAMON

Pattern Recognition 36 (2003) 1945–1959

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Eigenspace updating for non-stationary process and its application to face recognition

Xiaoming Liu[a], Tsuhan Chen[a,*], Susan M. Thornton[b]

[a] *Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213-3890, USA*
[b] *Sonic Foundry, Inc., 12300 Perry Highway, Wexford, PA 15090, USA*

Accepted 15 January 2003

Abstract

In this paper, we introduce a novel approach to modeling non-stationary random processes. Given a set of training samples sequentially, we can iteratively update the eigenspace to manifest the current statistics provided by each new sample. The updated eigenspace is derived based more on recent samples and less on older samples, controlled by a number of decay parameters. Extensive study has been performed on how to choose these decay parameters. Other existing eigenspace updating algorithms can be regarded as special cases of our algorithm. We show the effectiveness of the proposed algorithm with both synthetic data and practical applications on face recognition. Significant improvements have been observed on face images with different variations, such as pose, expression and illumination variations. We expect the proposed algorithm to have other applications in active recognition and modeling as well.

Keywords: Principal component analysis; Eigenspace updating; Non-stationary process; Face recognition

1. Introduction

The principal component analysis (PCA) [1] has attracted much attention among image analysis researchers. The basic idea is to represent images or image features in a transformed space where the individual features are uncorrelated. The orthonormal basis functions for this space, called the eigenspace, are the eigenvectors of the covariance matrix of the images or image features. PCA gives the optimal representation of the images or image features in terms of the mean square error.

PCA has been used extensively by researchers in many fields, such as data compression [2], feature extraction [3], and object recognition [4]. One of the successful applications of PCA is introduced in Ref. [5] and later made popular by Turk and Pentland [6]. They projected a face image into an eigenspace that is trained by all the images of multiple subjects, and performed face recognition in this eigenspace.

There are mainly two approaches to training the eigenspace in the literature. The first approach is to compute the eigenvectors given a set of training samples simultaneously, which we refer to as *batch training*. In this approach, PCA can be computationally intensive when it is applied in the image domain. The power method [7] is one approach to efficiently determine the dominant eigenvectors. Instead of determining all the eigenvectors, the power method obtains only the dominant eigenvectors, i.e., eigenvectors associated with the largest eigenvalues. Researchers have explored how to perform PCA more efficiently. Turk and Pentland [6] calculated the eigenvectors of an inner-product matrix instead of a covariance matrix, which is efficient in the case where the number of training

* Corresponding author. Tel.: +1-412-268-7536; fax: +1-412-268-3890.
E-mail addresses: xiaoming@andrew.cmu.edu (X. Liu), tsuhan@cmu.edu (T. Chen), sthornton@sonicfoundry.com (S.M. Thornton).

0031-3203/03/$30.00
doi:10.1016/S0031-3203(03)00057-8

samples is less than the dimension of the feature space. The second approach is to iteratively re-calculate the existing eigenvectors by taking the training samples one by one, which is called *eigenspace updating*, proposed for its computational efficiency compared to the batch training approach [8]. A few other researchers have proposed different methods of eigenspace updating [9,10], and suggested some interesting applications, such as salient view selections [10].

PCA is originally created to model multidimensional random variables. When extended to modeling random *processes*, traditional PCA works well as long as the random process under consideration is stationary. For non-stationary random processes, PCA needs to be adapted to model the time-varying statistics. To some extent, existing eigenspace updating methods [8–10] already accomplish this implicitly, because they all try to compute eigenvectors iteratively as samples come in one by one. In this paper, we propose a new eigenspace updating method to consider non-stationary random processes explicitly by modifying these methods in literature. In particular, our method puts more weight on recent samples than on older samples by using certain decay parameters, while traditional methods consider all samples equally and hence cannot effectively represent the most recent statistics of the data. We have also studied how to choose the decay parameters to model the time-varying statistics in the least mean squares sense.

For decades human face recognition has been an active topic in the field of object recognition. A general statement of this problem can be formulated as follows: given still or video images of a scene, identify one or more persons in the scene using a stored database of faces [11]. There are mainly two kinds of face recognition systems: the feature matching-based approach and the template matching-based approach. In the latter, applying PCA to obtain a face model (also known as the eigenface approach [6]) plays a fundamental role. It has good performance for the case of frontal face recognition with reasonable constraints on illumination, expression variations, etc. However, in practical applications, when large variations, which may be due to aging, changes in expressions and poses, and variations caused by illumination, etc., appear in the test face images, the traditional PCA algorithm degrades quickly in performance. Although some methods in literature work well for the specific variations being studied, their performance degrades rapidly when other variations are present [12]. In order to approach this general problem, we propose an updating-during-recognition scheme, which tries to make the recognition system more intelligent by learning the variations over time using the testing images. In this paper, we utilize our eigenspace updating method to learn the time-varying statistics of the face images and eventually enhance the recognition performance. We use individual PCA [13,14], instead of the universal PCA approach [6], as a baseline of our face recognition system.

1.1. Previous work

The eigenspace updating has a number of advantages. First, using the updating algorithm, we can determine the eigenvectors more efficiently than the batch training approach [8]. Second, the updating algorithm allows the construction of the eigenspace via a procedure that uses less storage, so it renders feasible some previous inaccessible problems, such as the training of a huge image data set [9]. Third, the availability of the training data may be constrained in some applications, such as online training [10]. In that case, we have to iteratively perform PCA instead of waiting for all the training data to be available.

Murakami and Kumar [8] proposed the first eigenspace updating algorithm. They iteratively generated the inner-product matrix and calculated the eigenvectors whenever there is a new training sample. Chandrasekaran et al. [10] also proposed an updating algorithm by performing the singular value decomposition (SVD). They showed the effectiveness of their algorithm in 3D object representation from 2D images, which is useful in active recognition and exploration. However, these two methods have limitations when used for classification because they assume the samples have zero mean. Hall et al. [9] addressed this issue and proposed an updating method where the mean is updated based on existing samples, and removed before PCA is performed. They showed that for classification, better performance could be obtained by their approach compared to those in Refs. [8,10]. Hall et al. [15] also proposed an algorithm to efficiently merge and split eigenspace models. All of these existing eigenspace updating methods, original designed to model the statistics of random vectors, also work for stationary random processes. They however cannot handle non-stationary processes to represent the time-varying statistics effectively, which is what our proposed algorithm tries to address.

Some researchers have tried to utilize the information provided by the new data when the system is in use to enhance the performance of face detection and tracking [16–19]. For example, Kurita et al. [16] iteratively updated the prior probabilities of the face location in the previous frames, which can guide and speed up the face detection on the current frame. Edwards et al. [17] described a method of updating the first order global estimation of the identity, which was integrated with an optimal tracking scheme. Wu et al. [18] proposed to build a subspace representation via the Gram–Schmidt orthogonalization procedure for the purpose of video compression. Weng et al. [19] proposed to incrementally derive discriminating features from training video sequences. Compared to the prior work, our work extends the idea of updating to face recognition and results in an updating-during-recognition scheme.

1.2. Paper outline

In Section 2, we introduce our eigenspace updating algorithms. Two algorithms aimed at different application

scenarios are presented in detail. When PCA is applied to the high-dimensional image domain, we use the algorithm based on updating an inner-product matrix. Otherwise, an algorithm based on updating a covariance matrix can be used.

In our eigenspace updating method, the decay parameters play a key role on how well the time-varying statistics can be modeled. Thus in Section 3, we theoretically and experimentally show how to choose the decay parameters based on the knowledge of model statistics.

In Section 4, we address the issue of iteratively updating the individual eigenspace for face recognition. Given one test image, we can use it to update the eigenspace when we have high confidence for its recognition result. Also we propose to use a twin-subspace method to alleviate some limitations and enhance the face recognition performance.

Experimental results using eigenspace updating methods are presented in Section 5. We conduct experiments on face databases containing different variations, such as poses, expressions and illuminations. We show that better performance can be obtained in these applications by using our eigenspace updating method.

In Section 6, we discuss the related issues for our work, such as the video-based recognition and the high order statistical model for face sequences. We provide conclusions in Section 7. Also in the appendix we compare our mean estimation algorithm with the Kalman filter [20].

2. Eigenspace updating with decaying memory

2.1. Updating based on the covariance matrix

Suppose there is a random process $\{\mathbf{x}_n\}$, where n is the time index, $\mathbf{x}_n$ is a column vector in a d-dimensional space, of which we want to find the eigenspace. Each sample will be available sequentially over time. If this random process is stationary, we can estimate its mean by the following equation:

$$\hat{\mathbf{m}} = \frac{\mathbf{x}_n + \mathbf{x}_{n-1} + \cdots + \mathbf{x}_1}{n}.$$

If $\mathbf{x}_n$ is a non-stationary random process, which implies that it has a time-varying mean $\hat{\mathbf{m}}_n$, we propose to estimate the mean at time n as

$$\hat{\mathbf{m}}_n = \frac{\mathbf{x}_n + \alpha_m \mathbf{x}_{n-1} + \alpha_m^2 \mathbf{x}_{n-2} + \cdots}{1 + \alpha_m + \alpha_m^2 + \cdots}, \tag{1}$$

where α_m is the *decay parameter*. It controls how much the previous samples contribute to the estimation of the current mean. Since α_m is in the range of 0–1, we have

$$1 + \alpha_m + \alpha_m^2 + \cdots = \frac{1}{1 - \alpha_m}. \tag{2}$$

Using (2) in (1), the resulting equation can be simplified to

$$\hat{\mathbf{m}}_n = \alpha_m \hat{\mathbf{m}}_{n-1} + (1 - \alpha_m)\mathbf{x}_n. \tag{3}$$

This equation reveals that based on the current sample and the previously estimated mean, we can obtain the new estimated mean in a recursive manner. How to choose α_m mainly depends on the knowledge of the random process. Note that α_m controls how fast we want to forget about the old samples. Therefore, if the statistics of the random process change fast, we choose a small α_m. If the statistics change slowly, a large α_m may perform better. In the next section, we will introduce how to choose these decay parameters based on the statistical knowledge of the samples.

After the mean of the random process has been estimated, we can estimate the covariance matrix, $\hat{\mathbf{C}}_n$, at each time n by

$$\hat{\mathbf{C}}_n = \frac{(\mathbf{x}_n - \hat{\mathbf{m}}_n)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^{\mathrm{T}} + \alpha_v(\mathbf{x}_{n-1} - \hat{\mathbf{m}}_{n-1})(\mathbf{x}_{n-1} - \hat{\mathbf{m}}_{n-1})^{\mathrm{T}} + \alpha_v^2(\mathbf{x}_{n-2} - \hat{\mathbf{m}}_{n-2})(\mathbf{x}_{n-2} - \hat{\mathbf{m}}_{n-2})^{\mathrm{T}} + \cdots}{1 + \alpha_v + \alpha_v^2 + \cdots},$$

where α_v is also a decay parameter, which is chosen based on how fast the covariance of a random process is changing. Now we can rewrite $\hat{\mathbf{C}}_n$ in a similar manner as $\hat{\mathbf{m}}_n$:

$$\hat{\mathbf{C}}_n = \alpha_v \hat{\mathbf{C}}_{n-1} + (1 - \alpha_v)(\mathbf{x}_n - \hat{\mathbf{m}}_n)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^{\mathrm{T}}. \tag{4}$$

Since we obtain $\hat{\mathbf{C}}_n$ at time n, we can perform PCA for $\hat{\mathbf{C}}_n$ and obtain the corresponding eigenvectors. We keep N eigenvectors corresponding to the N largest eigenvalues. In the recursive updating process, we only need to store the mean vector $\hat{\mathbf{m}}_n$ and the covariance matrix $\hat{\mathbf{C}}_n$. All the previous training samples can be discarded.

2.2. Updating based on the inner-product matrix

In many applications, PCA is applied directly in the image domain, such as face recognition. Suppose the face image has a size of 32×32, then the covariance matrix of an image set would be 1024×1024. It is very inefficient to store and update it using the algorithm introduced in Section 2.1. To solve this problem, we propose an updating algorithm based on the inner-product matrix.

Suppose at time n, we already have performed PCA for the random process at time $n-1$. Thus we have eigenvectors, $\varphi_{n-1}^{(i)}$, and eigenvalues, $\lambda_{n-1}^{(i)}$, of the covariance matrix, $\hat{\mathbf{C}}_{n-1}$. We can write

$$\hat{\mathbf{C}}_{n-1} = \lambda_{n-1}^{(1)} \varphi_{n-1}^{(1)} \varphi_{n-1}^{(1)\mathrm{T}} + \lambda_{n-1}^{(2)} \varphi_{n-1}^{(2)} \varphi_{n-1}^{(2)\mathrm{T}} + \cdots + \lambda_{n-1}^{(d)} \varphi_{n-1}^{(d)} \varphi_{n-1}^{(d)\mathrm{T}},$$

where eigenvalues, $\lambda_{n-1}^{(i)}$, have been sorted in the decreasing order and the superscript (i) indicates the order of

eigenvalues. By retaining only the first Q eigenvectors (with the largest eigenvalues), we can approximate $\hat{\mathbf{C}}_{n-1}$ as

$$\hat{\mathbf{C}}_{n-1} \approx \lambda_{n-1}^{(1)}\varphi_{n-1}^{(1)}\varphi_{n-1}^{(1)\mathrm{T}} + \lambda_{n-1}^{(2)}\varphi_{n-1}^{(2)}\varphi_{n-1}^{(2)\mathrm{T}} + \cdots + \lambda_{n-1}^{(Q)}\varphi_{n-1}^{(Q)}\varphi_{n-1}^{(Q)\mathrm{T}}. \quad (5)$$

The criteria for choosing Q vary, and depend on practical applications. We have tried three methods: (a) fix Q to be a constant value; (b) set a minimum threshold, and keep the first Q eigenvectors whose eigenvalues are larger than this threshold; (c) keep the eigenvectors corresponding to the largest eigenvalues, such that a specific fraction of energy in the eigenvalue spectrum is retained. These methods will result in different computational complexity for the updating algorithm.

Now we can use (3) to estimate the mean at time n. By replacing $\hat{\mathbf{C}}_{n-1}$ in Eqs. (4) with (5), we can obtain

$$\hat{\mathbf{C}}_n \approx \alpha_v\lambda_{n-1}^{(1)}\varphi_{n-1}^{(1)}\varphi_{n-1}^{(1)\mathrm{T}} + \alpha_v\lambda_{n-1}^{(2)}\varphi_{n-1}^{(2)}\varphi_{n-1}^{(2)\mathrm{T}} + \cdots + \alpha_v\lambda_{n-1}^{(Q)}\varphi_{n-1}^{(Q)}\varphi_{n-1}^{(Q)\mathrm{T}} + (1-\alpha_v)(\mathbf{x}_n - \hat{\mathbf{m}}_n)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^{\mathrm{T}}.$$

An equivalent formulation as above is that

$$\hat{\mathbf{C}}_n \approx \mathbf{B}_n\mathbf{B}_n^{\mathrm{T}},$$

where

$$\mathbf{B}_n = \left[\sqrt{\alpha_v\lambda_{n-1}^{(1)}}\varphi_{n-1}^{(1)} \quad \sqrt{\alpha_v\lambda_{n-1}^{(2)}}\varphi_{n-1}^{(2)} \quad \cdots \quad \sqrt{\alpha_v\lambda_{n-1}^{(Q)}}\varphi_{n-1}^{(Q)} \quad \sqrt{1-\alpha_v}(\mathbf{x}_n - \hat{\mathbf{m}}_n) \right]. \quad (6)$$

Based on the $\mathbf{B}_n$ matrix, an inner-product matrix can be formulated as

$$\mathbf{A}_n = \mathbf{B}_n^{\mathrm{T}}\mathbf{B}_n.$$

Furthermore, $\mathbf{A}_n$ can be described by the following equations:

$$(\mathbf{A}_n)_{i,j} = \alpha_v\sqrt{\lambda_{n-1}^{(i)}\lambda_{n-1}^{(j)}}\delta_{ij}, \quad i,j = 1,2,\ldots,Q,$$

$$(\mathbf{A}_n)_{i,Q+1} = (\mathbf{A}_n)_{Q+1,i} = \sqrt{\alpha_v(1-\alpha_v)\lambda_{n-1}^{(i)}}(\mathbf{x}_n - \hat{\mathbf{m}}_n), \quad i = 1,2,\ldots,Q,$$

$$(\mathbf{A}_n)_{Q+1,Q+1} = (1-\alpha_v)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^{\mathrm{T}}(\mathbf{x}_n - \hat{\mathbf{m}}_n). \quad (7)$$

Since the matrix $\mathbf{A}_n$ is usually a small matrix with the size of $Q+1$ by $Q+1$, we can determine its eigenvectors ψ_n by a direct method, which satisfies

$$\mathbf{A}_n\psi_n^{(i)} = \mathbf{B}_n^{\mathrm{T}}\mathbf{B}_n\psi_n^{(i)} = \lambda_n^{(i)}\psi_n^{(i)} \quad i = 1,2,\ldots,Q+1. \quad (8)$$

By pre-multiplying (8) with $\mathbf{B}_n$, we can obtain the eigenvectors of matrix $\hat{\mathbf{C}}_n$ as follows:

$$\varphi_n^{(i)} = \lambda_n^{(i)-1/2}\mathbf{B}_n\psi_n^{(i)} \quad i = 1,2,\ldots,Q+1, \quad (9)$$

where the term $\lambda_n^{(i)-1/2}$ is to make the resulting eigenvector to be a unit vector. Now we summarize the iterative updating algorithm outlined in this section:

Initialization:

1. Given the first two samples $\mathbf{x}_0$, $\mathbf{x}_1$, estimate the mean, $\mathbf{m}_1$, by (3), and construct the matrix
$$\mathbf{B}_1 = [\sqrt{\alpha_v}(\mathbf{x}_0 - \hat{\mathbf{m}}_1) \quad (\mathbf{x}_1 - \hat{\mathbf{m}}_1)].$$
2. Based on Eqs. (8) and (9), we can get the eigenvector, φ_1, and the eigenvalue, λ_1.

Iterative updating:

1. Get a new sample $\mathbf{x}_n$.
2. Estimate the mean, $\hat{\mathbf{m}}_n$, at time n by (3), and get the $\mathbf{B}_n$ matrix from (6).
3. Form the matrix $\mathbf{A}_n$ by (7) and calculate its eigenvectors, ψ_n, and eigenvalues, λ_n, by a direct method.
4. Sort the eigenvalues λ_n, and retain Q corresponding eigenvectors.
5. Obtain the eigenvectors, φ_n, at time n by (9).

We have mentioned three methods of choosing Q. If we use the second and the third methods, Q will increase as more and more training samples arrive till it reaches the intrinsic dimensionality of previous training samples. Due to the approximation in Eq. (5), among the Q eigenvectors, typically the first few eigenvectors are more precise than the others. Therefore, in practice if we need N eigenvectors for building an eigenspace, we would keep Q to be a number larger than N.

2.3. An example with synthetic data

In this section, we want to show that our updating algorithm can better model the statistics of a non-stationary random process, compared to the traditional eigenspace updating algorithms without decay parameters.

We generate 720 samples with a two-dimensional Gaussian distribution, whose mean is zero and variances in the horizontal and vertical direction are 1 and 7, respectively. If we associate each sample with a time index, we can obtain a random process. For each random variable in this random process, we incrementally rotate it by a certain degree in the two-dimensional space, and move its mean along the line, $x = y$. The first random variable rotates 0°, the last one rotates 90°, and all the others rotate in between. In another word, the synthetic data have the following statistics:

$$\begin{bmatrix} \mathbf{m}_x[n] \\ \mathbf{m}_y[n] \end{bmatrix} = \begin{pmatrix} n/15 \\ n/15 \end{pmatrix},$$

$$\mathbf{C}_{xy}[n] = \begin{pmatrix} 1 & 24\sin(n\pi/720) \\ 24\sin(n\pi/720) & 1 + 48\cos^2(n\pi/1440) \end{pmatrix}.$$

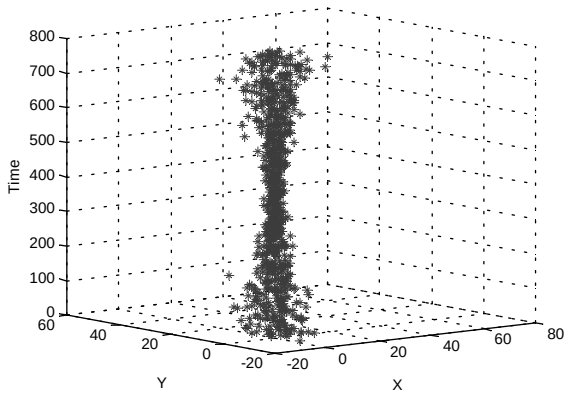

Fig. 1. A synthetic random process.

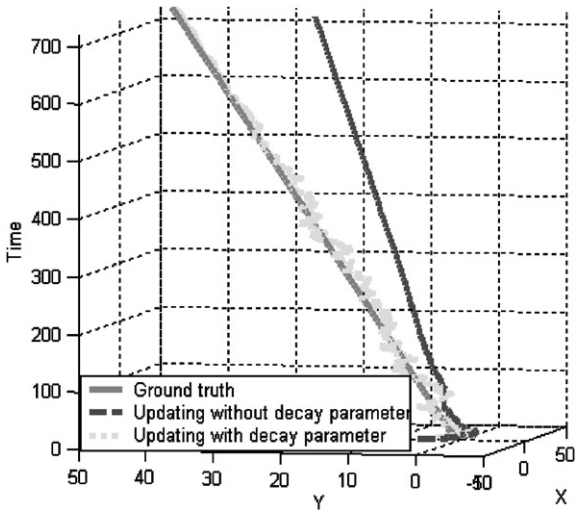

Fig. 2. Estimation of the mean for a random process.

One example of this synthetic data is shown in Fig. 1. We can see that the cluster of data keeps rotating and moving away from the origin over time.

We use the algorithm introduced in Section 2.1 to update the eigenspace, and show the estimation results of the mean compared to the ground truth in Fig. 2. The traditional updating algorithm without decay parameters is also applied on the same data. We can see from Fig. 2 that its estimation is much worse than our estimation. When the eigenspace is updated by a new random variable, we calculate the orientation of the first eigenvector with respect to the horizontal coordinate. Ideally the orientation should change from 90° to 0° according to the time coordinate. As shown in Fig. 3, our algorithm can successfully estimate the statistic of the time-varying random process. However, if we apply the traditional method without decay parameters on the same data, the resulting orientation is around 45° because it

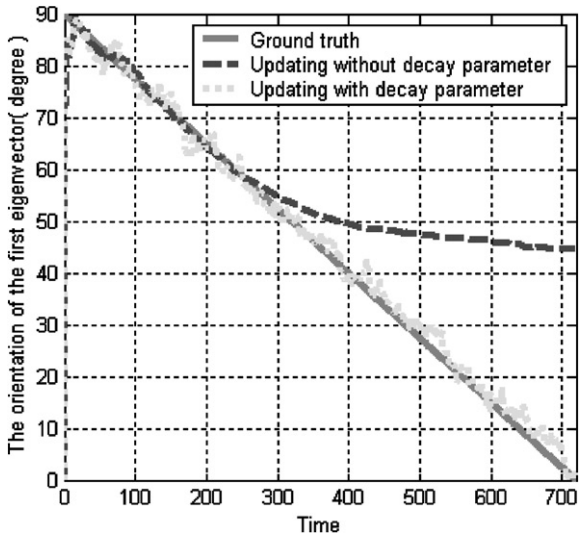

Fig. 3. Estimation of the variance for a random process.

considers all the previous samples equally. The traditional method works well in the beginning as it removes the mean as well. However, it quickly becomes worse because neither the mean nor the variance updating uses the decaying parameters.

3. Choosing the decay parameters

In the proposed eigenspace updating algorithm, we need to specify the decay parameters for both mean estimation and variance estimation. In practice, for a recognition system, there is usually a cross validation data available before the testing stage of the system. Thus based on the cross validation data, the optimal decay parameters specific to the application could be obtained by exhaustive search within the valid range, 0–1. If there is no cross validation data available, how do we determine decay parameters? We will answer it in this section.

Motivated by the Kalman filter, we can model the time-varying mean and variance as autoregressive (AR) random processes with certain parameters. Now the problem becomes, based on the models and the parameters, how do we find the optimal decay parameters without exhaustive search? We will address mean estimation and variance estimation separately.

3.1. Decay for mean estimation

Consider the model in Fig. 4, where the sample, x_n, is a scalar and generated by an AR(1) random process plus a white noise, v_n,

$$x_n = m_n + v_n, \tag{10}$$

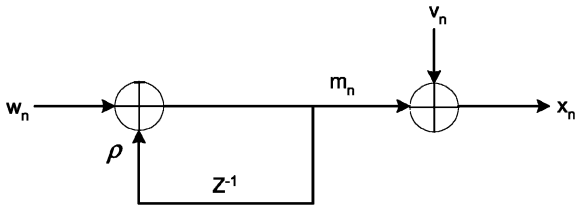

Fig. 4. AR(1) model for observed samples.

where the observation noise, v_n, has zero mean and the variance of r. The AR(1) process is generated by the following equation:

$$m_n = \rho m_{n-1} + w_n, \tag{11}$$

where the white noise, w_n, has zero mean and the variance of q. Based on the above two equations, we can see that x_n and m_n have the same mean. Given x_n, we can estimate its mean at each time instant by estimating the mean of m_n at that time instance, which is denoted as $\hat{m}_n$.

We consider that the mean is estimated via:

$$\hat{m}_n = \alpha_m \hat{m}_{n-1} + \beta_m x_n, \tag{12}$$

where α_m and β_m can take on any values between 0 and 1. Note that (12) is basically the same as (3) without the constraint that $\beta_m = 1 - \alpha_m$. Removing this constraint allows us a more comprehensive study for choosing the decay.

Now the problem becomes to find the optimal α_m and β_m, which can make the $\hat{m}_n$ as close to m_n as possible. Let us derive it by minimizing the estimation error, p_n.

$$p_n = E(m_n - \hat{m}_n)^2.$$

By extending the above equation, we have

$$p_n = \rho^2(1-\beta_m)^2 p_{n-1} + (1-\beta_m)^2 q + \beta_m^2 r + h^2 E(\hat{m}_{n-1})^2 - 2h\rho(1-\beta)E(\hat{m}_{n-1}(m_{n-1} - \hat{m}_{n-1})), \tag{13}$$

where $h = \alpha_m - \rho(1-\beta_m)$.

In order to have an explicit formulation for p_n, we let h be zero, i.e., α_m and β_m satisfy the following equation.

$$\rho = \frac{\alpha_m}{1-\beta_m}. \tag{14}$$

Thus p_n can be simplified as

$$p_n = \alpha_m^2 p_{n-1} + (1-\beta_m)^2 q + \beta_m^2 r. \tag{15}$$

When n goes to infinity, p_n converges to

$$p_{n\to\infty} = \frac{(1-\beta_m)^2 q + \beta_m^2 r}{1 - \rho^2(1-\beta_m)^2}. \tag{16}$$

By taking the derivative of $p_{n\to\infty}$ with respect to β_m to be 0, we can obtain the optimal value for β_m:

$$\beta_m = \frac{r(\rho^2-1) - q + \sqrt{(r(1-\rho^2)+q)^2 + 4\rho^2 qr}}{2r\rho^2}. \tag{17}$$

Based on Eq. (14), the optimal value for α_m is the following:

$$\alpha_m = \frac{r(\rho^2+1) + q - \sqrt{(r(1-\rho^2)+q)^2 + 4\rho^2 qr}}{2r\rho}. \tag{18}$$

We can see that as ρ increases, both α_m and β_m increase. However, α_m increases faster than β_m since there is one more term of ρ in the denominator of β_m. This means that as the random process changes more and more slowly, i.e., ρ gets larger and larger, the previous estimate, $\hat{m}_{n-1}$, should contribute more and more to the current estimate.

In order to show the effectiveness of our choice of decay parameters, we perform an experiment based on synthetic data. First, we synthesize a set of random processes using the AR(1) process in Eqs. (10) and (11). By taking these processes as observation samples and assuming we know the parameters in the AR(1) process, i.e., ρ, r and q, we can perform exhaustive search to find the optimal decay parameters which can generate the minimum estimation error. Also by using of (17) and (18), we can calculate the decay parameters for our estimate. From Fig. 5, we can see that our decay parameters are very close to the optimal decay resulting from exhaustive search.

All the above derivation works in the scalar case. When the sample, $\mathbf{X}_n$, is a vector, we can obtain the same results if we assume each element of $\mathbf{X}_n$ is independent. In the appendix, we will compare the estimation performance between our estimate and other estimates, such as the Kalman filter and exhaustive search, in terms of estimation errors.

3.2. Decay for variance estimation

Similar to the mean estimation, we study the case when the observation, x_n, is in the scalar form and it can be modeled as

$$x_n = \sqrt{c_n} v_n. \tag{19}$$

Here the white noise, v_n, has zero mean and variance of one. Thus c_n becomes the variance of x_n, and we assume c_n is generated from an AR(1) random process using (11).

Now the problem becomes given an observation sequence, x_n, estimate its variance, c_n, based on the knowledge of the parameter of the AR process, ρ, and the variance of the white noise w_n, q. We use the following equation as the estimate. Similar to our previous section, we use two parameters, α_v and β_v, to combine the information from the previous estimate, $\hat{c}_{n-1}$, and the current sample, x_n,

$$\hat{c}_n = \alpha_v \hat{c}_{n-1} + \beta_v x_n^2. \tag{20}$$

In order to find the optimal α_v and β_v, we can derive it by minimizing the estimation error. However, it turns out we need to make a very strict constraint in the derivation in order to obtain an explicit result. Experimentally, we found that the estimation performance of the derivation result under this strict constraint is not satisfying. Thus we want to solve this problem by an empirical method.

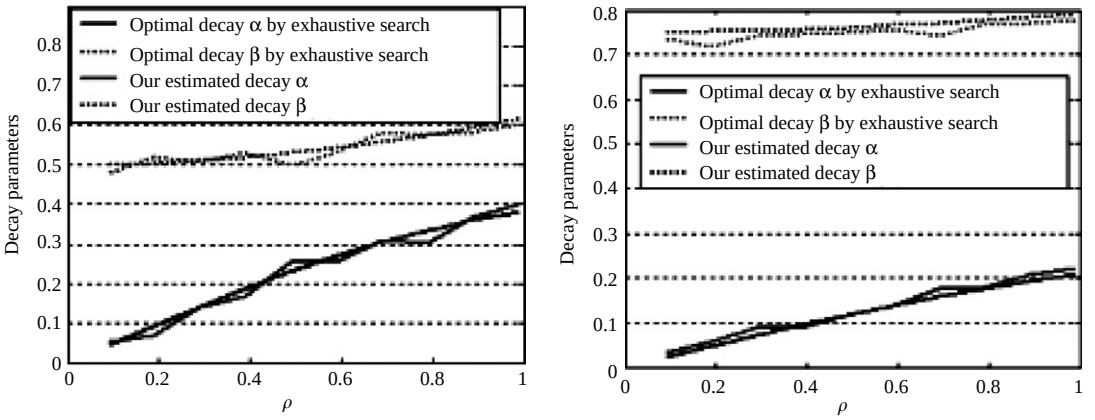

Fig. 5. Optimal decay parameters vs. our estimated decay parameters (left: $r/q = 3$; right: $r/q = 1$).

Basically, two parameters will affect the selection of decay parameters. The first one ρ, the parameter of the AR process, defines how fast the variance c_n changes over time. Since in most applications the variance does not change too fast, we will study the case where ρ is between 0.6 and 1. The second one q, the variance of the white noise w_n, defines how much variability the variance itself will have over time. The larger w_n, the larger range the variance will vibrate. In our experiments, we will change these two parameters, and observe the corresponding effect on the optimal decay parameters.

By fixing the above two parameters, we can synthesize the ground truth c_n and observation samples, x_n. The optimal decay parameters in the sense of minimal estimation error can be obtained by exhaustive search. Now changing the ρ to be other values, synthesizing data and performing estimation many times, we found the optimal α_v and β_v actually change very little, which means they are basically unaffected by ρ. For a fixed q, by tuning different ρ, we can obtain both the mean and variance of the optimal α_v and β_v. By varying q from 25 to 10 000, we can obtain four curves according to the above four statistics. We show the results in Fig. 6, where the horizontal axis represents the square root of q. From this figure, we can see that even though the standard deviation varies over a large range, the optimal decay parameters do not change significantly. The same experiment can be performed by fixing q and varying ρ. We plot the resulting optimal decay parameters according to different ρ in Fig. 7. Thus a good choice of our estimate is to choose the mean of optimal decay parameters as the value of α_v and β_v, where $\alpha_v = 0.85$ and $\beta_v = 0.13$.

We now conduct an experiment to compare the estimation performance of different approaches. In Fig. 8, we show the estimation performance of four approaches according to different ρ. The first one is exhaustive search by constraining that α_v and β_v sum to one. The second one is also exhaustive search, but both the optimal α_v and β_v are searched in the

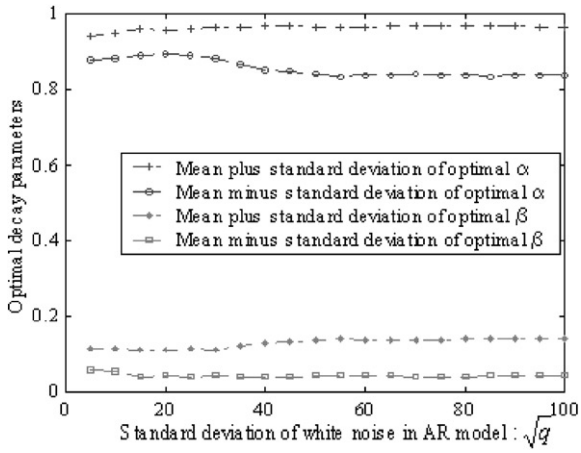

Fig. 6. Optimal decay parameters according to the different standard deviation of the white noise.

range 0–1. The third one is the sample variance, which is calculated from all the samples with equally weighting. The last one is our estimate where two fixed decay parameters are used. From this figure, we can see that α_v and β_v summing to one is not a bad constraint since the performance is only slightly worse than the unconstrained case. Also, when ρ is large, which indicates the variance changes slowly, our estimate works much better than the sample variance. But when the variance changes too fast, batch variance turns out to be better than ours because it is harder to estimate the variance for each time instance. Actually in practical applications, the variance tends to change slowly, where ρ is closer to 1. Similar to the mean estimate, when the samples are in the form of vectors, we can obtain the same results by extending our estimate to the vector form.

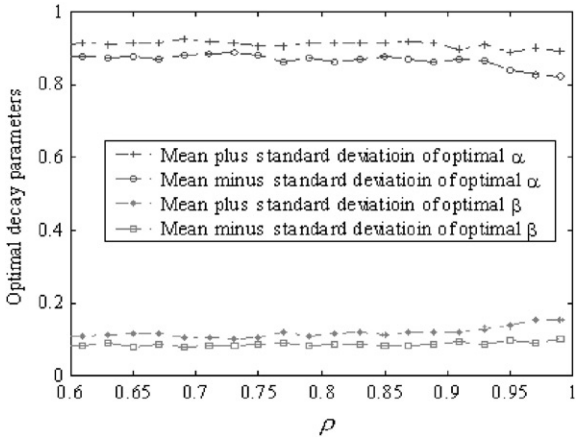

Fig. 7. Optimal decay parameters according to different ρ.

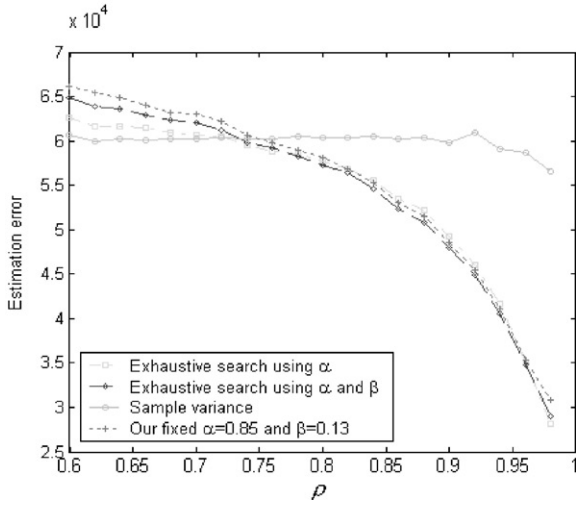

Fig. 8. Results of the variance estimation.

4. Face recognition based on updating individual PCA

When applied to face recognition, the proposed eigenspace updating algorithm results in an updating-during-recognition scheme. That is, the eigenspace for each subject is updated by test images while each of them being recognized. There are two reasons for doing this. First, in many applications it is not feasible to capture many training images for each subject containing enough variations for statistical modeling of that subject. Usually only a few images under the normal condition are available for training. Thus, it would be better if more and more images of that subject are used to update its model during the testing stage. Secondly, people change their appearance over time. Even if there are many images available for training, the system may not recognize faces when a subject changes the appearance due to aging, expression, pose, and illumination changes. A recognition system that is able to learn the changing appearance of the subject and adapt to it can achieve better performance.

In using our updating method for face recognition, we assume the test images are from a face sequence and there is continuity between consecutive frames. In Section 4.1, we introduce the scheme based on updating a single eigenspace model for each subject. Since this approach may suffer from slow learning, in Section 4.2 we also propose a twin-subspace approach to alleviating this problem.

4.1. Single-subspace updating scheme

Given a set of face images from K subjects for training, each subject has one individual eigenspace trained from his/her own images. When a test image arrives, it is projected into every individual eigenspace and assigned to the one that gives the minimal residue, which is defined by the difference between the test image and its projection in the eigenspace.

Now we need to decide whether to update the eigenspace model of the recognized subject, using the test image. First, by comparing the minimal residue with a pre-defined threshold, we can see whether the current model can represent the test image well. If it does, we do not perform updating since this test image does not bring enough new statistical information for the model. Second, we calculate the confidence measure as the difference between the residue of the second candidate and the residue of the top candidate. Then the confidence measure is compared with another pre-defined threshold. If the confidence measure is larger than the threshold, this test image will be utilized to update the assigned eigenspace using our updating method. Basically the larger the confidence measure, the more confidence we have about the current recognition result. Thus as time goes on, the eigenspace will adapt to the most recent statistics of the subject's appearance, and be able to recognize more "new looking" images from that subject.

One risk in this approach is that sometimes the eigenspace model is not updated by the test images with new appearance because of not-high-enough confidence measures, while in the same time the test images keep showing new appearances. In this case, it is likely the test images will not be correctly recognized because they show different appearance as the current model, which only represents out-of-date appearances. This is the problem with slow learning, i.e., the model does not learn fast enough in order to recognize the test images with new appearances. To alleviate this problem, we introduce the twin-subspace updating scheme in Section 4.2.

4.2. Twin-subspace updating scheme

In this scheme, we train two subspaces, the static model and the dynamic model, for each subject. The static model is trained from the original training images of that subject, and

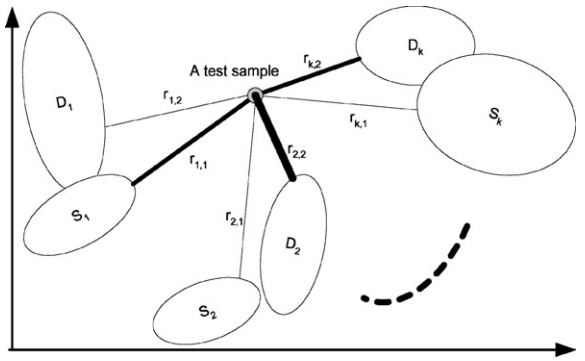

Fig. 9. Testing of the twin-subspace updating scheme.

the dynamic model is updated from the test images during the testing stage.

When one test image arrives, we calculate its residue to both subspaces for each subject. Then the smaller residue is considered as the distance between the test image and that subject. Eventually the test image is recognized as the subject with the minimal distance. Same as the previous section, we also make the decision of updating based on two thresholds. The first threshold filters out the test image without enough variations with respect to the current model. The second threshold is compared with the confidence measure, which is the difference between the top candidate and the second candidate in terms of distance. Test images with low confidence in its recognition result are rejected from being used for updating. In this scheme, only the dynamic model is updated by the test images, and the static model will never be changed once it is trained from the original training images. As illustrated in Fig. 9, each one of the K subjects has two models, the static one, S_i, and the dynamic one, D_i. The residues $r_{1,1}, r_{2,2}, \ldots, r_{k,2}$ are calculated as the distances between the test sample and each class. In this case Subject 2 is the recognition result because it has the minimal distance $r_{2,2}$. Suppose Subject k is the second candidate. The confidence measure $r_{k,2} - r_{2,2}$ is then utilized to decide whether this test sample will be used to update the dynamic model of Subject 2, D_2.

The main reason we propose this updating scheme is to capture different aspects of facial appearance. That is, the static model is used to capture the subject's more intrinsic appearance based on training images, while the dynamic model is used to capture the time-varying statistics on the appearance. The second reason is to deal with the slow learning problem. Because there are two models for each subject during testing, even the test image might not match well with the dynamic model because of the slow learning in the dynamic model, it is still possible that the static model will match with the test image and let the test image update the current dynamic model. Thus the dynamic model can learn the time-varying statistics and benefit future recognition.

In practice, many factors decide whether we should use the single subspace scheme or the twin-subspace scheme. For example, we should use the single subspace scheme when there is small amount of variations in the testing images. Also, when we need to deal with the recurrent type of variations, such as pose, expression, illumination, and facial hair variations, we should use the twin-subspace scheme. While for non-recurrent type of variation, such as aging, the single subspace scheme would be more proper because only the most current statistics, which is captured by the dynamic model, will be useful for future recognition.

5. Experimental results

We conduct experiments on face data sets that contain different variations, such as poses, illuminations and expressions. We will show that for all these variations, our algorithm can achieve much better performance than methods without updating, because we can model variations in a subject's appearance over time and thus improve the recognition performance. The methods we compare with are the individual PCA method without updating, and traditional eigenspace updating without decay.

In practical applications of face recognition, the human face usually undergoes different kinds of variations, most of which come from the pose, expression, illumination and the combination of them. In order to show the effectiveness of our algorithm in dealing with these variations, experiments are conducted on data sets with these three types of variations.

5.1. Pose data set

We collected a face database with 20 subjects. Each subject has 10 training images. The test images for each subject come from a video sequence, where the subject continuously shows different poses. Both the test and training images are of 32×32 grayscale images. There are 210 test images within one sequence for each subject. In Fig. 10, we show sample face images from six subjects in this data set, where the images in the same row belong to the same subject. A lot of pose variations can be observed from this data set. Also notice the registration error in some images. This is a very challenging data set for face recognition.

We show the experiment results in Fig. 11. The horizontal axis shows the index of the test images, and the vertical axis shows the recognition error rate based on the number of test images so far. We perform experiments on different random orders of the test sequence and show the average of them in the figure. Three algorithms have been tested on this data set. The first one is the individual PCA method, which works worst because there is no updating involving in the testing stage. The second is our updating method with dynamically estimated decay parameters, which has better performance than the individual PCA method. The third one is our twin-subspace method. It has significant improvement compared to the other two methods since it models the statistics more comprehensively for the changing appearance over time.

Fig. 10. Sample images of face sequences showing different poses.

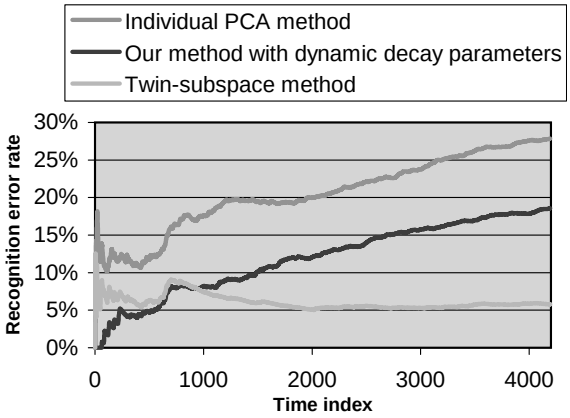

Fig. 11. Experiment results with different approaches on the pose data set.

In the previous experiment, we do not use eigenvectors in constructing the eigenspace for each training subject. So basically only the mean is used for recognition. Because the number of eigenvectors will affect the recognition performance, we also perform experiments with different number of eigenvectors. Table 1 shows different recognition error rates with respect to different number of eigenvectors used in constructing the individual eigenspace. Among these three methods, our twin-subspace updating method has the best performance and the individual PCA works the worst. From this experiment we can see that a proper updating method will work better than a non-updating method in face recognition. Also the twin-subspace method is a promising approach to deal with large variations, such as poses in this data set.

5.2. Expression data set

We collected another face database with 30 subjects. Each subject has 5 training images and 70 test images. Each image is of the size of 32×32 pixels. The test images for each subject come from a video sequence, where the subject shows varying expressions. The sample images from six subjects are shown in Fig. 12. We use the same test scheme as the pose data set. The result is shown in Fig. 13. We try both using fixed decay parameters and tuning the decay parameters online according to the changing statistics. Here we use the AR(1) random process as the model for face sequences. In all three methods we only update the mean and do not use any eigenvectors.

From this experiment we see that updating methods with decay parameters have better performance than the updating method without decay. Also dynamically tuning decay parameters during the testing stage enhances the modeling of time-varying statistics and hence improves the recognition performance.

5.3. PIE database

In this experiment, we use a subset of the CMU PIE database [21], which has 7 subjects. Each subject has 24 images, which have the size of 64×64 pixels, showing the same expression and pose while under continuous

Table 1
Recognition error rate with different number of eigenvectors

Number of eigenvectors	0	2	4	6
Individual PCA method	27.88%	19.62%	16.43%	14.76%
Our method with dynamic decay parameters	18.57%	10.67%	8.49%	6.87%
Our twin-subspace method	5.77%	4.83%	4.15%	3.98%

Fig. 12. Sample images of six subjects from the expression data set.

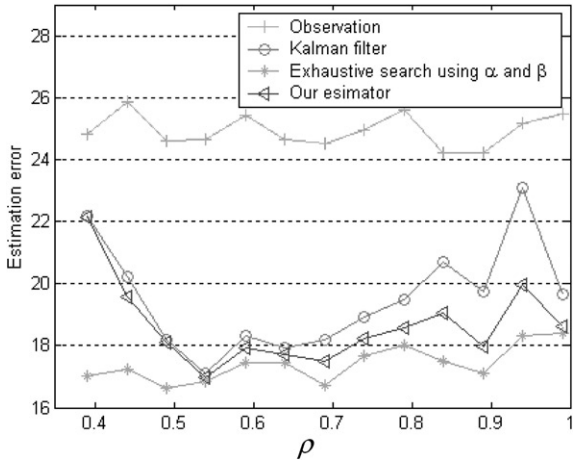

Fig. 13. Experiment results with different approaches on the expression data set.

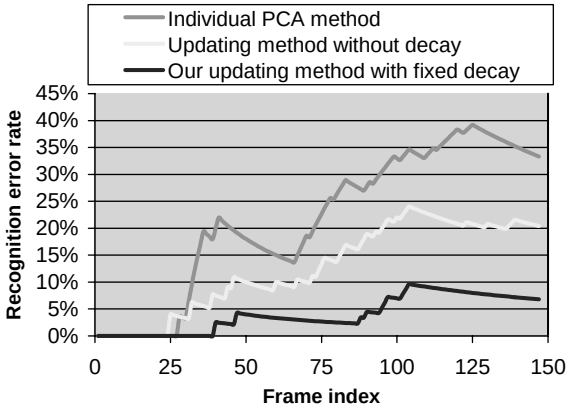

Fig. 15. Experiment results with different approaches on the PIE database.

varying illuminations. We use 3 images for training and the remaining 21 images for testing. One eigenvector is used for building eigenspace for each subject. Part of the test images for one subject are shown in Fig. 14. Since the number of test images per subject is small, during the testing only one image, instead of continuous few frames, is tested whenever a subject is randomly selected. The experiment result shown in Fig. 15 also indicates that our approach can achieve better performance compared to others.

Fig. 14. Images of one subject from PIE database.

6. Discussion

6.1. Frame- and video-based recognition

While in Section 5 we treat each test image independently and perform a frame-based recognition, we can also do a video-based recognition in the following two applications scenarios. One is that we recognize the human from the video sequence in a online fashion and do not know when the subject will leave or another subject will come in. In this case, we need to know the recognition results up to the current frame immediately. Many online recognition and verification systems of human faces belong to this case. We call this scenario as *online video*. The other is that we could offline process the video content, such as indexing of the meeting records or analyzing surveillance videos, where we are interested in the recognition results after all the frames of one sequence have been captured. This is called *offline video.* We illustrate these scenarios in Fig. 16.

For the online video, by using a face-tracking program [22], we can keep tracking human faces and crop the face region for recognition. With the face tracking, we also know whether the current frame and the previous frames belong to the same subject. An intuitive idea is to use majority voting to see which subject is mostly recognized among all the previous frames. Then a decision will be made on whether using the current frame to update the eigenspace. For the case of the offline video, we can still use the updating based on the majority voting in processing frames one by one. However, as shown in the third row of Fig. 16, once a sequence has done with the recognition, we can use all the frames in this sequence to update the eigenspace of the most recognized subject, while this is not feasible in the online video case because it needs to store all the previous frames in one sequence. We have also performed experiments for both the online video and the offline video cases, and the result shows that for the same database, the recognition performance can be significantly improved by using the video-based recognition.

6.2. AR(k) process for decay estimation

In Section 3, we solve the problem of determining decay parameters given the parameters of AR(1) process. However, in face recognition application, given a face sequence, how can we apply the theory in Section 3 on it, i.e., how can we determine whether a face sequence can be approximated by AR(1) or high order AR process; how do we estimate the parameters for an AR process?

The answer to the above questions involves two steps before applying our updating algorithm on the face sequence. One is the model selection. The other is model fitting. Model selection determines k in an AR(k) process. Model fitting estimates the parameters in a specific AR(k) process. There are many existing techniques to solve these two problems in signal processing literature [23]. For example, model selection can be done by finding k whose AR(k) has one pole that is close to 0, which means it can be approximated by AR($k-1$). Table 2 shows the corresponding model parameters by assuming five face sequences as AR(k) random processes with k equals to 1 or 2. We found for most face sequences with expression variations, AR(1) is a good statistical model, while for face sequences showing pose variations, some of them might need AR(2) to model them.

In Section 3, we assume that the observation samples come from the noised version of the AR(1) random process. However, what happen if the samples are actually intrinsic high order AR process, for example, AR(2) random process? In this case, how do we derive the relation between the model parameters, ρ_1, ρ_2, r, q, and the decay parameters, α_m, β_m?

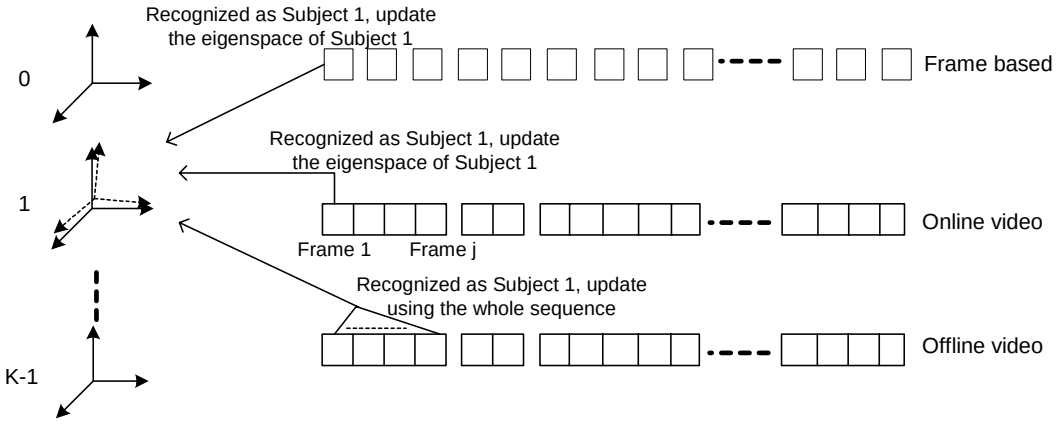

Fig. 16. Three application scenarios.

Table 2
Model parameters for real face sequences

	Expression 1	Expression 2	Expression 3	Pose 1	Pose 2
ρ in AR(1)	0.9998	0.9998	0.9998	0.9975	0.9998
ρ_1 in AR(2)	0.9997	0.9997	0.9502	0.9982	0.9347
ρ_2 in AR(2)	0.1084	−0.0209	−0.0158	0.3283	0.5659

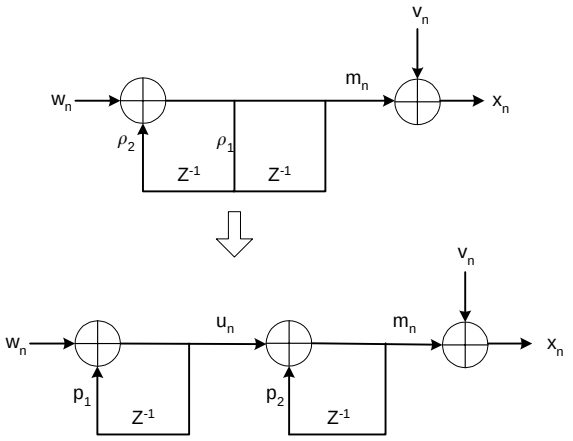

Fig. 17. AR(2) model for observed samples.

First of all, we can separate an AR(2) random process into two AR(1)-like random processes as follows:

$$H(Z) = \frac{m(Z)}{w(Z)} = \frac{1}{1-\rho_1 Z^{-1} - \rho_2 Z^{-1}}$$
$$= \frac{1}{1-p_1 Z^{-1}} \frac{1}{1-p_2 Z^{-1}} = \frac{u(Z)}{w(Z)} \frac{m(Z)}{u(Z)}$$
$$= H_1(Z) H_2(Z).$$

This is also illustrated in Fig. 17. Now if we compare two AR(1)-like random processes with the AR(1) random process in Fig. 4, we can see that u_n plays the similar role as w_n in Fig. 4. Thus given an AR(2) random process, we can calculate the variance of u_n and treat it as q. Then by using q, r and ρ_2 in Eqs. (17) and (18), we can obtain the decay parameters. Similarly for an AR(k) random process, we can separate it into two parts: an AR($k-1$) random process and an AR(1)-like random process, where the former contributes the noise signal for the latter.

One difference in solving decay parameters for AR(1) and AR(2) is that, u_n is not a white noise while w_n being a white noise is one assumption in deriving (17) and (18). However, if w_n is not a white noise, there will be a small no-zeros terms in the right side of the objective function, (13). Thus the solution provided in Eqs. (17) and (18) will become sub-optimal for the AR(2) case. We have performed simulation on estimating the mean of an AR(2) random process, and we found the estimation error of our estimate is very close to the one from exhaustive search.

7. Conclusions and future works

In this paper, we introduced a novel approach to updating the eigenspace for non-stationary random processes. Given a new training sample, we iteratively update the eigenspace to manifest the current statistics provided by the new sample. The updated eigenspace is based more on the recent samples and less on the older samples. Extensive study has been performed on how to choose the decay parameters in our updating method. We showed the effectiveness of our algorithm using both synthetic data and practical applications on face recognition. The experiments results indicate that the random processes in many practical applications are essentially non-stationary, which results in the significantly improved performance by our updating method compared to other methods in literature.

As a modeling tool, our eigenspace updating method can also be applied to other applications, for example, the detection of signal changing [24,25] and video coding. We have already applied it to the shot boundary detection [27] and the detection of facial expression changes. It is able to model the most recent statistics over time, and thus any change in signals can be detected from the residue between the new signal and the eigenspace.

In face recognition, many approaches have been proposed to deal with different variations. While each approach works well for the specific variation being studied, performance degrades rapidly when other variations are present. In practice, the test images usually undergo the mixture of variations, such as expressions, poses and illuminations. Trying to use a static model to cover all these variations is difficult. Using the proposed updating method is one solution. Instead of trying to model all variations at once, we try to dynamically model only the most recent variations.

There are many interesting directions to be explored further. For example, in our updating method, the eigenvector expansion is truncated as in (5), which results in an approximate representation for the covariance matrix. Can we have a better estimation for the covariance by adding a diagonal term that accounts for the discarded residue? Also, as the eigenspace only provides a subspace representation for a data set, it lacks a probabilistic measure for the samples in

the data set. Can we borrow the idea of probabilistic PCA [26] and update the eigenspace in a probabilistic framework, i.e., the resulting eigenspace can have a probabilistic interpretation for each sample? Further, while applying updating methods, a good scheme to decide when to perform updating is very critical and requires more study. Finally, how to take advantage of the timing information and perform the video-based recognition is also an interesting topic worth further study.

Acknowledgements

This work was supported by the US Department of Commerce, National Institute of standards and Technology Program, Cooperative agreement No. 70NANB8H4076. The authors would like to thank the anonymous reviewers for insightful comments. Thanks to Prof. B.V.K. Vijaya Kumar for fruitful discussion. Thanks also go to a number of volunteers in the Electrical and Computer Engineering Department of Carnegie Mellon University for helping us in data collection.

Appendix

In the Kalman filter, there are five iterative steps to perform the estimation. They can be described by the following five equations:

$$\hat{m}_n^- = \rho \hat{m}_{n-1}, \tag{A.1}$$

$$p_n^- = \rho p_{n-1} + q, \tag{A.2}$$

$$\hat{m}_n = \hat{m}_n^- + k_n(x_n - \hat{m}_n^-), \tag{A.3}$$

$$k_n = \frac{p_n^-}{p_n^- + r},$$

$$p_n = (1 - k_n)p_n^-. \tag{A.4}$$

By extending (A.3) with (A.1), we get

$$\hat{m}_n = \rho(1 - k_n)\hat{m}_{n-1} + k_n x_n. \tag{A.5}$$

If we compare the above equation with our estimate, we can find that k_n corresponds to β_m, and $\rho(1 - k_n)$ corresponds to α_m in our estimate. By combining (A.2) and (A.4), the converge formula of p_n when n goes infinity can be found. Since k_n only depends on p_{n-1}, q, and r, eventually we can obtain the converging formulation for k_n, which is exactly the same as β_m in Eq. (17). Hence $\rho(1 - k_n)$ has the same formulation as α_m in Eq. (18). From this, we can see that our estimate turns out to be the converging form of the Kalman filter.

We conduct the following experiment to show the estimation performance. Given different random processes synthesized by tuning different ρ in the AR(1) process, we can estimate the model parameters first. Then we can utilize the model parameters to derive the decay parameters for our updating method. For the AR(1) random process, we can estimate model parameters as following:

$$R_{xx}(k) = E(x_n x_{n+k}) = E((m_n + v_n)(m_{n+k} + v_{n+k}))$$

$$= E(m_n m_{n+k}) + E(v_n v_{n+k}) = R_{mm}(k) + R_{vv}(k),$$

$$R_{mm}(k) = \frac{q}{1 - \rho^2}\rho^{|k|}.$$

Basically by calculating $R_{xx}(0)$, $R_{xx}(1)$, up to $R_{xx}(k)$, we can estimate q, r and ρ based on the above two equations. Since there are estimation errors in estimating these model parameters, we are interested in how our estimate performs based on these estimated model parameters.

Then these model parameters are fed into both the Kalman filter and our estimate to estimate the mean of the random process. Since we have the ground truth of the mean, we can also perform exhaustive search for the decay parameters as well. In Fig. 18, we show the estimate errors of three different estimates and also the variance between the given samples and the ground truth. We can see that for different choices of ρ, our estimate performs better than the Kalman filter, especially in the region with large ρ.

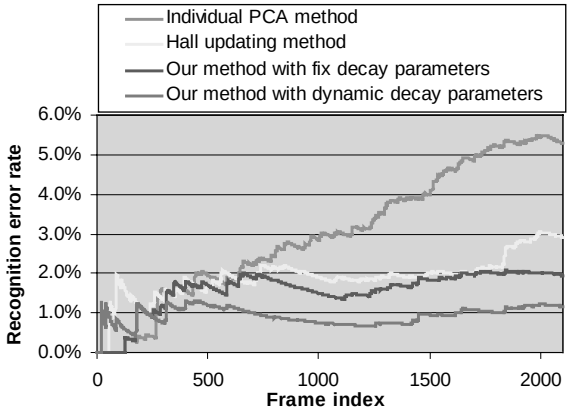

Fig. 18. Mean estimation based on estimated model parameters.

References

[1] Y.T. Chien, K.S. Fu, On the generalized Karhunen–Loeve expansion, IEEE Trans. Inf. Theory 13 (3) (1967) 518–520.
[2] A. Habibi, P.A. Wintz, Image coding by linear transformation and block quantization techniques, IEEE Trans. Commun. Technol. CoOM-19 (1971) 948–956.
[3] R.J. Wong, P.A. Wintz, Information extraction, SNR improvement, and data compression in multispectral imagery, IEEE Trans. Commun. Technol. CoOM-21 (1973) 1123–1131.
[4] E. Oja, Subspace Methods of Pattern Recognition, Wiley, Letchworth, Hertfordshire, England, New York, 1983.

[5] L. Sirovich, M. Kirby, Low dimensional procedure for the characterization of human faces, J. Opt. Soc. Am. 4 (3) (1987) 519–524.
[6] M. Turk, A. Pentland, Eigenfaces for recognition, J. Cognitive Neurosci. 3 (1) (1991) 71–86.
[7] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, Cambridge, 1985.
[8] H. Murakami, B.V.K.V. Kumar, Efficient calculation of primary images from a set of images, IEEE Trans. Pattern Anal. Mach. Intell. 4 (5) (1982) 511–515.
[9] P.M. Hall, D. Marshall, R.R. Martin, Incremental Eigenanalysis for Classification, Research Report Series No. 98001, Department of Computer Science, University of Wales Cardiff, UK, May 1998.
[10] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler, H. Zhang, An eigenspace update algorithm for image analysis, Graphical Models Image Process. 59 (5) (1997) 321–332.
[11] R. Chellappa, C.L. Wilson, S. Sirohey, Human and machine recognition of faces: a survey, Proc. IEEE 83 (5) (1995) 705–741.
[12] T. Sim, T. Kanade, Combining models and exemplars for face recognition: an illuminating example, Proceedings of the CVPR 2001 Workshop on Models versus Exemplars in Computer Vision, December 2001.
[13] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eigenfaces vs. fisherfaces: recognition using class specific linear projection, IEEE Trans. Pattern Anal. Mach. Intell. 20 (7) (1997) 711–720.
[14] X. Liu, T. Chen, B.V.K. Vijaya Kumar, Face authentication for multiple subjects using eigenflow, Pattern Recognition (special issue on Biometric) 36 (2) (2003) 313–328.
[15] P. Hall, D. Marshall, R. Martin, Merging and splitting eigenspace models, IEEE Trans. Pattern Anal. Mach. Intell. 22 (9) (2000) 1042–1049.
[16] T. Kurita, M. Tanaka, K. Hotta, H. Shimai, T. Mishima, Efficient face detection from news images by adaptive estimation of prior probabilities and ising search, Proceedings of 15th International Conference on Pattern Recognition, Vol. 2, 2000, pp. 917–920.
[17] G.J. Edwards, C.J. Taylor, T.F. Cootes, Learning to identify and track faces in image sequences, Proceedings of Third IEEE International Conference on Automatic Face and Gesture Recognition, 1998, pp. 260–265.
[18] Hsi-Jung Wu, D. Ponceleon, K. Wang, J. Normile, Tracking subspace representations of face images, Proceeding of 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 5, 1994, pp. 389–392.
[19] Juyang Weng, C.H. Evans, Wey-Shiuan Hwang, An incremental learning method for face recognition under continuous video stream, Proceedings of Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000, pp. 251–256.
[20] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME J. Basic Eng. 82 (1960) 35–45.
[21] T. Sim, S. Baker, M. Bsat, The CMU pose, illumination, and expression (PIE) database of human faces, Tech. report CMU-RI-TR-01-02, Robotics Institute, Carnegie Mellon University, January 2001.
[22] F.J. Huang, T. Chen, Tracking of multiple faces for human-computer interfaces and virtual environments, Proceeding of IEEE International Conference on Multimedia and Expo, New York, July 2000.
[23] James D. Hamilton, Time Series Analysis, Princeton University Press, Princeton, NJ, 1994.
[24] T. Otsuka, J. Ohya, Recognizing abruptly changing facial expressions from time-sequential face images, Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, 1998, pp. 808–813.
[25] H.J. Zhang, A. Kankanhalli, S. Smoliar, Automatic partitioning of full-motion video, Multimedia Systems 1 (1) (1993) 10–28.
[26] M.E. Tipping, C.M. Bishop, Mixtures of probabilistic principal component analyzers, Neural Comput. 11 (2) (1999) 443–482.
[27] X. Liu, T. Chen, Shot boundary detection using temporal statistics modeling, Proceedings of 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing, November 2001.