ELSEVIER

Image and Vision Computing 20 (2002) 1009–1016

image AND vision COMPUTING

www.elsevier.com/locate/imavis

Adding and subtracting eigenspaces with eigenvalue decomposition and singular value decomposition

Peter Hall[a,*], David Marshall[b], Ralph Martin[b]

[a]*Department of Computer Science, School of Mathematical Science, Bath University, Bath, UK*
[b]*Department of Computer Science, Cardiff University, Cardiff, UK*

Abstract

This paper provides algorithms for adding and subtracting eigenspaces, thus allowing for incremental updating and downdating of data models. Importantly, and unlike previous work, we keep an accurate track of the mean of the data, which allows our methods to be used in classification applications. The result of adding eigenspaces, each made from a set of data, is an approximation to that which would obtain were the sets of data taken together. Subtracting eigenspaces yields a result approximating that which would obtain were a subset of data used. Using our algorithms, it is possible to perform 'arithmetic' on eigenspaces without reference to the original data. Eigenspaces can be constructed using either eigenvalue decomposition (EVD) or singular value decomposition (SVD). We provide addition operators for both methods, but subtraction for EVD only, arguing there is no closed-form solution for SVD. The methods and discussion surrounding SVD provide the principle novelty in this paper. We illustrate the use of our algorithms in three generic applications, including the dynamic construction of Gaussian mixture models.

Keywords: Singular value decomposition; Eigenvalue decomposition; Dynamic updating and downdating; Gaussian mixture models

1. Introduction

This subject of this paper is incremental eigenanalysis; we provide algorithms for including new data, and another for removing data. An eigenspace comprises: the number of data points, their mean, the set of support vectors through the data, and a measure of the spread of the data over each support vector. Eigenspaces can be computed using either eigenvalue decomposition (EVD) of the covariance matrix of the data, or singular value decomposition (SVD) of the data itself. In either case, the same set of support vectors is produced. The spread values in EVD are proportional to the variance of the data, while in SVD spread values are proportional to the standard deviation of the data. When SVD is used, an eigenspace also includes information about data points projected into the eigenspace; this is absent from EVD computations. This difference will be significant later in the paper. This paper, uniquely, discussed both SVD and EVD. The principal novelty is provided by the discussions surrounding SVD.

Typically an eigenspace is *deflated*, which is to say that only 'significant' support vectors and spread values are retained in the eigenspace. The inclusion of new data is sometimes called *updating*, while the removal of data is sometimes called *downdating*. Rather than use data directly, we use eigenspace representations of the data, hence we *add* or *subtract* eigenspaces. We must make clear the difference between *batch* and *incremental* methods for computing eigenspace models. A batch method computes an eigenspace using all observations simultaneously. An incremental method computes an eigenspace model by successively updating an earlier model as new observations become available. Our operators for addition and subtraction are presented in Section 2.

Incremental eigenanalysis has been studied previously [1–4,7,14]. Each considers either EVD or else SVD approaches. Only one [2] considers downdating, and in that case EVD is used to remove only one data point. These authors either have ignored the fact that a change in data changes the mean, or else have handled it in an ad hoc way. This is a surprising omission when we consider that important functions such as the Mahalanobis distance, often used in classification applications, cannot be computed without the mean.

Our previous work considered both update and downdate of EVD with many data points, and allowed for a change of mean in a principled way [10]. Here, we also consider addition and subtraction operators that act on many data at

* Corresponding author.
E-mail address: pmh@cs.bath.ac.uk (P. Hall).

0262-8856/02/$ - see front matter
PII: S0262-8856(02)00114-2

once, they are *block* methods. These operators explicitly keep track of the mean in a principled way. We shown how to block update both EVD and SVD. We provide block downdating for EVD, but argue that downdating of SVD is not possible in general.

Applications of incremental methods are wide ranging both within computer vision and beyond. Focusing on computer vision, applications include: face recognition [13], modelling variances in geometry [6], and the estimation of motion parameters [4]. Our motivations for this work arose from several sources, one example being the construction of classification models for many images—too many to store all into memory at once. Intuition, confirmed by experiment, suggests it is better to construct the eigenspace from all the images rather than a subset of them, which is all that could be done using any batch method; hence the need for an incremental method (see Section 3). An example is a database of photographs for a security application in which images need to be added and deleted each year, yet not all images can be stored in memory at once (see Section 3). Our methods allow the database to be updated and downdated without recomputing the eigenspace ab initio. We are also interested in constructing dynamic Gaussian mixture models (GMMs) that is being able to add and subtract GMMs. For this, the ability to keep track of the mean while adding (or subtracting) eigenspaces is essential. A full discussion of the issues involved is beyond the scope of the paper, and is the subject of future work, but we present initial results (see Section 3) because of the potential of dynamic GMMs. For example, the mixture model used by Cootes and Taylor [5] can be brought into a dynamic learning framework and since our GMMs rely on a hierarchy of subspaces, so too can work such as that of Heap and Hogg [11], or Karaulova et al. [12].

2. Adding and subtracting eigenspaces

Before stating the problems which are our subject, we should explain in greater detail what we mean by the term *eigenspace*, with reference to both SVD and EVD.

Let $X = [x_1, \ldots, x_N]$ be a collection of N data points, each n dimensional. The EVD of the covariance of the data is defined by $(X - \mu\mathbf{1})(X - \mu\mathbf{1})^{\mathrm{T}} = (1/N)U\Lambda U^{\mathrm{T}}$ where μ is the data mean, $\mathbf{1}$ is a row N 1's, U is a $n \times n$ matrix of eigenvectors (support vectors), and Λ is and $n \times n$ diagonal matrix of eigenvalues (spread values). The ith eigenvalue is the scalar variance of the data about the mean in the direction of the ith eigenvector (the ith column), under the assumption that the data are Gaussian distributed. U is, necessarily, orthonormal.

It is often assumed that only those eigenvectors that correspond to large spread values are of interest, the others are discarded by deleting columns from the matrix U. Typically the number of non-zero eigenvalues is $p \leq min(n, N)$, this is the rank of the covariance matrix of $X - \mu$. In practice, p is chosen to include small values, in addition to zero values (see Ref. [9] for a discussion). This deflation leaves p eigenvectors in a $n \times p$ matrix U_{np}, and p eigenvalues in a diagonal matrix Λ_{pp}. We call p the dimension of the eigenspace. We hve $(X - \mu)(X - \mu)^{\mathrm{T}} \approx U_{np}\Lambda_{pp}U_{np}^{\mathrm{T}}$, because of deflation (here Λ is a diagonal matrix). We also have $U_{np}^{\mathrm{T}}U_{np} = I$, but $U_{np}U_{np}^{\mathrm{T}} \neq I$. The eigenvectors support a subspace of dimension p embedded in a space of dimension n.

We specify an EVD eigenspace as

$$\Omega(X) = (\mu(X), U(X)_{np}, \Lambda(X)_p, N(X)) \tag{1}$$

in which $\mu(X)$ is the data mean; $U(X)_{np}$ is a collection of p column eigenvectors; $\Lambda(X)_p$ is a vector of p eigenvalues, and N is the number of data points. The subscripts on each element identify its size, where we deem it helpful. $\Omega(X)$ may be interpreted as representing a multidimensional Gaussian distribution over a hyperplane, of dimension p, in some embedding space, of dimension n. Contours of equal likelihood generate hyperellipses of dimension p.

Turning now to SVD. The SVD of the same data, X is $X - \mu\mathbf{1} = U\Sigma V^{\mathrm{T}}$ in which U is a matrix of left singular vectors (support vectors), Σ is a $n \times N$ matrix that is non-zero only on its leading diagonal, these are the singular values (spread values), and V is a matrix of right singular vectors, which contain information about the data projected into eigenspace. The ith singular value is the standard deviation of the data along the ith left singular vector. Both U and V are orthonormal. EVD and SVD are related for the left singular vectors and eigenvectors are identical. Also it is easy to show that $N\Sigma^2 = \Lambda$. We can therefore specify an SVD eigenspace as

$$\Theta(X) = (\mu(X), U(X)_{np}, \Sigma(X)_p, V(X)_{Np}, N(X)) \tag{2}$$

This may be given exactly the same interpretation as $\Omega(X)$, provided the relation between Λ and Σ is borne in mind. We note that $\Theta(X)$ has greater information content due to the presence of $V(X)$, which carries information about the point coordinates in the eigenspace $U(X)$.

We can now state the problems of interest. Suppose we have another collection of observations $Y = [y_1, \ldots, y_M]$. These have the EVD eigenspace $\Omega(Y) = (\mu(Y), U(Y)_{nq}, \Lambda(Y)_q, N(Y))$ and the SVD eigenspace $\Theta(Y) = (\mu(Y), U(Y)_{nq}, \Sigma(Y)_q, V(Y)_{Mq}, N(Y))$ This collection is usually distinct from X, but such distinction is not a requirement. Notice that q eigenvectors and eigenvalues are kept in this model, and in general $q \neq p$ even if $Y = X$: deflation may occur in different ways.

The problem for addition, using EVD, is to compute the eigenspace for the concatenated pair of collections $Z = [X, Y]$

$$\Omega(Z) = (\mu(Z), U(Z)_{nr}, \Lambda(Z)_r, N(Z)) = \Omega(X) \oplus \Omega(Y) \tag{3}$$

with reference to $\Omega(X)$ and $\Omega(X)$ only: that is, define the algorithm for our $\oplus$ operator. We assume the original data are not available. In general, the number of

eigenvectors and eigenvalues kept, r, differs from both p and q. This implies that addition must account for a possible change in dimension of the eigenspace. The problem for SVD is exactly analogous: to define

$$\Theta(Z) = \Theta(X) \oplus \Theta(Y) \tag{4}$$

We wish to perform addition in the most parsimonious way possible.

The problem for subtraction is to compute $\Omega(X)$

$$\Omega(X) = \Omega(Z) \ominus \Omega(Y) \tag{5}$$

which is to remove the observations in Y from the eigenspace in Z. As in the case of addition, a possible change in the dimension of the eigenspace must be accounted for. We will argue that subtraction is possible for EVD only: that is, the SVD subtraction $\Theta(X) = \Theta(Z) \ominus \Theta(Y)$ is not possible in closed form.

2.1. Addition

We present solutions to $\Omega(Z) = \Omega(X) \oplus \Omega(Y)$ and to $\Theta(X) = \Theta(X) \oplus \Theta(X)$. Derivations for EVD available elsewhere [10], derivations for SVD are analogous.

Incremental computation of $N(Z)$ and $\mu(Z)$ is straightforward:

$$N(Z) = N(X) + N(Y) \tag{6}$$

$$\mu(Z) = (N(X)\mu(X) + N(Y)\mu(Y))/N(Z) \tag{7}$$

This is the same in either case. The general approach to addition is very similar for EVD and SVD; in fact the following discussion suffices for both, except for one step.

Since $U(Z)$ must support all data in both collections, X and Y, both $U(X)$ and $U(Y)$ must be subspaces of $U(Z)$. Generally, we might expect that these subspaces 'intersect' in the sense that $U(X)^{\mathrm{T}}U(Y) \neq 0$. The null space of each of $U(X)$ and $U(Y)$ may contain some component of the other, that is $H = U(Y) - U(X)(U(X)^{\mathrm{T}}U(Y)) \neq 0$. Both of these conditions are illustrated in Fig. 1. Furthermore, even if $U(X)$ and $U(Y)$ support the same subspace, then $U(Z)$ could still be of larger dimension. This is because some component, h say, of the vector joining the means, $\mu(X) - \mu(Y)$ may be in the null space of both subspaces, simultaneously. For example, $\mu(X)$, $U(X)$ and $\mu(Y)$, $U(Y)$ define a pair of planes parallel to the xy-plane, but separated in the z direction, as in Fig. 1.

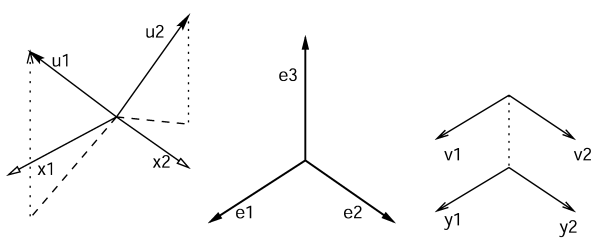

Fig. 1. An illustration of relationships between subspaces embedded in a larger space: intersecting subspaces (left), and parallel subspaces (right).

Momentarily putting to one side issues relating to changes in dimension, adding data acts to rotate the support vectors and scale the values relating to data spread. Hence, the new support vectors must be linear combination of the old. We deal with a change in dimension by constructing a basis sufficient span $U(Z)$, for which we use $U(X)$ augmented by v, v spans $[H, h]$, which is in the null space of $U(X)$. Note that v spans a t-dimensional subspace; $t \leq q + 1$. We have $U(Z) = [U(X), v]R$ where R is an orthonormal matrix. Addition for EVD and SVD diverge only in the manner in which R is computed.

For EVD, the following eigenproblem is solved

$$\begin{aligned}
&\frac{N(X)}{N(Z)}\begin{bmatrix} \Lambda(X)_{pp} & 0_{pt} \\ 0_{tp} & 0_{tt} \end{bmatrix} \\
&+ \frac{N(Y)}{N(Z)}\begin{bmatrix} G_{pq}\Lambda(Y)_{qq}G_{pq}^{\mathrm{T}} & G_{pq}\Lambda(Y)_{qq}\Gamma_{tq}^{\mathrm{T}} \\ \Gamma_{tq}\Lambda(Y)_{qq}G_{pq}^{\mathrm{T}} & \Gamma_{tq}\Lambda(Y)_{qq}\Gamma_{tq}^{\mathrm{T}} \end{bmatrix} \\
&+ \frac{N(X)N(Y)}{N(Z)^2}\begin{bmatrix} g_p g_p^{\mathrm{T}} & g_p \gamma_t^{\mathrm{T}} \\ \gamma_t g_p^{\mathrm{T}} & \gamma_t \gamma_t^{\mathrm{T}} \end{bmatrix} = R_{ss}\Pi_{ss}R_{ss}^{\mathrm{T}}
\end{aligned} \tag{8}$$

in which Π is diagonal and

$$g_p = U(X)^{\mathrm{T}}(\mu(X) - \mu(Y)) \tag{9}$$

$$G_{pq} = U(X)^{\mathrm{T}}U(Y) \tag{10}$$

$$H_{nq} = [U(Y) - U(X)G_{pq}] \tag{11}$$

$$h_n = (\mu(X) - \mu(Y)) - U(X)g_p \tag{12}$$

$$v_{nt} = \mathrm{Orthobasis}(\zeta[H_{nq}, h_n]) \tag{13}$$

$$\Gamma_{tq} = v_{nt}^{\mathrm{T}}U(Y)_{nq} \tag{14}$$

$$\gamma_t = v^{\mathrm{T}}(\mu(X) - \mu(Y)) \tag{15}$$

ζ is an operation that removes very small column vectors from the matrix, and Orthobasis computes a set of mutually orthogonal, unit vectors that support its argument; typically Gramm–Schmidt orthogonalization [8] is used to compute significant support vectors, v from $\zeta[H,h]$; these are 'outside' the eigenspace $\Omega(X)$. Note that while $v^{\mathrm{T}}v = I$, $vv^{\mathrm{T}} \neq I$. Also, G is the projection of the $\Omega(Y)$ eigenspace onto $\Omega(X)$ (the U vectors), while Γ is the projection of $\Omega(Y)$ onto the complementary space to $\Omega(X)$ (the v vectors). This complementary space must be determined to compute the new eigenspace $\Omega(Z)$, which argues in favour of adding and subtracting eigenspaces, rather than direct updating or downdating of data blocks.

Each matrix in the above eigendecomposition is of size $s = p + t \leq p + q + 1 \leq \min(n, M + N)$. Thus, we have eliminated the need for the original covariance matrices. Note this also reduces the size of the central matrix on the left hand side of Eq. (8). This is of crucial computational importance because it makes the eigenproblem tractable for problems in which n is very large, such as when each datum is an image.

When adding SVD models we must compute R using

$$R\Sigma V^{\mathrm{T}} = \begin{bmatrix} \Sigma(X)_{pp} V(X)^{\mathrm{T}}_{Np} & G_{pq}\Sigma(Y)_{qq} V(Y)^{\mathrm{T}}_{Mq} \\ 0_{tp} & (v^{\mathrm{T}}_{nt} U(Y)_{nq})\Sigma(Y)_{qq} V(Y)^{\mathrm{T}}_{Mq} \end{bmatrix} + \begin{bmatrix} U(X)^{\mathrm{T}}_{np}(\mu(X)-\mu(Z))\mathbf{1}_{N(X)} & U(X)^{\mathrm{T}}_{np}(\mu(Y)-\mu(Z))\mathbf{1}_{N(Y)} \\ v^{\mathrm{T}}_{nt}(\mu(X)-\mu(Z))\mathbf{1}_{N(X)} & v^{\mathrm{T}}_{nt}(\mu(Y)-\mu(Z))\mathbf{1}_{N(Y)} \end{bmatrix} \tag{16}$$

which is an $s\times(N+M)$ problem. This is the smallest sized problem, because it occupies the smallest dimension subspace possible. The number of columns cannot be reduced, because SVD explicitly maintains information about each data point, and $VV^{\mathrm{T}}\neq I$.

Each of the above decompositions directly yields the eigenvalues or singular values. The SVD expression also directly yields the required right singular values. In either case, the new support vectors must be found by rotation: $U(Z) = [U(X)v]R$. The model can then be deflated, if desired, to dimension $r \leq s$.

2.2. Subtraction

The algorithm for subtraction using EVD is very similar to that for addition. First compute the number of data, and their mean:

$$N(X) = N(Z) - N(Y) \tag{17}$$

$$\mu(X) = (N(Z)\mu(Z) - N(Y)\mu(Y))/N(X) \tag{18}$$

In this case, $U(Z)$ is a sufficient spanning set to rotate. To compute the rotation, we use the eigendecomposition:

$$\frac{N(Z)}{N(X)}\Lambda(Z)_{rr} - \frac{N(Y)}{N(X)}G_{rp}\Lambda(Y)_{pp}G^{\mathrm{T}}_{rp} - \frac{N(Y)}{N(Z)}g_r g_r^{\mathrm{T}} = R_{rr}\Lambda(X)_{rr}R^{\mathrm{T}}_{rr} \tag{19}$$

where $G_{rp}=U(Z)^{\mathrm{T}}_{nr}U(X)_{nq}$ and $g_r=U(Z)^{\mathrm{T}}_{nr}(\mu Y-\mu X)$. The eigenvalues we seek are the p non-zero elements on the diagonal of $\Lambda(X)_{rr}$. Thus, we can permute R_{rr} and $\Lambda(X)_{rr}$, and write without loss of generality:

$$R_{rr}\Lambda(X)_{rr}R^{\mathrm{T}}_{rr} = [R_{rp}R_{rt}]\begin{bmatrix}\Lambda(X)_{pp}0_{pt} \\ 0_{tp} \quad 0_{tt}\end{bmatrix}[R_{rp}R_{rt}]^{\mathrm{T}} = R_{rp}\Lambda(X)_{pp}R^{\mathrm{T}}_{rp} \tag{20}$$

where $p=r-q$. Hence, we need only identify the eigenvectors in R_{rr} with non-zero eigenvalues, and compute the $U(X)_{np}$ as:

$$U(X)_{np}=U(Z)_{nr}R_{rp} \tag{21}$$

Splitting must always involve the solution an eigenproblem of size r.

We now argue that subtraction is not possible for SVD models. Difficulties arise in two areas, even if we neglect a change in mean. The first difficulty comes from the (simplest) form of the problem which is $[ABC^{\mathrm{T}}, DEF^{\mathrm{T}}] = GHJ^{\mathrm{T}}$, where $X = ABC^{\mathrm{T}}$, $Y = DEF^{\mathrm{T}}$, and $Z = GHJ^{\mathrm{T}}$. We must obtain A, B, and C. By computing the inner product of both sides we obtain $AB^2A^{\mathrm{T}} + DE^2D^{\mathrm{T}} = GH^2G^{\mathrm{T}}$, which gives us an EVD problem from which we can compute A and B: we cannot produce A or B directly using SVD.

The second difficulty arises in computing C, when we note that the ordering of right singular vectors depends upon the ordering of data points in the matrix being decomposed. The left singular vectors and singular values are invariant to permutation of the data. To see this, we suppose P is a permutation matrix (obtained by randomly permuting rows or columns of the identity matrix, so that $PP^{\mathrm{T}} = P^{\mathrm{T}}P = I$), and note that given $Z = GHJ^{\mathrm{T}}$, then $ZP = GHJ^{\mathrm{T}}P = GH(P^{\mathrm{T}}J)^{\mathrm{T}}$. Therefore, in order to compute the right singular vectors, C, while downdating, we must have access to some matrix P which 'picks out' data elements in Z (or, equivalently, corresponding elements in J). Unfortunately no such information exists within the SVD model, and consequently computing C in a closed-form manner seems impossible. The only solution is to resort to search using data elements in J and F (for these specify data points in Z and Y, respectively). If search is the only solution, then we may simply downdate Z by building up X incrementally as elements in $Z\backslash Y$ are found, which is unsatisfactory in our opinion.

3. Properties of operators, and some applications

It can be shown [10] that the addition of exactly one new datum is a special case of the above addition, with $\Omega = (x, 0, 0, 1)$ or $\Theta = (x, 0, 0, 0, 1)$. In terms of its outcome, the addition of EVD eigenspaces is both commutative and associative (provided that in practice we allow for numerical errors, especially in association). SVD eigenspaces also commutes, up to a permutation of the right singular vectors. The null eigenspace is an additive identity. The addition of an eigenspace to itself yields an eigenspace, which is identical in all respects except the number of points (which doubles). As $N(X) \rightarrow \infty$ so the effect of addition becomes negligible. As both $N(X)$ and $N(Y)$ tend to infinity together, so the result tends to a stable state.

The time complexity for addition will shadow that used in computing the particular decomposition. Our experiments [10] demonstrate that the time taken is $O(s^3)$, where s is the size of the eigenproblem to be solved (we used a proprietary eigensolver). We also found that the time to compute the two eigenspaces ab initio and add them is about that of computing a large eigenspace using all the original data. However, it is much more efficient to add a pair of existing eigenspaces than to compute their sum ab initio. Similar remarks apply to splitting: removing a few data points is a comparatively efficient operation. The conclusion we reach is that addition and subtraction of eigenspaces is

no less efficient than batch methods, and in most cases is performed much more efficiently. Memory complexity is optimal for both EVD and SVD.

We measured accuracy of addition by adding a pair of eigenspaces and comparing the result with an eigenspace computed by concatenating data matrices. All data were Gaussian distributed. To compare models, we computed the Euclidean distances between origins and eigenvectors values (or singular value vectors). To measure the deviation of support vectors, U_{add} and U_{concat}, say, we used $|U_{\text{concat}}^{\text{T}} U_{\text{add}}| - 1$. The data we used was drawn from a Gaussian distribution. In both EVD and SVD, the accuracy in all measures usually was about 10^{-14} units. (No particular unit was used in the experiment, each datum being a random variable.) However, when adding a model with many eigenvectors to one with few, using EVD, the error in values and vectors peak sharply at about the number of vectors in the smaller model, but remained less than 10^{-11} units. The reason for the peak seems related to numerical instabilities in computing v, the basis in the null space of the smaller model.

The subtraction operator tends to instability as the number of points being removed rises, since in this case $N(X) \rightarrow \infty$, hence $1/N(X) \rightarrow \infty$. In the limit of all points being removed $N(X) = 0$, and an exception must be coded to return a null eigenspace. Unfortunately, we have found that prior scaling by $N(X)$ to be ineffective and have concluded that, in practice, subtraction is best used to remove a small fraction of the data points.

An obvious application of our methods is to build an eigenspace from many images—too many to all at once store into memory. We ran a simulation of this by building two eigenspaces: one using batch methods and another using our incremental methods. We were then able to compare the two models. The eigenspaces themselves turn out to be very similar, although differences between batch and incremental eigenspaces are greater in cases where eigenspaces are subtracted. Performance results bear out intuition: those images used to make the eigenspace had a much lower residue error than those not so used. As more images were added into the construction the maximum residue error for each image rose—but never so high as to reach the minimum residue error for images not used in eigenspace construction. Classification results follow a similar trend: each image is better classified by an eigenspace that uses all images.

We now present two more substantial applications of our methods. These are of a generic nature. The intention is to furnish the reader with a practically useful appreciation of the characteristics of our methods, and avoid the specific problems of any particular application.

3.1. Building an accurate eigenspace model

Here, we consider an image database application. The scenario is that of a university wishing to efficiently store photographs of its thousands of students for use in a security application of some kind, such as access to a laboratory. The students are to be identified from their facial appearance. Face recognition is well researched, and we do not claim to make a contribution, rather we aim to show how our methods might be used in a support role. In particular, we consider the case in which the database of images changes, as old students leave and new ones arrive.

We proceed in a very simple way: we construct an eigenspace of all those people who are to be recognized, and rely on the fact that eigenspaces do not generalize well to distinguish between those people in the set, and those not in the set. To allow for changes in pose, expression, and so on, we use several images of each individual.

Conventional batch methods cannot be used to construct an eigenspace because there are too many images to store into memory at once, so incremental methods are a prerequisite to our approach. Given that the database is subject to change we could reconstruct an eigenspace at each change, but we will use our incremental methods to effect the changes more efficiently; for which subtraction is required.

We used the Olivetti database of 400 faces[1] as our group of students. We constructed an eigenspace from a selection of 21 people, there being 10 photographs for each person. Each person in the entire database was then given a 'weight of evidence' between 0 (not in the database) and 1 (in the database). To compute the weight, we computed the maximum Mahalanobis distance (using Moghaddam and Pentland's method [13]) of any photograph used in constructing the database. Each photograph was then classified as 'in' if its Mahalanobis distance was less than this. Since each person has 10 photographs associated with them, we can then compute a weight for each person as the fraction of their photographs classified as in Fig. 2 shows the weight of evidence measure for the *second* year our hypothesized database has been running. The scenario is that in year one, only persons 0–21 inclusive were in, while in year two only persons 1–22 inclusive were in. The leftmost plot shows the measure for the images against a batch model. That on the right shows the same measure for the same images, but for a model incrementally computed from year one by first including any new students (person 22), and then removing old students (person 0). (The ordering used to make sure the fraction of images removed was minimized.)

We notice that both models produce some ambiguous cases, with weights between 0 and 1, and that the incrementally computed eigenspace gives rise to more of these cases than the eigenspace computed via batch methods. This result is in line with our earlier comments regarding the relative inaccuracy of subtraction. Even so, only those people in the database score 1, while everyone outside scored less than 1 and hence classification is still possible.

[1] http://www.cam-orl.co.uk/facedatabase.html.

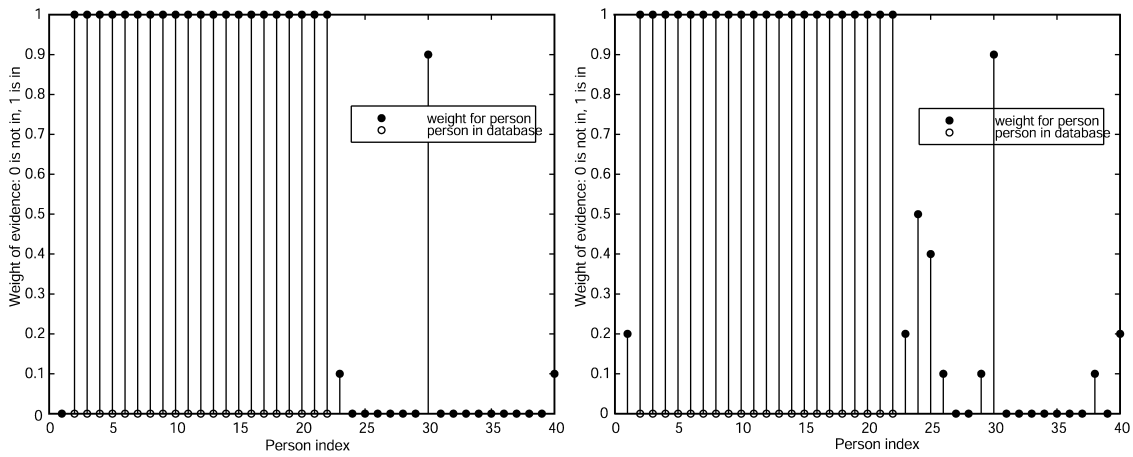

Fig. 2. Weight of evidence measures: year 2 batch (left), and year 2 incremental (right).

Given our observations, above, regarding previous measures when subtracting eigenspaces, we conclude that *additive* incremental eigenanalysis is safe for classification metrics, but that *subtractive* incremental eigenanalysis needs a greater degree of caution.

3.2. Dynamic Gaussian mixture models

We are interested in using our methods to construct dynamic GMMs. GMMs are useful n many computer vision contexts [5]. Our approach treats a GMM as a hierarchy of eigenspaces, which is a mechanism for improving the specificity of the data description [11,12]. To construct a hierarchy we first make an eigenspace, then project all data into it to reduce dimensionality, next construct a GMM using the projected data, and then represent each mixture as an eigenspace. Thus, each Gaussian in the mixture can be thought of as a hyperellipse, and each may have a different dimensions. The problem here is to merge two such GMMs.

As an example, we used photographs of two distinct toys, each photographed at 5° angles on a turntable. Hence we had 144 photographs. Examples of these photographs can be seen in Fig. 3. The photographs for each toy were input separately, and a hierarchy of eigenspaces constructed as described earlier; we used eighteen Gaussians in each mixture model, on the grounds that this would very probably produce too many Gaussians—a number which is later improve by merging.

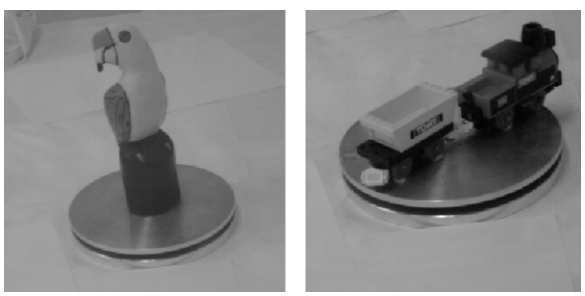

Fig. 3. Sample images of each toy used as source data in our dynamic GMM application.

Thus including the top-level eigenspace, each set of toy photographs was represented with nineteen eigenspace models. To merge the GMMs for the pair of toys we first added together the two top-most eigenspaces to make a complete eigenspace for all 144 photographs. Next, we transformed each of the GMM clusters into this space, thus bringing each of the 36 GMMs (18 from each individual hierarchy) into the same (large) eigenspace covering the *ensemble* of data. We then merged eigenspaces (Gaussian components), using a very simple criterion to merge based on reducing volume of hyperellipses, which is explained below. Hence, we were able to reduce the total number of Gaussians to 22 in the mixture. These clusters tend to model different parts of the cylindrical trajectories of the original data projected into the large eigenspace. Examples of cluster centres are shown in Fig. 4: the two models can be clearly seen in different positions. In addition, we found a few clusters occupying the space 'in between' the two toys—an example of which is seen in Fig. 4.

As mentioned earlier, we used a simple method based on volume to decide whether two eigenspaces should be merged. The procedure was as follows. First compute the volume of each of the eigenspaces, using the hyperellipse at one Mahalanobis distance. The volume of a hyperellipse with semi-axes $\mathbf{A}$ each element the square root of an eigenvalue), of dimension M, and at characteristic radius s (square root of the Mahalanobis distance) is

$$\frac{s^M |\mathbf{A}| \pi^{M/2}}{\Gamma\left(\frac{M}{2}+1\right)} \tag{22}$$

with $\Gamma(\cdot)$ the gamma function. We permanently merged a pair of eigenspaces in the GMM if the sum of their individual volumes was greater than their volume when merged. This measure suffers from problems of dimension: we should not

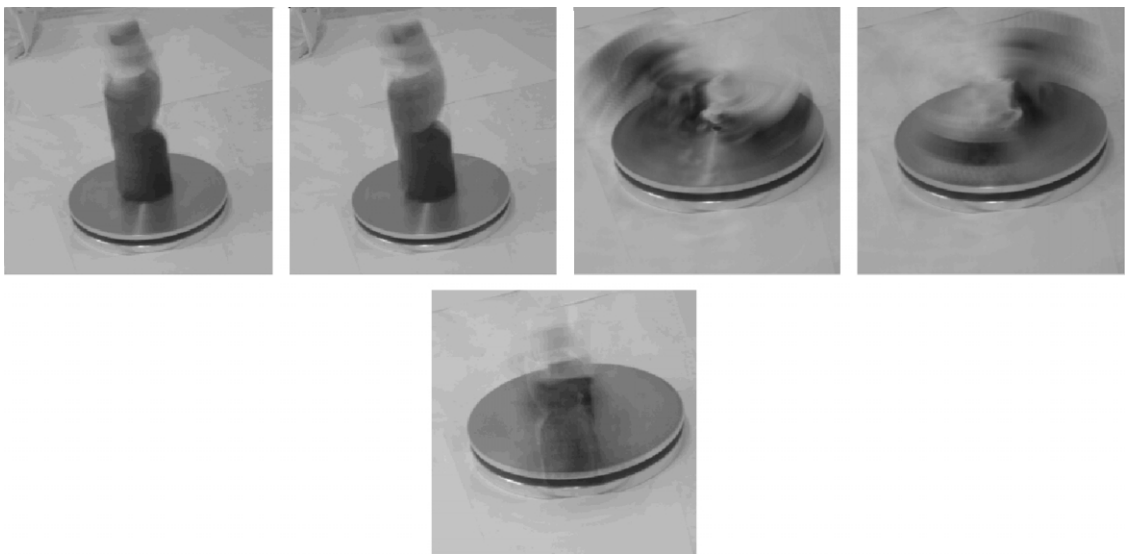

Fig. 4. Dynamic GMMs, showing 5 examples of the 22 cluster centres. These are arranged to show clusters for each toy (top row), and the clusters between them (bottom).

compare the volume of a p-dimensional hyperellipse with that of a q-dimensional hyperellipse. A solution is to use a characteristic length in place of volume, which for a p-dimensional hyperellipse of volume v is $v^{1/p}$.

Of course, the utility and properties of the final GMM is fully in line with any produced by conventional means, and hence can be used in any application that a conventional GMM is used. We conclude from these experiments that dynamic GMMs are a feasible proposition using our methods.

4. Conclusion

We have presented methods for adding and subtracting eigenspaces. We have discussed the form of our solutions, and shown that previous work is a special case of this work. Our contribution is to track the mean in a principled way, which makes our contribution novel. This is essential in classification applications, which makes our contribution important. This paper is unique in discussing block methods for both EVD and SVD.

Having experimentally compared eigenspaces, considered performance metrics of our algorithms, and having experimented with several more applications we have concluded that the addition of eigenspaces is stable and reliable. We advise that our methods be used carefully. Special are should be taken when subtracting eigenspaces: the way in which the results are to be used impacts on efficacy.

We should point out several omissions from this work. We have not performed any rigorous error analysis and hence any explanations we have for the behaviour of our algorithms are anecdotal in character. We have not fully worked through any particular application, and so can make general recommendations only. The important conclusion from that work is that updating the mean is crucial for classification results [9].

We would expect our methods to find much wider applicability than those we have already mentioned in this paper: updating image motion parameters [4], and selecting salient views [3] are two applications that exist already for incremental methods. We have experimented with image segmentation, building models of three-dimensional blood vessels, and texture classification. We believe that dynamic GMMs provide a very interesting future path for they enable useful representations [5,11]—and all their attendant properties—to be brought into a dynamic framework.

References

[1] J.R. Bunch, C.P. Nielsen, Updating the singular value decomposition, Numerische Mathematik 31 (1978) 111–129.
[2] J.R. Bunch, C.P. Nielsen, D.C. Sorenson, Rank-one modification of the symmetric eigenproblem, Numerische Mathematik 31 (1978) 31–48.
[3] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkler, H. Zhang, An eigenspace update algorithm for image analysis, Graphical Models and Image Processing 59 (5) (1997) 321–332. September.
[4] S. Chaudhuri, S. Sharma, S. Chatterjee, Recursive estimation of motion parameters, Computer Vision and Image Understanding 64 (3) (1996) 434–442. November.
[5] T.F. Cootes, C.J. Taylor, A mixture model for representing shape variations, Proceedings of British Machine Vision Conference (1997) 110–119.
[6] T.F. Cootes, C.J. Taylor, D.H. Cooper, J. Graham, Training models of shape from sets of examples, Proceedings of British Machine Vision Conference (1992) 9–18.
[7] R.D. DeGroat, R. Roberts, Efficient, numerically stabilized rank-one eigenstructure updating, IEEE Transactions on Acoustics, Speech, and Signal Processing 38 (2) (1990) 301–316. February.
[8] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins, Baltimore, MD, 1983.

[9] P. Hall, A.D. Marshall, R. Martin, Incrementally computing eigenspace models, Proceedings of British Machine Vision Conference, Southampton (1998) 286–295.
[10] P. Hall, A.D. Marshall, R. Martin, Merging and splitting eigenspaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (9) (1998) 1042–1049. September.
[11] T. Heap, D. Hogg, Improving specificity in PDMs using a hierarchical approach, Proceedings of British Machine Conference (1997) 80–89.
[12] J. Karaulova, P.M. Hall, A.D. Marshall, A hierarchical model for tracking people with a single video camera, British Machine Vision Conference (2000) 352–361.
[13] B. Moghaddam, A. Pentland, Probabilistic visual learning for object representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (7) (1997) 696–710. July.
[14] H. Murakami, B.V.K. Kumar, Efficient calculation of primary images from a set of images, IEEE Pattern Analysis and Machine Intelligence 4 (1982) 511–515. September.