Modelling Soft Tissue Deformations using Principal Component Analysis and Haptic Devices

Duncan F. Gillies and Michail Bourmpos

Department of Computing, Imperial College of Science, Technology and Medicine,
London, SW7 2BZ, UK, E-mail: dfg@doc.ic.ac.uk , mb1901@doc.ic.ac.uk

ABSTRACT

This paper describes a new technique to incorporate accurate mechanical deformations of soft tissue into interactive training simulators designed to teach medical procedures. The technique is based on pre-computing a representative set of deformations accurately, then encoding these using principal component analysis such that they can be reconstructed quickly using a small number of shape parameters. Learning algorithms provide the link between haptic devices and the shape parameters. The investigation has been carried out with simulated data representing deformations to an elastic sphere, and it has been shown that real time performance with realistic deformations can be achieved.

KEYWORDS

Active Shape models, Haptic devices, simulation training, soft tissue deformation

1. Introduction

Recently much attention has been paid to simulating deformations of soft tissue. Applications that have been proposed include pre-operative prediction, medical instrument design and simulation training using computer graphics. However there is a computational problem in applying modern modelling techniques in simulation training. If the model is to be sufficiently accurate, it cannot be computed in real time. Finite element analysis is a suitable tool for simulation, but to make it work in real time it is necessary to use a small number of elements, and linear elastic behaviour. Under these conditions the results will be highly inaccurate since it has been established that the behaviour of, for example, muscle tissue under deformation is highly non linear and exhibits visco-elastic behaviour and stress relaxation [1]. All these effects can be incorporated in a full finite element simulation, but create huge computational demands.

To get round this problem we are investigating a method whereby correct finite element solutions can be pre-computed and encoded into a compact representation so that they can be used later in an interactive simulation. The difficulty here is that the number of degrees of freedom is great. It is clearly not feasible to pre-compute every deformation that can be produced by manipulating living tissue. However, in a training context it is possible to compute a representative set of deformations that accurately illustrate correct ways of carrying out procedures and train the student in these correct methods.

For example, consider the process of laryngoscopy, in which the tongue is compressed and displaced to the side of the mouth to give the anaesthetist a view of the larynx. The correct procedure utilises only three degrees of freedom: moving the instrument into the mouth, moving down and moving to the right. To achieve a view of the vocal chords the tongue must be deformed in a very particular way. It is feasible to define and compute a comprehensive range of deformed tongue shapes that would be encountered when carrying out this procedure correctly. For training purposes accurate simulation of these deformations is clearly necessary for demonstrating the correct procedure. However other, incorrect, attempts at the procedure need not be represented so accurately, since in training the student will be taught to avoid these actions.

Even with this restriction, we have a problem in data handling and representation. If a shape is represented by a set of points, from which a surface may be reconstructed, then a very large data set is required to represent a reasonable range of deformations. Accordingly we have been investigating the use of principal component analysis to encode the shape deformations.

A further problem concerns the use of a haptic device to simulate a medical instrument and provide tactile feedback. We need to determine how to associate the movements and forces of a haptic device with the deformations found in a simulation of a medical procedure. Our proposed solution is a trainable device to match the haptic device's outputs to the desired parameters used by the algorithm to control the shape of the deformable object. We investigated three possible algorithms to achieve this, namely linear prediction, fuzzy logic and neural networks. In each case the models are trained from known, matching data, so that it can produce a realistic correspondence between any movement of the haptic device and an actual deformation.

The work here investigates these ideas in a simulated application in which an elastic sphere is deformed by applying point forces. The ideas are equally applicable to constructing simulations from the results of comprehensive finite element studies.

2. Shape Models

Principal component analysis (PCA) is a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components [2]. The purpose of PCA is to determine factors that cause variation in the data set. Principal components can be ordered by their contribution to the total variance, the first making the largest contribution. In many practical cases only a few components are required to describe the majority of the variance.

Principal components are linear combinations of the observed variables. For instance, the first principal component in a data set of observed variables X_j, j=1,2,..,p can be written:

$$PC_1 = w_{(1)1}X_1 + w_{(1)2}X_2 + ... + w_{(1)p}X_p$$

where the weights $w_{(1)1}, w_{(1)2}, ..., w_{(1)p}$ are chosen to maximize the ratio of the variance of PC_1 to the total variation, subject to the constraint

$$\sum_{j=1}^{p} w_{(1)j}^2 = 1.$$

Given a set of observations $X' = (X_1, X_1, ..\ X_p)$ with variance-covariance matrix Σ, and for all j zero mean of X_j, the first principal component PC$_1$ is given by the vector of coefficients $W = (w_1, w_2, ..., w_p)$ having the property that the variance of WX' is maximum over the class of all linear combinations of X_j's, and subject to $WW' = 1$. It can be shown that the W coefficients must satisfy the p simultaneous linear equations:

$$(\Sigma - \lambda_{(1)}I)W_{(1)} = 0 \qquad \textbf{(2.1)}$$

where $\lambda_{(1)}$ is the Lagrange multiplier.

If the solution to these equations is to be other than the null vector, the value of $\lambda_{(1)}$ must be chosen so that:

$$|\Sigma - \lambda_{(1)}I| = 0 \qquad \textbf{(2.2)}$$

So we find that $\lambda_{(1)}$ is the largest eigenvalue of Σ and the required solution of W is the corresponding eigenvector of Σ, denoted by $W_{(1)}$. Hence the first principal component can be written as $PC_1 = W_{(1)}^{'}X$. Following the same procedure for the other principal components we see that they are the eigenvectors of Σ ordered by the corresponding eigenvalue.

A shape model can be defined as a vector containing all the coordinate values of a set of characteristic points from which a geometric shape can be re-constructed. Thus if the geometry of an object, for example the tongue, can be constructed from 100 anatomical surface points, then the shape model will have 300 variables being the (x,y,z) values of each point. Given a representative set of data we can compute the co-variance matrix for those 300 points, and hence the principal components.[3-6]

It is possible to represent the same data in either the original variable space or the space formed by the principal components. If we use the vector ***b*** to represent the shape model in the PCA space, and ***m*** to represent the mean of the original shape vectors ***X*** then:

$$b = \Phi^T(X - m). \qquad \textbf{(2.3)}$$

where $\mathbf{\Phi}$ is a matrix whose columns are the principal components of the covariance matrix [2,3]. The inverse transformation is:

$$X = m + \Phi \cdot b \qquad \textbf{(2.4)}$$

By controlling the components of vector ***b*** we can control the actual shape of a deformable object defined by ***X***. Since ***b*** is ordered by the contribution that each principal component makes to the variance, we can represent the majority of the possible shapes that a deformable object can assume by manipulating only a small number of its elements, the rest being set to zero. This is much faster than having to control hundreds of variables representing the actual positions in space of each point on the object. Furthermore, the calculations required are trivial. As we can see from equation (2.4), if t is the number of principal components used, we only need *p.t* multiplications and *p.(t-1)* additions in the calculation of $\Phi \cdot b$ and another *p* additions are required to compute *X*.

3. Haptic device models

We need to establish the relation between the outputs of a haptic device and the shape parameters of a deformable object. Each action on a haptic device corresponds to a specific output, for example a force vector where the magnitude of the force, the direction and the point of action are defined. Having the above vector as input we must somehow produce the vector of shape parameters ***b*** that creates the appropriate realistic deformation. Three possible ways of doing this were investigated.

3.1 Linear Model

We assume that there is a model of the form:

$$\Delta b = A \cdot f + B \qquad \textbf{(2.5)}$$

that describes the relation between the deviation in shape parameters and the force vector output from the haptic device. Second or third order models could also be used.

$$\Delta b = A \cdot f^2 + B \cdot f + C \qquad \textbf{(2.6)}$$

$$\Delta b = A \cdot f^3 + B \cdot f^2 + C \cdot f + D \qquad \textbf{(2.7)}$$

The only thing required after the formulation of the equations is the calculation of matrices A, B, C, D. We can find the optimal value of the matrices, A, B, C and D using the least squares fitting algorithm.

$$x \cong (A^T \cdot A)^{-1} \cdot A^T \cdot B \qquad \textbf{(2.8)}$$

In our case the vector **x** is the vector of coefficients describing the model.

3.2 Fuzzy Models

The Takagi-Sugeno form of a fuzzy model [7] was the second method investigated. It uses the following type of rules:

If

$$X_i(1) \text{ is } A_m^1 \text{ AND} \ldots \text{AND } X_i(r-1) \text{ is } A_m^{r-1}$$

$$\text{then } Y_m = f_i = a_0^m + a_1^m X_i(1) + \ldots + a_{r-1}^m X_i(r-1)$$

In the Takagi-Sugeno models the rule's consequent is a linear function of the input variables. The parameters $a_0^m, \ldots, a_{r-1}^m$ are trained to acquire the best possible values.

The training method used is called ANFIS (Adaptive-Network-based Fuzzy Inference System or Adaptive Neuro-Fuzzy Inference System) [8]. It is an architecture that could operate as a basic stage for the creation of a fuzzy knowledge base, comprised of if-then rules. The membership functions in each rule are set in such a way, that the (input, output) pairs are approximated satisfactorily.

ANFIS applies two techniques in updating parameters. For premise parameters that define membership functions, it employs gradient descent to fine-tune them. For consequent parameters that define the coefficients of each output equation, it uses the least-squares method to identify them. This approach is thus called a hybrid learning method [9,10].

3.3 Neural Network models

An artificial neural network is a network of many simple processors "units", each possibly having a small amount of local memory. The units are connected by communication channels "connections", which usually carry numeric data, encoded by various means. The units operate only on their local data and on the inputs they receive via the connections [11-13].

Most neural networks have a "training" rule whereby the weights of connections are adjusted on the basis of data. If trained carefully, neural networks may exhibit some capability for generalization beyond the training data, that is, to produce approximately correct results for new cases that were not used for training [14-16]. Furthermore, hidden layers enable neural networks to handle complex nonlinear problems. The most widely used training method for feed-forward neural networks is backpropagation[17,18]. In backpropagation 'learning' is the supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection.

4. Experiments

4.1. Shape Models

In order to test the method proposed we created shape models which encode deformations to a sphere. A variety of different deformations to this shape were made, simulating effects such as elastic deformation from an applied force. Different data sets were created to test how noise, the number of variables used and the number of different deformations in the training set effect the method.

Data sets were constructed using 441 and 961 points (1323 and 2883 variables). Up to 10 different deformations were used, simulating for example applied force at point *(x,y,z)* on the surface. For each deformation 11 different stages were constructed for 11 different magnitudes of the applied force. Three stages are shown in figure 3.1.

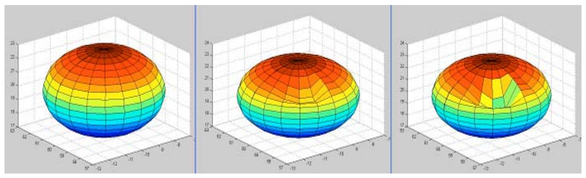


Figure 3.1 Different Stages of one deformation

Each member of a data set is the complete shape vector containing the co-ordinates defining a surface grid of points. A data set contains a member representing each stage of each deformation simulated. The principal components are found for this data.

We need to use sufficient principal components, so that most of the variation of the data is included and the reconstruction of the sphere is achieved with minor errors. We chose the number of components using a 'scree test' (Cattell(1966) [1]). The eigenvalues are plotted in successive order of their magnitude and then an elbow in the curve is identified by applying a straight edge to the bottom portion of the eigenvalues to see where they fit an approximate line. The number of components retained is given by the point, at which the components curve intersects the straight line, formed by the smaller eigenvalues.

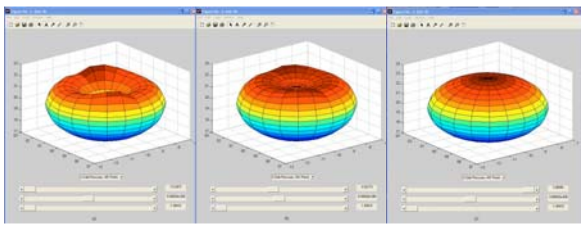
Having decided the number of principal components to retain and assuming that this number is *k*, we form matrix $\mathbf{\Phi}$, whose columns are the eigenvectors corresponding to the *k* largest eigenvalues of covariance matrix. Then using equation (2.3) we calculate all values of vector ***b***, the shape parameter vector, for every deformation in the data set. Finding the maximum and minimum value of each one of the *k* parameters in vector ***b***, we can determine the range of values the shape parameters have. By assigning the values of each one of the shape parameters to a separate slide bar, we change their values and reconstruct the sphere using equation (2.4).

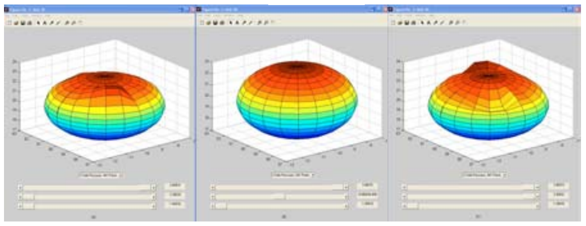
4.2. Creating the Haptic Device

In order to train and test the models that represent the relation between a haptic device and the appropriate shape parameters we had to create data sets of force vectors. For each of the deformations defined we specified a force vector, consisting of the point of application and the magnitude. In practice, this information could all be obtained from finite element simulations. We used just simulated data for this study. A further simplification was to fix the direction of the applied force, so that it could be specified by four values rather than six.

Applying the least squares algorithm we constructed the first, second and third order linear models, for the above data sets. For the fuzzy model, we first construct the actual model using 4 inputs (the elements of the force vector) and 1 output for a single shape parameter. This means that *k* different fuzzy models are created for the *k* shape parameters in vector ***b***. The number of rules in each model is heuristically decided so that it remains small but accurate. Using ANFIS we now train each model to produce the final trained fuzzy models.

Finally, for the neural network we must choose the number of layers, the number of neurons in each layer and the activation functions. Four input neurons and *k* output neurons were used. We chose 2 hidden layers with 7 neurons for the first layer and 9 for the second. The activation functions in each layer were chosen to be 'tan-sig' functions. Applying backpropagation we trained the neural network to acquire the final model.


(a)


(b)

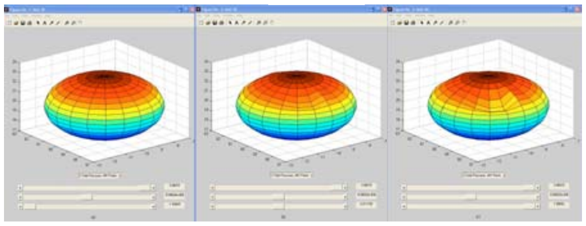

(c)

Figure 4.1 Shape reconstruction from data with four different deformations. (a) Shows variation in the first component magnitude, (b) the second and (c) the third

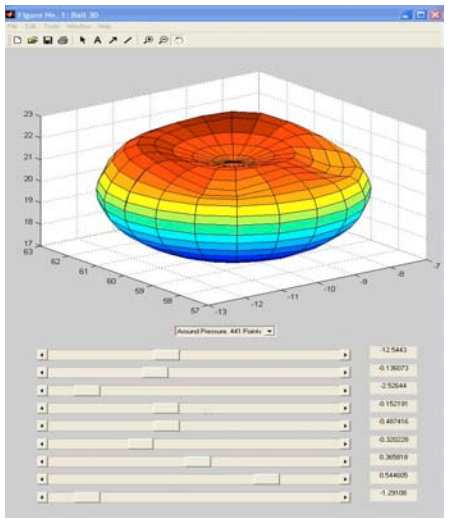


Figure 4.2 Complex shape reconstruction combining 10 different deformations

4.3 Reconstructions

The reconstructions can be studied using slider bars to set the values of the principal components. Not surprisingly the number of principal components that we need to produce an accurate reconstruction corresponds to the number of different deformations that we included in the data. Figure 4.1 shows some typical results. Here we created a shape model that included four different deformations. The images show the effect of varying the value of one principal component while keeping the other fixed. Figure 4.2 shows complex deformations that can be achieved by combining ten individual deformations.

The data sets used for figures 4.1 and 4.2 represent the sphere using 441 points on the surface. We have observed that, when the number of deformations in the data sets remains the same but the points representing the surface of the sphere changes, in our case from 441 to 961, the number of principal components extracted remains the same. Furthermore, each component extracted from the first data set controls the shape of the sphere in exactly the same way, although the sign of the component may be different.

All data sets created and used have been corrupted with white noise of some small magnitude. The errors we face, having added noise, are of rank 10^{-5}. These tests indicate that reasonable noise in the data does not affect PCA and in turn the algorithm.

Finally, we will discuss the calculations required and the speed of reconstructing any deformation, in order to prove the suitability of the proposed technique for real-time 3D graphical representations. The early calculations of the covariance matrix, its eigenvectors and eigenvalues are very time-consuming. The time rises with the number of points used to reconstruct the 3D object. However, this is an 'off-line', one-time only calculation and therefore does not give us any problems for real-time reconstruction. The calculations required for this real-time reconstruction are a matrix-vector multiplication and a vector addition. The actual time for these calculations was measured to be less then 0.01sec, even for the larger data set.

Therefore we can safely conclude that PCA could be effectively used for real-time graphical representations of 3D deformable objects.

4.4 Reconstruction from a haptic device

We shall now examine the behaviour of the models used to match the artificial outputs of a haptic device to the desired values of shape parameters and in turn into reliable graphical representations.

The results of the linear prediction models were disappointing. The error in some cases was huge and the model did not manage to capture the behaviour of our system.

For the fuzzy model the results were much more encouraging. The estimates produced were satisfactory as shown in figure 4.6. The mean squares error was small and measured at 0.003. As a result the graphical representation of the deformation of the sphere (figure 4.7) is more than satisfactory.

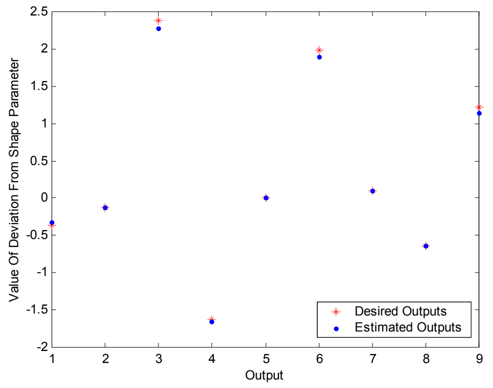


Figure 4.6: Desired and estimated output of the fuzzy model

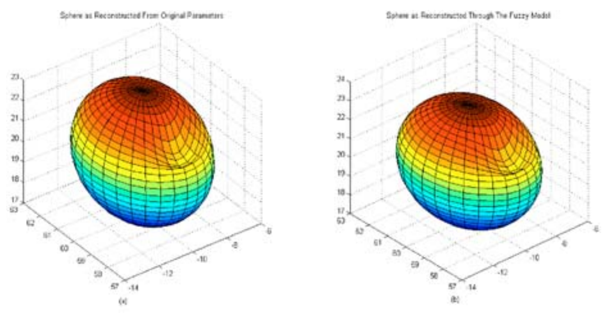

Sphere as Reconstructed From Original Parameters
Sphere as Reconstructed Through The Fuzzy Model
(a)
(b)

Figure 4.7: Reconstructions of the sphere from original (a) and estimated (b) data, using a fuzzy model

The time taken to use the fuzzy model and reconstruct the object is small. The time to load the models was approximately 0.45sec, but this is a one-time only procedure occurring at the beginning of the simulation. Therefore this delay is acceptable. The time delay of simulating the fuzzy model and reconstructing the three dimensional data was less than 0.01sec. This demonstrates that the algorithm can provide real-time 3D graphical representations.

The neural network results were the most successful (figure 4.8). The mean squared error was 9.10^{-5} and as a result the graphical representations are almost identical (figure 4.9).

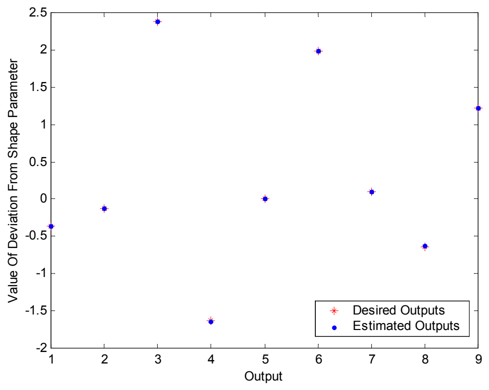


Figure 4.8: Desired and estimated output of the neural network

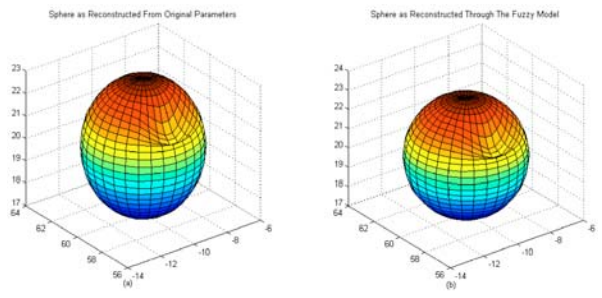

Sphere as Reconstructed From Original Parameters
Sphere as Reconstructed Through The Fuzzy Model
(a)
(b)

Figure 4.9: Reconstructions of the sphere f original (a) and estimated (b) data, using a ne

Loading the neural network is much faster than loading the fuzzy model. However, the time required to simulate the neural network, produce the estimates and reconstruct the three-dimensional data rises to something over 0.01secs. However it is still acceptable for real time performance.

We must note that both the fuzzy model and the neural network have been tested with great success on data used, or similar to those used, during training. However, their response to a totally unknown input force vector is not so satisfactory. For the fuzzy model the output might not be calculated at all, be of zero value - if no rule is fired - or have great errors and in some cases become unstable. In those cases, the shape parameters possibly obtain values out of their feasible range.

On the other hand, the neural network will never become unstable, since the range of parameters is required for the creation of the neural network. Therefore, even for values of the parameters that have errors, stability will be maintained.

We believe that the above problem could be easily overcome if the identification of the models was done using larger data sets. In that case the models would grow, either in terms of rules for the fuzzy model, or in terms of neurons within the hidden layers for the neural network.

Computational costs would also increase, but are likely to remain affordable.

5. Conclusions

Principal component analysis proved to be a powerful tool for reducing the variables needed to create a three-dimensional graphical representation of a deformable object. We have demonstrated how, using PCA, we can control the shape of an artificial ball, in a fast, accurate and realistic manner.

We have examined how the resulting system behaves under data sets containing noise and have reached the conclusion that noise - in reasonable levels - does not affect the performance of the algorithm. Therefore, the reconstruction of the shape of an object using PCA produces insignificant errors, even when using noisy data sets.

Reconstruction of the original variables turned out to be very fast, allowing for the algorithm to be used in real-time three-dimensional graphical representations.

A procedure is necessary for translating the outputs of a haptic device into suitable shape parameters, so that a realistic correspondence between actions on the device and the deformations of the object would be achieved. Three models were tested as means of automating this procedure. Both fuzzy model and a neural network proved to be able of producing very good estimates and in an extremely fast manner.

Finally we can conclude that the results of this work encourage us to believe that the ultimate goal of the training simulator is in grasp for the near future.

The next stage of the work is to combine the results of our finite element studies, in which we identified an accurate way to simulate the compression of the tongue [1], with the present techniques in order to build a proper simulator for teaching laryngoscopy. This will enable us to explore the ideas more fully.

REFERENCES

[1] Rodrigues M A F, Gillies D F and Charters P "A biomechanical Model of the Upper Airways for Simulating Laryngoscopy" Computer Methods in Biomechanics and Biomedical engineering **4**(2)127-148 (2001)

[2] W.R. Dillon and M. Goldstein, *'Multivariate Analysis: Methods And Applications'*, John Wiley & Sons, 1984

[3] A.F. Frangi, D. Rueckert, J.A. Schnabel and W.F. Niessen, *'Automatic 3D ASM Construction via Atlas-based Landmarking and Volumetric Elastic Registration'*

[4] D. Rueckert, A.F. Frangi and J.A. Schnabel, *'Automatic Construction of 3D Statistical Deformation Models of the Brain using Non-rigid Registration '*, 2001

[5] T.F. Cootes, C.J. Taylor, D.H. Cooper and J. Graham, *'Active Shape Models - Their Training and Application'*, Computer Vision And Image Understanding, Vol. 61, No 1, pp 38-59, January 1995

[6] A. Lanitis, C.J. Taylor and T.F. Cootes, *'Automatic Interpretation and Coding of Face Images using Flexible Models'*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No 7, pp 743-756, July 1997

[7] T. Takagi and M. Sugeno, *'Fuzzy Identification of Systems and it's Application to Modeling and Control'*, IEEE Transactions on Systems, Man and Cybernetics, vol. 15, pp. 116-132, 1985

[8] Jyh–Shing Roger Jang, *'ANFIS : Adaptive – Network – Based Fuzzy Inference System'*, IEEE Transactions on Systems, Man and Cybernetics, vol. 23, No 3, May/June 1993

[9] Li–Xin Wang and J.M. Mendel, *'Back – Propagation Fuzzy System As Nonlinear Dynamic System Identifiers'*, Proc. of 1992 IEEE International Conference on Fuzzy Systems, San Diego, CA, pp. 1283-1290, March 8-12, 1992

[10] C.T. Ln and C.S.G. Lee, *'Real-Time Supervised Structure/Parameter Learning For Fuzzy Neural Network'*, Proc. of 1992 IEEE International Conference on Fuzzy Systems, San Diego, CA, pp. 1283-1290, March 8-12, 1992

[11] M.T. Hagan, , H.B. Demuth, and M.H. Beale, *'Neural Network Design'*, Boston, MA: PWS Publishing, 1996.

[12] J.A.Anderson, *'An Introduction to Neural Networks'*, Cambridge, The M.I.T. Press, 1995

[13] S. Haykin, *'Neural Networks: A Comprehensive Foundation',* Prentice Hall, 1999

[14] R.P. Lippman, *'An introduction to computing with neural nets'*, IEEE ASSP Magazine, pp. 4-22, 1987.

[15] C.M. Bishop, *'Neural Networks for Pattern Recognition',* Oxford University Press, 1995

[16] J. Hertz, A. Krogh, and R.G. Palmer, *'Introduction to the Theory of Neural Computation',* Addison-Wesley, 1991

[17] S. Russell and P. Norvig, *'Atrificial Intelligence, A Modern Approach'*, Prentice Hall, 1995

[18] D. E. Rumelhart, G. E. Hinton and R. J. Williams, *'Learning internal representations by error propagation'*, Parallel Data Processing, vol.1, Cambridge, MA: The M.I.T. Press, pp. 318-362, 1986