Non-linear Point Distribution Modelling using a Multi-layer Perceptron

P.D. Sozou, T.F. Cootes, C.J. Taylor and E.C. Di Mauro
Department of Medical Biophysics
University of Manchester
M13 9PT
email: pds@sv1.smb.man.ac.uk

Objects of the same class sometimes exhibit variation in shape. This shape variation has previously been modelled by means of point distribution models (PDMs) in which there is a linear relationship between a set of shape parameters and the positions of points on the shape. A polynomial regression generalization of PDMs, which succeeds in capturing certain forms of non-linear shape variability, has also been described. Here we present a new form of PDM, which uses a multi-layer perceptron to carry out non-linear principal component analysis. We compare the performance of the new model with that of the existing models on two classes of variable shape: one exhibits bending, and the other exhibits complete rotation. The linear PDM fails on both classes of shape; the polynomial regression model succeeds for the first class of shapes but fails for the second; the new multi-layer perceptron model performs well for both classes of shape. The new model is the most general formulation for PDMs which has been proposed to date.

1. INTRODUCTION

Objects of the same class are often not identical in shape. This presents a challenge for a number of medical and industrial applications which involve locating objects in images. In such cases deformable models, which allow for variability in the shapes of imaged objects, are often appropriate. The most important issues are those of *generality* and *specificity*. A deformable shape model should be sufficiently general to fit to valid unseen examples of the object(s) it represents, but sufficiently specific that it cannot fit to invalid 'impostor' shapes. It is also useful for many applications to have a *parsimonious* model, in which the allowed variation in the objects of interest can be represented by a small number of shape parameters.

Cootes et al. [1] have described how shape variability for a class of objects can be represented by a Point Distribution Model (PDM). (See [1] also for general references on deformable models.) In a PDM, objects are defined by landmark points which are placed in the same way on each of a set of examples. A statistical analysis yields estimates for the mean shape and the main 'modes of variation' for the class. Each mode changes the shape by moving the landmarks along straight lines passing through their mean positions. New shapes are created by modifying the mean shape with weighted sums of the modes.

For flexible or articulated objects these linear models are not sufficiently specific or parsimonious – they can adopt implausible shapes, very different from those in the training set, and have more modes of variation than the true number of degrees of freedom needed to describe the objects' variation. We have recently [2] described a polynomial regression PDM (PRPDM) which succeeds in capturing limited non–linear shape variability.

In this paper we present a new method for modelling non–linear shape variability which involves using a multi–layer perceptron to perform non–linear principal component analysis. We first give a brief description of linear PDMs and PRPDMs, and show how they can fail. We then describe the new multi–layer perceptron PDMs (MLPPDMs) , and show how they can be trained from sets of example shapes. The new method is shown to perform well on both data which involves limited non–linear shape variability and data involving full rotation.

2. LINEAR POINT DISTRIBUTION MODELS

A shape in a 2–D image may be represented by the position $\mathbf{x}$ of a set of n landmark points [$\mathbf{x} = (x_1, y_1 \; .... x_n, y_n \;)$], and is thus described by a point in $2n$–dimensional landmark space. Given a number of examples of shapes belonging to the same class, we will have a distribution of points in landmark space.

A linear PDM is created by carrying out a simple principal component analysis of this distribution [1]. The t eigenvectors corresponding to the largest t eigenvalues give a set of basis vectors for a flexible model. A new example is generated by adding to the mean shape a superposition of these basis vectors, weighted by a set of t shape parameters $(b_1, b_2, \; ... \; b_t)$. We describe the basis vectors as the *modes of variation* of the shape.

The principal component analysis can be thought of as performing two statistical functions. Firstly, by retaining only the first few eigenvectors which explain most of the variance, it enables a parsimonious model to be created, with a small number of shape parameters. Thus it performs linear dimension reduction. Secondly, the distribution of the training examples is linearly independent in the t eigenvectors which are retained, giving the most specific (i.e. the most compact) model possible for a linear transformation of landmark space.

The linear model is not appropriate if there are non–linear dependencies in the data. An example is where one sub–part of a modelled object can rotate about another. The linear paths traced out by the landmark points, as each parameter b_j is varied, provide a poor approximation to the circular arcs which are required to model the shape variation accurately.

2.1. Tadpole Data

The tadpoles shown in Fig. 1 are examples from a synthetic class of shapes. We have previously described in detail how they are generated [2].

Training a linear PDM on 200 tadpoles, we found that three modes of variation were needed to explain 95% of the variance in the training data [2]. Space does not permit us to show these modes of variation, but we reproduce in Fig. 2 the scattergrams of the second and third shape parameters plotted against the first. A weak non-linear relationship exists between b_2 and b_1, and there is a very clear non-linear relationship between b_3 and b_1. Thus the assumption of independence of the shape parameters does not hold. This means that a combination of shape parameters can be chosen, each within its normal range, such that a shape very different from any in the training set is generated.

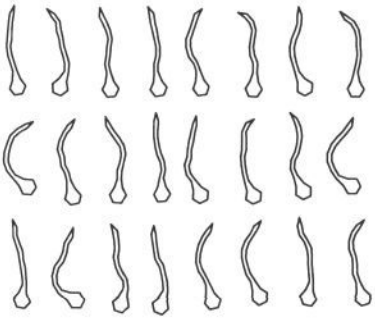
Fig. 1.– Tadpole training data.

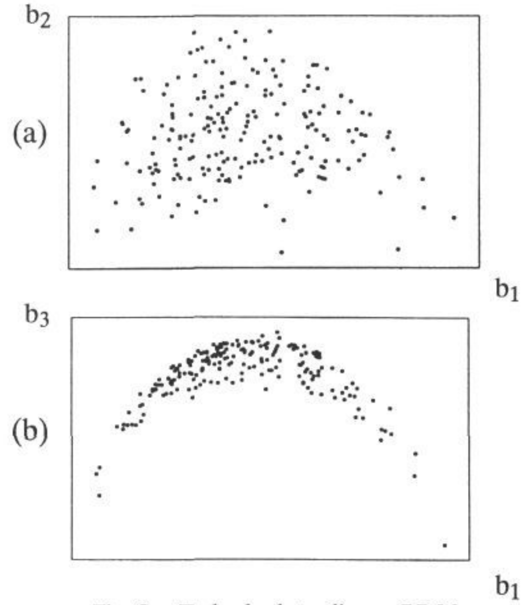

Fig. 2.– Tadpole data, linear PDM: scattergrams: (a) b_2 vs b_1, (b) b_3 vs b_1.

2.2. Watch Data

The watch data were obtained from 30 video images of a Seiko quartz watch, set to random times between 12.00 and 1.00. One of the images is shown in Fig. 3. Landmark points were placed at either end of each of the 11 runes, at the four corners of the date box, at the ends of the hour and minute hands, and at the centre of the watch, giving a total of 29 landmark points. Examples of the training data are shown in Fig. 4.

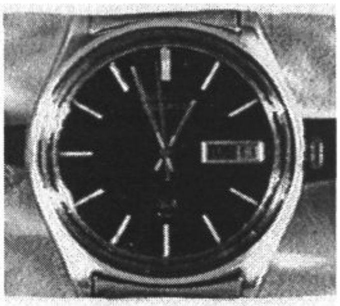
Fig. 3.– Example of watch image.

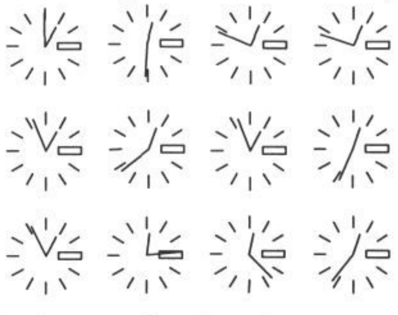
Fig. 4.– Examples of watch training data.

For the linear PDM, two modes of variation are needed to explain 99% of the variance in the data. They are illustrated in Fig. 5, which shows how the shape changes as each shape parameter is varied. We see that in this model the minute and hour hands change length, something which does not occur in the training examples. Thus the model is not specific. Fig. 6 is a scattergram of the first two shape parameters (b_2 against b_1) for the training data. There is a very clear non–linear relationship between b_2 and b_1, with the points lying approximately on a circle. As with the tadpole example, implausible shapes can be generated from apparently legal combinations of parameters.

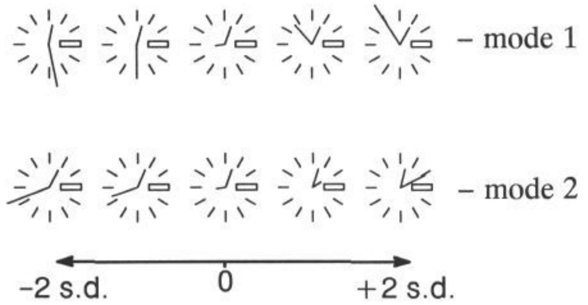

Fig. 5.– Two variation modes (mean + /– two standard deviations) of linear PDM trained on the watch data. Note the changing length of the hour and the minute hands.

3. THE POLYNOMIAL REGRESSION MODEL

We have recently described [2] a modified class of models known as a Polynomial Regression PDM (PRPDM). The approach is motivated by noting that the eigen–analysis used to extract the modes of variation for a standard PDM can be conceptualised (however implemented) as a process of sequentially fitting modes. The basis for the polynomial regression model is to further reduce the residuals $\mathbf{r}_i$, once a linear mode has been extracted, by fitting a polynomial along the direction of the mode. For example in Fig. 2, b_2 and b_3 could be modelled by polynomials in b_1. This approach succeeds where there is limited non–linear shape variability.

3.1. Tadpole Data

When a PRPDM was trained on the tadpole data, using polynomials of degree 2 (i.e quadratics) for each mode, two modes of variation were sufficient to describe 95% of the variance in the training set [2]. A scattergram of the two shape parameters of the PRPDM is shown in Fig. 7; the shape parameters appear to be more independent than for the linear PDM.

3.2. Watch Data

Training a PRPDM on the watch data yields a model which is very similar to the linear model. We again require two modes of variation to explain 99% of the training data, and the modes of variation cause variation in the length of the minute hand, which is not found in the training data. This occurs even when polynomials of degree 6 are used. Thus the PRPDM fails. The reason for this failure is not difficult to fathom. The formulation of the PRPDM requires that the second eigenvector can be modelled as a function of the first; a circular or nearly circular dataset cannot be modelled in this way. Hence the PRPDM does not give a specific model; as with the linear model trained on the watch data, implausible shapes can be generated from apparently legal combinations of shape parameters.

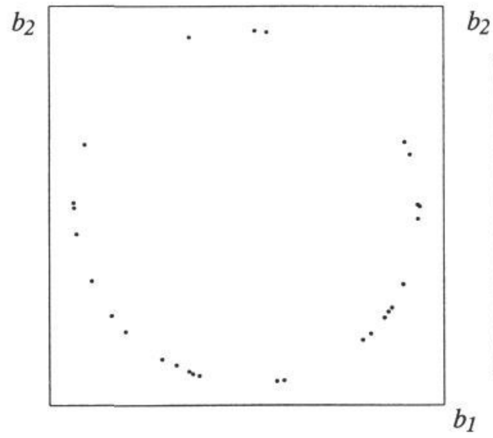

Fig. 6.– Watch data, linear PDM: scattergram of b_2 against b_1.

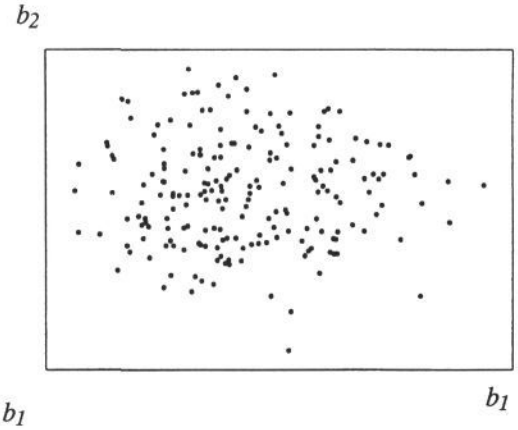

Fig. 7.– Tadpole data, PRPDM: scattergram of b_2 against b_1.

4. THE MULTI–LAYER PERCEPTRON MODEL

As we have seen above, for the PRPDM to work well requires that the second and subsequent eigenvector components of the training data can be modelled satisfactorily as functions of the first. This restricts the classes of variable object to which the model can be applied. A model which is capable of fitting to any arbitrary non–linear structure in the data is preferable.

The use of multilayer perceptrons for carrying out non–linear principal component analysis has been described by Kramer [3]. His approach involves training a multilayer perceptron (MLP) to give a set of outputs which are as close as possible to the inputs, over a training set of examples. We have implemented a 5–layer perceptron of this form. The structure is illustrated in Fig. 8. Consecutive layers are exhaustively linked. The key feature is a middle "bottleneck" layer with a small number of neurons. During signal propagation, the signal must pass through the bottleneck. In order to achieve outputs equal to the inputs, the MLP is forced to code the data into a number of components equal to the number of neurons in the bottleneck layer and thus effects a *non–linear dimension reduction.* We do not specify the representation to be used; instead, the MLP learns the appropriate representation during training. The first layer of the MLP is the input layer. We refer to the second layer as the mapping layer. Increasing the number of neurons in this layer increases the complexity of the non–linear relationship which can be modelled, but at the risk of "overfitting", i.e. learning the stochastic variations in the data. In this way the number of neurons in the mapping layer is analogous to the number of parameters in a regression model. The fourth layer is a "de–mapping" layer. We use the same number of neurons for this as for the mapping layer. The fifth layer is the output layer. We use a linear transfer characteristic for the neurons in the input, bottleneck and output layers. For the mapping and de-mapping layers we use a sigmoid transfer characteristic:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Rather than feeding the landmark point coordinates directly into the MLP, we first reduce the dimensionality by performing a linear principal component analysis on the data and take the first few principal components. Since the sigmoidal neurons work best when the typical range of the data is within the bounds of 0 and 1, we transform the input data to the MLP by shifting and scaling it to have a mean of 0.5 in each component and an overall variance of 0.15 for all the cases described here. Note that we use the same scale factor for all the principal components, i.e we do not "pre-whiten" the data. We store all the parameters of the eigenvector transformation and of the data shift and scaling operations.

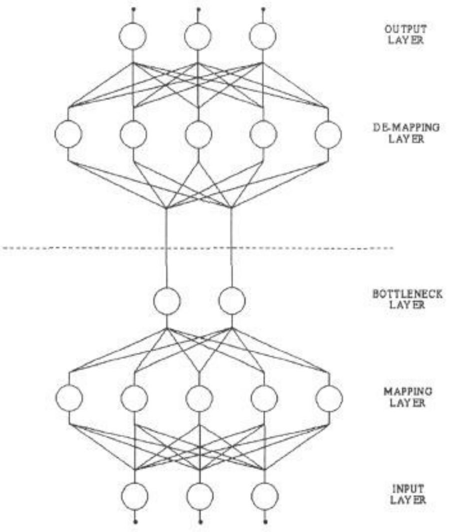

Fig. 8.– Structure of the multi-layer perceptron. The full five layer structure is trained to give a set of inputs which match the outputs as closely as possible over a training set. After training, the MLP is split along the dashed line. The lower part (first three layers) forms the encoder, used to calculate the shape parameters corresponding to a new shape example. The upper part (top two layers) forms the decoder, used to calculate the landmark co-ordinates corresponding to a set of shape parameters.

The weights on the links connecting neurons in successive layers are adjustable. Each neuron also has an input bias. For each neuron in the input layer, the input bias is fixed at zero; it is adjustable for the other neurons. The network is trained by adjusting the weights of the links feeding into and input biases of the second, third, fourth and fifth layers. We train the network using the delta-rule backpropagation algorithm [4]. As suggested by Haykins [5], we use a lower gain nearer the top (output end) of the MLP.

In the examples presented here, we have used a gain of 0.08, 0.04, 0.02 and 0.01 in the second, third, fourth and fifth layers respectively and momentum of 0.9 throughout. We use sequential training, i.e. the weights are adjusted after each training example is presented. The training algorithm seeks to make the errors (differences between inputs and outputs) as small as possible. The training algorithm was normally run for 100,000 presentations of the complete training set; the total sum–of–squares error over all the training examples would typically stop decreasing significantly after between 10000 and 20000 presentations of the complete training data.

Once the MLP has been trained, we split it into an encoder and a decoder (Fig. 8). We use the encoder to obtain the coded representations of the training set. To obtain the shape parameters (b–values) for the training examples, we shift these codes to have a mean of zero in each component. If there is more than one component we apply a further principal component analysis, to ensure that the final shape parameters are linearly independent. The training algorithm may be summarised as follows:

(i) Let the vector of landmark positions for each training example i be denoted by $\mathbf{r}_i$, the mean value of the $\mathbf{r}_i$ be denoted by $\bar{\mathbf{r}}$, and let the covariance matrix of the training set $\{ \mathbf{r}_1, \mathbf{r}_2, \ldots \mathbf{r}_N \}$ about the mean be denoted by $\mathbf{S}$. Find the first t eigenvectors of the covariance matrix of the training data, so as to explain most of the variance in the training data.

(ii) Transform the training data to a coordinate system with the axes lying along the eigenvectors, with t ordinates. The new coordinates can then be labelled as:

$$\mathbf{e}_i \;=\; (e_{i,1}, e_{i,2} \,\ldots\, e_{i,t}) \;=\; \mathbf{P}^{\mathrm{T}}(\mathbf{r}_i - \bar{\mathbf{r}})$$

where $\mathbf{P} = (\mathbf{p}_1 | \mathbf{p}_2 | \ldots | \mathbf{p}_t)$, and $\mathbf{p}_i$ is the ith eigenvector of $\mathbf{S}$.

(iii) Transform the $\mathbf{e}_i$ by multiplying by a scale factor f (which we choose to give an overall variance of 0.15 over the training set) and adding a shift $\mathbf{s}$ (to give a mean of 0.5 in each component over the training set), to give a new vector $\mathbf{q}_i$:

$$\mathbf{q}_i \;=\; f\,\mathbf{e}_i + \mathbf{s}$$

(iv) Set up the MLP, using a five–layer structure of the form shown in Fig. 8. Train the MLP so as to minimise the sum–of squares differences between inputs $\mathbf{q}_i$ and outputs $\mathbf{o}_i$ over the training set.

(v) Split the MLP into an encoder and a decoder (Fig. 8).

(vi) For each training example i, compute the coded representation $\mathbf{c}_i$, obtained by applying the MLP input $\mathbf{q}_i$ to the encoder.

(vii) Compute $\bar{\mathbf{c}}$, the mean of the $\mathbf{c}_i$. Store $\bar{\mathbf{c}}$. Subtract $\bar{\mathbf{c}}$ from each of the $\mathbf{c}_i$, and if the codes have more than one component, carry out a further principal component analysis, to obtain a matrix of $\mathbf{M}$ of eigenvectors. The shape parameters for the training examples are given by

$$\mathbf{b}_i = (\mathbf{c}_i - \bar{\mathbf{c}}) \qquad \text{(one component)}$$

$$\mathbf{b}_i = \mathbf{M}^{\mathrm{T}}(\mathbf{c}_i - \bar{\mathbf{c}}) \qquad \text{(more than one component)}$$

This completes the building of the model.
To compute the shape parameters for a new example shape, we transform the shape coordinates using the eigenvector transformation matrix **P**, and the scale and shift terms *f* and **s** and which were stored during training. We apply the transformed shape data to the encoder, and compute the codes. Finally we apply shift $\bar{\mathbf{c}}$ to the data and, if there is more than one component, the transformation **M**.
To compute a shape instance from a set of shape parameters, we begin, if there is more than one shape parameter, by reversing the final eigenvector transformation **M** of the data. We then add the shift vector $\bar{\mathbf{c}}$ to obtain the coded representation. We then use the decoder (Fig. 8) to obtain the outputs **o**. Finally, we reverse the scaling and shift transformation *f* and **s** and reverse the eigenvector transformation **P**, to obtain a vector of landmark positions.

4.1. Tadpole Data

We have trained an MLPPDM on the tadpole data, using three inputs to the multi-layer perceptron from the initial principal component analysis, six neurons in the mapping and de-mapping layers, and two neurons in the bottleneck layer, giving two modes of variation. These two modes explain more than 95% of the variance in the training data. The model yields modes of variation and a scattergram of shape parameters which are very similar to those of the PRPDM trained on the same data.

4.2. Watch Data

We have trained an MLPPDM on the watch data, using three inputs to the multi-layer perceptron from the initial principal component analysis, six neurons in the mapping and de-mapping layers, and one neuron in the bottleneck layer, giving a single mode of variation. This single mode explains more than 99% of the variance in the training data, and is shown in Fig. 9.

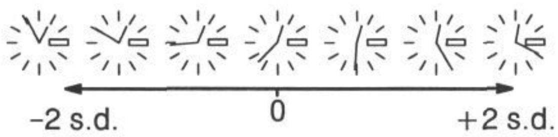

Fig. 9.– Single mode of variation (mean + /– two standard deviations) of MLPPDM trained on the watch data.

The model correctly captures the single degree of freedom of the variation in the data, and hence is a parsimonious model. There is virtually no spurious variation in the length of the hands, thus the model is more specific than the linear and polynomial regression models trained on the same data; it does not allow implausible shapes to be generated.

5. MODEL PERFORMANCE ANALYSIS

5.1. Sensitivity to Number of Neurons in the Mapping and De–mapping Layers

For the both the watch and tadpole examples, increasing the number of neurons in the mapping and de–mapping layers from 6 to 8 does not significantly change the model. When the number is reduced to 4, there is some deterioration of the model – we see some slight spurious variation in the size of the tadpole's head, and the watch model exhibits slight variation in the length of the minute hand, but both models still perform much better than the linear model.

5.2. Behaviour with a Small Number of Training Examples

When we reduce the number of tadpole training examples to 10, both the MLPPDM and the PRPDM display some degradation, in the form of spurious variation in the size of the head and the width of tail. This spurious variation is greater for the PRPDM – i.e. the multi–layer perceptron model performs better. On the watch data, however, the MLPPDM performs poorly with just 10 training examples, displaying wide variation in the length of both the hour and minute hands.

6. DISCUSSION

We have described MLPPDMs – shape models which use a multilayer perceptron to carry out non–linear principal component analysis of data derived from example shapes. We have compared the performance of MLPPDMs with the existing linear and polynomial regression models on two classes of shape data – synthetic tadpoles, which exhibit limited bending, and real data derived from images of a watch, which exhibits full rotation.

With the tadpole data, the linear model fails. Both the polynomial regression and multilayer perceptron models succeed with 200 training examples. The multilayer perceptron model performs better than the polynomial regression model when trained on just 10 examples. With the watch data, both the linear and polynomial regression models fail. The MLPPDM succeeds in generating a specific and parsimonious model with 30 examples, but does not succeed after training with 10 examples. Thus the MLPPDM appears to outperform the PRPDM for modelling non–linear shape variation. Heap and Hogg [6] describe using polar co–ordinates for sub–parts of a PDM, allowing them to rotate. Although potentially useful, this is not as general a model as the MLP–based approach described here.

An MLPPDM is computationally inexpensive to use, but takes significantly longer to train than a linear or polynomial regression PDM. We are investigating more efficient training algorithms.

In our present implementation, the user specifies the number of neurons in each layer of the multi–layer perceptron. The most important choice is the number of neurons in the bottleneck layer. This determines the number of modes of variation, and should in principle be equal to the intrinsic dimensionality of the data. Methods for estimating this dimensionality [7] could thus aid the choice of this number. The model is not overly sensitive to the number of neurons in the mapping and de–mapping layers. It would, however, be useful if the appropriate network configuration could be determined automatically. This could be done by trial and error, building models with different numbers of neurons, and subjecting them to statistical tests for properties such as non-linear dependencies in the data [8-11]. An alternative approach involves building a network with a surplus number of neurons, and using regularization methods to excise neurons that are not needed for explaining the variance in the data [12].

The use of PDMs in image interpretation has been described [1]. Recognition and location of the modelled objects is treated as an optimisation problem - the model instance which is best supported by the evidence is found, leading directly to an interpretation. We are exploring a similar approach to using MLPPDMs for image search. This would allow a broader range of applications to be addressed using flexible shape models.

REFERENCES

1. T.F. Cootes, C.J. Taylor, D.H. Cooper and J. Graham, "Active shape models - their training and application", *Computer Vision and Image Understanding* 61, 38-59 (1995).
2. P.D. Sozou, T.F Cootes, C.J. Taylor and E.C. Di Mauro, "A non-linear generalisation of PDMs using polynomial regression", *Proc. British Machine Vision Conference* 1994 , BMVA press, 397-406 (1994). Also published in *Image and Vision Computing,* 13, 5, 451-457 (1995) .
3. M.A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks", *AIChE Journa*l 37, 233-243 (1991).
4. D. E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning representations by back-propagating errors", *Nature* 323, 533-536 (1986).
5. S. Haykin, *Neural Networks - a Comprehensive Foundation* (MacMillan, New York, 1994).
6. T. Heap and D. Hogg, "Extending the Point Distribution Model Using Polar Co-ordinates", *Internal Report* 95 5, Computer Studies, Leeds University (1995).
7. P.J. Verveer and R.P.W. Duin, "An Evaluation of Intrinsic Dimensionality Estimators", *IEEE Trans. on PAMI* 17, 81-86 (1995).
8. B.D. Ripley, "Tests of 'randomness' in spatial point patterns", *J. R. Statist. Soc.* B 41, 368-374 (1979).
9. P.J. Diggle, *Statistical Analysis of Spatial Point Patterns* (Academic Press, London, 1983).
10. A.D. Barbour and M. Costi, "Correlation tests for non-linear alternatives", *J. R. Statist. Soc.* B 55, 541-548 (1993).
11. A. Azzalini and A. Bowman, "On the use of non-parametric regression for checking linear relationships", *J. R. Statist. Soc.* B 55, 549-557 (1993).
12. D. DeMers and G. Cottrell, "Non-linear dimensionality reduction", *Proceedings of Neural Information Processing Systems* 5, 580-587 (1993).