Partial Eigenvalue Decomposition of Large Images Using Spatial Temporal Adaptive Method

Hiroshi Murase and Michael Lindenbaum

Abstract— **Finding eigenvectors of a sequence of real images has usually been considered to require too much computation to be practical. Our spatial temporal adaptive (STA) method reduces the computational complexity of the approximate partial eigenvalue decomposition based on image encoding. Spatial temporal encoding is used to reduce storage and computation, and then, singular value decomposition (SVD) is applied. After the adaptive discrete cosine transform (DCT) encoding, blocks that are similar in consecutive images are consolidated. The computational economy of our method was verified by tests on different large sets of images. The results show that this method is 6 to 10 times faster than the traditional SVD method for several kinds of real images. The economy of this algorithm increases with increasing correlation within the image and with increasing correlation between consecutive images within a set. This algorithm is useful for pattern recognition using eigenvectors, which is a research field that has been active recently.**

I. Introduction

EIGENVECTOR analysis is important in many fields including signal processing, image analysis [1], [13], and pattern matching [2]. The eigenvectors form an orthogonal basis for the representation of individual images in the image set. Although a large number of eigenvectors may be required for very accurate reconstruction of the image, a much smaller number of eigenvectors is generally sufficient to represent the image. This is the basis of the image compression or coding technique known as the Karhunen–Loeve transform. Eigenvectors are also very attractive for machine vision. In particular, pattern recognition using image eigenvectors has been an active research area recently. This idea has been applied to several recognition problems, including handwritten character recognition [3], face recognition [4], [5], object recognition [6], and illumination planning [7].

In practice, however, computing eigenvectors of high-resolution images of large sample sizes often requires prohibitive amounts of computation. For example, eigenvectors of more than 1000 images, where each image has more than 16 384 (128 × 128) pixels, are typical in object recognition [6]. Calculating the eigenvectors of such a set using a traditional method is not easy on common workstations.

A wide variety of computational methods are found in the literature. Some of the methods suitable for very large data sets may be considered to be iterative gradient search algorithms [9], [15]. In addition, there are specialized methods for specific problems, such as decomposition of matrices having a Toeplitz structure [10], matrices arising in stochastic systems [8], or dynamic methods for use with data that arrive sequentially [11]. A useful approach for finding eigenvalue decomposition when the dimension of the vectors is larger than the number of the vectors in the set is singular value decomposition (SVD) [11], [12]. However, when the sample vectors are large images, this approach requires a large amount of computation. This approach is reviewed in the following section. For extremely large data sets, such as those arising in image recognition applications, the computation as a function of data size can still be unacceptable.

In this paper, we present a method called the spatial temporal adaptive (STA) method for computing the approximate partial eigenvalue decomposition of large image sets. Computation and storage requirements are reduced by the spatial temporal adaptive image encoding, and then singular value decomposition is applied. The coding algorithm is based on the traditional adaptive discrete cosine transform (DCT) encoding [16] followed by consolidation of the DCT coefficients of blocks that are similar in consecutive images. The STA algorithm provides results that approximate the ideal results in a predictable and controllable way. The economy of our algorithm increases with increasing correlation within the image and with increasing correlation between consecutive images within a set.

First, we review previous methods applicable to the present task. Then, we present the STA algorithm followed by a short analysis. Experimental results for various sets of images are then given, and the advantages of the new method are demonstrated.

II. Computing the Partial Eigenvector Expansion

We begin by introducing some terminology and then show the known methods. We assume the data to be a set of m vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ representing m images. Each component $x_1, \ldots, x_n$ of an image $\boldsymbol{x}$, where n is the number of pixels in the image, represents the intensity of one pixel. For convenience of notation, the images are combined into an $n \times m$ matrix $\boldsymbol{X}$, with each image $\boldsymbol{x}_i$ forming a column of $\boldsymbol{X}$, that is

$$\boldsymbol{X} = [\boldsymbol{x}_1 \quad \cdots \quad \boldsymbol{x}_m]. \tag{2.1}$$

The position of the image within the set will be referred to here as a "time" coordinate, even though not all image sets

Manuscript received January 12, 1993; revised March 10, 1994. The associate editor coordinating the review of this paper and approving it for publication was Prof. Robert M. Haralick.

The authors are with NTT Basic Research Laboratories, Atsugi-Shi, Kanagawa, Japan.

IEEE Log Number 9410202.

are time sequences. If the rank of the matrix $\boldsymbol{X}\boldsymbol{X}^T$ is K, then the K nonzero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^T$, $\lambda_1, \ldots, \lambda_K$, and their associated eigenvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K$ satisfy the fundamental eigenvalue relationship

$$\lambda_i \boldsymbol{e}_i = \boldsymbol{R}\boldsymbol{e}_i, \quad i = 1, \ldots, K \tag{2.2}$$

where $\boldsymbol{R}$ is the square, symmetric matrix computed from $\boldsymbol{X}$ and its transpose $\boldsymbol{X}^T$

$$\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^T. \tag{2.3}$$

The K eigenvectors form a basis, and the columns (original images) may be represented by their projection on these vectors. This special basis, which is adapted to the data, is also known as the Karhunen–Loeve or principal component basis. Often, the image data is approximated by retaining only the $k \leq K$ largest eigenvalues $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k|$ and their associated eigenvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k$. This partial set of k eigenvectors spans a subspace in which $\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_m$ are the projections of the original vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. What makes this partial basis special is that these projections are the best approximation of the original vectors with respect to the l^2 norm, that is, every other approximation of the m original vectors as a linear combination of k basis vectors gives an average l^2 error larger than the error approximated by using the eigenvectors.

A. Conjugate Gradient Method

A practical approach to computing the partial eigenvector decomposition is to use iterative methods. A relatively efficient method for finding the largest eigenvalue is to use the conjugate gradient method for finding maximal values of a function with a suitable scalar function [9]. This method overcomes the convergence problems associated with the original power method [14]. The function to be maximized is the Raleigh quotient $F(\hat{\boldsymbol{e}})$ defined as

$$F(\hat{\boldsymbol{e}}) = (\hat{\boldsymbol{e}}^T \boldsymbol{R}\hat{\boldsymbol{e}})/(\hat{\boldsymbol{e}}^T\hat{\boldsymbol{e}}). \tag{2.4}$$

Clearly, when $F(\hat{\boldsymbol{e}})$ takes on a maximum value, $F(\hat{\boldsymbol{e}})$ will be equal to $\lambda_{\max}$, and $\hat{\boldsymbol{e}}$ will be colinear with $\boldsymbol{e}_{\max}$. Changing $\boldsymbol{R}$ to find the next eigenvector involves updating the data by removing the dimension associated with the first eigenvector and then recalculating $\boldsymbol{R}$. Let $\boldsymbol{R}_s$ be the updated matrix used to calculate the sth eigenvector. Then

$$\begin{aligned} \boldsymbol{R}_1 &= \boldsymbol{R} \\ \boldsymbol{R}_s &= \boldsymbol{R}_{s-1} - \hat{\lambda}_{s-1}\hat{\boldsymbol{e}}_{s-1}\hat{\boldsymbol{e}}_{s-1}^T. \end{aligned} \tag{2.5}$$

This algorithm performs well, and the amount of computation required grows approximately linearly with the number of elements in $\boldsymbol{R}$ when n is large. However, the number of elements varies with the square of n; therefore, the initial computation of $\boldsymbol{R}$, which requires a dot product between two m-dimensional vectors for each element, becomes the limiting step for many practical problems. This problem can be avoided by a modification [14] that eliminates the need to use $\boldsymbol{R}$. The modified algorithm (the *Implicit* R algorithm) eliminates the explicit computation of $\boldsymbol{R}$ by using the data matrix $\boldsymbol{X}$ in each iteration. For many problems, this modification, however, is not faster. A more detailed explanation can be found in the recent survey [15] and in Section IV of this paper, which analyzes the performance of the different algorithms.

B. Singular Value Decomposition Algorithm

Although $\boldsymbol{R}$ is an $n \times n$ matrix, the matrix

$$\tilde{\boldsymbol{R}} = \boldsymbol{X}^T\boldsymbol{X} \tag{2.6}$$

is $m \times m$ and is much smaller in practical problems. The matrix $\tilde{\boldsymbol{R}}$ has eigenvalues $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_K$ and associated eigenvectors $\tilde{\boldsymbol{e}}_i, \ldots, \tilde{\boldsymbol{e}}_K$ related to those of $\boldsymbol{R}$ by

$$\left.\begin{aligned} \lambda_i &= \tilde{\lambda}_i \\ \boldsymbol{e}_i &= \tilde{\lambda}_i^{-1/2}\boldsymbol{X}\tilde{\boldsymbol{e}}_i \end{aligned}\right\}_{i=1,\ldots,K}. \tag{2.7}$$

The relation (2.7) was developed within the framework of the *singular value decomposition* theory [11]. We call $\tilde{\boldsymbol{e}}$ a coeigenvector and note that it is of length m, whereas the eigenvector $\boldsymbol{e}$ is of length n. When m is much smaller than n, using the above relations to compute the eigenvectors can greatly reduce the computational load. The coeigenvectors' elements are later used as weights for computing the true eigenvectors as linear combinations of the input images. This approach of first computing the coeigenvectors and then using them to compute the eigenvectors is referred to as the *singular value decomposition* (SVD) algorithm. This method performs well for some problems, but the limiting step with many large data sets is still the computation of $\tilde{\boldsymbol{R}}$.

As we shall see in the next section, our algorithm uses an efficient representation for computing the $\tilde{\boldsymbol{R}}$ matrix. Then, it calculates the coeigenvectors and converts them to the eigenvectors using the SVD algorithm. These sequences are, for example, often obtained in the visual learning stage in pattern recognition. Our method is suitable for the calculation of large image sequences for three reasons. First, we use a representation that is efficient for natural gray-level images. (Color is a natural and easy extension.) Second, our representation adapts itself to the similarity between different images in the given data set and uses it to improve the computation efficiency. Finally, the calculation of the $\tilde{\boldsymbol{R}}$ matrix is done not element by element but by computing contributions to blocks of elements at each step.

III. Spatial Temporal Adaptive (STA) Algorithm

Our approach is not to use the given pixel data itself but rather to use the DCT coefficients. The calculation is mainly done in the DCT (frequency) domain. The following equations show that both approaches are the same. Let $\boldsymbol{U}$ be an orthogonal matrix ($\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$). Generally, for any orthogonal transformation $\boldsymbol{U}$ of the data $\boldsymbol{X}$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{X} \tag{3.1}$$

and the eigenequation

$$\lambda\tilde{\boldsymbol{e}} = \boldsymbol{X}^T\boldsymbol{X}\tilde{\boldsymbol{e}} \tag{3.2a}$$

can be rewritten as

$$\lambda \tilde{e} = X^T U^T U X \tilde{e} \quad (3.2b)$$

$$= Y^T Y \tilde{e}. \quad (3.2c)$$

Thus, any orthogonal transformation U that is convenient for computing $Y^T Y$ may be applied without changing the computed $\tilde{e}$. The DCT is an orthogonal transformation, and it represents images efficiently. This fact is our motivation.

The images are approximated by the small number of the DCT coefficients [18]. In other words, we use the approximate images instead of the accurate images. The first two stages in the algorithm are thus dedicated to approximating the data efficiently. In the first stage, every image is divided into square blocks with 8 pixels on a side, and each block is represented adaptively by a partial collection of its discrete cosine transform (DCT) coefficients. We use a variation of the method of Chen and Smith [16], allocating more coefficients to blocks that have a high variance. The second step looks at blocks at the same location in different images and, if the difference between blocks is small enough, merges them into spatio-temporal volumes. The third stage uses the spatio-temporal volumes to calculate the matrix $\tilde{R}$, but unlike the traditional methods, it does not calculate every component of the matrix separately. Instead, it computes contributions to blocks of components together. These contributions lead to a faster calculation of the $\tilde{R}$ matrix. The algorithm then uses the conjugate gradient method to compute the coeigenvectors and finally calculates the eigenvectors adaptively using the spatio-temporal volume structure. The main algorithm is as follows.

STA Algorithm

Preprocessing

1) Divide each image into 8 × 8 pixel blocks, and find its DCT coefficients and ac energy.

Coefficient Selection

2) Using the first image and an energy histogram, find three energy thresholds that classify the blocks into four equal classes.
3) Estimate the variance of the DCT coefficients in each of these classes, and find the required bit allocation for each coefficient. If the rate is higher than 1, this coefficient is selected.

Merging Blocks into Spatio-Temporal Volumes

4) For each of the images starting from the second one, compare each block with the corresponding block in the same place in the previous image. If the mean square difference is less than ϵ_t, merge these two blocks into a spatio-temporal volume. Otherwise, start a new spatio-temporal volume with this block.

Find the $\tilde{R}$ Matrix from the Spatio-Temporal Volume Description

5) For each block location k, evaluate the dot product of every pair of spatio-temporal volumes related to that location and add it to the corresponding terms of the matrix $\tilde{R}^k$. Sum all the matrices $\tilde{R}^k$ to get the matrix $\tilde{R}$.

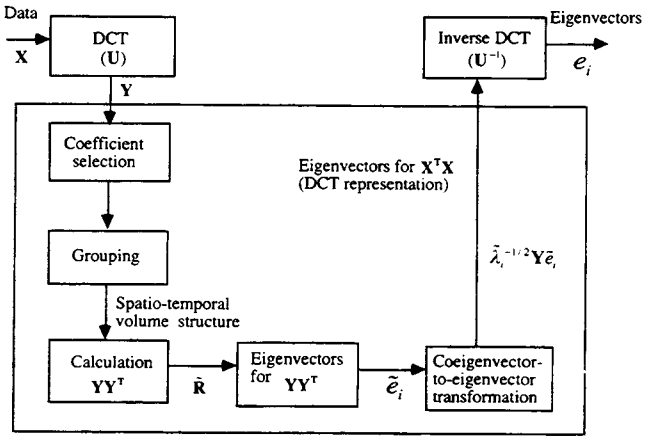


Fig. 1. Block diagram for the spatial temporal adaptive (STA) method.

Complete the Eigenvector Calculation

6) Find the coeigenvectors $\tilde{e}_1, \ldots, \tilde{e}_k$ that maximize the Raleigh quotient of the matrices $\tilde{R}_1, \tilde{R}_2, \ldots$ defined by (2.5) using the conjugate gradient method.
7) Compute the DCT representation of the eigenvectors $\tilde{e}_1, \ldots, e_k$ from the coeigenvectors using the spatio-temporal volume structure.
8) Transform the derived eigenvectors $e_1, \ldots, e_k$ in the DCT domain into the image domain using inverse DCT.

In the rest of this section, we describe the STA algorithm in detail. A block diagram is given in Fig. 1.

The nonstationarity of visual images implies that representing them efficiently requires local adaptation. We follow the classical and frequently used approach of Chen and Smith [16], who gave an efficient method for coding an image using an efficient representation of the block DCT coefficients. They start by dividing an image into blocks and finding the DCT coefficients of each block. These DCT coefficients are much less correlated than the gray levels and are thus easier to code efficiently. The blocks are divided into four classes of equal size, according to their ac energy. We took this number of classes (four) from Chen and Smith's paper [16]. The variances of the DCT coefficients, which are assumed to be independently distributed Gaussian random variables, are estimated for each class. Optimal mean square error (MSE) representation of a set of Gaussian independent random variables with known variance is a problem that has already been solved in rate distortion theory [17]; therefore, deciding how many bits to use to represent each coefficient is straightforward. Because the goal of Chen and Smith's algorithm is to encode the image efficiently, it proceeds by optimal Lloyd–Max quantization and obtains the quantized coefficient as the efficient representation. Here, we assume that DCT coefficients have a Gaussian distribution because this rough assumption is sufficient for our purpose. However, we can easily change this bit allocation step to a more accurate method for nonGaussian data.

A. Coefficient Selection

Rather than trying to reduce the computational cost by using less precision of representation, we try only to reduce the number of floating point calculations. Thus, we select only

13	13	21	13	21	21	21	24	21	21	24	24	4	4	4	4
13	13	21	21	21	24	21	24	13	24	21	24	13	4	4	4
13	13	24	13	13	21	21	13	24	24	24	21	24	4	4	4
21	21	21	21	13	24	24	24	24	24	4	21	24	4	4	4
21	21	21	21	21	24	21	21	24	24	4	21	24	4	4	4
21	21	13	21	13	21	21	21	21	24	4	24	13	4	4	4
13	13	13	13	13	4	24	21	21	24	4	24	4	4	4	4
21	13	21	21	13	4	21	4	13	21	4	4	4	4	4	4
13	4	13	4	13	13	24	24	24	24	4	4	13	4	4	4
24	24	24	24	24	24	24	4	24	24	4	4	21	21	21	21
13	13	13	13	13	24	24	21	24	13	13	13	24	21	4	13
21	13	21	21	24	21	24	24	24	24	24	24	24	24	24	24
21	21	21	21	24	24	24	24	24	24	24	13	21	21	4	4
21	21	13	21	13	13	24	13	24	4	4	21	13	13	13	13
13	21	13	21	13	4	4	21	13	4	4	13	13	13	13	13
13	4	13	13	4	4	4	13	4	4	4	13	13	4	4	13

Fig. 2. Number of coefficients retained in each block for the first image of the "walking" set (the upper image in Fig. 4(a)). The image size is 128 × 128 pixels. This image is represented by 16 × 16 blocks, where each block size is 8 × 8 pixels.

coefficients whose allocated rate is one bit or more, and we discard the rest. The variance statistics and the selection maps are derived from only one of the images, which is thus assumed to be characteristic of the rest. This assumption could be avoided by calculating average values of all the blocks in the image set or calculating them from a randomly selected subset of the blocks.

Suppose that for some particular class out of four classes, the variances of the coefficients $c_1, c_2, \ldots$ are $\sigma_1^2, \sigma_2^2, \ldots$. For Gaussian coefficients and a prespecified average block distortion ε_s, the optimal bit allocation can be calculated as

$$\text{RATE}(c_i) = \max\left\{\log_2\left(\frac{\sigma_i^2}{\theta}\right), 0\right\} \tag{3.3a}$$

where θ is the biggest number that satisfies the relation [16]–[18]

$$\varepsilon_s \leq \sum_i \min\{\sigma_i^2, \theta\}. \tag{3.3b}$$

This calculation is applied for four different classes.

The result of this stage is four different selection maps: one for each of the classes. The number of coefficients retained depends on the spatial average block distortion ε_s, which determines the quality of the approximation. For example, to achieve an average approximation SNR of 29 dB in the first image, the number of retained coefficients is reduced from 64 to 24, 21, 13, and 4 in the four corresponding classes. As an illustration, Fig. 2 gives the number of coefficients retained in 8 × 8 pixel blocks for the first image in the "walking" set shown in Fig. 3(a). Because the dot product between two blocks represented by n_1 and n_2 coefficients can be calculated in the DCT transform domain with only $\min\{n_1, n_2\}$ multiplications, we save a substantial amount of computation time if we simply use the results of this stage to straightforwardly calculate the elements of the $\tilde{R}$ matrix. In the actual program, we use pointer tables that indicate which coefficients should be multiplied; therefore, we need not scan all coefficients every time. In addition, we can save even more by using the correlation between images in the set, and so, we proceed to a grouping stage that merges blocks that are in corresponding locations on adjacent images.

B. Using Interimage Redundancy by Merging Similar Corresponding Blocks

The member $\tilde{r}_{ij}$ of the matrix $\tilde{R}$ is a dot product between the ith and jth images in the given set

$$\tilde{r}_{ij} = \boldsymbol{x}_i \bullet \boldsymbol{x}_j = \sum_{l=1}^{N_b} \boldsymbol{b}_i^l \bullet \boldsymbol{b}_j^l \qquad i, j = 1, 2, \ldots, m, \tag{3.4}$$

where $\boldsymbol{b}_i^l$ is the DCT coefficient vector representing the lth block in the ith image, and N_b is the number of blocks in an image. This means that the single dot product between two images can be decomposed into a summation of dot products between lower dimensional coefficient vectors that correspond to the blocks. This is the key to using local similarity to save computation.

For many practical sets of images, such as a set describing isolated objects in different positions or a time sequence set taken with a stationary camera, blocks in the same location on adjacent images are often very similar. Many dot product terms in (3.4) are therefore often identical, and repeated calculation of these dot products can be avoided if we identify the similarities between blocks.

A simple and efficient way is to do this is by first merging a continuous series of nearly identical consecutive blocks into "spatio-temporal volumes." The volume here is a set of blocks. We choose the first block in a certain location to be the representative of that location's first spatio-temporal volume. Then, we compare this representative block with the corresponding block from the second image, and if their mean square difference is smaller than some threshold ε_t, we add the second block to the spatio-temporal volume. Otherwise, the volume is terminated, and a new volume is begun with the second block as its representative. The coefficients considered for finding this difference are the unions of the subsets of coefficients retained for these two blocks. This process continues until all m blocks at this location have been checked and grouped into spatio-temporal volumes; then, we proceed to group the blocks in other locations.

The result of this stage is a separate representation for each possible block location. Each representation is a connected list of spatio-temporal volumes that are described by the selected DCT coefficients of its representative block and the time indices of the images in which it begins and ends (see Fig. 4 for an illustration of the spatio-temporal volume structure).

C. Finding the $\tilde{R}$ Matrix from the Spatio-Temporal Volume Description

We use the spatio-temporal volume description to efficiently calculate the $\tilde{R}$ matrix. Let $\tilde{R}^l$ be partial $\tilde{R}$ matrices whose elements are dot products between blocks in the same location l but from images i and j.

$$\begin{aligned}\tilde{R}^l &= \{\tilde{r}_{ij}^l \mid i, \quad j = 1, 2, \ldots, m\} \\ &= \{\boldsymbol{b}_i^l \bullet \boldsymbol{b}_j^l \mid i, \quad j = 1, 2, \ldots, m\}.\end{aligned} \tag{3.5}$$

(a) (b) (c) (d) (e)

Fig. 3. Five sets of image sequences used in the experiments: (a) Walking; (b) talking; (c) bicycle; (d) eraser; (e) toy. Each set is composed of 256 images, each of 128 × 128 pixels. This figure shows two image examples from each set.

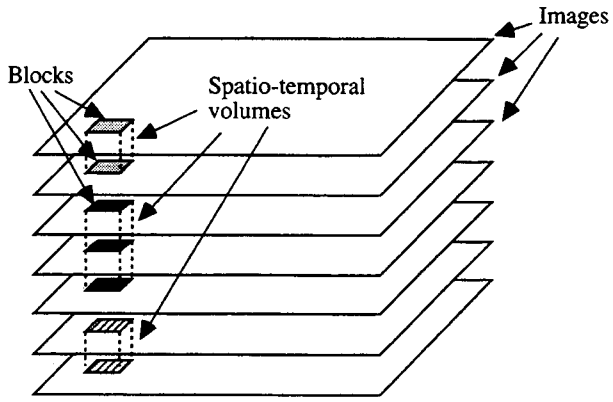


Fig. 4. Spatio-temporal volume structure derived by the grouping process.

Clearly, the linearity of the dot product implies that

$$\tilde{\boldsymbol{R}} = \sum_{l=1}^{N_b} \tilde{\boldsymbol{R}}^l. \tag{3.6}$$

Thus, instead of calculating $\tilde{\boldsymbol{R}}$ element-by-element, we can calculate it as the sum of the partial matrices $\tilde{\boldsymbol{R}}^l$.

We calculate each partial matrix $\tilde{\boldsymbol{R}}^l$ only from spatio-temporal volumes that correspond to location l. For each pair of such volumes, we form the dot product between their representative blocks and add it to the corresponding elements of the matrix. If one spatio-temporal volume includes blocks from images starting in the i_S^1th image and ending in the i_T^1th and another volume includes blocks from images starting in the i_S^2th image and ending in the i_T^2th, then all the elements $\tilde{r}_{ij}^l$ of the matrix $\tilde{\boldsymbol{R}}^l$ that satisfy $i_S^1 \le i \le i_T^1$; $i_S^2 \le j \le i_T^2$ or $i_S^2 \le i \le i_T^2$; $i_S^1 \le j \le i_T^1$ are identical and are equal to the dot product between the corresponding representative blocks.

The symmetry of $\tilde{\boldsymbol{R}}$ is used, and only elements $\tilde{r}_{ij}$ for which $i \ge j$ are actually calculated. Thus, if the number of spatio-temporal volumes corresponding to a particular location l is much smaller than the number of images m, computation is substantially reduced. This adds to the initial benefit of using a subset of the coefficients to calculate each dot product. Only the intersections of the subsets of coefficients retained for the two representing blocks should be used, and because the larger subset usually contains the smaller one, the number of coefficients used is the minimum. Section IV of this paper uses a simple model to estimate the computational saving more quantitatively.

D. Finding the Coeigenvectors

After finding the matrix $\tilde{\boldsymbol{R}}$, we proceed according to the *singular value decomposition algorithm* described in Section II. We start by finding the coeigenvectors of $\boldsymbol{R}$, $\tilde{\boldsymbol{e}}_1, \ldots, \tilde{\boldsymbol{e}}_k$, which are the eigenvectors of $\tilde{\boldsymbol{R}}$ and by finding their associated eigenvalues $\lambda_1, \ldots, \lambda_k$. The conjugate gradient method is used to find the vector that maximizes the Raleigh quotient $F(\hat{\boldsymbol{e}})$ (see (2.4)) of $\tilde{\boldsymbol{R}}$. This maximizing vector is the first eigenvector. Then, $\tilde{\boldsymbol{R}}$ is updated using (2.5), and the process is repeated until the specified number of coeigenvectors has been calculated.

E. Finding the Eigenvectors

The eigenvectors are calculated as linear combinations of the input data, weighted by the corresponding coeigenvector elements. Instead of using the raw input data as in (2.7), we use the adaptive spatio-temporal structure to reduce the computation at this stage as well. This is explained by the following simple equation. We can rewrite (2.7) using (3.1) as follows:

$$\boldsymbol{e}_i = \tilde{\lambda}_i^{-1/2} \boldsymbol{X} \tilde{\boldsymbol{e}}_i \tag{3.7a}$$

$$= \tilde{\lambda}_i^{-1/2} \boldsymbol{U}^{-1} \boldsymbol{Y} \tilde{\boldsymbol{e}}_i \tag{3.7b}$$

$$= \boldsymbol{U}^{-1} (\tilde{\lambda}_i^{-1/2} \boldsymbol{Y} \tilde{\boldsymbol{e}}_i), \quad i = 1, \ldots, K \tag{3.7c}$$

where $\boldsymbol{Y}$ is the compact representation of images, and $\boldsymbol{U}^{-1}$ is an inverse DCT transformation (see Fig. 1). The calculation using (3.7c) is much more efficient than that using (2.7)

The linear combination is calculated for each block location separately using the partial DCT coefficient representation. For each spatio-temporal volume, instead of adding all the blocks included in it—weighted by their corresponding coeigenvector elements—we add the representative block weighted by the sum of these coeigenvector elements. This is similar to the procedure calculating $\tilde{R}$ described in Section III-C. Finally, an inverse DCT transformation changes the eigenvectors into a representation of image intensity.

IV. Comparative Performance Analysis of the Algorithms

Here, we introduce a simple model for the image set and use it to evaluate the number of multiplications needed by our new STA algorithm and the conventional SVD algorithm. Reference to the *Implicit* R method [14] is also included.

Let n be the image size, m be the number of images, k be the number of required eigenvectors, and i be the number of iterations. (Twelve iterations usually suffice.) The total number of multiplications required by the SVD method is

$$N_{\mathrm{SVD}} = N_{\tilde{R}}^{\mathrm{direct}} + N_{\mathrm{coeigenvectors}} + N_{\mathrm{eigenvectors}} \quad (4.1a)$$

$$= \frac{1}{2}nm^2 + kim^2 + knm. \quad (4.1b)$$

The STA algorithm depends on approximating the images. Therefore, its performance depends on the degree of approximation, represented by the number of coefficients retained in the DCT approximation and represented by the length of the spatio-temporal volumes. Here, we show a rough estimation of the number of multiplications (a more detailed estimation is given in [19]). Let the parameters β_a and β_b be average compression factors in the first DCT approximation (step 1 in the algorithm) and the second DCT approximation (step 7 in the algorithm), respectively. The performance also depends on the variation between images expressed by parameter α, which is the probability of a new spatio-temporal volume being created. Detailed analysis yields the following expression in which each term reflects the computational effort (number of multiplications) of some stage in the STA algorithm

$$N_{\mathrm{STA}} = N_{\mathrm{DCT}} + N_{\tilde{R}}^{\mathrm{adaptive}} + N_{\mathrm{coeigenvectors}} + N_{\mathrm{eigenvectors}}^{\mathrm{adaptive}} + N_{\mathrm{IDCT}} \quad (4.2a)$$

$$= 1.25mn + nm(1+\alpha^2)\beta_a + kim^2 + mn(1+k\alpha)\beta_b + 1.25kn. \quad (4.2b)$$

For comparison, *Implicit* R method requires

$$N_{\mathrm{Implicit\_}R} = kimn. \quad (4.3)$$

multiplications.

To compare the various methods using the model derived in this section, we inserted the parameters that are typical of the "walking" set of images used for the demonstration in the next section (see Fig. 3(a)). These parameters were chosen to allow very little degradation in the quality of the derived eigenvectors (The next section defines this degradation quantitatively.) The average number of iterations was between 10 and 14, depending on the images; therefore, we set $i = 12$ for the model. Using this model, the number of multiplications was plotted for small (128 × 128) and big (1024 × 1024) images as a function of the number of images in the set (Fig. 5(a)). We found that the number of multiplications computed from the model roughly matched the time actually required for the algorithms. This demonstrates that the model is valid for predicting their performance. We also analyzed the storage requirements [19]. The theoretical requirements are plotted in Fig. 5(b). These graphs show the advantages of the STA algorithm.

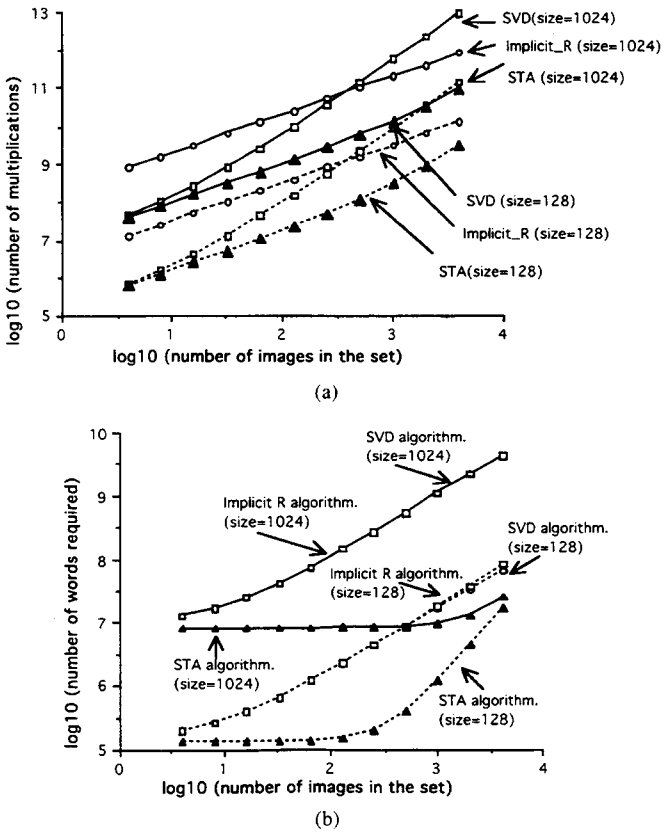


Fig. 5. (a) Number of multiplications and (b) the storage requirements for small (128 × 128) and large (1024 × 1024) images, plotted as a function of the number of images in the set.

The advantages of our method become even more apparent when higher dimensional data is used. The gap between the computation time of the classic SVD method and our new STA method increases for larger sets and larger images. For example, when the number of images in the set is more than a few thousand, the STA method for 1024 × 1024 pixel images is faster than the SVD method for 128 × 128 pixel images. For a very large number of images, the Implicit-R method becomes competitive—first with the SVD method but also later with the STA method. However, when the speed of the Implicit-R method becomes comparable with that of the STA method, the amount of main memory required becomes prohibitively large. For such large amounts of data, the new STA method provides a much more efficient tradeoff between main memory requirements and the time required to use external memory.

TABLE I
SUMMARY OF RESULTS

Method	Set	Parameters		Quality			Result
		Approx. SNR_1 (dB)	Approx. SNR_2 (dB)	Reconst. error True / Approx.	Sub-space criterion	Relative SNR (dB)	Total computing (s)
SVD	*	∞	∞	---	1	∞	1141
STA	walking	29.1 25.1	41.6 41.6	0.02610 / 0.02704 0.02610 / 0.02725	0.981 0.870	42.5 37.4	164 135
	toy	22.9 22.8	42.0 42.0	0.08601 / 0.08636 0.08601 / 0.08665	0.999 0.996	45.0 41.0	129 116
	eraser	25.9 21.9	48.4 48.4	0.14094 / 0.14100 0.14094 / 0.14108	0.999 0.985	46.8 34.8	170 138
	talking	34.1 26.7	48.1 48.1	0.00517 / 0.00522 0.00517 / 0.00532	0.993 0.951	53.3 46.8	188 140
	bicycle	21.6 18.9	38.5 38.5	0.39772 / 0.40012 0.39772 / 0.40182	0.983 0.958	35.7 32.0	184 148

* This row is for the SVD algorithm, which gave the same results for each image set.

V. EMPIRICAL DEMONSTRATION OF THE STA METHOD

A. Data

We demonstrated our method on some typical sets of images. We considered two types of data: fixed and changing views of a moving object and changing views of a stationary object. In the "walking" and "talking" image sets, the camera was stationary while the object moved. (Figs. 3(a) and (b) show two images from each set). In the "bicycle" set, both the camera and the bicycle moved (Fig. 3(c)). In the "eraser" and "toy" image sets, the object was rotated while the camera was fixed. The "eraser" set of images was derived from a single image given different 2-D rotations by software (Fig. 3(d)), and the toy was given real 3-D rotations (Fig. 3(e)). The image sets tested included 256 images, each with 128 × 128 pixels.

We describe in detail the results for the "walking" set under different parameters and criteria but only summarize numerical results for the other image sets in Table I.

B. Parameters

The parameters that affect our algorithm's performance are the degrees of approximation allowed in the various stages. We found experimentally that the best result was achieved by making the spatial MSE ε_s allowed in the coefficient selection stage equal to the temporal MSE ε_t allowed in the grouping stage. We also found, however, that in the final stage—finding the eigenvectors from the coeigenvectors—approximation should be finer. The degree of approximation in each stage was measured by the average SNR, which reflects the errors due to selecting only some coefficients when approximating the first image. After defining the criteria used in experimentally evaluating the algorithm-derived eigenvectors, we will give some guidelines for choosing the approximation parameters according to the desired performance.

C. Criteria

We compared the quality of the eigenvectors obtained by the STA with the eigenvectors obtained by the SVD method, which are denoted *true eigenvectors* $e_1^T, e_2^T, \ldots, e_k^T$ and to the original data. Direct comparison between each of the true eigenvectors e_i^T and the corresponding *approximated eigenvector* e_i^A obtained using the STA method may be misleading because the order of two eigenvectors with very close eigenvalues can be changed by even a small approximation. We chose instead to compare the approximated partial eigenvectors with the true partial eigenvectors. Because the approximated eigenvectors may not be exactly orthogonal, an orthogonalization process that provides orthogonalized approximated eigenvectors $v_1^A, v_2^A, \ldots, v_k^A$ is applied when needed. We think this potential nonorthogonality, which is in any case small, is not a major deficiency. This is because applications that involve extracting only a few eigenvectors usually do not require reconstruction, and representation and classification tasks are not strongly affected by slight nonorthogonality. If exact orthogonality is needed, orthogonalization can be performed with less than one additional second of computation time.

The first evaluation criterion is the degree to which k approximated eigenvectors span the subspace spanned by the first k true eigenvectors. This is measured by the double sum

$$\text{Subspace Criterion} = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{k} v_i^A \bullet e_j^T \qquad (5.1)$$

which is 1 if the entire subspace is spanned and is smaller than 1 otherwise.

The subspace criterion might seem too conservative because it weights all the eigenvectors equally. A criterion that weights them according to their contribution to the image is the *reconstruction error criterion*, which specifies how much of the original energy of the images is preserved in the approximated eigenvectors. Let x_j be the jth image in the set. Then, x_j^A, which is the image approximated from its projections on the k approximated eigenvectors, is given by

$$x_j^A = \sum_{i=1}^{k} (x_j \bullet v_i^A) v_i^A. \qquad (5.2)$$

Denoting the energy of an image x_j by $\|x_j\|^2$, the reconstruction error criterion is computed as

$$\text{reconstruction error criterion} = \left\{ \frac{\|x_j^A - x_j\|^2}{\|x_j\|^2} \right\} \qquad (5.3)$$

where $\{\}$ is the average over all images $\{x_j,\ j = 1, \ldots, m\}$ in the set. By the energy packing optimality of the Karhunen–Loeve expansion, only the true eigenvectors $e_1^T, e_2^T, \ldots, e_k^T$ will preserve the maximal energy and yield a minimal reconstruction error. Thus, the reconstruction error achieved by the approximated eigenvectors is a measure of how close they are to the true eigenvectors. Another measure is the *relative SNR criterion*, which is defined as

$$\text{relative SNR criterion} = 10 \log_{10} \left\{ \frac{\|x_j^T\|}{\|x_j^A - x_j^T\|} \right\}. \qquad (5.4)$$

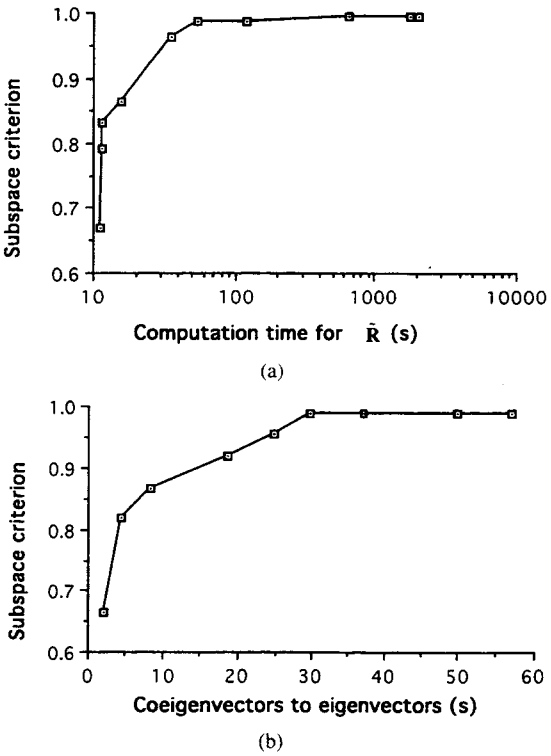


Fig. 6. Tradeoff between the quality of eigenvectors (measured by the subspace criterion) and the computation time for the "walking" image set: (a) Criterion as a function of the computation time for different approximations in the first stage; (b) criterion as a function of the eigenvector computation time for different approximations in the second stage.

(Here, x_j^T is the image approximated by the true eigenvectors and defined by an equation similar to (5.2).) This criterion is closely related to an observer's ability to distinguish between images spanned by the true eigenvectors and the corresponding images spanned by the approximate eigenvectors. We found that when this criterion was 35 dB or more, the two approximations were visually indistinguishable.

The parameters that allow significant computational savings while keeping these criteria high depend on the images and should be found experimentally, but the following lower bounds on the approximation SNR's can be used as rough guidelines. We found that to achieve a certain *relative* SNR, the approximation in the coeigenvector-to-eigenvector transformation stage should be fine enough that the approximation SNR of the second stage SNR_2 is not lower than the desired *relative* SNR. This is justified intuitively because approximating the images should result in a similar approximation of their linear combinations: the eigenvectors. Interestingly, the approximation allowed in the first stage is much coarser. We found that if the approximation error in this stage is roughly 20 times smaller (13 dB) than the reconstruction error using true eigenvectors, then using the approximated eigenvectors for reconstruction increases the reconstruction error by only 1%. With the "walking" set, for example, the first eight eigenvectors give a reconstruction error of 0.026, which corresponds to a reconstruction SNR of 16 dB. Thus, according to our guideline, the approximation SNR of the first stage SNR_1 was chosen to be 13 dB higher, i.e., 29 dB. The low approximation SNR in the first stage allows a very efficient spatio-temporal volume structure representation, which yields significant computational gain. Even though the images are approximated with a large error in the first stage (29 dB), a finer approximation in the second stage—which requires much less time to calculate—yields a high relative SNR (up to 48 dB in this example).

```
Class = 1                    Class = 2

1 1 1 1 1 1 0 0              1 1 1 1 1 1 0 0
1 1 1 1 1 1 0 0              1 1 1 1 0 0 0 0
1 1 1 1 1 0 0 0              1 1 1 1 1 0 0 0
1 1 1 1 0 0 0 0              1 1 1 0 0 0 0 0
1 1 0 0 0 0 0 0              1 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0              1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0

Class = 3                    Class = 4

1 1 1 1 1 0 0 0              1 1 0 0 0 0 0 0
1 1 1 1 0 0 0 0              1 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0              1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0              0 0 0 0 0 0 0 0
```

Fig. 7. Coefficient selection maps for the four energy classes ("walking" image set). The number of coefficients selected in each class is {24, 21, 13, 4}.

```
 4  4 25 33 28 32 35 68 28 24 22 10  4  2  2  2
 8  7 32 41 45 79 50 25 28 54 43 27  8  3  4  4
 8 12 20 17 12 32 11 18 36 34 68 44 32  1  5  5
47 30 23 32 31 25 29 48 36  5  1 59 31  2  5  2
23 43 44 77 51 48 63 137 96 26  4 43 14  2  2  2
23  3  4 28 21 47 55 80 74 12  1 17  1  4  1  1
 3  5  1  4  3 66 162 139 162 95  1  5  1  4  1  2
 1  5  2  2  1 98 72 19 74 156  1  1  1  2  2  2
 4  2  2  2 16 106 131 70 153 201 16  1  1  4  2  6
 2  2  2  2 12 123 204 115 182 216 19  1  1  8  4  4
 2  2  2  2  9 124 152 64 182 177 15  2  5  8 25 15
 2  2  2  2  9 63 79 93 187 82  8  4  6  4  7  3
 2  3  2  3  6 40 106 121 181 28  5  2  1  5  1  2
 2  2  4  3  2 32 81 88 133  2  1  2  1  2  1  3
14 16  2  5  1 15 72 140 108  2  2  2  1  1  4  4
39 25  5  2  4  9 65 105 76  1  1  9 11 19 12 14

Average = 32.4
```

Fig. 8. Number of spatio-temporal volumes created at each block location ("walking" image set). A lot of volumes are created at moving parts in the image.

D. Results

An Apollo DN10000 workstation was used for all the experiments. Approximations with high errors could of course be computed more quickly, but then, the approximated eigenvectors did not span the optimal subspace or pack the maximal energy. (Fig. 6 shows the tradeoff between computation time of the $\tilde{R}$ matrix and the reconstruction error for the "walking" image set.) As a conservative condition, we used approximation parameters that yielded eigenvectors that gave a reconstruction error only 1% higher than the reconstruction error using true eigenvectors.

With these conservative parameters, coefficient selection maps (Fig. 7) were derived by the STA method. The number of coefficients selected in each class was {24, 21, 13, 4}. The number of coefficients selected for each block was adapted to the amount of spatial activity at that block's location (see

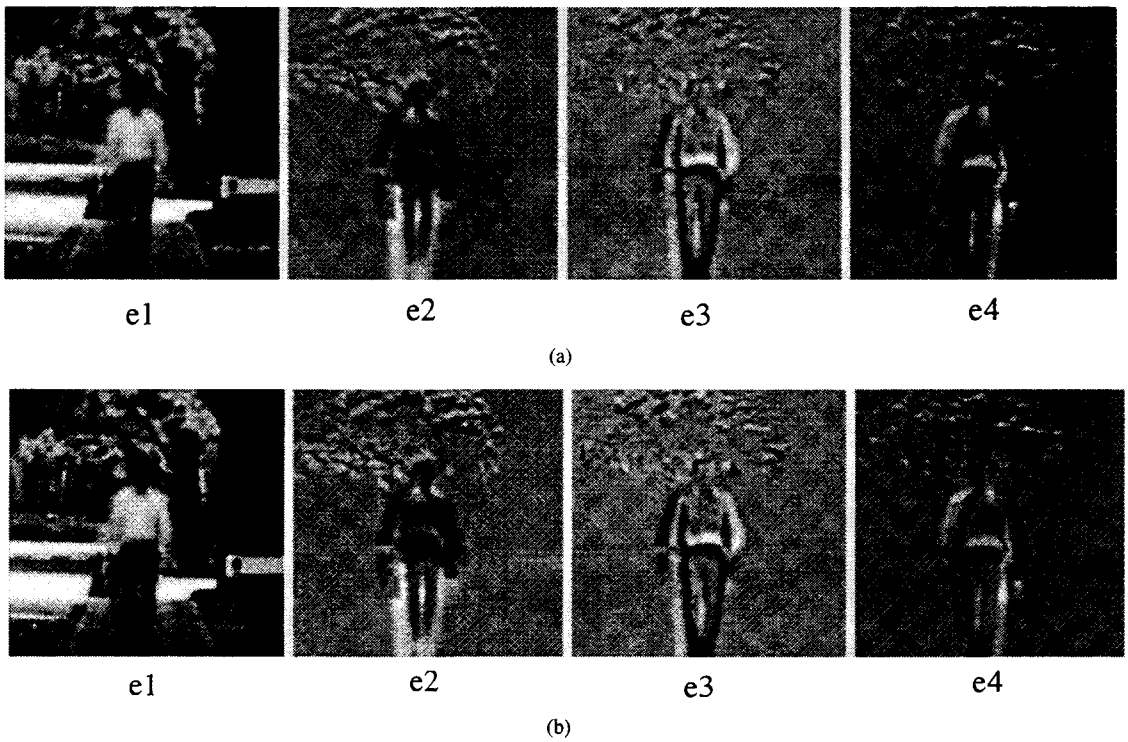


Fig. 9. First four true eigenvectors derived by the (a) SVD method and the first four approximated eigenvectors derived by the (b) STA method ("walking" image set).

Fig. 2). Then, the grouping process created the spatio-temporal volume structure. The number of volumes created at each location (Fig. 8) depends on the local change between the images: more spatio-temporal volumes were created at the center of the image, where the person was walking, and at the top of the image, where the trees moved. The number of spatio-temporal volumes created will depend on the approximation parameter, but a significant saving of unnecessary calculations is clear even with the conservative parameter chosen for this experiment—on average, 32.4 volumes represented the 256 blocks at each location.

Because the approximation parameter used in transforming the coeigenvectors into eigenvectors is much smaller than that used in the calculation of $\tilde{R}$, more DCT coefficients are selected, and more spatio-temporal volumes are created. In fact, for the "walking" example with the conservative criterion, the selection of coefficients yielded relatively small savings in computation, and the grouping stage did not produce any saving. The number of coefficients selected were {61, 59, 48, 26}, and on average, 255.3 spatio-temporal volumes represented the 256 blocks at each location. For other image sets and with a less severe criterion, this stage was calculated more efficiently, but the higher approximation SNR required in the second stage means that the saving will always be smaller than in the first stage.

The first few approximated eigenvectors were not visually distinguishable from the true eigenvectors (Fig. 9). The most time-consuming part of the computation, which used to be the calculation of $\tilde{R}$, was reduced from 1080 to 54 s (Fig. 10). The time needed to compute the eigenvectors from the co-eigenvectors was reduced from 47 to 37 s. However, a DCT

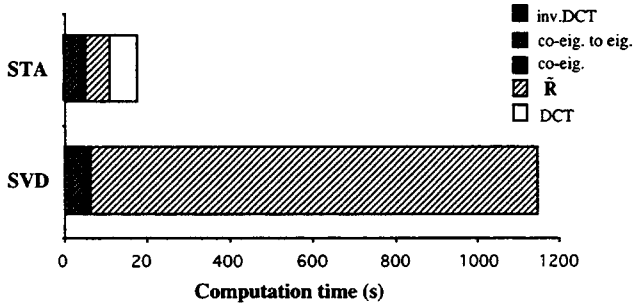


Fig. 10. Computation time for the different stages in the STA and SVD algorithms ("walking" image set).

computation time of 68 s must be added to both the input and the output times of the algorithm.

The results for each image set are quantitatively summarized in Table I, which lists the three criteria and the computation times for different approximations. The SVD method took 1141 s to compute the partial eigenvalue decomposition for any of the tested sets. The STA method clearly saves a substantial amount of computation while keeping the approximated eigenvectors very close to the true eigenvectors. In all cases tested, including the motion sequence taken with a moving camera, the time required for the STA method (116 to 188 s) was less than that needed for the SVD method (1141 s).

E. Potential Improvements Using Hardware

With the Apollo DN10000 workstation, it was possible to process 1–2 frames/s using software only. The most time-consuming stage in the STA algorithm is the standard block DCT, which may be eliminated if input images are already

available in block DCT representation. This transform can be computed using on-the-shelf components that can process 2–3 frames/s. (Commercially available chips can calculate the DCT of 256 images—128 × 128 pixels—in less than a second.) Using a vector processor to calculate dot products could reduce the calculation time even more. Finally, efficient parallel implementation is straightforward because the computationally expensive parts of each stage can be divided into independent blocks.

VI. Conclusion

The use of eigenvector techniques is limited by the high computational cost of finding the eigenvectors; therefore, for large sets of high-dimensional data, like image sets, the computational burden has been prohibitive. The spatial temporal adaptive (STA) algorithm presented here allows fast calculation of approximated eigenvectors of large image sets. Spatial temporal adaptive encoding exploits redundancy both within an image and between the images in a set before singular value decomposition (SVD) is applied. The STA algorithm provides results that approximate the ideal results in a predictable and controllable way. It allows a tradeoff to be made between computation time and the quality of the eigenvectors. We demonstrated theoretically and empirically that the STA method yields a large saving of computing time. For the different kinds of images used and for the high-quality approximation we demanded, the partial eigenvector expansion could be computed within 116–188 s, compared with 1141 s using the SVD algorithm. This computational advantage is expected to increase with increasing image size and set size. For large amounts of data that require a prohibitively large amount of memory, the STA method can provide an efficient tradeoff between main memory requirements and the time required to access external memory. The STA method thus makes it practical to use eigenvector techniques to analyze large sets of real gray-level images.

References

[1] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* London: Academic, 1990.

[2] E. Oja, *Subspace Methods of Pattern Recognition.* Hertfordshire, UK: Research Studies, 1983.

[3] H. Murase, F. Kimura, M. Yoshimura, and Y. Miyake, "An improvement of the auto-correlation matrix in the pattern matching method and its application to handprinted 'HIRAGANA' recognition," *IECE Trans.*, vol. J64-D, no. 3, 1981.

[4] L. Sirovich and M. Kirby, "Low dimensional procedure for the characterization of human faces," *J. Opt. Soc. Amer.*, vol. 4, no. 3, pp. 519–524, 1987.

[5] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," *Proc. IEEE Conf. Comput. Vision Patt. Recogn.*, Maui, June 1991, pp. 586–591.

[6] H. Murase and S. K. Nayar, "Learning object models from appearance," in*Proc. AAAI-93, Amer. Assoc. Artificial Intell.*, Washington DC, July 1993, pp. 836–843.

[7] ——, "Illumination planning for object recognition using parametric eigenspaces," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 16, no. 12, pp. 1219–1227, Dec. 1994.

[8] A. K. Jain, "A fast Karhunen–Loeve transform for a class of random processes," *IEEE Trans. Commun.*, pp. 1023–1029, Sept. 1976.

[9] R. Haimi-Cohen and A. Cohen, "Gradient-type algorithms for partial singular value decomposition," *IEEE Trans. Patt. Analy. Machine Intell.*, vol. PAMI-9, no. 1, Jan. 1987.

[10] Y. H. Hu and S. Y. Kung, "Toeplitz eigensystem solver," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, no. 4, pp. 1264–1271, Oct. 1985.

[11] H. Murakami and V. Kumar, "Efficient calculation of primary images from a set of images," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-4, no. 5, pp. 511–15, Sept. 1982.

[12] J. A. McLaughlin and J. Raviv, "Nth order autocorrelation in pattern recognition," *Inform. Contr.*, vol. 12, pp. 121–142, 1968.

[13] A. Rosenfeld and A. Kak, *Digital Picture Processing.* New York: Academic, 1976.

[14] S. Shlien, "A method for computing the partial singular value decomposition," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-4, no. 6, pp. 671–676, Nov. 1982.

[15] X. Yang, T. K. Sarkar, and E. Arvas, "A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 10, pp, 1550–55, Oct. 1989.

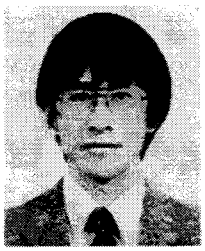
[16] W. H. Chen and H. Smith, "Adaptive coding of monochrome and color images," *IEEE Trans. Commun.*, vol. COM-25, pp. 1285–1292, 1977.

[17] T. Berger, *Rate Distortion Theory—A Mathematical Basis for Data Compression.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[18] A. N. Netravaly and B. G. Haskell, *Digital Pictures Representation and Compression.* New York: Plenum, 1988.

[19] H. Murase and M. Lindenbaum, "Spatial temporal adaptive method for partial eigenstructure decomposition of large images," NTT Tech. Rep. 6527, Mar. 1992.

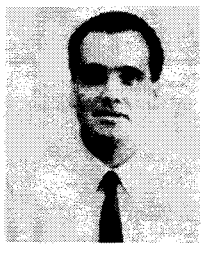
[20] W. H. Chen, H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004–1009, 1977.



Hiroshi Murase (M'87) received the B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Nagoya, Japan, in 1978, 1980, and 1987, respectively.

From 1980 to the present, he has been engaged in pattern recognition research at Nippon Telegraph and Telephone Corporation (NTT). From 1992 to 1993, he was a visiting research scientist at Columbia University, New York, NY, USA. His research interests include computer vision, character recognition, image recognition, and pattern analysis.

Dr. Murase was awarded an IECEJ Shinohara Award in 1986, a Telecom System Award in 1992, and the IEEE CVPR Best Paper Award in 1994. He is a member of the IEICEJ, IPSJ, and AVIRG.



Michael Lindenbaum was born in Israel in 1956. He received the B.Sc., M.Sc., and D.Sc. degrees from the Department of Electrical Engineering at the Technion, Israel, in 1978, 1987 and 1990 respectively.

From 1978 to 1985, he served in the IDF. He did his post-doctoral work at the NTT Basic Research Labs in Tokyo, Japan, and since October 1990, he has been with the Department of Computer Science at NTT. His main research interest is computer vision and especially object recognition.