Adding and subtracting eigenspaces

Peter Hall*, David Marshall†, Ralph Martin†
* School of Mathematical Sciences, Bath University
† Department of Computer Science, Cardiff University
peter,dave,ralph@cs.cf.ac.uk

Abstract

This paper provides two algorithms; one for adding eigenspaces, another for subtracting them, 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. We illustrate the use of our algorithms in three generic applications, including the dynamic construction of Gaussian mixture models.

1 Introduction

This subject of this paper is incremental eigenanalysis: we provide an algorithm for including new data into an eigenspace, and another for removing data. An eigenspace comprises: the number of data points, their mean, the eigenvectors, and the eigenvalues that result from the eigenvalue decomposition (EVD) of the data covariance matrix. Typically the eigenspace is *deflated*, which is to say that only "significant" eigenvectors and eigenvalues 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. Our methods are presented in Section 2.

We must make clear the difference between *batch* and *incremental* methods for computing eigenspace models. A batch method computes an eigenmodel using all observations simultaneously. An incremental method computes an eigenspace model by successively updating an earlier model as new observations become available. In either case, the observations used to construct the eigenspace model are the *training observations*; that is, they are assumed to be instances from some class. This model may then used to decide whether further observations belong to the class.

Incremental eigenanalysis has been studied previously [1, 2, 3, 4, 7, 13], but surprisingly 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. Only our previous work allows for a change of mean [9], where we allowed for the inclusion of only a single new datum. In contrast, our algorithms here handle block update and downdate, so many datum can be included

BMVC 1999 doi:10.5244/C.13.45

or removed in a single step. They explicitly allow the mean to vary in a principled and accurate manner, and this is important. Consider, for example, that functions such as the Mahalanobis distance, often used in classification applications, cannot be computed without the mean; previous solutions cannot be used in this case.

Applications of incremental methods are wide ranging both within computer vision and beyond. Focusing on computer vision, applications include: face recognition [12], modelling variances in geometry [6], and the estimation of motion parameters [4].

Our motivations for this work arose from several sources, the construction of classification models for many images — too many to fit all of them into memory at once — for example. Intuition, confirmed by experiment, suggests it is better to construct the eigenspace model from all the images rather than a subset of them as batch methods would allow; 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 kept in memory at once. Our methods allow the database to be updated and downdated on a year by year basis without recomputing the eigenmodel *ab initio*. (In any case, there may be so many images that batch methods cannot be used — incremental methods are then necessary.) An example security application appears in Section 3.

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 are beyond the scope of the paper, and are 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].

2 Adding and subtracting eigenspaces

We now state the problems which are our subject.

Let $X = [x_1, \ldots x_N]$ be a collection of N data points, each n dimensional. The eigenmodel of these data is

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

in which: $\mu(X)$ is their mean; $U(X)_{np}$ is a collection of p eigenvectors, each a column in an $n \times p$ matrix; $\Lambda(X)_p$ is a collection of p eigenvalues, one per eigenvector, and N is the number of data points. The subscripts on each element identify its size, where necessary.

Typically $p \leq \min(n, N)$ is the rank of the covariance matrix of X, though this depends on details of how the model is deflated (see our previous work [9] for a discussion). We call p the dimension of the eigenspace. We have that $U^T U = I$, but usually $UU^T \neq I$, so the eigenvectors support a subspace of dimension p in a space of dimension n. The eigenvalues measure the standard deviation of the data over each of the eigenvectors, under the assumption that the data are Gaussian distributed. Hence $\Omega(X)$ may be regarded 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.

Another collection of observations $Y = [y_1, \ldots y_M]$ has an eigenmodel

$$\Omega(Y) = (\mu(Y), U(Y)_{nq}, \Lambda(Y)_q, N(Y)) \tag{2}$$

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 is to compute the eigenmodel for the pair of collections $Z = [X, Y]$

$$\begin{aligned} \Omega(Z) &= (\mu(Z), U(Z)_{nr}, \Lambda(Z)_r, N(Z)) & (3) \\ &= \Omega(X) \oplus \Omega(Y) & (4) \end{aligned}$$

with reference to $\Omega(X)$ and $\Omega(Y)$ only; that is, define the algorithm for the $\oplus$ 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 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 eigenmodel in Z. As in the case of addition, a possible change in the dimension of the eigenspace must be accounted for.

This paper has space only to present the solutions to these problems, and derivations are available elsewhere [10].

2.1 Addition

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

$$\begin{aligned} N(Z) &= N(X) + N(Y) & (6) \\ \mu(Z) &= (N(X)\mu(X) + N(Y)\mu(Y))/N(Z) & (7) \end{aligned}$$

Computing eigenvectors and eigenvalues depends upon properties of the subspaces that the eigenvectors $U(X)$, $U(Y)$, and $U(Z)$ support; properties we describe next.

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 $U(X)^T U(Y) \neq 0$. The null space of each of $U(X)$ and $U(Y)$ may contain some component of the other, that is to say $H = U(Y) - U(X)(U(X)^T U(Y)) \neq 0$. Both of these conditions are illustrated in Figure 1. Furthermore, even if $U(X)$ and $U(Y)$ are each a basis for the same subspace, $U(Z)$ could 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 Figure 1.

Putting (temporarily) to one side issues relating to changes in dimension, adding data acts to rotate the eigenvectors and scale the eigenvalues. Hence, the new eigenvectors must be a linear combination of the old. When adding, we deal with a change in dimension by constructing a basis sufficient to span $U(Z)$, for which we use $U(X)$, and a basis,

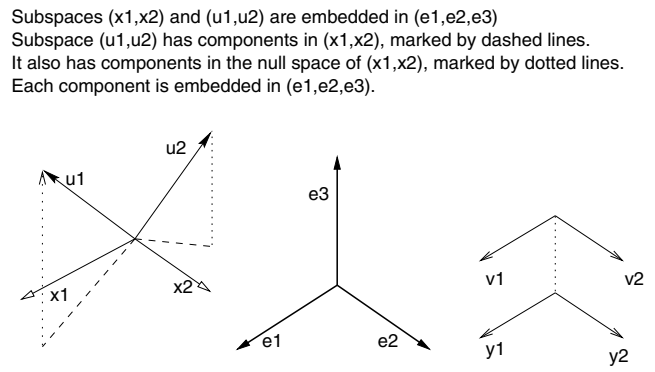

Figure 1: An illustration of relationships between subspaces embedded in a larger space.

ν that spans $[H, h]$, which is in the null space of $U(X)$ with respect to $U(Z)$. We have $U(Z) = [U(X), \nu]R$ where R is an orthonormal (rotation) matrix to be found by solving the following eigenproblem:

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

in which

$$\begin{aligned}
g_p &= U(X)^T(\mu(X) - \mu(Y)) && (9)\\
G_{pq} &= U(X)^T U(Y) && (10)\\
H_{nq} &= [U(Y) - U(X)G_{pq}] && (11)\\
h_n &= (\mu(X) - \mu(Y)) - U(X)g_p && (12)\\
\nu_{nt} &= \mathsf{Orthobasis}(\zeta[H_{nq}, h_n]) && (13)\\
\Gamma_{tq} &= \nu_{nt}^T U(Y)_{nq} && (14)\\
\gamma_t &= \nu^T(\mu(X) - \mu(Y)) && (15)
\end{aligned}$$

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 orthogonalisation is used to compute significant support vectors, ν from $\zeta[H, h]$; these are "outside" the eigenmodel $\Omega(X)$. [8]. Note that while $\nu^T\nu = I$, $\nu\nu^T \neq I$. 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 ν vectors). This complementary space must be computed to compute the new eigenspace $\Omega(Z)$, which argues

in favour of adding and subtracting eigenspace, rather than direct updating or downdating of data blocks.

Given the above decomposition, we can complete our computation of $\Omega(Z)$:

$$\Lambda(Z)_s = \text{diag}(\Pi_{ss}) \quad (16)$$

$$U_{ns}(Z) = [U_{np}\nu_{nt}]R_{ss} \quad (17)$$

The eigenmodel can then be deflated, if desired, to dimension $r \leq s$.

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. 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.

2.2 Subtraction

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

$$N(X) = N(Z) - N(Y) \quad (18)$$

$$\mu(X) = (N(Z)\mu(Z) - N(Y)\mu(Y))/N(X) \quad (19)$$

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

$$\frac{N(Z)}{N(X)}\Lambda(Z)_{rr} - \frac{N(Y)}{N(X)}G_{rp}\Lambda(Y)_{pp}G_{rp}^T - \frac{N(Y)}{N(Z)}g_r g_r^T = R_{rr}\Lambda(X)_{rr}R_{rr}^T \quad (20)$$

where $G_{rp} = U(Z)_{nr}^T U(X)_{nq}$ and $g_r = U(Z)_{nr}^T(\mu Y - \mu X)$. 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:

$$\begin{aligned} R_{rr}\Lambda(X)_{rr}R_{rr}^T &= [R_{rp}R_{rt}] \begin{bmatrix} \Lambda(X)_{pp} & 0_{pt} \\ 0_{tp} & 0_{tt} \end{bmatrix} [R_{rp}R_{rt}]^T \\ &= R_{rp}\Lambda(X)_{pp}R_{rp}^T \end{aligned} \quad (21)$$

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:

$$U(X)_{np} = U(Z)_{nr}R_{rp} \quad (22)$$

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

2.3 Some comments on the solutions

Previously we presented a method for adding a single point, x, to an eigenspace [9]. It can be shown [10] that the previous result is a special case of the above addition, with $(x, 0, 0, 1)$ as either operand. In terms of its outcome the above addition is both commutative and associative (provided that in practice we allow for numerical errors, especially

in association). The null eigenspace $(0, 0, 0, 0)$ is an additive identity. As $N(X) \to \infty$ so the effect of adding $\Omega(Y)$ becomes negligible, and vice-verse. 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 eigenvalue decomposition. Our experiments [10] demonstrate that time is proportional to the cube of s, the size of the eigenproblem to be solved (we used a proprietary implementation). We also found that the time to compute 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 new eigenspace to an existing one rather than compute the large eigenspace using all the original data. 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.

We have compared the angular deviation of eigenvectors, change in eigenvalues, accuracy of data representation in the least-squares sense, and classification performance [10]. The incremental methods for adding generally compare very well with batch methods, with discrepancies being a minimum when the two eigenspaces added are of about the same size; the exception is the discrepancy in eigenvalues, which shows a maximum of about one part in 10^5 at that point. Reasons for this behaviour are the subject of future work — we have not yet undertaken a rigorous analysis of errors.

The subtraction operator tends to instability as the number of points being removed rises, since in this case $N(X) \to 0$, hence $1/N(X) \to \infty$. 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.

3 Applications

An obvious application of our methods is to build an eigenspace for many images — too many to all at once fit into memory. We ran a simulation of this by building two eigenmodels: 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. However, as more images were used in the construction, so the maximum residue error for each image rises — but not 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 substantive 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. This problem 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 will proceed in a very simple way: construct an eigenmodel of all those people who are to be recognised, and rely on the fact that eigenmodels do not generalise well to new instances exclude others. To allow for changes in pose, expression, and so on, we will use several images of each individual.

Conventional batch methods cannot be used to construct an eigenmodel because not all images can fit into memory at once, so incremental methods are a pre-requisite to our approach. Given that the database is subject to change we could reconstruct an eigenmodel at each change, but we will use our incremental methods to effect the changes more efficiently; for which subtraction is required.

We will use the Olivetti database of 400 faces [14] as our group of students. We constructed an eigenmodel from a selection of 20 people, there being 10 photographs for each person. Each person in the entire database was then give 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 [12]) of any photograph used in constructing the database. Each photograph was then judged 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.

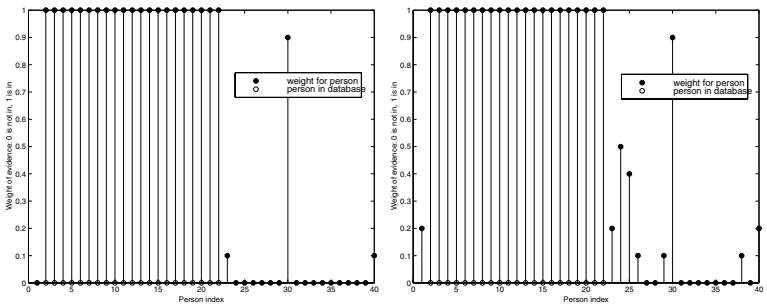

Figure 2: Weight of evidence measures: year 2 batch (left), and year 2 incremental (right).

Figure 2 show the "weight of evidence" measure for the second year our hypothesised database has been running. 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 1 by including new and removing old students.

We notice that both models produce some false positives in the sense that some people who should not be classified an in have a weight larger than zero. We notice 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.

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. Gaussian mixture models are useful in 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]. To construct a hierarchy we fist make a eigenmodel, then project all data into it to reduce dimensionality, construct a GMM using the project data, and represent each mixture as an eigenmodel. Thus, each Gaussian in mixture can be thought of as a hyperellipse, and each may have a unique dimension. The problem here is to merge two such GMMs.

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 3. The photographs were input in four groups of thirty-six photographs.

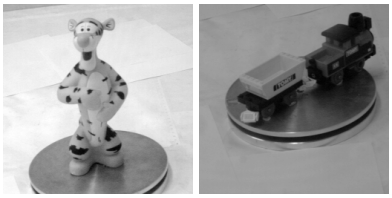

Figure 3: Sample images of each toy used as source data in our dynamic GMM application.

To merge the GMMs we first added added together the four eigenspaces to make a complete eigenspace. Next we transformed the GMM clusters into this space, thus bringing each of the thirty-six GMMs (eighteen from individual eigenspace) into the same eigenspace and covering the *ensemble* of data. As before, each Gaussian in the mixture model was represented by an eigenmodel. Hence, we were able to reduce the number of total Gaussians in the mixture by merging eigenmodels. We used a very simple criteria to merge based on reducing volume of hyperellipses.

We thus generated a final GMM with a total of twenty-two Gaussians. 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 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 Figure 4.

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.

Figure 4: Dynamic Gaussian Mixture Models, 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).

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.

Having conducted experiments comparing eigenspaces and some of their performances metrics, and (reasonably generic) applications, 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 — any statistical method may be misapplied. Especial care should be taken when subtracting eigenspaces: the way in which the results are to be used impacts on efficacy.

We should point to the 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 (in terms of approximating the "batch" version) are anecdotal in character. We have not compared our method to singular value decomposition (SVD) techniques. Thus far we have been able to block update SVD models, at the expense of keeping all the right singular values — but have been unable to find a solution for block downdating, except via EVD. Presenting our SVD solution, explaining why block downdating is not directly possible, and comparing EVD with SVD would unduly extend this paper. We have not fully worked through any particular application as yet and so can make general recommendations only. We have also omitted comparisons with other incremental methods: most deal with adding one new data point. However, in a previous paper, dealing with that very issue, we presented experimental evidence to show that updating the mean is crucial for classification results [9].

We would expect our methods to find much wider applicability than those we have mentioned, updating image motion parameters [4], selecting salient views [3] are two applications that exist already. 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 for it enables useful

representations [5, 11] — and all their attendant properties — to be brought into a dynamic framework.

References

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

[2] J. R. Bunch, C. P. Nielsen, and D. C. Sorenson. Rank-one modification of the symmetric eigenproblem. *Numerische Mathematik*, 31:31–48, 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] S. Chaudhuri, S. Sharma, and S. Chatterjee. Recursive esitmation of motion parameters. *Computer Vision and Image Understanding*, 64(3):434–442, November 1996.

[5] T.F. Cootes and C.J. Taylor. A mixture model for representing shape variations. In *Proc. British Machine Vision Conference*, pages 110–119, 1997.

[6] 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.

[7] R. D. DeGroat and R. Roberts. Efficient, numerically stablized rank-one eigenstructure updating. *IEEE Transactions on acoustics, speech, and signal processing*, 38(2):301–316, February 1990.

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

[9] P. Hall, D. Marshall, and R. Martin. Incrementally computing eigenspace models. In *Proc. British Machine Vision Conference*, pages 286–295, Southampton, 1998.

[10] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspaces. Technical Report CS–98002, Deptartment of Computer Science, Cardiff University of Wales, 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] 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.

[13] 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.

[14] The Olivetie face database at `http://www.cam-orl.co.uk/facedatabase.html`