ISSN 1369-1961

Department of Computer Science

Technical Report No. 739

Modelling Face Space with Gaussian Mixtures using Modified K-Means

Peter Loft & Shaogang Gong

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

July 1997

Technical Report CS-739, ISSN 1369-1961, Department of Computer Science, Queen Mary and Westfield College

Modelling Face Space with Gaussian Mixtures using Modified K-Means*

Peter J. Loft and Shaogang Gong

Department of Computer Science, Queen Mary and Westfield College

E-mail : {pjl,sgg}@dcs.qmw.ac.uk

ABSTRACT

Contrary to the conventional assumption that object representations for recognition are in 3D and viewpoint invariant, most psychophysical experiments have shown consistently that recognition is viewpoint dependent and only partially aided by stereo and depth. In particular, it has been shown that we do not need to understand the 3D shape in order to recognise human faces, although shape information is important for gender discrimination. In this work, we investigate view-based representation schemes for modelling face space. In particular, a method was exploited that first approximates the non-linear face pose space with a set of linear pose subspaces using Gaussian mixtures and second, learns to recognise faces from image sequences of head rotations in depth using networks of Hyper Basis Functions. We conclude with supporting experimental results and discussions on future work.

1 Introduction

Much of the work in face recognition has been on static images at or near frontal view taken under constrained conditions. This is quite different from the human system in which people are identified in cluttered dynamic scenes at variety of pose angles caused by head rotations in depth. In this work, we investigate computationally efficient and plausible solutions to the problem of face recognition under head rotations in depth. More specifically, we exploit a method that first approximates the non-linear face pose space with a set of linear subspaces using Gaussian mixtures and second learns to recognise faces from image sequences of head rotations in depth using networks of Hyper Basis Functions. The work uses results from previous work on real-time detection, tracking and normalisation of face images (see Figure 2 and Figure 1) in a given dynamic scene [15, 21, 22].

If robust face recognition is to be performed, it has been commonly assumed that face models must exhibit invariance under changes in a variety of conditions. Whereas illumination, scale, translational and small image plane rotations can be dealt with to some extent through a process of normalisation, rotations in depth (changes in pose) cannot be so easily dealt with. Head rotations in depth introduce non-linear transformations in the image plain causing difficulties in recognition by correspondence. A popular approach to the problem is to find a view invariant representation scheme such as building a full scale 3D face model [6].

* The research was in part funded by a QMW Computer Science Bursary Scheme.

Figure 1: *A girl is tracked for recognition as she approaches the camera. Bounding boxes for the Kalman filtered motion cluster and the associated tracked face region are shown overlaid on every* 15^{th} *frame. The face is shown centred and normalised in scale every* 5^{th} *frame.*

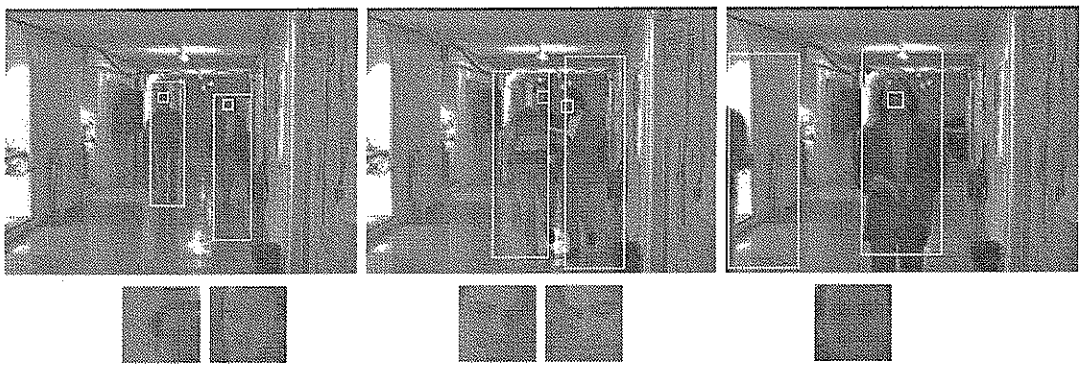

Figure 2: *Two people are tracked as they walk along a corridor.*

However, contrary to the conventional assumption that object representations for recognition are in 3D and viewpoint invariant, most psychophysical experiments have shown consistently that recognition is viewpoint dependent and only partially aided by stereo and depth [8]. In particular, it has been shown that we do not need to understand the 3D shape in order to recognise human faces, although shape information is important for gender discrimination [7]. A novel and increasingly important concept using view-based representation is

supported by recently suggested models for recognition based on view interpolation [34].

If a view invariant 3D model is computationally unnecessarily complex and expensive for recognition, on the other hand, a view-based 2D representation relying on a set of views has been shown to be computationally far more efficient [1, 10, 27, 28, 32, 33]. However, despite recent work on view-based representation, the issue of modelling face space for robust recognition under large head rotations in depth is still largely unaddressed [3, 16, 17].

Section 2 introduces the non-linear Gaussian mixture method and HBF network together with the supervised method to determine the final layer weights of the network. In section 3, we first describe a modified K-means algorithm used to estimate the parameters of the Gaussian Mixture model and second present an representation scheme used to form the input to the neural network. Section 4 describes the data set used and a number of experiments performed. Section 5 concludes with a discussion and describes further work and areas for improvement.

2 Hyper Basis Function Networks

A Hyper Basis Function (HBF) network is a two layer hybrid class of feedforward network as shown in Figure 3, in which the hidden layer consists of basis functions of Gaussian clusters[1] with each function having an associated cluster mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The hidden layer acting as a Gaussian mixture model used to model the density distribution of the input data.

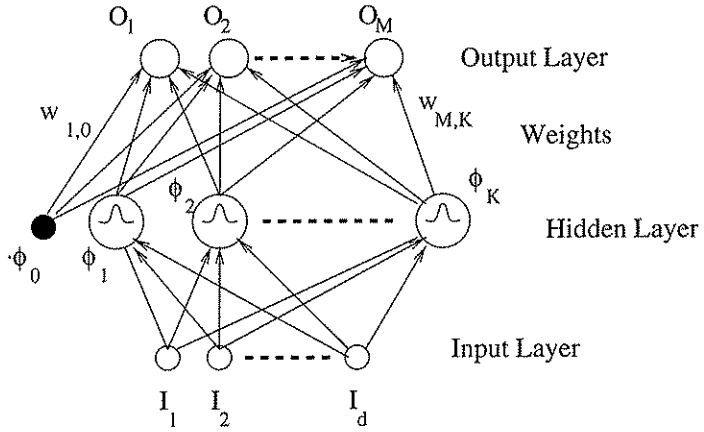

Figure 3: An HBF network with d inputs and $K+1$ hidden units, K of which are Gaussian basis functions ϕ_k. The additional one is a bias unit (in black). The number of outputs can include the number of known face classes plus gender, unknown and non-face classes.

The first layer of weights (i.e. the parameters of the basis functions) are obtained through unsupervised training while the final layer of weights through supervised learning. This method of network training is particularly attractive especially for face recognition where large quantities of labelled data may be unavailable. Hence the hidden layer can therefore be trained using large quantities of unlabelled data and then having set the parameters of the basis functions the final layer of weights can be trained using a smaller quantity of labelled data. In addition such a network is faster to train than a typical two layer feedforward network.

[1]Basis functions, hidden units and clusters will be used interchangeably.

2.1 Gaussian Mixture Models for Face Space

A Gaussian mixture model is used to model the density distribution of the input vectors for a HBF network. This is achieved through the linear superposition of Gaussian clusters placed in the input vectors subspace, with the basis functions associated with hidden unit k taking the form:

$$\phi_k(\mathbf{x}) = \exp\left[-\frac{1}{2}\Delta^2(\mathbf{x})\right] \tag{1}$$

where

$$\Delta^2(\mathbf{x}) = (\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) \tag{2}$$

is the *Mahalanobis* distance (M-distance) from $\mathbf{x}$ to $\boldsymbol{\mu}_k$. For the problem at hand, a set of generalised Gaussian density functions can be established in such a way that each function is tuned towards a specific pose subspace that is approximately linear. Therefore, for face recognition in a non-linear face space induced by head rotation in depth, the Gaussian mixture model captures the underlying non-linear pose distribution before a HBF network is trained to perform face recognition in each linear pose subspace independently. A similar approach was employed for face detection in static images [31].

An attempt to perform face recognition in such a non-linear face space was proposed by Howell and Buxton [16] using networks of Radial Basis Functions (RBF) [20] in which each hidden unit is assigned to a specific training face vector whilst the output units provide simple linear weighting. However, this type of RBF network can at best perform exact *interpolation* on the training data. There was no attempt to model the non-linear face space and the essence of the system is based on memorising given face image examples in the training set with a degree of radial Gaussian overlapping. There are a number of problems with such an approach. Firstly, since the number of hidden units equals the number of training face vectors[2], the network is expensive to extend to the large training sets required for any adequate modelling of the non-linear face space. Secondly, due to the presence of noise and the constraints on the number of face images that can be practically used to capture the non-linear face space of all poses, such a network would always give poor generalisation [5].

2.1.1 Gaussian Parameter Estimation

The nature of the covariance matrix determines the type of Gaussian for a mixture model. The more flexible the Gaussian the less number of basis functions are required to model the density distribution of the face space. However, the number of parameters required to be determined will be greater which introduces a number of problems that will be discussed later in section 3. Three types of Gaussian cluster are shown in Figure 4. A number of techniques have been proposed for estimating parameters of a Gaussian mixture model [5, 24, 29, 31]. Two commonly used methods are:

1. K-Means clustering
2. Expectation maximisation (EM) clustering

[2]With appearance-based approaches, we also refer to face images as face vectors in the hyper-dimensional image space.

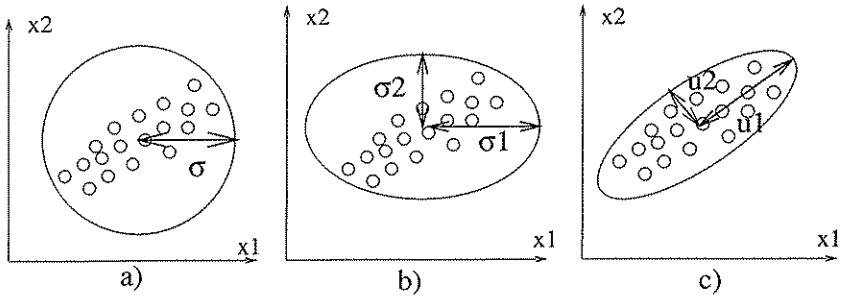

Figure 4: Gaussian clusters with three types of covariance matrix: (a) A scaled identity radial covariance matrix σI, (b) A diagonal covariance matrix $\texttt{diag}[\sigma_1, \sigma_2]$ where σ_1 denotes the variance of the cluster in the first dimension (x1) and σ_2 the variance of the cluster parallel with the second dimension (x2) and, (c) A full covariance matrix where u1 is the first eigenvector with eigenvalue λ_1. The variance of the Gaussian along this eigenvector is equal to $\sqrt{\lambda_1}$. Similarly, u2 is the second eigenvector and the variance along this direction is $\sqrt{\lambda_2}$.

2.1.2 Traditional K-Means Algorithm

The K-Means algorithm originally introduced by Moody and Darken [25] has been commonly adopted for Gaussian cluster parameter estimation. The algorithm works by partitioning the data $\{\mathbf{x}^n\}$ into K disjoint subsets in order to minimise the following function:

$$J = \sum_{j=1}^{K} \sum_{n \in C_j} \| \mathbf{x^n} - \mu_j \|^2 \tag{3}$$

where $\boldsymbol{\mu}_j$ is the mean of the data in the jth clusters C_j. The algorithm is an iterative procedure after which the covariance matrices of the basis functions are set based upon the number of data assigned to it, as follows:

1. Choose the number of Gaussian clusters to be K.
2. Collect X training face vectors from sequences under head rotations in depth.
3. Initialise the mean vector $\boldsymbol{\mu}_k$ of each of K clusters to be a randomly chosen face vector x_i in set X.
4. For each vector $\mathbf{x}_i$ in X, assign it to cluster C_j where a $dist(C_j, \mathbf{x}_n)$ measure, e.g. Euclidean, is the least so that
$$\forall k. \, dist(C_j, \mathbf{x}_i) < dist(C_k, \mathbf{x}_i), k \in \{1, ..K\}, j \neq k, \tag{4}$$
5. Each cluster is assigned a set of face vectors where its new mean is calculated from.
6. Repeat 4. until either there is no change in the means' positions for all the clusters or if a predefined number of iteration has been reached.
7. Calculate the covariance matrix of each cluster based on the number of face vectors N_k assigned to it
$$\sigma_k = \frac{1}{N_k - 1} \sum_{x \in C_k} (\mathbf{x}_k - \boldsymbol{\mu}_k)^2$$

2.1.3 The EM Algorithm

A problem encountered with the K-Mean algorithm is that the clusters are disjoint, i.e. each cluster can only estimate its parameters using the face vectors closest to it. In face space, it is more likely that certain pose clusters overlap and therefore they should be able to "share points". An alternative approach therefore is not to explicitly restrict the data available for estimating each individual cluster. The Expectation Maximisation (EM) algorithm [9] is a means to achieve this. This method is essentially an iterative method for monotonically reducing the negative log likelihood of a given data set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n, \ldots, \mathbf{x}_N\}$

$$-\ln L = \sum_{n=1}^{N} \ln p(\mathbf{x}_i)$$

where $p(\mathbf{x})$ is the *mixture distribution* formed by a linear combination of the probability density estimates of K clusters in the mixture model and is given by

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(x|k)P(k) \tag{5}$$

where $P(k)$ is the prior probability of the k^{th} Gaussian. With the EM algorithm, it is common to first use the K-Means method to initialise the means and covariance of the clusters in order to speed up the clustering process. Then, iterative approximation of the optimal clusters' parameters are performed with the following update rules: Given K clusters in a mixture model, whilst σ_i^k is the i^{th} diagonal[3] element of the covariance matrix of the k^{th} cluster, x_i^n is the i^{th} component of $\mathbf{x}_n$ and, μ_i^k is the i^{th} component of $\boldsymbol{\mu}_k$, then

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{n=1}^{N} \mathbf{x}_n \, P^{old}(k|\mathbf{x}_n)}{\sum_{n=1}^{N} P^{old}(k|\mathbf{x}_n)} \tag{6}$$

$$\sigma_i^{k\ new} = \frac{\sum_{n=1}^{N} (x_i^n - \mu_i^k) \, P^{old}(k|\mathbf{x}_n)}{\sum_{n=1}^{N} P^{old}(k|\mathbf{x}^n)} \tag{7}$$

$$P(k)^{new} = \frac{1}{N} \sum_{n=1}^{N} P^{old}(k|\mathbf{x}_n) \tag{8}$$

and the probability of a given vector $\mathbf{x}$ falling into the k^{th} cluster is

$$P(k|\mathbf{x}) = \frac{p(\mathbf{x}|k)P(k)}{p(\mathbf{x})} \tag{9}$$

For the use of "new" parameters above, the new values are calculated in the following order: mean, covariance and prior probability.

[3]This particular implementation of the algorithm is based upon a Gaussian with a diagonal covariance matrix.

2.2 Supervised Training of Final Layer Weights

Having determined the parameters of the basis functions, the second stage of network training is a supervised procedure in order to determine the output weights w_{jk}. The output units of a HBF network are linear[4] and take the form:

$$O_j(\mathbf{x}) = \sum_{k=1}^{K} w_{jk}\,\phi_k(\mathbf{x}) + w_{j0} \tag{10}$$

where w_{j0} are the biases.

Two methods for the supervised training of the output layer can be used. The first being the Widrow-Hoff delta rule [35] where the network weights are updated at the end of each full presentation of training data, with the final outputs having gone through a Sigmoid activation function. Training using the delta rule can take tens of minutes to achieve sensible results with further training having negligible effect on classification results. Testing during training is thus used to provide an indication of when to cease training. Alternatively, one can use a faster approach using the pseudo-inverse technique which finds an exact solution to the least squares problem. This is done with singular value decomposition (SVD) which is very faster allowing weight training in a fraction of a second. An exact solution does not, however, necessarily mean that the training data is classified 100% due to the presence of noise and the fact that the number of basis functions used was less than the number of data points. The pseudo-inverse method will now be described in more detail.

2.2.1 The Pseudo-Inverse Method

The pseudo-inverse method computes the output weights through matrix inversion using SVD. A sum-of-squares error function is used as a measure of the error between actual target values $\boldsymbol{t}_i$ for input vector $\boldsymbol{x}^i$ and those computed at the network outputs for all training patterns:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\{\sum_{k=0}^{K} w_{jk}\,\phi_k^i - t_k^i\}^2 \tag{11}$$

where $E(\boldsymbol{w})$ is a quadratic function of weights $\boldsymbol{w}$ and M are the number of network outputs, K the number of basis functions and N the amount of training data. Differentiating with respect to the weights and writing in matrix notation

$$(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})\mathbf{W}^{\mathrm{T}} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{T} \tag{12}$$

where $\mathbf{\Phi}$ is a matrix of basis outputs, $\mathbf{W}$ the weight matrix required to be determined and $\mathbf{T}$ the matrix of actual target values. The pseudo-inverse method determines $\mathbf{\Phi}^{\dagger}$ such that the weight matrix, which forms the final layer weights, can be found by

$$\mathbf{W}^{\mathrm{T}} = \mathbf{\Phi}^{\dagger}\mathbf{T} \tag{13}$$

where

$$\mathbf{\Phi}^{\dagger} = (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}} \tag{14}$$

and $(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}$ found through singular value decomposition.

[4]These can also be Sigmoid functions.

3 A Modified K-Means Algorithm

As described in section 2.1.3, a popular approach to find the basis function parameters is to use the EM algorithm. However, this method requires posterior probability estimates and is more complex and expensive than the K-Means. Instead, we adopt here a modified K-Means algorithm whereby hyper-ellipsoidal Gaussians rather than isotropic ones are fitted to the data, as in the more traditional K-Means algorithm described in section 2.1.2.

The traditional K-Means algorithm partitions the data into K disjoint subsets though a process of assigning data points to the nearest basis centre and a continual re-estimation of the basis centres $\boldsymbol{\mu}$. At the point where no further partitioning occurs the covariance matrix of each basis function is then set. A modified K-Means algorithm, which is similar to that of Sung and Poggio [31], also relies on a continual re-estimation of the basis centres. However, the normalised Mahalanobis distance metric is used to partition the data rather than the Euclidean distance metric thus allowing elliptical clusters to form. The algorithm also differs in that a continual re-estimation of a clusters shape through its covariance matrix also occurs. The modified K-Means algorithm is as follows.

1. Choose the number of clusters to be K.
2. **Initialisation**: Set the means $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_k$ of the basis functions as follows:
   (a) Randomly assign each $\mathbf{x}$ in X to a cluster and then compute the K means $\boldsymbol{\mu}_k$.
   (b) Re-partition X to the nearest mean using Euclidean distance.
   (c) Re-compute the means.
   (d) Approximate covariance matrices with
   $$\boldsymbol{\Sigma}_k \approx \frac{1}{N_k} \sum_{i=1}^{N_k} [\,(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^{\mathrm{T}}] \tag{15}$$
   where $k = 1, \ldots, K$ and N_k is the number of training vectors in cluster k.
   (e) If the number of vectors within a cluster becomes too small, discard that cluster, create a new cluster[5] through "division" of the largest cluster and goto 2.(b), else continue.
3. **Recursion**: Re-estimate $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ as follows **(Outer Loop)**
   (a) Compute the eigenvectors $\mathbf{u}_j$ and eigenvalues λ_j of $\boldsymbol{\Sigma}_k$ and then estimate $\boldsymbol{\Sigma}_k^{-1}$ by
   $$\boldsymbol{\Sigma}_k^{-1} = \mathbf{U}^{\mathrm{T}} \boldsymbol{\Lambda}^{-1} \mathbf{U}$$
   where $\mathbf{U}$ is the orthogonal eigenvector matrix of $\boldsymbol{\Sigma}_k$ and $\boldsymbol{\Lambda}$ is the corresponding diagonal eigenvalue matrix.
   (b) Re-estimation of $\boldsymbol{\mu}_k$ **(Inner Loop)**

[5] This is necessary because the covariance matrices are likely to become singular with small clusters. In order to keep the number of clusters constant, on the other hand, the largest clusters are "split".

i. Re-partition X with current $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$, where $k = 1, \ldots, K$, using normalised M-distance given by

$$M(\mathbf{x}, \boldsymbol{\mu}_k) = \frac{1}{2}[(d \ln 2\pi) + \ln|\boldsymbol{\Sigma}_k| + (\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)] \tag{16}$$

ii. Re-compute each $\boldsymbol{\mu}_k$.

iii. If the number of vectors within a cluster becomes too small, discard that cluster, create a new one through "division" of the largest cluster. Else continue.

iv. Goto 3.(b)i. until a pre-set number is reached.

(c) Goto 3.(a) until a pre-set number is reached.

4. Re-estimate $\boldsymbol{\Sigma}_k$ using Approximation (15).

5. With current means $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ forming the parameters of the basis functions train the final layer weights using a supervised technique.

3.1 Normalised Mahalanobis Distance

The normalised Mahalanobis distance given by (16) is used as a distance metric to partition the training data set. It is the negative natural logarithm of a multi-dimensional Gaussian distribution centred at $\boldsymbol{\mu}$ with covariance matrix $\boldsymbol{\Sigma}$, and originates directly from a probability density distribution that integrates to unity. The normalised Mahalanobis distance is used for two main reasons:

1. It allows elliptical clusters to form by reducing the penalty of pattern differences along a clusters major axes of distribution, i.e. the eigenvectors corresponding to the largest eigenvalues. Partitioning with Euclidean distance favours hyper-spherical clusters of equal size and results in the splitting of large and also elongated clusters.

2. It is used for stability reasons. The Mahalanobis distance given by (2) could be used to produce hyper-ellipsoidal clusters. However has a number of undesirable properties in that it produces very large or very small clusters. The Mahalanobis distance tends to be smaller for long clusters with large covariance matrices than small ones which results in the larger clusters overwhelming the smaller ones. This does not tend to occur with the normalised M-distance.

3.2 Input Representation — The "Curse of Dimensionality"

Since the estimation of basis parameters is an unsupervised procedure, it does not proffer an optimal solution in the sense that unsupervised learning is purely based on the data distribution taking no account of target labels. In addition a clustering algorithm that works on one type of data distribution may not work on another. The choice of data representation will have an important effect on the clustering result.

Using face images directly as the input to the network is infeasible. This phenomenon is due to the "The Curse of Dimensionality" [2] which occurs with limited data of high dimensionality. In such cases the data distribution is sparse and as a result, the representation of the mapping from data space to an output variable poor.

For a HBF with input vector of dimension d, the covariance matrix $\mathbf{\Sigma}_k$ for hidden unit k is of size $d \times d$, with $d(d+1)/2$ parameters (covariance elements) required to be determined. If the number of patterns per basis function is small compared with d, $\mathbf{\Sigma}_k$ is likely to be singular and modelling of the data distribution with a full covariance matrix becomes infeasible. Due to the amount of data available, with on average less than sixty pattern vectors per cluster being used to determine the covariance parameters, there exist two main options[6] for reducing the number of parameters.

The first is the use of radial Gaussians or a diagonal covariance matrix for which the multi-dimensional Gaussians major axes of elongation align with the input vector space axes. However, this is likely to result in a poor model of the data distribution. This is likely to be the case when dealing with face/head pose varying data. The second method is to keep full covariance Gaussians but use a smaller input dimension d. It should be noted that when the number of vectors assigned to a basis function becomes too small, the estimation of the covariance matrix becomes less accurate and is likely to become singular. In fact, when the number of vectors in a cluster is less than the input dimension, the calculation of $\mathbf{\Sigma}^{-1}$ through Principal Component Analysis (PCA) becomes infeasible. As a result, in clustering small clusters should be discarded when the number of vectors assigned to them becomes too small. In order to keep the number of clusters constant the largest clusters are therefore "split". This will be explained in more detail later in section 3.4.1.

We choose to reduce the dimensionality of the original 56×56 grey level images through PCA, to ten dimensions. The choice of reduction to only a ten dimensional vector will result in significant information loss including perhaps that associated with between class variation. However, due to the amount of data available the choice of such a drastic dimensionality reduction is a necessary one. This is in order to determine the covariance matrix associated with the basis functions with reasonable degree of accuracy. It can be seen from the above that the number of basis functions to be selected is dependent on the amount of data available and its dimensionality. If there are too many basis functions the average number of vectors per cluster becomes small and any attempt to estimation of full covariance Gaussians will be likely to fail.

3.3 PCA for Dimensionality Reduction

Dimensionality reduction is achieved via the projection of a face vector $\mathbf{x}$ onto the first N eigenvectors $\mathbf{u}_k$ of the covariance matrix obtained from a given face database, $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_P\}$, where P is the number of face vectors in X and $N < d$, where d is the original image dimension:

$$\omega_k = \frac{\mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu})}{\lambda_k} \tag{17}$$

where λ_k are the eigenvalues and $\boldsymbol{\mu}$ is the mean face vector of X. The division by the eigenvalues is employed to give equal variance along each principal component axis. This prevents projections onto principal components with large eigenvalues overwhelming projections onto those with smaller ones. These eigenvectors capture the global statistical variation among the images. Thus the pattern vector $\mathbf{\Omega}(\mathbf{x})$ which constitutes all the weights for the N projections provides a representation of a face vector $\mathbf{x}$:

$$\mathbf{\Omega}(\mathbf{x}) = [\,\omega_1\, \omega_2\, \ldots\, \omega_N\,] \tag{18}$$

[6]Low resolution face images could be used. However, this has its limitations since there will be a minimal image resolution one must have in order to perform any sensible recognition.

3.4 Implementation

The number of basis functions is set at the onset and an initial random assignment of training data to basis functions is made. The initial positioning of cluster prototypes and the number of hidden units used both play an important part in the networks ability to accurately represent the underlying data distribution and thus its ability to classify new data. A number of runs of a particular test were undertaken to find the best set up for a particular training and test data set. This was achieved by altering the network parameters such as the initial clustering positions and the number of clusters used. However, various schemes [23, 36, 19, 12, 26] have been examined to try and build invariance to the initial clustering position and to allow the number of hidden units to grow/decrease as the clustering process evolves. This would thus provide a more automatic clustering procedure without the need to run a number of tests in order to find the best network and conditions for the specific problem at hand.

The second stage in HBF network training is supervised in order to determine the network output weights. Two methods for the supervised training were implemented, these being the Widrow-Hoff delta rule and the pseudo inverse method. Both supervised methods make use of a target vector $\mathbf{t}_i$, attached to each training input vector[7] that describes the required output of each of the HBF output units. The target vector has individual elements that take the values of $\pm$ 0.5. For a vector associated with a class c_j among n classes in the training data, the target vector takes the form:

$$\mathbf{t}_i = [m|f\ f|o\ c_1\ c_2\ \ldots\ c_j\ \ldots\ c_n] \tag{19}$$

where,

$$c_k = \begin{cases} +0.5 & \text{for } k = j \\ -0.5 & \text{for all } k \neq j \end{cases}$$

for $k = 1, \ldots, n$. The target vector has $n + 2$ values in total with the first two values indicating gender, + 0.5 male and -0.5 female, and the second face/object class, +0.5 and -0.5 respectively.

Various problems arose during preliminary testing that need to be addressed. These encompass numerical as well as more general problems mainly found due to the nature of the input data.

3.4.1 Splitting Clusters

The reason for splitting large clusters to accommodate smaller ones has been mentioned briefly in section 3.2 in that it discourages very small clusters for which the computation of the covariance matrix might become infeasible. The method for performing this will now be described. During clustering, prior to the computation of the covariance matrices, the number of vectors per clusters is checked and those containing less than a desired number (user defined) are discarded. This is achieved by "splitting" the larger clusters by assigning a new mean $\boldsymbol{\mu}_{new}$, to the basis function of the small cluster through perturbation of the largest cluster mean vector $\boldsymbol{\mu}_{max}$, such that:

$$\boldsymbol{\mu}_{new} = (1 + \delta)\boldsymbol{\mu}_{max}, \qquad where \qquad \delta > 0$$

Re-partitioning then continues using this new mean value. However care does need to be taken as to the choice of δ and ideally should be small, $\delta = 0.01$ for the data used. Due to the scale of the input vector used [-1, 1], a problem arose when δ was too big. This actually resulted in a new mean being assigned to a basis function

[7] With the limited data available all data is used in both supervised and unsupervised training.

that was so far from the maximum clusters mean that "splitting" of the largest cluster did not actually occur. This was due to the data remaining nearer to its mean value than the new mean resulting in the algorithm became unstable.

3.4.2 Input Vector Scaling

The nature of the input data also results in numerical problems which were solved through scaling of the input vector. Eigenvalues produced through PCA on a cluster, for input data in the range [-1, 1], are small and of a similar range. In fact to floating point precision they were very close to zero and when saved in ASCII format on many occasions became zero. This is a major problem in that the estimation of $\mathbf{\Sigma}^{-1}$, in the computation of the normalised Mahalanobis distance given by (16). The calculation requires division by eigenvalues resulting in various inaccuracies and problems, particularly of course when eigenvalues are zero! Un-normalised eigenvectors could be used however floating point overflow occurs and a similar scaling is required.

An additional problem found was that eigenvalues in this range produce in some cases negative normalised Mahalanobis distances. This is due to the fact that the natural log of the determinant of the covariance matrix in the computation of the normalised Mahalanobis distance is in the sum of the natural log of the eigenvalues, which for eigenvalues in the above range is negative. In some cases this is sufficient to make the normalised Mahalanobis distance negative. This is not a major problem however due to the fact that when partitioning data the smallest normalised Mahalanobis distance determines the cluster to which data is assigned which in such cases is the most negative value. However it is more intuitively pleasing for a distance metric to remain non negative which can be achieved by the scaling of the input vector. This is solved through scaling the input vector by a large value β for example. This results in a similar scaling of the data variance within a cluster by β, and thus eigenvalues by β^2. For sufficiently large β ($\beta = 1000$ used) the natural log of the eigenvalues become non negative which guarantees positive normalised Mahalanobis distance achieving a more sensible distance metric. To guarantee a positive normalised Mahalanobis distance the eigenvalues produced through PCA on a cluster are required to be greater than 1. Input data in the range [-1000, 1000] should achieve this. It should be noted that the Mahalanobis distance actually remains constant with scaling as the covariance matrices scale accordingly however, numerical problems that could effect the calculation of $\mathbf{\Sigma}^{-1}$, which requires division by eigenvalues, are removed.

One further numerical problem found was due to the nature of the outputs of the basis functions given by (1). The values of the basis function outputs (1) were again very close to zero due to the fact that the Mahalanobis distance given by (2) is non negative and tends to be large. This output is used in the final supervised training and such small values presents problems, particularly when the pseudo inverse matrix inversion technique is used to train final layer weights. The scaling of the input data has no effect on the Mahalanobis distance therefore this would not effect the basis function output. However, a method that provides a solution is to scale the Mahalanobis distance by a value γ for example, prior to taking the exponential in the computation of the basis function output:

$$\phi_j(\mathbf{x}) = \exp\left[-\frac{1}{2\gamma}\mathbf{\Delta}^2(\mathbf{x})\right] \tag{20}$$

During training and testing it was found that a division by a large number, $\gamma = 1000$, generally resulted in larger basis function outputs allowing supervised training to occur without numerical breakdown. A more automatic choice of γ could be made through automatic inspection of the Mahalanobis distance range and scaling accordingly. The various tests performed after having trained the HBF network will now be discussed.

4 Experiments

The image data available consists of three sets of rotating head sequences of known faces (see Figure 5). The faces undergo continuous rotations in depth. A number of face recognition experiments were performed on these data sets. A further set of random head and non-face image data are also available but were not used.

4.1 Data Sets

1. Tracked head rotation (set A with 356 images): Sequences of 6 different people whose pose angle varies from -90 to +90 degrees. This data was obtained from a motion based tracker [21] with sets varying in length from 56 to 60 frames. There is extreme variation, particularly in scale and translation, amongst the frames.

2. Smooth head rotation (set B with 563 images): Sequences of 7 different people whose pose angle varies from -90 to + 90 degrees. This data was obtained via a person rotating their head in a smooth manner while seated. The data was obtained under more constrained conditions, in terms of lighting a background, with image sequences varying in length from 66 to 87 frames. Six of the people are represented within set A above but an extra class called "Jon" exists in this data set.

3. Labelled head rotation (set C with 95 images). Sequences of 5 different people all present in the above two sets. There are 19 frames per person taken at 10 degree increments thus representing a -90 to +90 degree pose rotation. This set was obtained at different times to the above sets therefore lighting would be expected to be different and the heads at a slightly different scale. It should be noted that "lorna" in this set has a significantly different hair style than in the data sets above.

4. Random head and non-face data (set D): Sequences of random head movements, about all 3 axes (x,y,z), of people not contained in the above 3 sets. In addition a further sequence of continuous rotating non-face objects exists. This data can be used to "carve" out (model) the regions around the -90 to +90 degree rotating face-pose distribution of the above 3 data sets. This should go someway to preventing data not from the specified classes being classified as such. Such a scheme would form the basis of an intruder detection system for example.

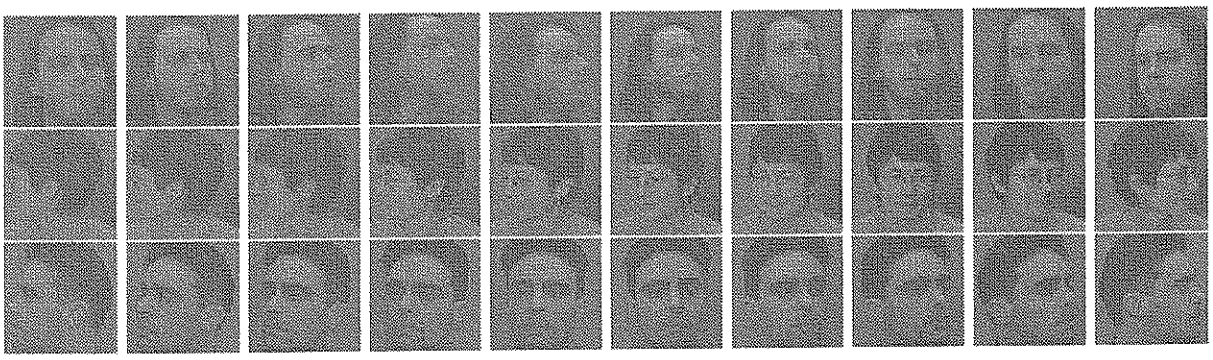

Figure 5: Top row: Tracked head rotation, tracked face images from every tenth frame of a head rotation sequence. Middle row: Smoothed head rotation. Bottom row: Labelled head rotation, second frame of a labelled head sequence rotating in depth with 10 degrees increments.

4.2 Results

Two kinds of training and testing were performed. With the first type, the training and testing data were distinct, i.e. from different sets. In the second type, the training and testing data were obtained through the division of a much larger set. This was done through "sampling" from a sequence at regular intervals (i.e. the removal of specific frames from a set at regular intervals). It should be noted that no preprocessing, such as Difference of Gaussian (DoG) filtering, Gabor wavelet transforms (GWT) or image registration [16, 13], was performed on the data other than PCA for dimensionality reduction.

4.2.1 *Experiment 1 - Setting Up HBF Networks*

If the number of face images available for estimating parameters of a basis function is small compared with the dimension of the face space, its covariance matrix is likely to be singular and modelling density distribution with a full covariance matrix becomes infeasible. Experiments were then performed in which the parameters of a HBF network were varied in order to find suitable configurations for the particular training and testing data. Such variations included the initial positioning of training data vectors to basis functions, together with the number of basis functions themselves which varied between five and fifteen. The number of inner and outer loops (10×10 by default) within the clustering algorithm were also altered but did not seem to play a major role in classification performance. The dimensionality of the input vector was kept constant and restricted to ten dimensions.

The input vector was scaled by 1000 prior to clustering, for reasons explained in section 3.4.2, and the Mahalanobis distance divided by 1000 prior to taking the exponential, in the computation of the basis function outputs. In some cases however the Mahalanobis distance was divided by a larger number resulting in increased performance. As mentioned previously though this scaling factor could be expected to be found through automatic means. The pseudo-inverse method was mainly used to train the final layer weights although the Widrow-Hoff delta rule, which was the first supervised method implemented, was tested as well. Test pattern vectors were obtained through projection onto the eigenvectors obtained from the "image" data.

In the following, a set of experiments were performed for face recognition using the image sequences from the data set previously described consisting of head rotations in depth. The experiments were aimed to evaluate the networks ability to generalise, and to cope with scale variation and translational shift within the test data. It should be noted that some results are presented alongside those obtained through experimentation on the same data elsewhere [14]. Those experiments included the Eigenface and Fisherface linear approaches plus three HBF networks employing radial Gaussians (HBF-R), diagonal Gaussians (HBF-D) and a decoupled input HBF (DI-HBF) employing again radial Gaussian. Both HBF-R and DI-HBF were trained using the traditional K-means algorithms while the HBF-D network was trained using the EM algorithms. The Modified K-Means algorithm employing full covariance Gaussians is represented by HBF-FULL.

4.2.2 *Experiment 2 - Generalisation Ability*

This test was undertaken to test the networks ability to generalise to new data. The network was trained with labelled image data from set C plus synthesised data obtained through x,y image plane shifts of up to 15% on the head images. Two sequences were selected from set B with the same person looking quite different ("aleka" and "lorna") in the test and training sequences. A correct classification rate of 68% was achieved for

the test set with 32 out of 59 and 49 out of 59 frames correctly recognised for "aleka" and "lorna" respectively. However, on average "lorna" repeatedly scored a better average of correct matches, of approximately 45 frames recognised with "aleka" more like 20 frames correct. In addition to the differences in scale and translation which could effect overall results, there is a slightly greater scale difference between the two sequences of "aleka" than "lorna". This could explain why "lorna" was repeatedly better recognised. The results are presented in Table 1.

Method	On Training Data	On Testing Seqs
HBF-FULL	75%	68 %
HBF-R	84%	51%
HBF-D	77%	33%
DI-HBF	84%	32%
Eigenface	16%	0%
Fisherface	18%	0%

Table 1: Results on generalisation.

4.2.3 *Experiment 3 - On Translational Shift & Scale Variation*

This experiment was conducted to test the networks ability to cope with translational and scale variation. Again, the networks were trained using the labelled sequences in set C. Two test sequences were obtained from the tracked data of set A, both having a fair amount of scale and translational variations within the images, especially for the second testing sequence which contained quite severe offsets. The results can be found in Table 2. It can be seen that 60% correct classification rate was achieved signifying that the network is able to cope with large translational and scale variation.

Method	On Training Data	On Testing Seqs
HBF-FULL	66%	60%
HBF-R	84%	54%
HBF-D	77%	58%
DI-HBF	81%	50%
Eigenface	16%	0%
Fisherface	18%	8%

Table 2: Results on Scale and Offset.

A further experiment to test this was performed in which the smooth rotating sequences of set B formed training set while set A formed the training data set. The results are shown below in Table 3. It can be seen that a high training data classification rate was achieved with a reasonable rate for the test data set. Of the six face sequences that formed the test set four sequences achieved greater than 50% individual correct classification. Of the two that did not achieve this, one sequence was totally miss-classified. This we feel was largely due to the scale of the head within the training data set being very much different to that within the test set. This was even with all the scale variation within the test images.

The PCA representation used to form the input to the HBF is not invariant to such factors as scale and translational variation although both experiments show that the network is able to cope better with scale rather than translational variation. It should be noted that the test image data does show extreme variation in head translation and scale for profile views with such images tending to be those that were incorrectly classified.

Network	On Training Data	On Testing Seqs
HBF-FULL	99%	54%
HBF-R	95%	30%
HBF-D	96%	58%
DI-HBF	99.5%	69%

Table 3: Scale and Offset using data set B as training data.

4.2.4 *Experiment 4 - Eliminating Scale and Translational Variation*

Further experiments were undertaken where both the training and testing data were taken from set B. The testing data were formed through the extraction of different frames from the same sequences used for training. Hence scale and translational variation have been removed. The results are shown in Table 4. The images used to form the testing data were different from those for training.

Forming Testing Data	On Training Data	On Testing Data
Every 5^{th} frame	93%	93%
Every 2^{nd}, 3^{rd}, 7^{th} and 8^{th} frame	88%	90%
Every 3^{rd}, 4^{th}, 5^{th}, 6^{th} and 7^{th} frame	91%	93%

Table 4: The Testing set was formed by extracting every n^{th} frame per ten frames from sequences in set B. The training data comprised of the remaining frames in the sequences.

These tests show good results with over 90% correct recognition achieved. All 7 sequences in set B were correctly classified for more than 50% of all images. In fact the failures in such cases tended to occur when the corresponding head images were at or near profile views. This could be due to the limited data available to model the pose distribution at the profile views. The network was not able to accurately model the full pose-distribution with poor results where discontinuities exist and where data were scarce. The network shows reasonably high classification rates with its ability to classify correctly even when five consecutive frames out of every ten form the test set is quite good. This test show perhaps more the networks ability to interpolate data rather than generalise. However it could be argued that the network does have the ability to generalise when "new" data is normalised and registered. This would be the case when testing data is obtained under the same conditions as the training data through the division of a much larger set.

5 Conclusion

Overall the HBF networks implemented, particularly the one employing a full covariance Gaussian HBF-FULL, shows promise in its ability to model the non-linear face pose distribution due to head rotations in depth. Certainly better recognition rates were achieved than those obtained for the linear methods based on the Eigenface and Fisherface. These two methods which do not in any way estimate the density distribution of the data performed poorly. The results for the three HBF networks were comparable with the network employing a full covariance Gaussian performing slightly better overall. This was even with the lower dimensional input vector and the relatively smaller number of hidden units used[8]. The DI-HBF implemented certainly shows promise with its recognition results expected to improve with the introduction of hyper-ellipsoidal Gaussians through the use of a full covariance matrix rather than the radial Gaussians employed at present.

It should be noted that for most tests training data was not classified very highly. In such circumstances test data would not be expected to achieve good classification results. Training of the output weights using SVD requires linear outputs. With only a small number of basis functions being used to model the distribution the basis function outputs would be expected to show a high degree of non linearity resulting in poor training of the output layer weights and thus poor classification. Training using the Widrow-Hoff delta rule for which the outputs undergo a non linear transform does seem to show an increase in classification performance however the time to train the network does restrict its use.

Overall though, preliminary results presented have shown that HBF networks with Gaussian mixtures provides an effective way to model non-linear face pose distributions in face space and is a computationally efficient method of performing recognition under large pose variations. However, more accurate estimation of the Gaussian mixture parameters is required and thus more accurate modelling of the data density distribution.

5.1 Improvements and Additional Work

Various schemes exist which would be expected to provide increased classification results, some of which have been discussed in section 3.2 , with more data expected to solve many of the problems. For one it would allow a more accurate estimation of the Gaussian mixture parameters and thus better modelling of the data distribution. The dimensionality of the input vector could then be extended resulting in more of the across class variation to be retained. Increased data would also help to remove biases that exist due to the differing amounts available per training class and to some extent also provide invariance to such factors as scale and translational variation which is not the case at present. As well as building invariance in to the network this could also be achieved prior to network training through preprocessing. Howell and Buxton [16] and Gong *et. al.*[13, 22] for example highlight how preprocessing such as GWT and DoG filtering provide invariance to such factors as scale, global lighting etc and therefore increased classification rates should follow.

However, a method that may provide a partial solution in the absence of significant amounts of data is proposed by Mao *et. al.* [18]. A hyper-ellipsoidal clustering method is presented in which clustering is performed using the following regularized Mahalanobis distance (RMD)

$$\Delta_{RMD}(\mathbf{x}) = (\mathbf{x} - \mu_k)^{\mathrm{T}}[(1-\lambda)(\Sigma_k + \eta I)^{-1} + \lambda I](\mathbf{x} - \mu_k) \tag{21}$$

[8]The input dimensionality of the other networks implemented in citeGong-etc:97 varied from 20 to 50 and the number of hidden units varied from 20 to 40.

where

$$\Sigma_k^{-1*} = (1-\lambda)(\Sigma_k + \eta I)^{-1} + \lambda I \tag{22}$$

is the regularized inverse of the covariance matrix Σ_k and I, the identity matrix. Clustering is performed taking each cluster size into consideration with λ, a parameter to control the amount the RMD deviates from the squared Euclidean distance. Hence, a larger value of λ is employed in situations when the amount of data in a cluster is small or not significantly larger than the dimensionality of the input vector. In such situations, as highlighted in section 3.2, Σ_k is likely to be singular and thus the computation of Σ_k^{-1} unreliable. Mao *et. al.* also suggest that the use of the RMD avoids the problem of large and small clusters forming, a problem with the Mahalanobis distance. The normalised Mahalanobis distance was used to overcome this problem in the algorithm that was implemented however, the introduction of a regularized normalised Mahalanobis distance may provide a partial solution to this problem of insufficient data. This is something that will need to be investigated further.

Better segmentation of image data would also be expected to provide several benefits. At present the images used contain significant amounts of background as well as shoulder, hair etc. Segmenting and using the face rather than the present images would allay fears that environmental factors influence classification. Segmented data would also reduce the effects of translational variation in addition to which it would be hoped that the input vector, obtained through PCA, would better represent the data and across class variation.

5.1.1 Basis Function Number and Initial Position Invariance

One of the major problems of clustering algorithms and the K-Means method in particular is in choosing the appropriate number of bais functions and their initial position. This makes a big difference to final classification results with various schemes [23, 36, 19, 12, 26] having been proposed to provide some invariance. For example Gath *et. al.* [12] describe a fuzzy K-Means clustering algorithm which employs unsupervised tracking of basis function centres to provide invariance to initial clustering positioning. In addition the clustering process is iterative for which the number of basis functions is increased until an optimum value of performance is reached. Various stopping rules for fuzzy clustering [4] and hard clustering[9] [23] exist which are used to produce a cluster validity criteria to determine the optimal number of clusters in the data under examination. Such rules however, tend to be for clustered data for which the clusters are disjoint and are represented by only one basis functions. Certainly the thirty stopping rules examined in [23] were tested on such disjoint clustered data.

In addition to such stopping rules other techniques have been applied. Xu *et. al.* [36] present a clustering algorithm called "Rival Penalized Competitive Learning" (RPCL) to combat the main problem of the inappropriate choice of the number of basis functions and their initial positioning within the K-Means method. This algorithm has also been shown to be effective for disjoint clustered data and relies on a choice of the number of basis functions being greater than the number of clusters actually present within the data. This method works by making sure that each cluster in the data set is learnt by only one basis function through adapting the winning basis function to that of a new input vector while de-learning its nearest rival. This results in the rival basis functions mean being pushed away from that of the winning basis function. Thus by choosing the number of basis functions larger than the number of clusters within the data set there will be spare basis functions, or "dead units" as they are known, which do not represent the data set. These spare units can be easily identified and culled from the network making network learning computationally more effective and classification better than the traditional K-Means not employing this RPCL scheme.

[9]Each data point is assigned to one cluster only

Morroquin *et. al.* [19] provide additional extensions to the K-Means algorithm. These methods could be beneficial for face recognition under pose variation as they are shown to work well in regions of sparse and varying data distribution and for multi-class data containing very complex decision boundaries. The necessary algorithm being used to sample the inter-class boundary manifold directly which contains the relevant locations for classification purposes. The various methods presented take into account the past dynamic behaviour of cluster centres through the introduction of state variables and results in the number of clusters adapting to the local density distribution of the data with clusters being "split"[10] in regions of high data density.

As well as looking at various schemes for improving the clustering process it is worth mentioning that pose estimation has been treated here essentially as a pattern recognition task. There clearly exist, however, a variety of spatial and temporal contextual cues such as body pose and continuity of pose change which could be used [3, 11, 15, 30]. This will be the focus of further work at QMW.

6 Acknowledgments

We would like to thank both Eng-Jon Ong and Stephen McKenna for many helpful discussions on this work and for providing some of the diagrams and pictures used.

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. In *ECCV*, Cambridge, England, 1996.

[2] R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[3] D. J. Beymer. Face recognition under varying pose. AI Memo 1461, MIT, Cambridge, Massachusetts, 1993.

[4] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* New York: Plenum, 1981.

[5] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford Press, Oxford, England, 1995.

[6] V. Bruce, A.M. Coombes, and R. Richards. Describing the shapes of faces using surface primitives. *Image and Vision Computing*, 11, 1993.

[7] V. Bruce, P. Healy, M. Burton, T. Doyle, A. Coombes, and A. Linney. Recognising facial surfaces. *Perception*, 20:755–769, 1991.

[8] H. Bülthoff, S. Edelman, and M. Tarr. How are three-dimensional objects represented in the brain? *Cerebral Cortex*, 5:247–260, 1995.

[9] 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):1–38, 1977.

[10] In the algorithm that has been implemented some clusters are "split" but for different reasons

[10] K. Etemad and R. Chellappa. Discriminant analysis for recognition of human face images. In *Int. Conf. on Audio- and Video-Based Biometric Person Authentication*, pages 127–142, Crans-Montana, 1997.

[11] P. Foldiak. Learning invariance from transformation sequences. *Neural Computation*, 3, 1991.

[12] I. Gath and A.B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 11(7):773–781, July 1989.

[13] S. Gong, S. J. McKenna, and J. J. Collins. An investigation into face pose distributions. In *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, 1996.

[14] S. Gong, E.J. Ong, and P.J. Loft. On appearance-based methods for face recognition under large rotations in depth. Submitted to ICCV'98, 1997.

[15] S. Gong, A. Psarrou, I. Katsoulis, and P. Palavouzis. Tracking and recognition of face sequence. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, Hamburg, November 1994.

[16] J. Howell and H. Buxton. Face recognition using radial basis function neural networks. In *British Machine Vision Conference*, Edinburgh, Scotland, September 1996.

[17] M. Lando and S. Edelman. Generalization from a single view in face recognition. Tech. Report CS-TR95-02, Weizman Institute, Israel, 1995.

[18] J. Mao and A.K. Jain. A self-organizing networks for hyperellipsoidal clustering (hec). *IEEE Transactions on Neural Networks*, 7(1):16–29, January 1996.

[19] J.L. Marroquin and F. Girosi. Some extensions of the k-means algorithm for image segmentation and pattern classification. Technical Report Paper No. 079, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, January 1993.

[20] J.C. Mason and M.G. Cox. *Algorithms for Approximation*, chapter Radial basis functions for multivariate interpolation: a review. Oxford: Clarendon Press, 1987.

[21] S. McKenna and S. Gong. Tracking faces. In *IEEE Second Internatioanl Conference on Automatic Face and Gesture Recognition*, Killington, Vermont, US, October 1996.

[22] S. McKenna and S. Gong. Real time face pose estimation. Submitted to the Academic Press International Journal on Real Time Imaging, Special Issue on Visual Monitoring and Inspection, March 1997.

[23] G.W. Milligan and M.C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, June 1985.

[24] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *IEEE ICCV*, Cambridge, Massachusetts, June 1995.

[25] J.E. Moody and C.J Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[26] R.P. Nikhil, J.C. Bezdek, and C.-K. Tsao, E. Generalized clustering networks and kohenen's self-organizing scheme. *IEEE Transactions on Neural Networks*, 4(4):549–557, July 1993.

[27] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *IEEE CVPR*, Seattle, July 1994.

[28] T. Poggio and S. Edelman. A network that learns to recognize three-dimensioanl objects. *Nature*, 343, January 1990.

[29] T. Poggio and F. Girosi. Netwroks for approximation and learning. *Proceedings of The IEEE*, 78(9), September 1990.

[30] A. Psarrou, S. Gong, and H. Buxton. Spatio-temporal trajectories and face signatures on partially recurrent neural networks. In *IEEE ICNN*, Perth, Australia, November 1995.

[31] K. Sung and T. Poggio. Example-based learning for view-based human face detection. Technical Report AI Memo 1512, CBCL 103, MIT, 1995.

[32] D. L. Swets and J. Weng. Discriminant analysis and eigenspace partition tree for face and object recognition from views. In *Proc. 2nd Int. Conf. on Automatic Face and Gesture Recognition*, pages 192–197, 1996.

[33] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neuroscience*, 3(1), 1991.

[34] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE PAMI*, 13(10):992–1006, October 1991.

[35] B. Widrow and M.E. Hoff. Adaptive switching circuits. In *IRE WESCON Convention Record*, volume 4, pages 96–104, New York, 1960. Reprinted in Anderson and Rosenfeld (1988).

[36] L. Xu, A. Krzyzak, and E. Oja. Rival penalized competitive learning for clustering analysis, rbf net and curve detection. *IEEE Transactions on Neural Networks*, 4(4):636–649, July 1993.