Merging and splitting eigenspace models

Peter Hall, David Marshall, Ralph Martin

***Abstract*— We present new deterministic methods that given two eigenspace models, each representing a set of n-dimensional observations will: (1) merge the models to yield a representation of the union of the sets; (2) split one model from another to represent the difference between the sets; as this is done, we accurately keep track of the mean.**

These methods are more efficient than computing new eigenspace models directly from the observations when the eigenmodels are dimensionally small compared to the total number of observations.

Such methods are important because they provide a basis for novel techniques in machine learning, using a dynamic split-and-merge paradigm to optimally cluster observations.

Here we present a theoretical derivation of the methods, empirical results relating to the efficiency and accuracy of the techniques, and three general applications, including the on-line construction of Gaussian mixture models.

Keywords: Eigenspace models, principal component analysis, model merging, model splitting, merge-and-split.

I. Introduction

The contributions of this paper are: (1) a method for merging eigenspace models; (2) a method for splitting eigenspace models; in both of which we explicitly and accurately *keep track of the mean* of the observations. Methods for merging (updating) or splitting (downdating) eigenspace models exist [1], [2], [3], [4], [5] but they generally fail to handle a change in the mean adequately. The methods we provide do update the mean properly, which is of crucial importance in classification problems — in such problems the mean represents the centre of a cluster of observations in a given class. Such classification problems were our original reason for investigating eigenspace model updating.

Eigenspace models have a wide variety of applications, for example: classification for recognition systems [6], characterising normal modes of vibration for dynamic models, such as the heart [7], motion sequence analysis [8], and the temporal tracking of signals [4]. Our motivation for this work arose in the context of building models of blood vessels for x-ray interpretation [9], and building eigenspace models for many images [10]. As an example, an image database of employees may require frequent changes to its records: our methods permit both the addition and deletion of new images, without the need to recompute the eigenspace model *ab initio*.

We would also like to incrementally build Gaussian mixture models [11], [12], which use separate Gaussian distributions to describe data falling into several clusters or classes. Updating the means is a prerequisite in this case, as the mean represents the centre of the distribution for each class; classification is based on the Mahalanobis distance, which measures the distance from the mean in units of standard deviation. Building such models dynamically is, perhaps, the most significant application of our methods. (Currently the EM Algorithm [13] is used for building such models.)

This paper is primarily concerned with deriving a new theoretical framework for merging and splitting eigenspaces, and an empirical evaluation of these new techniques, rather than their particular application in any area. However, we demonstrate our methods in three ways: building a database from many images; a security application; and the dynamic construction of Gaussian-mixture models.

All authors are with the Department of Computer Science, University of Wales, Cardiff, PO Box 916, Cardiff CF2 3XF, Wales UK: peter@cs.cf.ac.uk

An eigenspace model is a statistical description of a set of N observations in n-dimensional space; such a model may be regarded as a multi-dimensional Gaussian distribution. From a geometric point of view, an eigenspace model can be thought of as a hyperellipsoid that characterises a set of observations: its centre is the mean of the observations; its axes point in directions along which the spread of observations is maximised, subject to them being orthogonal; the surface of the hyperellipsoid is a contour that lies at one standard deviation from the mean. Often, the hyperellipsoid is almost flat along certain directions, and thus can be modelled as having lower dimension than the space in which it is embedded.

Eigenspace models are computed using either eigenvalue decomposition (EVD) (also called principal component analysis) or singular-value decomposition (SVD). We wish to distinguish between *batch* and *incremental* computation. In batch computation all observations are used simultaneously to compute the eigenspace model. In an incremental computation, an existing eigenspace model is updated using new observations.

Previous research in incremental computation of eigenspace models has only considered adding *exactly one* new observation at a time to an eigenspace model [1], [2], [3], [4], [5], [14]. A common theme of these methods is that none require the original observations to be retained. Rather, a description of the hyperellipsoid is sufficient information for incremental computation of the new eigenspace model. Each of these previous approaches allows for a change in dimensionality of the hyperellipsoid, so that a single additional axis is added if necessary. Only our previous work allows for a shift of the centre of the hyperellipsoid [14], other methods keep it fixed at the origin. This proves crucial if the eigenspace model is to be used for classification, as demonstrated in [14]: a set of observations whose mean is far from the origin is clearly not well modelled by a hyperellipsoid centred at the origin.

When using incremental methods previous observations need not be kept — thus reducing storage requirements and making large problems computationally feasible. Incremental methods *must* be used if not all observations are available simultaneously. For example, a computer may lack the memory resources required to store all observations. This is true even if *low-dimensional* methods are used to compute the eigenspace [5], [15]. (We will mention low-dimensional methods later, in Section II-B, but they give an advantage when the number of observations is less than the dimensionality of the space, $N < n$, which is often true when observations are images.) Even if all observations are available, it is usually faster to compute a new eigenspace model by incrementally updating an existing one rather than by using batch computation [3]. This is because the incremental methods typically compute p eigenvectors, with $p \leq \min(n, N)$. The disadvantage of incremental methods is their accuracy compared to batch methods. When only a few incremental updates are made the inaccuracy is small, and is probably acceptable for the great majority of applications [14]. When many thousands of updates are made, as when eigenspace models are incremented with a single observation at a time, the inaccuracies build up, although methods exist to circumvent this problem [4]. In contrast, our methods allow a whole new *set* of observations to be added in a single step, thus reducing the total number of updates to an existing model.

Section II defines eigenspace models in detail, standard methods for computing them, and how they are used for representing and classifying observations. Section III discusses merging of eigenspace models, while Section IV addresses splitting. Section V presents empirical results, and Section VI presents some applications of the work. Section VII gives our conclusions.

II. Eigenspace models

In this section, we describe what we mean by *eigenspace models*, briefly discuss standard methods for their batch computation, and how observations can be represented using them. Firstly, we establish our notation for the rest of the paper.

Vectors are columns, and denoted by a single underline. Matrices are denoted by a double underline. The size of a vector, or matrix, is often important, and where we wish to emphasise this size, it is denoted by subscripts. Particular column vectors within a matrix are denoted by a superscript, and a superscript on a vector denotes a particular observation from a set of observations, so we treat observations as column vectors of a matrix. As an example, $\underline{\underline{A}}^i_{mn}$ is the ith column vector in an $(m \times n)$ matrix. We denote matrices formed by concatenation using square brackets. Thus $[\underline{\underline{A}}_{mn} \underline{b}]$ is an $(m \times (n+1))$ matrix, with vector $\underline{b}$ appended to $\underline{\underline{A}}_{mn}$ as a last column.

A. Theoretical background

Consider N observations, each a column vector $\underline{x}^i \in \Re^n$. We compute an eigenspace model as follows:

The mean of the observations is

$$\underline{\bar{x}} = \frac{1}{N}\sum_{i=1}^{N} \underline{x}^i \tag{1}$$

and their covariance is

$$\begin{aligned} \underline{\underline{C}}_{nn} &= \frac{1}{N}\sum_{i=1}^{N}(\underline{x}^i - \underline{\bar{x}})(\underline{x}^i - \underline{\bar{x}})^T \\ &= \left(\frac{1}{N}\sum_{i=1}^{N}\underline{x}^i(\underline{x}^i)^T\right) - \underline{\bar{x}}\,\underline{\bar{x}}^T \end{aligned} \tag{2}$$

Note that $\underline{\underline{C}}_{nn}$ is real and symmetric.

The axes of the hyperellipsoid, and the spread of observations over each axis are the eigenvectors and eigenvalues of the eigenproblem

$$\underline{\underline{C}}_{nn}\underline{\underline{U}}_{nn} = \underline{\underline{U}}_{nn}\underline{\underline{\Lambda}}_{nn} \tag{3}$$

or, equivalently, the eigenvalue decomposition of $\underline{\underline{C}}_{nn}$ is

$$\underline{\underline{C}}_{nn} = \underline{\underline{U}}_{nn}\underline{\underline{\Lambda}}_{nn}\underline{\underline{U}}^T_{nn} \tag{4}$$

where the columns of $\underline{\underline{U}}_{nn}$ are eigenvectors, and $\underline{\underline{\Lambda}}_{nn}$ is a diagonal matrix of eigenvalues. The eigenvectors are orthonormal, so that $\underline{\underline{U}}^T_{nn}\underline{\underline{U}}_{nn} = \underline{\underline{I}}_{nn}$, the $(n \times n)$ identity matrix.

The ith eigenvector $\underline{U}^i$ and ith eigenvalue $\underline{\underline{\Lambda}}^{ii}_{nn}$ are associated; the eigenvalue is the length of the eigenvector, which is the ith axis of the hyperellipsoid. Typically, only $p \leq \min(n, N)$ of the eigenvectors have significant eigenvalues, and hence only p of the n eigenvectors need be retained. This is because the observations are correlated so that the covariance matrix is, to a good approximation, rank-degenerate: small eigenvalues are presumed to be negligible. Thus an eigenspace model often spans a p-dimensional subspace of the n-dimensional space in which it is embedded.

Different criteria for discarding eigenvectors and eigenvalues exist, and these suit different applications and different methods of computation. Three common methods are: (1) stipulate p as a fixed integer, and so keep the p largest eigenvectors [5]; (2) keep those p eigenvectors whose size is larger than an absolute threshold [3]; (3) keep the p eigenvectors such that a specified fraction of energy in the eigenspectrum (computed as the sum of eigenvalues) is retained.

Having chosen to discard certain eigenvectors and eigenvalues, we can recast Equation 4 using block form matrices and vectors. Without loss of generality, we can permute the eigenvectors and eigenvalues such that $\underline{\underline{U}}_{np}$ are those eigenvectors that are kept, and $\underline{\underline{\Lambda}}_{pp}$ their eigenvalues. If $d = n - p$, then $\underline{\underline{U}}_{nd}$ and $\underline{\underline{\Lambda}}_{dd}$ are those discarded. We may rewrite Equation 4 as:

$$\begin{aligned} \underline{\underline{C}}_{nn} &= \underline{\underline{U}}_{nn}\underline{\underline{\Lambda}}_{nn}\underline{\underline{U}}^T_{nn} \\ &= [\underline{\underline{U}}_{np}\underline{\underline{U}}_{nd}]\begin{bmatrix} \underline{\underline{\Lambda}}_{pp} & \underline{\underline{0}}_{pd} \\ \underline{\underline{0}}_{dp} & \underline{\underline{\Lambda}}_{dd} \end{bmatrix}[\underline{\underline{U}}_{np}\underline{\underline{U}}_{nd}]^T \\ &= \underline{\underline{U}}_{np}\underline{\underline{\Lambda}}_{pp}\underline{\underline{U}}^T_{np} + \underline{\underline{U}}_{nd}\underline{\underline{\Lambda}}_{dd}\underline{\underline{U}}^T_{nd} \end{aligned} \tag{5}$$

Hence

$$\underline{\underline{C}}_{nn} \approx \underline{\underline{U}}_{np}\underline{\underline{\Lambda}}_{pp}\underline{\underline{U}}^T_{np} \tag{6}$$

with error $\underline{\underline{U}}_{nd}\underline{\underline{\Lambda}}_{dd}\underline{\underline{U}}^T_{nd}$, which is small if $\underline{\underline{\Lambda}}_{dd} \approx \underline{\underline{0}}_{dd}$.

Thus, we define an *eigenspace model*, Ω, as the mean, a (reduced) set of eigenvectors, their eigenvalues, and the number of observations:

$$\Omega = (\underline{\bar{x}}, \underline{\underline{U}}_{np}, \underline{\underline{\Lambda}}_{pp}, N) \tag{7}$$

B. Low-dimensional computation of eigenspace models

Low-dimensional batch methods are often used to compute eigenspace models, and are especially important when the dimensionality of the observations is very large compared to their number. Thus, they may be used to compute eigenspace models that would otherwise be infeasible. Incremental methods also use a low dimensional approach.

In principle, computing an eigenspace model requires that we construct an $(n \times n)$ matrix, where n is the dimension of each observation. In practice, the model can be computed by using an $(N \times N)$ matrix, where N is the number of observations. This is an advantage in applications like image processing where, typically, $N \ll n$.

We show how this can be done by first considering the relationship between eigenvalue decomposition and singular value decomposition. This leads to a simple derivation for a low-dimensional batch method for computing the eigenspace model. The same results were obtained, at greater length, by [5], see also [15].

Let $\underline{\underline{Y}}_{nN}$ be the set of observations shifted to the mean, so that $\underline{\underline{Y}}^i = \underline{x}^i - \underline{\bar{x}}$. Then a SVD of $\underline{\underline{Y}}_{nN}$ is:

$$\underline{\underline{Y}}_{nN} = \underline{\underline{U}}_{nn}\underline{\underline{\Sigma}}_{nN}\underline{\underline{V}}^T_{NN} \tag{8}$$

where $\underline{\underline{U}}_{nn}$ are the left singular vectors, which are identical to the eigenvectors previously given; $\underline{\underline{\Sigma}}_{nN}$ is a matrix with singular values on its leading diagonal, with $\underline{\underline{\Lambda}}_{nn} = \underline{\underline{\Sigma}}_{nN}\underline{\underline{\Sigma}}^T_{nN}/N$; and $\underline{\underline{V}}_{NN}$ are right singular vectors. Both $\underline{\underline{U}}_{nn}$ and $\underline{\underline{V}}_{NN}$ are orthonormal matrices.

This can now be used to compute eigenspace models in a low-dimensional way, as follows:

$$\begin{aligned} \underline{\underline{Y}}^T_{nN}\underline{\underline{Y}}_{nN} &= \underline{\underline{V}}_{NN}\underline{\underline{\Sigma}}^T_{nN}\underline{\underline{\Sigma}}_{nN}\underline{\underline{V}}^T_{NN} \\ &= \underline{\underline{V}}_{NN}\underline{\underline{S}}_{NN}\underline{\underline{V}}^T_{NN} \end{aligned} \tag{9}$$

is an $(N \times N)$ eigenproblem. $\underline{\underline{S}}_{NN}$ is the same as $\underline{\underline{\Lambda}}_{nn}/N$, except for the presence of extra trailing zeros on the main diagonal of $\underline{\underline{\Lambda}}_{nn}$. If we discard the small singular values, and their singular vectors, following the above, then remaining eigenvectors vectors are

$$\underline{\underline{U}}_{np} = \underline{\underline{Y}}_{nN}\underline{\underline{V}}_{Np}\underline{\underline{\Sigma}}^{-1}_{pp}. \tag{10}$$

This result formed the basis of the incremental technique developed by Murakami and Kumar [5] but they did not allow for a change in origin, nor does their approach readily generalise to merging and splitting. Chandrasekaran *et. al.* [3] observe that a solution based on the matrix product $\underline{\underline{Y}}^T_{nN}\underline{\underline{Y}}_{nN}$, as above, is likely to lead to inaccurate results because of conditioning problems, and they develop a method for incrementally updating SVD solutions with a single observation. SVD methods have proven more accurate (see [14]) and can be generalised for block updating, with a change of mean, provided all right singular vectors are maintained. We have not seen this published by others, but do have a derivation which is too long to include in this paper.

We can also perform downdating of the SVD, but only via EVD, in which a reduced set of left singular vectors is computed first. We allow for a shift of mean and block downdating. Again a derivation is too long for this paper, but here we provide a brief explanation of the difficulty. Given SVD for three data sets such that $[\underline{\underline{APB}}^T, \underline{\underline{CQD}}^T] = \underline{\underline{ERF}}^T$, the problem is to compute the SVD $\underline{\underline{APB}}^T$. This requires that terms on the left hand side be separated, which is difficult. Our approach follows that of Bunch *et. al* [2], who are the only authors we know to address the problem, and multiply on the right by the transpose of the matrices, hence taking an inner product and converting the problem to EVD, as follows: $[\underline{\underline{APB}}^T, \underline{\underline{CQD}}^T][\underline{\underline{APB}}^T, \underline{\underline{CQD}}^T]^T = \underline{\underline{ERF}}^T(\underline{\underline{ERF}}^T)^T$ leading to $\underline{\underline{AP}}^2\underline{\underline{A}}^T + \underline{\underline{CQ}}^2\underline{\underline{C}}^T = \underline{\underline{ER}}^2\underline{\underline{E}}^T$. We conclude that a *general* dynamic change of SVD is not possible directly; this is not the case for EVD.

SVD methods were actually proposed quite early in the development of incremental eigenproblem analysis [2]. This early work included a proposal to delete single observations, but did not extend to merging and splitting. SVD also formed the basis of a proposal to incrementally update an eigenspace with several observations at one step [8]. However, contrary to our method, a possible change in the dimension of the solution eigenspace was not considered. Furthermore, none of these methods considered a change in origin.

Our incremental method is based on the matrix product $\underline{\underline{C}}_{nn} = \underline{\underline{Y}}_{nN}\underline{\underline{Y}}^T_{nN}$, and specifically its approximation as in Equation 6. It is a generalisation of our earlier work [14], which now appears naturally as the special case of adding a single observation.

C. Representing and classifying observations

High-dimensional observations may be approximated by a low-dimensional vector using an eigenspace model. Eigenspace models may also be used for classification. We briefly discuss both ideas here, prior to using them in our results section.

An n-dimensional observation $\underline{x}_n$ is represented using an eigenspace model $\Omega = (\bar{\underline{x}}, \underline{\underline{U}}_{np}, \underline{\underline{\Lambda}}_{pp}, N)$ as a p-dimensional vector $\underline{g}_p$:

$$\underline{g}_p = \underline{\underline{U}}^T_{np}(\underline{x}_n - \bar{\underline{x}}) \tag{11}$$

This shifts the observation to the mean, and then represents it by components along each eigenvector. This is called the Karhunen-Loève transform [16].

The n-dimensional *residue vector* is defined by:

$$\begin{aligned}\underline{h}_n &= \underline{x}_n - \underline{\underline{U}}_{np}\underline{g}_p \\ &= \underline{x}_n - \underline{\underline{U}}_{np}(\underline{\underline{U}}^T_{np}(\underline{x}_n - \bar{\underline{x}}))\end{aligned} \tag{12}$$

and $\underline{h}_n$ is orthogonal to every vector in $\underline{\underline{U}}_{np}$. Thus, $|\underline{h}_n|$ is the residue error in the representation of $\underline{x}_n$ with respect Ω.

The likelihood associated with the same observation is given by:

$$\begin{aligned}P(\underline{x}|\Omega) &= \frac{\exp(-\frac{1}{2}(\underline{x} - \bar{\underline{x}})^T\underline{\underline{C}}^{-1}_{nn}(\underline{x} - \bar{\underline{x}}))}{(2\pi)^{n/2}\det(\underline{\underline{C}}_{nn})^{1/2}} \\ &= \frac{\exp(-\frac{1}{2}(\underline{x} - \bar{\underline{x}})^T\underline{\underline{U}}_{nn}\underline{\underline{\Lambda}}^{-1}_{nn}\underline{\underline{U}}^T_{nn}(\underline{x} - \bar{\underline{x}}))}{(2\pi)^{n/2}\det(\underline{\underline{\Lambda}}_{nn})^{1/2}}\end{aligned} \tag{13}$$

Clearly, the above definition cannot be used directly in cases where $N \leq n$, as $\underline{\underline{C}}_{nn}$ is then rank degenerate. In such cases we use an alternative definition due to Moghaddam and Pentland [6] which is beyond the scope of this paper.

III. Merging Eigenspace models

We now turn our attention to one of the two main contributions of this paper, merging eigenspace models.

We derive a solution to the following problem. Let $\underline{\underline{X}}_{nN}$ and $\underline{\underline{Y}}_{nM}$ be two sets of observations. Let their eigenspace models be $\Omega = (\bar{\underline{x}}, \underline{\underline{U}}_{np}, \underline{\underline{\Lambda}}_{pp}, N)$ and $\Psi = (\bar{\underline{y}}, \underline{\underline{V}}_{nq}, \underline{\underline{\Delta}}_{qq}, M)$ respectively. The problem is to compute the eigenspace model $\Phi = (\bar{\underline{z}}, \underline{\underline{W}}_{nr}, \underline{\underline{\Pi}}_{rr}, P)$, for $\underline{\underline{Z}}_{n(N+M)} = [\underline{\underline{X}}_{nN}\underline{\underline{Y}}_{nM}]$ using only Ω and Ψ.

Clearly, the total number of new observations is $P = N + M$.

The combined mean is:

$$\bar{\underline{z}} = \frac{1}{(N+M)}\left(N\bar{\underline{x}} + M\bar{\underline{y}}\right) \tag{14}$$

The combined covariance matrix is:

$$\begin{aligned}\underline{\underline{E}}_{nn} &= \frac{1}{(N+M)}\left(\sum_{i=1}^{N+M}(\underline{z} - \bar{\underline{z}})(\underline{z} - \bar{\underline{z}})^T\right) \\ &= \frac{1}{(N+M)}\left(\sum_{i=1}^{N}\underline{x}^i(\underline{x}^i)^T + \sum_{i=1}^{M}\underline{y}^i(\underline{y}^i)^T\right) - \underline{z}\underline{z}^T \\ &= \frac{1}{(N+M)}(N\underline{\underline{C}}_{nn} + N\bar{\underline{x}}\bar{\underline{x}}^T + M\underline{\underline{D}}_{nn} + M\bar{\underline{y}}\bar{\underline{y}}^T) - \underline{z}\underline{z}^T \\ &= \frac{N}{(N+M)}\underline{\underline{C}}_{nn} + \frac{M}{(N+M)}\underline{\underline{D}}_{nn} + \\ &\quad \frac{NM}{(N+M)^2}(\bar{\underline{x}} - \bar{\underline{y}})(\bar{\underline{x}} - \bar{\underline{y}})^T\end{aligned} \tag{15}$$

where $\underline{\underline{C}}_{nn}$ and $\underline{\underline{D}}_{nn}$ are the covariance matrices for $\underline{\underline{X}}_{nN}$ and $\underline{\underline{Y}}_{nM}$, respectively.

We wish to compute the s eigenvectors and eigenvalues that satisfy:

$$\underline{\underline{E}}_{nn} = \underline{\underline{W}}_{ns}\underline{\underline{\Pi}}_{ss}\underline{\underline{W}}^T_{ns} \tag{16}$$

where some eigenvalues are subsequently discarded to give r non-negligible eigenvectors and eigenvalues. The problem to be solved is of size s, and this is necessarily bounded by

$$\max(p, q) \leq s \leq p + q + 1 \tag{17}$$

We explain the perhaps surprising additional 1 in the upper limit later (Section III-A.1), but briefly, it is needed to allow for the vector difference between the means, $\bar{\underline{x}} - \bar{\underline{y}}$.

A. Method of solution

This problem may be solved in three steps:

1) Construct an orthonormal basis set, $\underline{\underline{\Upsilon}}_{ns}$, that spans both eigenspace models and $\bar{\underline{x}} - \bar{\underline{y}}$. This basis differs from the required eigenvectors, $\underline{\underline{W}}_{ns}$, by a rotation, $\underline{\underline{R}}_{ss}$, so that:

$$\underline{\underline{W}}_{ns} = \underline{\underline{\Upsilon}}_{ns}\underline{\underline{R}}_{ss} \tag{18}$$

2) Use $\underline{\underline{\Upsilon}}_{ns}$ to derive an *intermediate* eigenproblem. The solution of this problem provides the eigenvalues, $\underline{\underline{\Pi}}_{ss}$, needed for the merged eigenmodel. The eigenvectors, $\underline{\underline{R}}_{ss}$, comprise the linear transform that rotates the basis set $\underline{\underline{\Upsilon}}_{ns}$.
3) Compute the eigenvectors $\underline{\underline{W}}_{ns}$, as above, and discard any eigenvectors and eigenvalues using the chosen criteria (as discussed above) to yield $\underline{\underline{W}}_{nr}$ and $\underline{\underline{\Pi}}_{rr}$.

We now give details of each step.

1) Construct an orthonormal basis set: To construct an orthonormal basis for the combined eigenmodels we must choose a set of orthonormal vectors that span three subspaces: (1) the subspace spanned by eigenvectors $\underline{\underline{U}}_{np}$; (2): the subspace spanned by eigenvectors $\underline{\underline{V}}_{nq}$;(3) the subspace spanned by $(\underline{\bar{x}} - \underline{\bar{y}})$. The last of these is a single vector. It is necessary because the vector joining the centre of the two eigenspace models need not belong to either eigenspace. This accounts for the additional 1 in the upper limit of the bounds of s in Equation 17. To see this consider a pair of two-dimensional eigenspaces which are embedded in a three-dimensional space. The eigenvectors for each eigenspace could define parallel planes that are separated by a vector perpendicular to each of them. Clearly, a merged model should be a 3D ellipse, and the vector between the origins of the models must contain a component perpendicular to both eigenspaces.

A sufficient spanning set is:

$$\underline{\underline{\Upsilon}}_{ns} = [\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}] \tag{19}$$

where $\underline{\underline{\nu}}_{nt}$ is an orthonormal basis set for that component of the eigenspace of Ψ which is orthogonal to the eigenspace of Ω, and in addition accounts for that component of $(\underline{\bar{x}} - \underline{\bar{y}})$ orthogonal to both eigenspaces; $t = s - p$.

To construct $\underline{\underline{\nu}}_{nt}$ we start by computing the residues of each of the eigenvectors in $\underline{\underline{V}}_{nq}$ with respect to the eigenspace of Ω:

$$\underline{\underline{G}}_{pq} = \underline{\underline{U}}^T_{np} \underline{\underline{V}}_{nq} \tag{20}$$

$$\underline{\underline{H}}_{nq} = \underline{\underline{V}}_{nq} - \underline{\underline{U}}_{np} \underline{\underline{G}}_{pq} \tag{21}$$

The $\underline{\underline{H}}_{nq}$ are all orthogonal to $\underline{\underline{U}}_{np}$ in the sense that $(\underline{\underline{H}}^i)^T \underline{\underline{U}}^j = 0$ for all i, j. In general, however, some of the $\underline{\underline{H}}_{nq}$ are zero vectors, because such vectors represent the intersection of the two eigenspaces. These zero vectors are removed to leave $\underline{\underline{H}}_{nq'}$. We also compute the residue $\underline{h}$ of $\underline{\bar{y}} - \underline{\bar{x}}$ with respect to the eigenspace of Ω, using Equation 12.

$\underline{\underline{\nu}}_{nt}$ can now be computed by finding an orthonormal basis for $[\underline{\underline{H}}_{nq'}, \underline{h}]$, which is sufficient to ensure that $\underline{\underline{\Upsilon}}_{ns}$ is orthonormal. Gramm-Schmidt orthonormalisation [17] may be used to do this:

$$\underline{\underline{\nu}}_{nt} = \text{Orthonormalise}([\underline{\underline{H}}_{nq'}, \underline{h}]) \tag{22}$$

2) Forming a intermediate eigenproblem: We now form a new eigenproblem by substituting Equation 19 into Equation 18, and the result together with Equation 15 into Equation 16 to obtain:

$$\begin{aligned} &\frac{N}{(N+M)} \underline{\underline{C}}_{nn} + \frac{M}{(N+M)} \underline{\underline{D}}_{nn} + \\ &\frac{NM}{(N+M)} (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T = \\ &[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}] \underline{\underline{R}}_{ss} \underline{\underline{\Pi}}_{ss} \underline{\underline{R}}^T_{ss} [\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T \end{aligned} \tag{23}$$

Multiplying both sides on the left by $[\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}]^T$, on the right by $[\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}]$ and using the fact that $[\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}]^T$ is a left inverse of $[\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}]$ we obtain:

$$\begin{aligned} &[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T \Big(\frac{N}{(N+M)} \underline{\underline{C}}_{nn} + \frac{M}{(N+M)} \underline{\underline{D}}_{nn} + \\ &\frac{NM}{(N+M)^2} (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T\Big) [\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}] = \underline{\underline{R}}_{ss} \underline{\underline{\Pi}}_{ss} \underline{\underline{R}}^T_{ss} \end{aligned} \tag{24}$$

which is a new eigenproblem whose solution eigenvectors constitute the $\underline{\underline{R}}_{ss}$ we seek, and whose eigenvalues provide eigenvalues for the combined eigenspace model. We do not know the covariance matrices $\underline{\underline{C}}_{nn}$ or $\underline{\underline{D}}_{nn}$, but these can be eliminated as follows:

The first term in Equation 24 is proportional to:

$$[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T \underline{\underline{C}}_{nn} [\underline{\underline{U}}_{np} \underline{\underline{\nu}}] = \begin{bmatrix} \underline{\underline{U}}^T_{np} \underline{\underline{C}}_{nn} \underline{\underline{U}}_{np} & \underline{\underline{U}}^T_{np} \underline{\underline{C}}_{nn} \underline{\underline{\nu}}_{nt} \\ \underline{\underline{\nu}}^T_{nt} \underline{\underline{C}}_{nn} \underline{\underline{U}}_{np} & \underline{\underline{\nu}}^T_{nt} \underline{\underline{C}}_{nn} \underline{\underline{\nu}}_{nt} \end{bmatrix} \tag{25}$$

By Equation 6, $\underline{\underline{U}}^T_{np} \underline{\underline{C}}_{nn} \underline{\underline{U}}_{np} \approx \underline{\underline{\Lambda}}_{pp}$. Also, $\underline{\underline{U}}^T_{np} \underline{\underline{\nu}}_{nt} = \underline{\underline{0}}_{pt}$ by construction, and again, using Equation 6 we conclude:

$$[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T \underline{\underline{C}}_{nn} [\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}] \approx \begin{bmatrix} \underline{\underline{\Lambda}}_{pp} & \underline{\underline{0}}_{pt} \\ \underline{\underline{0}}_{tp} & \underline{\underline{0}}_{tt} \end{bmatrix} \tag{26}$$

The second term in Equation 24 is proportional to:

$$[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T \underline{\underline{D}}_{nn} [\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}] = \begin{bmatrix} \underline{\underline{U}}^T_{np} \underline{\underline{D}}_{nn} \underline{\underline{U}}_{np} & \underline{\underline{U}}^T_{np} \underline{\underline{D}}_{nn} \underline{\underline{\nu}}_{nt} \\ \underline{\underline{\nu}}^T_{nt} \underline{\underline{D}}_{nn}, \underline{\underline{U}}_{np} & \underline{\underline{\nu}}^T_{nt} \underline{\underline{D}}_{nn} \underline{\underline{\nu}}_{nt} \end{bmatrix}. \tag{27}$$

We have $\underline{\underline{D}}_{nn} \approx \underline{\underline{V}}_{nq} \underline{\underline{\Delta}}_{qq} \underline{\underline{V}}^T_{nq}$, which on substitution gives the right hand side as

$$\begin{bmatrix} \underline{\underline{U}}^T_{np} \underline{\underline{V}}_{nq} \underline{\underline{\Delta}}_{qq} \underline{\underline{V}}^T_{nq} \underline{\underline{U}}_{np} & \underline{\underline{U}}^T_{np} \underline{\underline{V}}_{nq} \underline{\underline{\Delta}}_{qq} \underline{\underline{V}}^T_{nq} \underline{\underline{\nu}}_{nt} \\ \underline{\underline{\nu}}^T_{nt} \underline{\underline{V}}_{nq} \underline{\underline{\Delta}}_{qq} \underline{\underline{V}}^T_{nq} \underline{\underline{U}}_{p} & \underline{\underline{\nu}}^T_{nt} \underline{\underline{V}}_{nq} \underline{\underline{\Delta}}_{qq} \underline{\underline{V}}^T_{nq} \underline{\underline{\nu}}_{nt} \end{bmatrix}$$

From Equation 20 we have $\underline{\underline{G}}_{pq} = \underline{\underline{U}}^T_{np} \underline{\underline{V}}_{nq}$. Set $\underline{\underline{\Gamma}}_{tq} = \underline{\underline{\nu}}^T_{nt} \underline{\underline{V}}_{nq}$. We obtain:

$$[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T \underline{\underline{D}} [\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}] = \begin{bmatrix} \underline{\underline{G}}_{pq} \underline{\underline{\Delta}}_{qq} \underline{\underline{G}}^T_{pq} & \underline{\underline{G}}_{pq} \underline{\underline{\Delta}}_{qq} \underline{\underline{\Gamma}}^T_{tq} \\ \underline{\underline{\Gamma}}_{tq} \underline{\underline{\Delta}}_{qq} \underline{\underline{G}}^T_{pq} & \underline{\underline{\Gamma}}_{tq} \underline{\underline{\Delta}}_{qq} \underline{\underline{\Gamma}}^T_{tq} \end{bmatrix} \tag{28}$$

Now consider the final term in Equation 24:

$$\begin{aligned} &[\underline{\underline{U}}_{np} \underline{\underline{\nu}}_{nt}]^T (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T [\underline{\underline{U}}_{np}, \underline{\underline{\nu}}_{nt}] = \\ &\begin{bmatrix} \underline{\underline{U}}^T_{np} (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T \underline{\underline{U}}_{np} & \underline{\underline{U}}^T_{np} (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T \underline{\underline{\nu}}_{nt} \\ \underline{\underline{\nu}}^T_{nt} (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T \underline{\underline{U}}_{np} & \underline{\underline{\nu}}^T_{nt} (\underline{\bar{x}} - \underline{\bar{y}})(\underline{\bar{x}} - \underline{\bar{y}})^T \underline{\underline{\nu}}_{nt} \end{bmatrix} \end{aligned} \tag{29}$$

Setting $\underline{g}_p = \underline{\underline{U}}^T_{np} (\underline{\bar{x}} - \underline{\bar{y}})$, and $\underline{\gamma}_t = \underline{\underline{\nu}}^T_{nt} (\underline{\bar{x}} - \underline{\bar{y}})$, this becomes:

$$\begin{bmatrix} \underline{g}_p \underline{g}^T_p & \underline{g}_p \underline{\gamma}^T_t \\ \underline{\gamma}_t \underline{g}^T_p & \underline{\gamma}_t \underline{\gamma}^T_t \end{bmatrix} \tag{30}$$

So, the new eigenproblem to be solved may be approximated by

$$\begin{aligned} &\frac{N}{(N+M)} \begin{bmatrix} \underline{\underline{\Lambda}}_{pp} & \underline{\underline{0}}_{pt} \\ \underline{\underline{0}}_{tp} & \underline{\underline{0}}_{tt} \end{bmatrix} + \\ &\frac{M}{(N+M)} \begin{bmatrix} \underline{\underline{G}}_{pq} \underline{\underline{\Delta}}_{qq} \underline{\underline{G}}^T_{pq} & \underline{\underline{G}}_{pq} \underline{\underline{\Delta}}_{qq} \underline{\underline{\Gamma}}^T_{tq} \\ \underline{\underline{\Gamma}}_{tq} \underline{\underline{\Delta}}_{qq} \underline{\underline{G}}^T_{pq} & \underline{\underline{\Gamma}}_{tq} \underline{\underline{\Delta}}_{qq} \underline{\underline{\Gamma}}^T_{tq} \end{bmatrix} + \\ &\frac{NM}{(N+M)^2} \begin{bmatrix} \underline{g}_p \underline{g}^T_p & \underline{g}_p \underline{\gamma}^T_t \\ \underline{\gamma}_t \underline{g}^T_p & \underline{\gamma}_t \underline{\gamma}^T_t \end{bmatrix} = \underline{\underline{R}}_{ss} \underline{\underline{\Pi}}_{ss} \underline{\underline{R}}^T_{ss} \end{aligned} \tag{31}$$

Each matrix is of size $s \times s$, where $s = p + t \leq p + q + 1 \leq \min(n, M + N)$. 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. This is of crucial computational importance because it makes the eigenproblem tractable in cases where the dimension of each datum is large, as is the case for image data.

3) Computing the eigenvectors: The matrix $\underline{\underline{\Pi}}_{ss}$ is the eigenvalue matrix we set out to compute. The eigenvectors $\underline{\underline{R}}_{ss}$ comprise a rotation for $\underline{\underline{\Upsilon}}_{ns}$. Hence, we use Equation 18 to compute the eigenvectors for $\underline{\underline{\Pi}}_{ss}$. However, not all eigenvectors and eigenvalues need be kept, and some ($s - r$ of them) may be discarded using a criterion as previously discussed in Section II. This discarding of eigenvectors and eigenvalues will usually be carried out each time a pair of eigenspace models is merged.

Notice that there are two sources of error. The first is rounding error introduced because of finite machine precision. The second source of error is introduced by truncation of the eigenmodel, i.e. the discarding of eigenvectors and eigenvalues. This is the dominant source, and its precise behaviour deserves further investigation, both theoretically and empirically. However, our experience backs up our intuition: discarding more eigenvectors and eigenvalues worsens the approximation. (Furthermore, note that we require no lower bound on the number of eigenvectors, which is in contrast to SVD methods where all right singular vectors must be kept to block update while shifting the mean.)

B. Discussion on the form of the solution

We now briefly justify that the solution obtained is of the correct form by considering several special cases.

First, suppose that both eigenspace models are null, that is each is specified by $(\underline{0}, \underline{\underline{0}}, \underline{\underline{0}}, 0)$. Then the system is clearly degenerate and null eigenvectors and zero eigenvalues are computed.

If exactly one eigenspace model is null, then the non-null eigenspace model is computed and returned by this process. To see this, suppose that Ψ is null. Then, the second and third matrices on the left-hand side of Equation 31 both disappear. The first matrix reduces to $\underline{\underline{\Lambda}}_{pp}$ exactly ($t = 0$), and hence the eigenvalues remain unchanged. In this case, the rotation $\underline{\underline{R}}_{ss}$ is the identity matrix, and the eigenvectors are also unchanged. If instead Ω is a null model, then only the second matrix will remain (as $N = 0$). Also $\underline{\underline{\nu}}_{nt}$ and $\underline{\underline{V}}_{nq}$ will be related by a rotation (or else identical). The solution to the eigenproblem then computes the inverse of any such rotation, and the eigenspace model remain unchanged.

Suppose Ψ has exactly one observation, then it is specified by $(\underline{y}, \underline{\underline{0}}, \underline{\underline{0}}, 1)$. Hence, the middle term on the left of Equation 31 disappears, and $\underline{\underline{\nu}}_{nt}$ is the unit vector in the direction $\underline{y} - \underline{\bar{x}}$. Hence $\underline{\gamma}_t = |\underline{y} - \underline{\bar{x}}|$ is a scalar, and the eigenproblem becomes

$$\frac{N}{(N+1)}\begin{bmatrix} \underline{\underline{\Lambda}}_{pp} & \underline{\underline{0}}_{p1} \\ \underline{\underline{0}}_{1p} & 0 \end{bmatrix} + \frac{N}{(N+1)^2}\begin{bmatrix} \underline{g}_p\underline{g}_p^T & \underline{g}_p\gamma \\ \gamma\underline{g}_p^T & \gamma^2 \end{bmatrix} \tag{32}$$

which is exactly the form obtained when one observation is explicitly added, as we have proven elsewhere [14]. This special case has interesting properties too: if the new observation lies within the subspace spanned by $\underline{\underline{U}}_{np}$, then $\gamma = 0$ and any change in the eigenvectors and eigenvalues can be explained by rotation and scaling caused by $\underline{g}_p\underline{g}_p^T$. Furthermore, in the unlikely event that $\underline{\bar{x}} = \underline{\bar{y}}$, then the right matrix disappears altogether, in which case the eigenvalues are scaled by $N/(N+1)$, but the eigenvectors are unchanged. Finally, as $N \to \infty$, then $N/(N+1) \to 1$ and $N/(N+1)^2 \to 0$, indicating a stable model in the limit.

If Ω has exactly one observation, then it is specified by $(\underline{x}, \underline{\underline{0}}, \underline{\underline{0}}, 1)$. Thus the first matrix on the left of Equation 31 disappears. Then $\underline{\underline{G}}_{pq}$ is a zero matrix, and $\underline{\underline{\nu}}_{nt} = [\underline{\underline{V}}_{nq}\underline{\underline{h}}]$, where $\underline{\underline{h}}$ is the component of $\underline{\bar{y}} - \underline{\bar{x}}$ which is orthogonal to the eigenspace of Ψ. Hence the eigenproblem is:

$$\frac{M}{(M+1)}\begin{bmatrix} 0 & \underline{0} \\ \underline{0}^T & \underline{\underline{\Gamma}}_{tq}\underline{\underline{\Delta}}_{qq}\underline{\underline{\Gamma}}_{tq}^T \end{bmatrix} + \frac{M}{(1+M)^2}\begin{bmatrix} 0 & 0 \\ 0 & \underline{\gamma}_t\underline{\gamma}_t^T \end{bmatrix} \tag{33}$$

Given that in this case $\underline{\underline{\Gamma}}_{tq} = [\underline{\underline{V}}_{nq}\underline{\underline{h}}]^T\underline{\underline{V}}_{nq}$, then $\underline{\underline{\Gamma}}_{tq}\underline{\underline{\Delta}}_{qq}\underline{\underline{\Gamma}}_{tq}^T$ has the form of $\underline{\underline{\Delta}}_{qq}$, but with a row and column of zeros appended. Also, $\underline{\gamma}_t = [\underline{\underline{V}}_{nq}\underline{\underline{h}}]^T(\underline{\bar{x}} - \underline{\bar{y}})$. Substitution of these terms shows that in this case too, the solution reduces to the special case of adding a single new observation: Equation 33 is of the same form as Equation 32, as can readily be shown.

If the Ω and Ψ models are identical, then $\underline{\bar{x}} = \underline{\bar{y}}$. In this case the third term on the left of Equation 31 disappears. Furthermore, $\underline{\underline{\Gamma}}_{tq}$ is a zero matrix, and $\underline{\underline{G}}_{pq} = \underline{\underline{U}}^T_{np}\underline{\underline{U}}_{np}$ is the identity matrix, with $p = q$. Hence, the first and second matrices on the left of Equation 31 are identical, with $N = M$, and they reduce to the matrices of eigenvalues. Hence, adding two identical eigenmodels yields a third which is identical in every respect, except for a change in the number of observations.

Finally, notice that for fixed M, as $N \to \infty$ so the solution tends to the Ω model. Conversely, for fixed N as $M \to \infty$ so the solution tends to the Ψ model. If M and N tend to ∞ simultaneously, then the final term loses its significance.

C. Algorithm

Here, for completeness, we now express the mathematical results obtained above, for merging models, in the form of an algorithm for direct computer implementation; see Figure 1.

```
Function Merge( $\bar{x}, U, \Lambda, N, \bar{y}, V, \Delta, M$ )
returns ($\bar{z}, W, \Pi, P$)

BEGIN
    $P$ = $N$ + $M$
    $\bar{z}$ = ($N\bar{x}$ + $M\bar{y}$)/$P$
    difforg = $\bar{x}$ - $\bar{y}$
    $G$ = $U^T V$
    $H$ = $V - UG$
    for each column vector of $H$
        discard this column,
        if it is of small magnitude.
    endfor
    $g$ = $U^T$difforg
    $h$ = difforg - $Ug$
    $\nu$ = orthonormalbasis for $[H, h]$
    $\gamma$ = $\nu^T$difforg
    $\Gamma$ = $\nu^T$V
    $p$ = size of $\Lambda$
    $q$ = size of $\Delta$
    $t$ = number of basis vectors in $\nu$
    $A$ = construct LHS of Equation 31
    $\Pi$ = eigenvalues of $A$
    $R$ = eigenvectors of $A$
    $W$ = $[U\nu]R$
    discard small eigensolutions, as appropriate
END
```

Fig. 1. Algorithm for merging two eigenspace models.

D. Complexity

Computing an eigenspace model of size N as a single batch may incur a computational cost $O(N^3)$. Our experimental results bear this out, though there are faster methods available using SVD [18]. Examination of our merging algorithm shows that it also requires an intermediate eigenvalue problem to be solved, as well as other steps;

again overall giving cubic computational requirements. Nevertheless, let us suppose that Ω, with N observations, can be represented by p eigenvectors, and that Ψ, with M observations, can be represented by q eigenvectors. Typically p and q are (much) less than N and M, respectively.

To compute an overall model with the batch method requires $O((N+M)^3)$ operations. Assuming that both models to be merged are already known, our merging method requires at most $O((p+q+1)^3)$ operations; the problem to be solved becomes smaller the greater the amount of overlap between the eigenspaces of Ω and Ψ. (In fact, the number of operations required is $O(s^3)$: see the end of Section III-A.1.)

If one, or both, of the models to be merged are unknown initially, then we incur an extra cost of $O(N^3)$, $O(M^3)$, or $O(N^3+M^3)$, which reduces any advantage. Nevertheless, in one typical scenario, we might expect Ω to be known (an existing large database of N observations), while Ψ is a relatively small batch of M observations to be added to it. In this case, the extra penalty, of $O(M^3)$, is of little significance compared to $O((N+M)^3)$.

Overall, while an exact analysis is complicated and indeed data dependent, we expect efficiency gains in both time and memory resources in practice.

Furthermore, if computer memory is limited, subdivision of the initial set may be unavoidable in order to reduce eigenspace model computation to a tractable problem.

IV. Splitting eigenspace models

Here we show how to split two eigenspace models. Given an eigenspace model $\Phi = (\underline{z}, \underline{\underline{W}}_{nr}, \underline{\underline{\Pi}}_{rr}, P)$ we remove $\Psi = (\underline{y}, \underline{\underline{V}}_{nq}, \underline{\underline{\Delta}}_{qq}, M)$ from it to give a third model $\Omega = (\underline{x}, \underline{\underline{U}}_{np}, \underline{\underline{\Lambda}}_{pp}, N)$. We use $\underline{\underline{\Pi}}_{rr}$, because $\underline{\underline{\Pi}}_{ss}$ is not available in general. We ask the reader to carefully note that splitting means removing a subset of observations; the method is the inverse of merging in this sense. However, it is impossible to regenerate information which was discarded when the overall model was created (whether by batch methods or otherwise). Thus, if we split one eigenspace model from a larger one, the eigenvectors of the remnant must still form some subspace of the larger.

The derivation (and algorithm) for splitting follow in a very straightforward way by analogy from those of merging. Therefore we state the results for splitting without proof. Clearly, $N = P - M$. The new mean is:

$$\bar{\underline{x}} = \frac{P}{N}\bar{\underline{z}} - \frac{M}{N}\bar{\underline{y}} \tag{34}$$

As in the case of merging, new eigenvalues and eigenvectors are computed via an intermediate eigenproblem. In this case it is:

$$\frac{P}{N}\underline{\underline{\Pi}}_{rr} - \frac{M}{N}\underline{\underline{G}}_{rp}\underline{\underline{\Delta}}_{pp}\underline{\underline{G}}_{rp}^T - \frac{M}{P}\underline{g}_r\underline{g}_r^T = \underline{\underline{R}}_{rr}\underline{\underline{\Lambda}}_{rr}\underline{\underline{R}}_{rr}^T \tag{35}$$

where $\underline{\underline{G}}_{rp} = \underline{\underline{W}}_{nr}^T\underline{\underline{V}}_{nq}$ and $\underline{g}_r = \underline{\underline{W}}_{nr}^T(\bar{\underline{y}} - \bar{\underline{x}})$.

The eigenvalues we seek are the q non-zero elements on the diagonal of $\underline{\underline{\Lambda}}_{rr}$. Thus we can permute $\underline{\underline{R}}_{rr}$ and $\underline{\underline{\Lambda}}_{rr}$, and write without loss of generality:

$$\begin{aligned}\underline{\underline{R}}_{rr}\underline{\underline{\Lambda}}_{rr}\underline{\underline{R}}_{rr}^T &= [\underline{\underline{R}}_{rp}\underline{\underline{R}}_{rt}]\begin{bmatrix}\underline{\underline{\Lambda}}_{pp} & \underline{\underline{0}}_{pt}\\ \underline{\underline{0}}_{tp} & \underline{\underline{0}}_{tt}\end{bmatrix}[\underline{\underline{R}}_{rp}\underline{\underline{R}}_{rt}]^T\\ &= \underline{\underline{R}}_{rp}\underline{\underline{\Lambda}}_{pp}\underline{\underline{R}}_{rp}^T \end{aligned} \tag{36}$$

where $p = r - q$.

Hence we need only identify the eigenvectors in $\underline{\underline{R}}_{rr}$ with non-zero eigenvalues, and compute the $\underline{\underline{U}}_{np}$ as:

$$\underline{\underline{U}}_{np} = \underline{\underline{W}}_{nr}\underline{\underline{R}}_{rp} \tag{37}$$

In terms of complexity, splitting must always involve the solution of an eigenproblem of size r. An algorithm for splitting may readily be written out using a similar approach to that for merging.

V. Results

This section describes various experiments that we carried out to compare the computational efficiency of a batch method with our new methods for merging and splitting, and to compare the eigenspace models produced.

We compared models in terms of Euclidean distance between the means, mean angular deviation of corresponding eigenvectors, and mean **relative** absolute difference between corresponding eigenvalues. In doing so, we took care that both models had the same number of dimensions.

As well as the simple measures above, other performance measures may be more relevant when eigenspace models are used for particular applications, and thus other tests were also performed. Eigenspace models may be used for approximating high-dimensional observations with a low-dimensional vector; the error is the size of the residue vector. The sizes of such residue vectors can readily be compared for both batch and incremental methods. Eigenspace models may also be used for classifying observations, giving the likelihood that an observation belongs to a cluster. Different eigenspace models may be compared by relative differences in likelihoods. We average these differences over all corresponding observations.

We used a database of 400 face images (each of $112 \times 92 = 10304$ pixels) available on-line [1] in the tests reported here ; similar results were obtained in tests with randomly generated data. The gray levels in the images were scaled into the range $[0,1]$ by division only, but no other preprocessing was done. We implemented all functions using commercially available software (Matlab) on a computer with standard configuration (Sun Sparc Ultra 10, 300 Hz, 64 Mb RAM).

The results we present used up to 300 images, as the physical resources of our computer meant that heavy paging started to occur beyond this limit for the batch method, although such paging did not affect the incremental method.

For all tests, the experimental procedure used was to compute eigenspace models using a batch method [15], and compare these to models produced by merging or splitting other models also produced by the batch method. In each case, the largest of the three data sets contained 300 images. These were partitioned into two data sets, each containing a multiple of 50 images. We included the degenerate cases when one model contained zero images. Note that we tested both smaller models merged with larger ones, and vice-versa.

The number of eigenvectors retained in any model, including a merged model, was set to be 100 as a maximum, for ease of comparing results. (Initial tests using other strategies indicate that the resulting eigenspace model is little effected.)

A. Timing

When measuring CPU time we ran the same code several times and chose the smallest value, to minimise the effect of other concurrently running process.

Initially we measured time taken to compute a model using the batch methods, for data sets of different sizes. Results are presented in Figure 2 and show cubic complexity, as predicted.

1) Merging: We then measured the time taken to merge two previously constructed models. The results are shown in Figure 3. This shows that time complexity is approximately symmetric about the point $N = 150$, half the number of input images. This result may be surprising because the algorithm given for merging is not symmetric with respect to its inputs, despite that fact that the mathematical solution is independent of order. The approximate symmetry in time-complexity can be explained by assuming independent eigenspaces with a fixed upper-bound on the number of eigenvectors: suppose the numbers of eigenvectors in the models are N and M. Then complexities of the main steps are approximately as follows: computing a new spanning set, $\underline{\nu}$ is $O(M^3)$; solving an eigenproblem is $O(N^3+M^3)$; rotating the new eigenvectors is $O(N^3+M^3)$. Thus the time complexity, under the stated conditions, is approximately $O(N^3+M^3)$, which is symmetric.

[1] The Olivetti database of faces: `http://www.cam-orl.co.uk/facedatabase.`

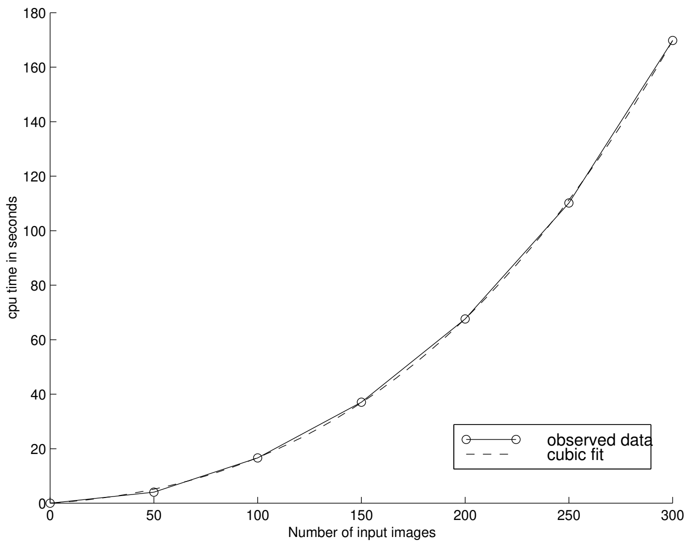


Fig. 2. Time to compute an eigenspace model with a batch method versus the number of images, N. The time is approximated by the cubic: $5.3 \times 10^{-4} N + 8.6 \times 10^{-4} N^2 + 2.8 \times 10^{-6} N^3$.

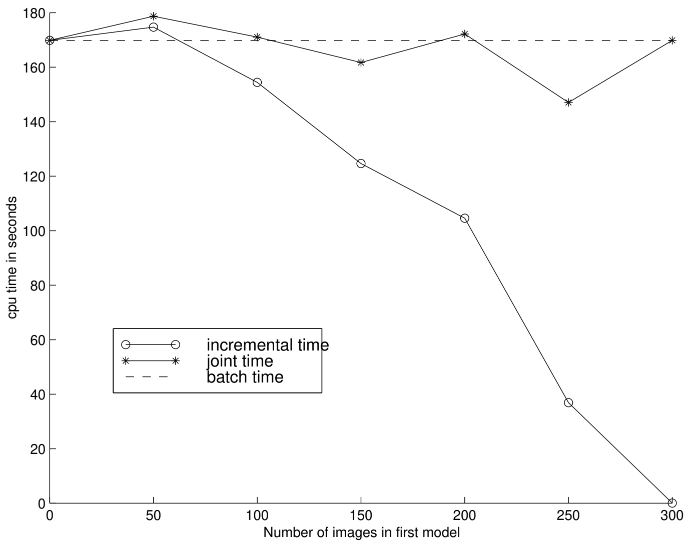


Fig. 4. Time to make a complete eigenspace model for a database of 300 images. The incremental time is the addition of the time to construct only the eigenspace to be added. The joint time is the time to compute both eigenspace models and merge them.

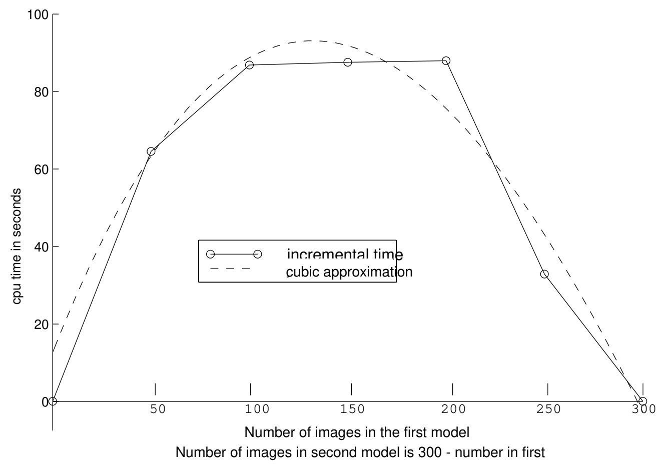


Fig. 3. Time to merge two eigenspace models of images $\Phi = \text{merge}(\Omega, \Psi)$, versus the number of images, N, in Ω. The number of images in Φ is $300 - N$. Hence, the total number of different images used to compute Φ is constant 300.

Next, the times taken to compute an eigenspace model from 300 images in total, using the batch method and our merging method, are compared in Figure 4. The *incremental time* is the time needed to compute the eigenspace model to be merged, and merge it with a precomputed existing one. The *joint time* is the time to compute both smaller eigenmodels and then merge them. As might be expected, incremental time falls as the additional number of images required falls. The joint time is approximately constant, and very similar to the total batch time.

While the incremental method offers no time saving in the cases above, it does use much less memory. This could clearly be seen when a model was computed using 400 images: paging effects set in when a batch method was used and the time taken rose to over 800 seconds. The time to produce an equivalent model by merging two sub-models of size 200, however, took less than half that.

2) Splitting: Time complexity for splitting eigenspaces should depend principally on the size of the large eigenspace which from which the smaller space is being removed, and the size of the smaller eigenspace should have little effect. This is because the size of the intermediate eigenproblem to be solved depends on the size of the larger space, and therefore dominates the complexity. These expectations are borne out experimentally. We computed a large eigenmodel using 300 images, as before. We then removed smaller models of sizes between 50 and 250 images inclusive, in steps of 50 images. At most, 100 eigenvectors were kept in any model.The average time taken was approximately constant, and ranged between 9 and 12 seconds, with a mean time of about 11.4 seconds. These figures are much smaller than those observed for merging because the large eigenspace contains only 100 eigenvectors. Thus the matrices involved in the computation were of size (100×100), whereas in merging the size was at least (150×150), and other computations were involved (such as computing an orthonormal basis).

B. Similarity and performance

The measures used for assessing similarity and performance of batch and incremental methods were described above.

1) Merging: We first compared the means of the models produced by each method using Euclidean distance. This distance is greatest when the models to be merged have the same number of input images (150 in this case), as fall smoothly to zero when either of the models to be merged is empty. The value at maximum is typically very small, and we measured it to be 3.5×10^{-14} units of gray level. This compares favourably with the working precision of Matlab, which is 2.2×10^{-16}.

We next compared the directions of the eigenvectors produced by each method. The error in eigenvector direction was measured by the mean angular deviation, as shown in Figure 5. Ignoring the degenerate cases, when one of the models is empty, we see that angular deviation has a single minimum when the eigenspace models were built with about the same number of images. This may be because when a small model is added to a large model its information tends to be swamped.

These results show angular deviation to be very small on average.

The sizes of eigenvalues from both methods were compared next. In general we observed that the smaller eigenvalues had larger errors, as might be expected as they contain relatively little information and so are more susceptible to noise. In Figure 6 we give the mean absolute difference in eigenvalue. This rises to a single peak when the number of input images in both models is the same. Even so, the maximal value is small, 7×10^{-3} units of gray level. The largest eigenvalue is typically about 100.

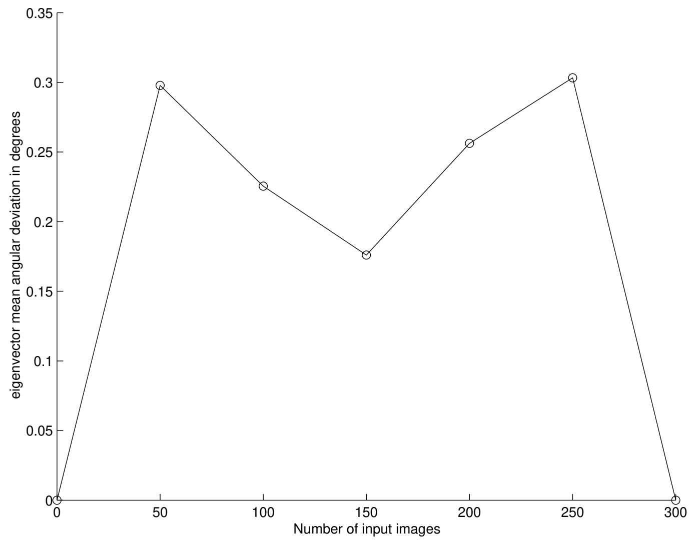


Fig. 5. Angular deviation between eigenvectors produced by batch and incremental methods versus the number of images in the first eigenspace model.

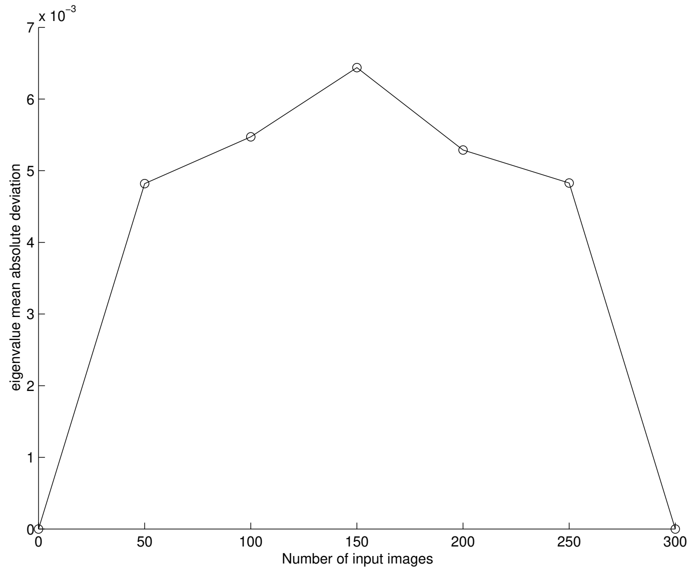


Fig. 6. Difference between eigenvalues produced by batch and incremental methods versus the number of images in the first eigenspace model.

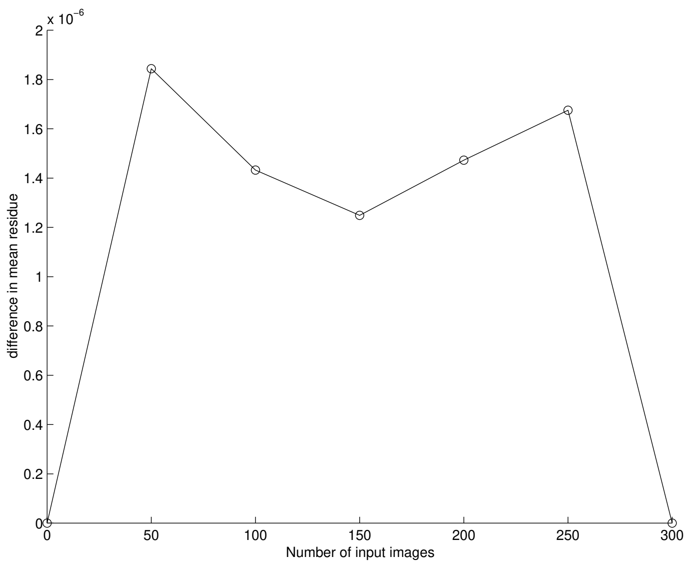


Fig. 7. Difference in reconstruction errors per pixel produced by batch and incremental methods versus the number of images in the first eigenspace model.

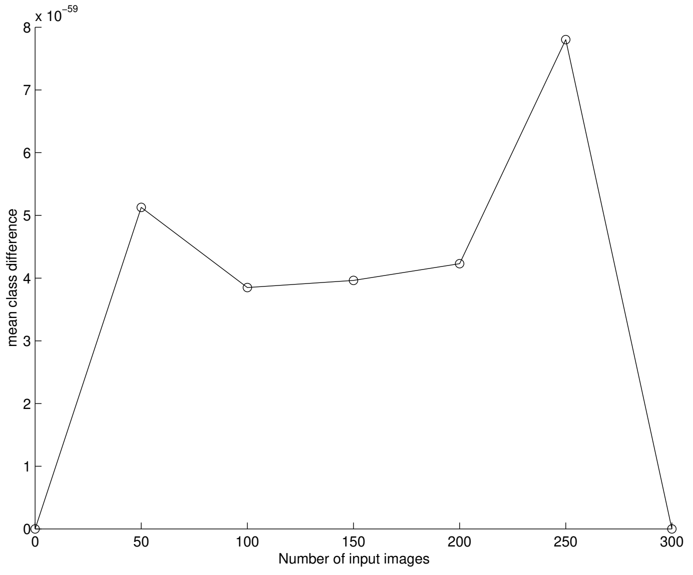


Fig. 8. Difference in likelihoods produced by batch and incremental methods versus the number of images in the first eigenspace model.

We now turn to performance measures. The merged eigenspaces represent the image data with little loss in accuracy, as measured by the mean difference in residue error, Figure 7. This performance measure is typically small, about 10^{-6} units of gray level per pixel, clearly below any noticeable effect.

Finally we compared differences in likelihood values (Equation 13) produced by the two methods. This difference is again small, typically of the order 10^{-59}, as Figure 8 shows; this should be compared with a mean likelihood over all observations of the order 10^{-55}. Again the differences in classifications that would be made by these models would be very small.

2) Splitting: Similar measures for splitting were computed using exactly those conditions described for testing the timing of splitting, and for exactly those characteristics described for merging. In each case a model to be subtracted was computed by a batch method, and removed from the overall model by our splitting procedure. Also, a batch model was made for purposes of comparison with the residual data set. In all that follows the phrase "size of the removed eigenspace" means the number of images used to construct the eigenspace removed from the eigenspace built from 300 images.

The Euclidean distance between the means of the models produced by each method grows monotonically as the size of the removed eigenspace falls, and never exceeds about 1.5×10^{-13} gray-level units. Splitting is slightly less accurate in this respect than merging.

The mean angular deviation between corresponding eigenvector directions rises in similar fashion, from about 0.6 degrees when the size of the removed eigenspace is 250, to about 1.1 when the removed eigenmodel is of size 100. This represented a maximum in the deviation error, because an error of about 1 degree was obtained when the removed model is of size 50. Again, these angular deviations are somewhat larger than those for merging.

The mean difference in eigenvalues shows the same general trend. Its maximum is about 0.5 units of gray level, when the size of the removed eigenspace is 50. This is a much larger error than in the case of merging, but is still relatively small compared to a maximum eigenvalue of about 100. As in the case of merging, the deviation in eigenvalue grows larger as the size (importance) of the eigenvalue falls.

Difference in reconstruction error rises as the size of the removed eigenspace falls. Its size is of the order 10^{-4} units of gray level per pixel, which again is negligible.

The difference in likelihoods is significant, the relative difference in some cases being factors of 10 or more. After conducting further experiments, we found that this relative difference is sensitive to the errors introduced when eigenvectors and eigenvalues are discarded. This is not a surprise, given that likelihood differences are magnified exponentially. We found that changing the criteria for discarding eigenvectors very much reduced: relative difference in likelihood of the order 10^{-14} were achieved in some cases. We conclude that should an application require not only splitting, but also require classification, then eigenvectors and eigenvalues must be discarded with care. We suggest keeping eigenvectors whose corresponding eigenvalues exceed a threshold.

Overall the trend is clear; accuracy and performance grew worse, against any measure we used, as the size of the eigenmodel being removed falls.

VI. Applications

We now turn to applications of our methods. We have experimented with building point distribution models [19] of three dimensional blood vessels and texture classification, while others have used similar methods for updating image motion parameters [8], selecting salient views [3], and building large image databases [3]. In this paper we feel it is appropriate to discuss applications that are more general in nature; the intention is to furnish the reader with a practically useful appreciation of the characteristics of our methods, and avoid the particular diversions of any specific application. We chose, therefore, building large databases of images, a security application, and the dynamic construction of Gaussian mixture models.

A. Building a large database

An obvious application of our methods is to build an eigenspace for many images, when there are too many to store in memory at once. This might arise in the case of very large databases, and has been previously suggested [3]. Intuition suggests that images in the database will be better represented by the model is if all of them are used in its construction; EVD (and SVD) fits a hyperplane to the data in the least squares sense.

To test this we built eigenmodels using all images and a subset of images, using both batch and incremental methods, using a small test set to allow comparisons to the ideal case. As might be expected from the experiments above, the batch and incremental eigenspaces turned out to be very similar, when comparing either the models built from a subset of images, or else those models built from them all. In both cases, the models built from a subset of images represented those images used in construction very well — they had a very low residue error. However, those images not used in construction were badly represented — having a high residue error. When all images were used to make the eigenmodel the overall fit was much improved: those images in the previous subset were slightly less well represented — but those not in that subset were much better represented.

This result confirms that EVD (and SVD) models do not generalise well. Classification results follow a similar trend: each image is better classified by an eigenspace that uses all images. We conclude that eigenmodels should always be constructed from as much data as possible, and in some cases incremental methods provide the only option for this.

We now turn our attention to applications of a more substantive nature.

B. A security application

Here we aim to show that our methods are useful in classification applications. This is because we update the mean, and many statistical computations that are commonly used in classification (such as the Mahalanobis distance) require an accurate mean.

We consider a security application based on identification. The scenario is that of a company wishing to efficiently store photographs of its thousands of employees for security reasons, such as admitting entry to a given building or a laboratory. We chose to the store the data using an eigenmodel — the images can be projected into the eigenmodel and stored with tens rather than thousands of numbers. Conventional batch methods cannot be used to make the eigenmodel because not all images can fit into memory at once. Additionally, the database requires changing each year, as employees come and go.

Our methods allow the eigenspace to be constructed and maintained. An initial eigenmodel is constructed by building several eigenspaces, each as large as possible, and merging them: the data is too large to do otherwise. Thereafter, the eigenmodel can be maintained by simply merging or splitting eigenmodels as required. (Note that splitting means removal of images from the database.)

We illustrate this with the data base of faces used previously. We constructed an eigenmodel from a selection of 21 people, there being 10 photographs for each person. To recognise an individual a new photograph was given a "weight of evidence" between 0 (not in the database) and 1 (in the database). To compute this weight we used the maximum Mahalanobis distance (using Moghaddam and Pentland's method [6]) of all photographs used to construct the eigenmodel. Each new photograph was then judged as *in* if its Mahalanobis distance was less than this maximum. 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.

We recognise this as a rather crude measure, but its merits are two fold: first it provides an economic alternative to extensive image processing (aligning faces, segmenting shape from texture, and so on); second this measure is sufficient for use to demonstrate that we can update image databases for classification using *some* measure — and this is our aim here.

We initialised the eigenmodel with the first twenty-one people (200 images). We then made a change by adding the twenty-second person and removing the first — arbitrary but convenient choices. Figure 9 show the "weight of evidence" measured after this change. The upper plot shows the measure for the images against a batch model. The lower plot shows the same measure for the same images. We notice that both models produce some false positives in the sense that some people who should not be classified as in have a weight larger than zero. We notice too that the incrementally computed eigenspace gives rise to more false positives than the eigenmodel computed via batch methods — in line with earlier observations on subtraction. However, the weight-of-evidence factor is less than one in every case, no matter how the eigenmodel was computed, and this fact (or some other more sophisticated test and pre-processing) could be used to eliminate false positives — but the point here is not to develop a fully operational and robust security application but to demonstrate the potential of our methods in classification.

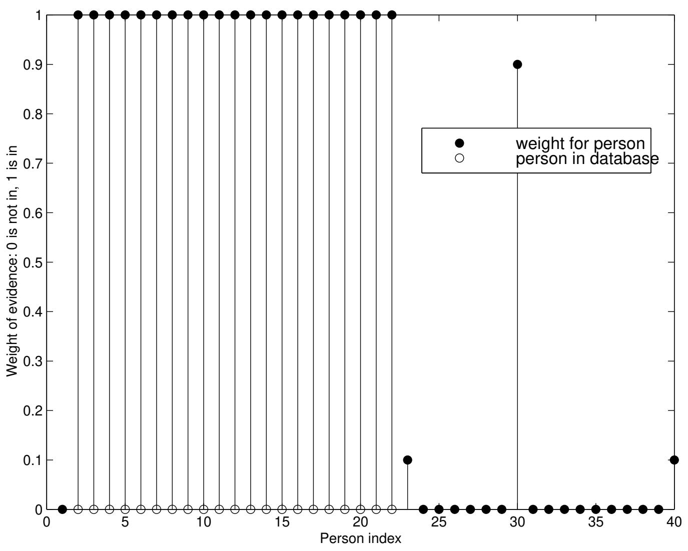
We conclude that *additive* incremental eigenanalysis is safe for classification metrics, but that *subtractive* incremental eigenanalysis needs a greater degree of caution.

C. Dynamic Gaussian mixture models

We are interested in using our methods to construct dynamic Gaussian mixture models (GMM's). Such models are increasingly common in the vision literature, and a method for their dynamic construction would be useful. The methods presented in this paper make this possible. We note that block updating and maintainance of the mean are prerequisites for dynamic GMMs.

Here we focus on merging existing GMMs, and show how to construct a dynamic GMM from a library of photographs. Our aim here is not to discuss the issues surrounding dynamic GMMs in full, for that would unduly extend this paper, but instead we seek to demonstrate that dynamic GMMs are feasible using our methods.

We partition data into sets, and for each set construct a GMM as follows: first use all the data in a set to build an eigenmodel (using incremental methods if necessary). Second project each datum in the set into the eigenmodel. Thirdly, construct a GMM from the projected



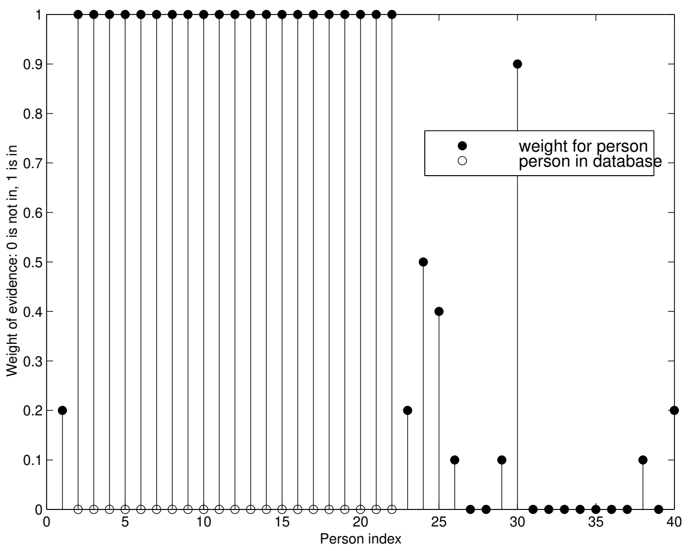


Fig. 9. Weight of evidence measures after a change: batch (top), incremental (below).

data, using the EM algorithm for this [13]. Finally, represent each Gaussian in the mixture using an eigenmodel. Hence, each GMM is a hierarchy of eigenspace models. In this regard they are similar to a hierarchy of models proposed to improve the specificity of eigenmodels [11]. No two Gaussians need have the same dimension.

To merge GMMs we first merged their base eigenspaces. Second we transformed all dependent eigenmodels from each previous model into the new basis eigenspace. Finally we merged those new dependent eigenspaces that were sufficiently close. We found that a simple volume measure to be adequate for most cases. The volume of a hyperellipse with semi-axes $\underline{A}$ (each element the square root of an eigenvalue), of dimension M, and at characteristic radius s (square root of the Mahalanobis distance) is

$$\frac{s^M|\underline{A}|\pi^{M/2}}{\Gamma(\frac{M}{2}+1)}$$

We permanently merged a pair of eigenmodels in the GMM if the sum of their individual volumes was greater than their volume when merged. We found this works well enough, dimensionality problems notwithstanding.

As an example, we used photographs of two distinct toys, each photographed at 5 degree angles on a turntable. Hence we had 144 photographs. Examples of these photographs can be seen in Figure 10. The photographs were input in four groups of thirty-six photographs. For each group we made an eigenmodel, projected the photographs into the eigenmodel, and used these projections to construct a GMM of eighteen clusters. The Gaussians making up the mixture were represented by an eigenmodel. Hence we had four GMMs, which we wanted to merge into a large GMM.

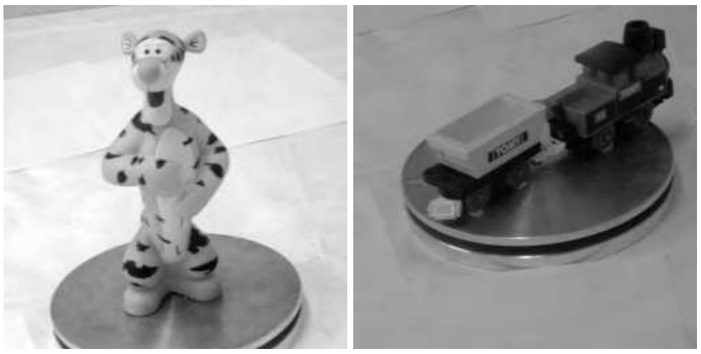


Fig. 10. Example images of each toy.

To merge the GMMs we first added added together the four eigenspaces to make a complete eigenspace. Next we transformed each of the GMM clusters into this space, thus bringing the *ensemble* of clusters into a common space. Each Gaussian cluster in the mixture model in the new space was represented by an eigenmodel. We then merged the cluster, pairwise, using our volume criterion. Hence we were able to reduce the number of Gaussians in the mixture to twenty-two.

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 Figure 11, where the two toys can be clearly seen in different positions. These clusters may be used to identify the toy and its pose, for example. (Murase and Nayar [20], and Borotschnig *et. al* [21] recognise pose using eigenmodels.) In addition, we found a few clusters occupying the space "in between" the two toys — an example of which is seen in Figure 11. This artifact of clustering appears to derive from the high dimensionality of the space that the clusters are in, rather than being a side-effect of our method. Notice that these clusters might in future be removed because no picture matches well against them.

We conclude from these experiments that dynamic GMMs are a feasible proposition using our methods. We note that the ability to merge complete spaces while updating the mean are prerequisites of dynamic GMMs. Also, dynamic GMMs are likely to be an important application and deserve further attention.

VII. Conclusion

We have shown that merging and splitting eigenspace models is possible, allowing sets of new observations to be processed as a whole. The theoretical results are novel, and our experimental results show that the methods are wholly practical, computation times are feasible and often advantageous compared to batch methods. Batch and incremental eigenspaces are very similar so performance characteristics, such as residue error, differ little. Our methods are useful in many applications, and we have illustrated a few of a general nature.

We have concluded that the merging of eigenspaces is stable and reliable, but advise caution when splitting. Thus splitting is the principle weakness of our methods and it is interesting to ask whether the process can be made more reliable.

We should point to several omissions from this work, each providing an avenue for further work. We have not performed analytic error analysis, relying instead on experiment. Most of the errors arise from discarding eigenvectors and eigenvalues. To the best of our knowledge the work in unique, and so we have been not compared our method to others. However, in a previous paper we considered the inclusion

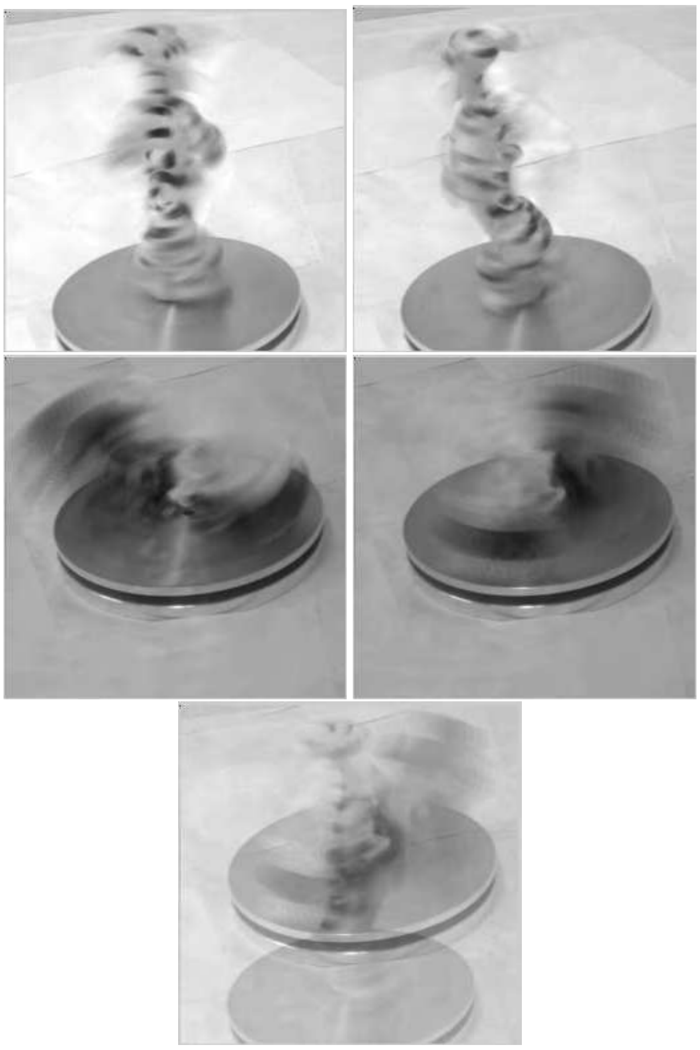


Fig. 11. Dynamic Gaussian Mixture Models. The top line shows 2 examples of the original 144 photographs. Below are 5 examples of the 22 cluster centres. These are arranged to show clusters for each toy, and the space between them.

of a single new datum, and were able to make comparisons [14]. The conclusion there was that SVD tends to be more accurate, but that updating the mean is crucial for classification applications. We note in this paper we have demonstrated that adding one datum each time is much less accurate than adding complete spaces. We have omitted our derivation for block update/downdate of SVD, with change of mean, for want of space. However, we have indicated that downdating SVD seems to require an EVD step. We have presented general applications rather than become embroiled in any particular application, which allows us to highlight important generic applications.

We would expect our methods to find much wider applicability than those we have mentioned; updating image motion parameters [8], selecting salient views [3], and building large image databases [3] are two applications that exist already. We now use our methods routinely to construct eigenmodels that would be impossible by any other means, and this has allowed us to experiment with image compression methods. Also, we have experimented with image segmentation, building models of three-dimensional blood vessels, and texture classification. We believe that dynamic Gaussian mixture models provide a very interesting future path.

References

[1] J. R. Bunch, C. P. Nielsen, and D. C. Sorenson. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31:31–48, 1978.

[2] J. R. Bunch and C. P. Nielsen. Updating the singular value decomposition. *Numerische Mathematik*, 31:111–129, 1978.

[3] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkler, and H.Zhang. An eigenspace update algorithm for image analysis. *Graphical Models and Image Processing*, 59(5):321–332, September 1997.

[4] R. D. DeGroat and R. Roberts. Efficient, numerically stablized rank-one eigenstructure updating. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 38(2):301–316, February 1990.

[5] H. Murakami and B.V.K.V Kumar. Efficient calculation of primary images from a set of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 4(5):511–515, September 1982.

[6] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):696–710, July 1997.

[7] C. Nastar and N. Ayache. Frequency-based nonrigid motion analysis: application to four dimensional medical images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(11):1067–1079, November 1996.

[8] S. Chaudhuri, S. Sharma, and S. Chatterjee. Recursive esitmation of motion parameters. *Computer Vision and Image Understanding*, 64(3):434–442, November 1996.

[9] Peter Hall, Milton Ngan, and Peter Andreae. Reconstruction of vascular networks using three-dimensional models. *IEEE Transactions on Medical Imaging*, 16(6), December 1997.

[10] P.M. Hall. Drawing by example. In *16th Eurographics UK conference*, pages 159–167, Leeds, 1998.

[11] T. Heap and D. Hogg. Improving specificity in pdms using a heirarchical approach. In *Pooc. British Machine Conference*, pages 80–89, 1997.

[12] G. E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Trans. on Neural Networks*, 8(1):65–74, January 1997.

[13] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[14] To be included.

[15] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Journal Optical Society of America, A*, 4(3):519–524, March 1987.

[16] Y.T. Chien and K.S. Fu. On the generalised karhunen-loeve expansion. *IEEE Trans. on Information Theory*, IT-13:518–520, 1967.

[17] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins, 1983.

[18] M. Gu and S.C. Eisenstat. A stable and fast algorithm for updating the singular value decomposition. Technical Report YALE/DCS/RR-996, Department of Computer Science, Yale University, 1994.

[19] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Training models of shape from sets of examples. In *Proc. British Machine Vision Conference*, pages 9–18, 1992.

[20] Hi. Murase and S. K. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[21] H. Borotschnig, L.s Paltta, M.d Prantl, and A. Pinz. Active object recognition in parametric eigenspace. In *British Macine Vision Conference*, pages 629–638, 1998.