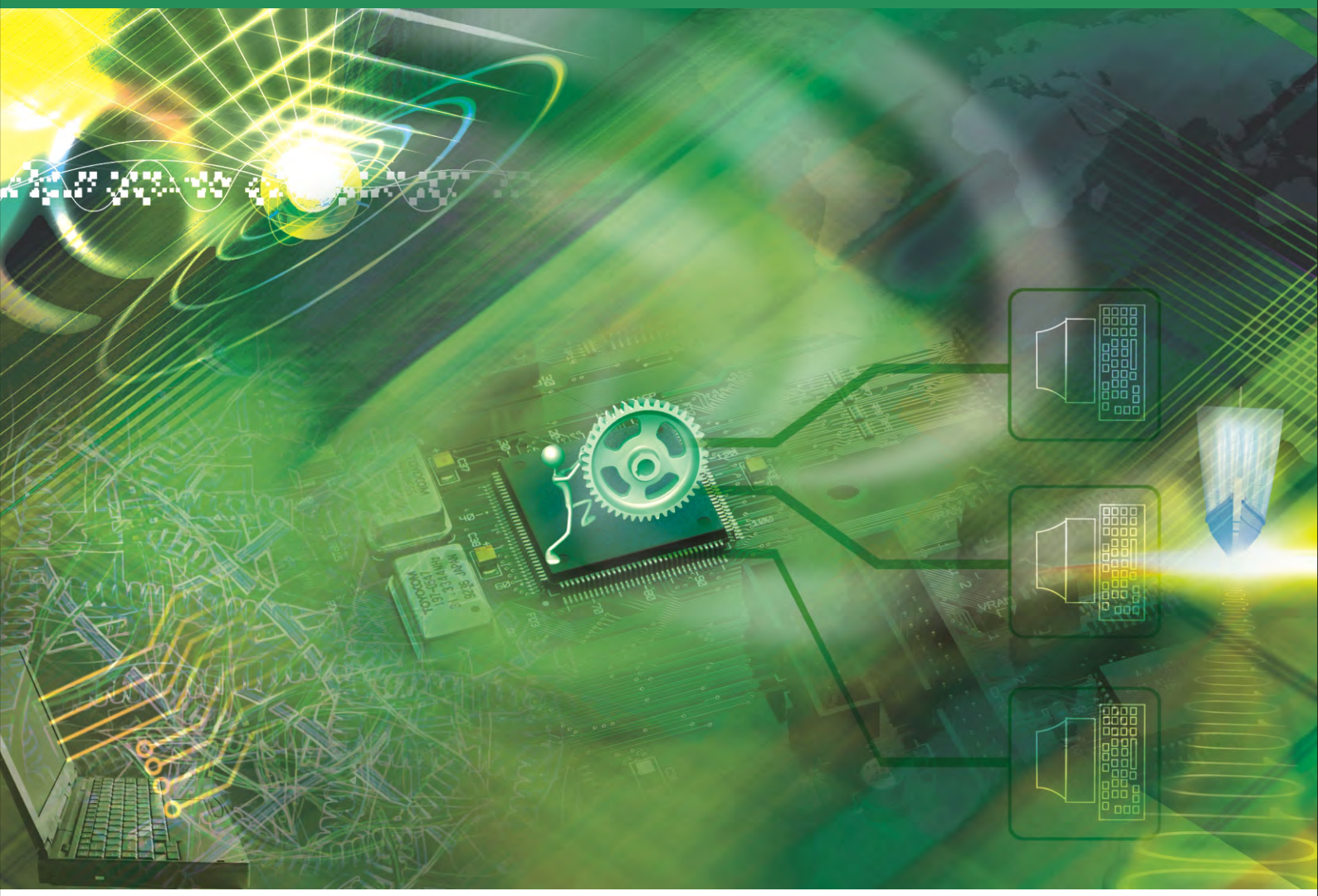
ISSN: 2150-8402 (Print), 2150-8410 (Online) Volume 2, Number 1, February 2010

Journal of Intelligent Learning Systems and Applications



www.scirp.org/journal/jilsa

Scientific Research

Journal Editorial Board

ISSN: 2150-8402 (Print), 2150-8410 (Online)

http://www.scirp.org/journal/jilsa

Editor-in-Chief

Editorial Board

Journal of Intelligent Learning Systems and Applications, **2010**
Published Online February 2010 in SciRes (http://www.SciRP.org/journal/jilsa)

CONTENTS

Vol. 2 No. 1 **February 2010**

Journal of Intelligent Learning Systems and Applications (JILSA)

Journal Information

SUBSCRIPTIONS

Journal of Intelligent Learning Systems and Applications (Online at Scientific Research Publishing, www.SciRP.org) is published quarterly by Scientific Research Publishing, Inc., USA.

E-mail: service@scirp.org

Subscription rates: Volume 2 2010

Print: $50 per copy.

Electronic: free, available on www.SciRP.org.

To subscribe, please contact Journals Subscriptions Department, E-mail: service@scirp.org

Sample copies: If you are interested in subscribing, you may obtain a free sample copy by contacting Scientific Research Publishing, Inc. at the above address.

SERVICES

Advertisements

Advertisement Sales Department, E-mail: service@scirp.org

Reprints (minimum quantity 100 copies)

Reprints Co-ordinator, Scientific Research Publishing, Inc., USA.

E-mail: service@scirp.org

COPYRIGHT

PRODUCTION INFORMATION

For manuscripts that have been accepted for publication, please contact:

E-mail: jilsa@scirp.org

Journal of Intelligent Learning Systems and Applications, **2010, 2: 1-11**
doi:10.4236/jilsa.2010.21001 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

A New Neural Network Structure: Node-to-Node-Link Neural Network

S. H. Ling

Centre for Health Technologies, University of Technology Sydney, Sydney, Australia
Email: steve.ling@uts.edu.au

Received October 17th, 2009; accepted January 8th, 2010.

ABSTRACT

This paper presents a new neural network structure and namely node-to-node-link neural network (*N-N-LNN*) *and it is trained by real-coded genetic algorithm* (*RCGA*) *with average-bound crossover and wavelet mutation* [1]. *The N-N-LNN exhibits a node-to-node relationship in the hidden layer and the network parameters are variable. These characteristics make the network adaptive to the changes of the input environment, enabling it to tackle different input sets distributed in a large domain. Each input data set is effectively handled by a corresponding set of network parameters. The set of parameters is governed by other nodes. Thanks to these features, the proposed network exhibits better learning and generalization abilities. Industrial application of the proposed network to hand-written graffiti recognition will be presented to illustrate the merits of the network.*

Keywords*: *Genetic Algorithm, Hand-Written Graffiti Recognition, Neural Network

1. Introduction

Neural networks can approximate any smooth and continuous nonlinear functions in a compact domain to an arbitrary accuracy [2]. Three-layer feed-forward neural networks have been employed in a wide range of applications such as system modelling and control [2], load forecasting [3–5] prediction [6], recognition [1,4,5,7–10], etc. Owing to its specific structure, a neural network can realize a learning process [2]. Learning usually consists of two steps: designing the network structure and defining the learning process. The structure of the neural net-work affects the non-linearity of its input-output relationship. The learning algorithm governs the rules to optimize the connection weights. A typical structure has a fixed set of connection weights after the learning process. However, a fixed set of connection weights may not be suitable to learn the information behind the data that are distributed in a vast domain separately.

Traditionally, two major classes of learning rules, namely the error correction rules [11] and gradient methods [2], were used. The error correction rules [11], such as the α-LMS algorithm, perception learning rules and May's rule, adjust the network parameters to correct the network output errors corresponding to the given input patterns. Some of the error correction rules are only applicable to linear separable problems. The gradient rules [2], such as the MRI, MRII, MRIII rules and back-propagation techniques, adjust the network parameters based on the gradient information of the cost function. One major weakness of the gradient methods is that the derivative information of the cost function is needed, meaning that it has to be continuous and differentiable. Also, the learning process is easily trapped in a local optimum, especially when the problem is multimodal and the learning rules are network structure dependent. To tackle this problem, some global search evolutionary algorithms [12], such as the real-coded genetic algorithm (RCGA) [1,13], is more suitable for searching in a large, complex, non-differentiable and multimodal domain. Recently, neural or neural-fuzzy networks trained with RCGA are reported [5,6,14]. The same GA can be used to train many different networks regardless of whether they are feed-forward, recurrent, wavelet or of other structure types. This generally saves a lot of human efforts in developing training algorithms for different types of networks.

In this paper, modifications are made to neural networks such that the parameters of the activation functions in the hidden layer are changed according to the network inputs. To achieve this, node-to-node links are introduced in the hidden layer. The node-to-node links interconnect the hidden nodes with connection weights. The structure of the N-N-LNN is shown in Figure 1.

Conceptually, the introduction of the node-to-node links increases the degree of freedom of the network model. It should be noted that the way to realize the node-to-node links is also governed by the tuning algorithm. The resulting neural net-work is found to have better learning and generalization abilities. The enhancements are due to the fact that the parameters in the activation functions of the hidden nodes are allowed to change in order to cope with the changes of the network inputs of different operating sub-domains. As a result, the N-N-LNN seems to have a dedicated neural network to handle the inputs of different operating sub-domain. This characteristic is good for tackling problems with input data sets distributed in a large spatial domain. In this paper, hand-written graffiti recognition (which is a pattern recognition problem with a large number of data set) is given to show the merit of the proposed network. The proposed network is found to perform well experimentally.

This paper is organized as follows: The N-N-LNN will be presented in Section 2. In Section 3, the training of the parameters of the N-N-LNN using RCGA [1] will be discussed. The application example on hand-written graffiti recognition system will be given in Section 4. A conclusion will be drawn in Section 5.

2. Node-to-Node Link Neural Network Model

A neural network with node-to-node relations between nodes in the hidden layer is shown in Figure 1. An inter-node link with weight $\tilde{m}_i$ is connected from the $(i+d_m)$-th node to the i-th node. Similarly, an inter-node link with weight $\tilde{r}_i$ is connected from the $(i-d_r)$-th node to the i-th node, i = 1, 2, …, n_h.. d_m and d_r are the node-to-node distance, i.e. if d_m =2, the link with weight $\tilde{m}_3$ will be connected from node 5 to node 3. Similarly, if d_r = 3, the link with weight $\tilde{r}_6$ will be connected from node 3 to node 6. As a result, the total number of node-to-node links is $2\times n_h$, where n_h is the total number of hidden nodes. An example of the node-to-node link connections is shown in Figure 2. The node-to-node relationship enhances the degree of freedom of the neural network model if it is made adaptive to the changes of the inputs. Consequently, the learning and the generalization abilities of the N-N-LNN can be increased.

Figure 3 illustrates the inadequacy of a traditional neural network. In this figure, S1 and S2 are the two sets of data in a spatial domain. To solve a mapping problem using a neural network, the weight of the network can be trained to minimize the error between the network outputs and the desired values. However, the two data sets

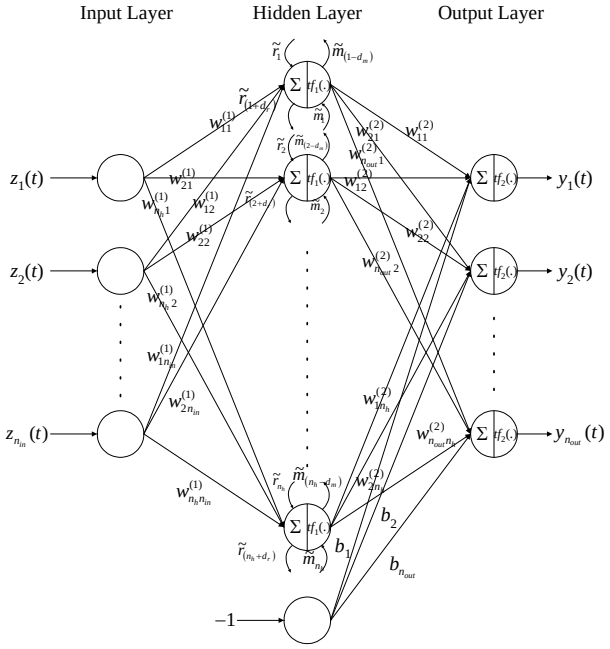


Figure 1. Variable node-to-node link neural network

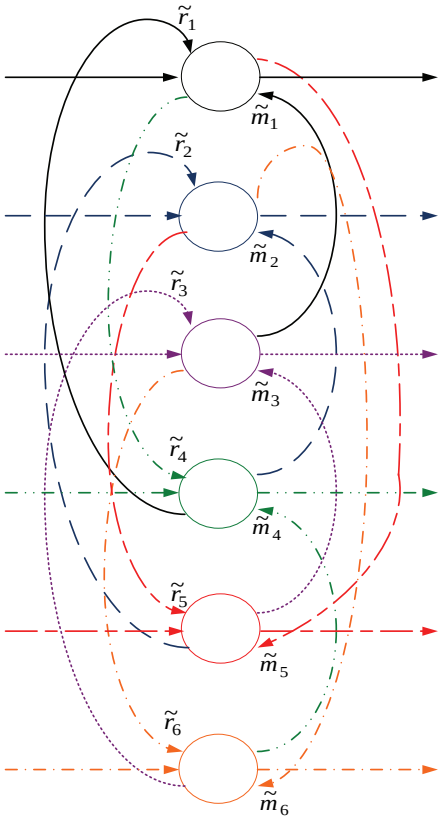


Figure 2. Example node-to-node link connections in the hidden layer (number of hidden node = 6, d_m =2, d_r = 3)

are separated too far apart for a single neural network to

model. As a result, the neural network may only model the data set S (average of S1 and S2) after the training (unless we employ a large number of network parameters.) To improve the learning and generalization abilities of the neural network, the proposed N-N-LNN adopts a structure as shown in Figure 4. It consists of two units, namely the parameters-set (PS) and the data-processing (DP) neural networks. The PS is realized by the node-to-node links which store the parameters (m, r are the parameters which will be described later) governing how the DP neural network handles the input data. Referring back to Figure 3, when the input data belongs to S1, the PS will provide the parameters (network parameters corresponding to S1) for the DP neural network to handle the S1 data. Similarly, when the input data belongs to S2, the DP neural network will obtain another set of parameters to handle them. In other words, it operates like two individual neural networks handling two different sets of input data. Consequently, the proposed N-N-LNN is suitable for handling large numbers of data.

Referring to Figure 1, $z(t)=\begin{bmatrix} z_1(t) & z_2(t) & \cdots & z_{n_{in}}(t) \end{bmatrix}$ denotes the input vector; n_{in} denotes the number of input nodes; t denotes the current input number which is a non-zero integer; $w_{ij}^{(1)}$, i = 1, 2, …, n_h; j = 1, 2, …, n_{in}, denote the connection weights between the j-th node of the input layer and the i-th node of the hidden layer; n_h denotes the number of hidden nodes; $w_{ki}^{(2)}$, k = 1, 2, …, n_{out}; i = 1, 2, …, n_h, denote the connection weights between the i-th node of the hidden layer and the k-th node of the output layer; nout denotes the number of output nodes. $\tilde{m}_i$ and $\tilde{r}_i$ are the connection weights of the links between hidden nodes (there are $2n_h$ inter-node links); d_m is the node-to-node distance between the ($i+d_m$)-th node and the i-th node, d_r is the node-to-node distance between the i-th node and the ($i-d_r$)-th node. b_k denotes the bias of the output nodes; $tf_1(\cdot)$ and $tf_2(\cdot)$ denote the activation functions of the hidden and output nodes respectively. $\mathbf{y}(t)=\begin{bmatrix} y_1(t) & y_2(t) & \cdots & y_{n_{out}}(t) \end{bmatrix}$ denotes the output vector. The input-output relationship of the proposed neural network is governed by the following equation:

$$y_k(t)=tf_2\left(\sum_{i=1}^{n_h} w_{ki}^{(2)} f_{s_i}(\mathbf{z}(t))-b_k\right),\ \mathrm{k}=1,2,\ldots,n_{out} \quad (1)$$

Figure 5 shows the proposed neuron at node i of the hidden layer. Its output $f_{s_i}(\cdot)$ is given by

$$f_{s_i}(\mathbf{z}(t))=tf_1\left(\chi_i(t),\tilde{m}_i(t),\tilde{r}_i(t)\right),\ i=1,2,\ldots,n_h \quad (2)$$

$$\chi_i(t)=\sum_{j=1}^{n_{in}} w_{ij}^{(1)} z_j(t), \quad (3)$$

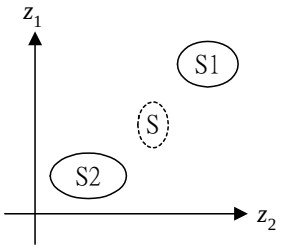


Figure 3. Diagram showing two sets of data in a spatial domain

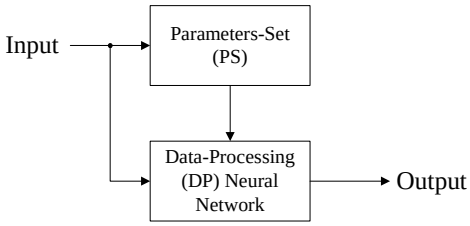


Figure 4. Proposed structure of the neural network

$$\tilde{m}_i(t)=m_i\sum_{j=1}^{n_{in}} \tilde{w}_{(i+d_m)j}^{(1)} z_j(t), \quad (4)$$

$$\tilde{r}_i(t)=r_i\sum_{j=1}^{n_{in}} \tilde{w}_{(i-d_r)j}^{(1)} z_j(t), \quad (5)$$

where m_i and r_i are parameters to be trained. Referring to Figure 4, these parameters are stored in the PS.

$$\tilde{w}_{(i+d_m)j}^{(1)}=\begin{cases} w_{((i+d_m)-n_h)j}^{(1)} & \text{for } i+d_m>n_h \\ w_{(i+d_m)j}^{(1)} & \text{otherwise} \end{cases}, \quad (6)$$

$$\tilde{w}_{(i-d_r)j}^{(1)}=\begin{cases} w_{(n_h+(i-d_r))j}^{(1)} & \text{for } i-d_r<1 \\ w_{(i-d_r)j}^{(1)} & \text{otherwise} \end{cases}, \quad (7)$$

$$tf_1\left(\chi_i(t),\tilde{m}_i(t),\tilde{r}_i(t)\right)=\frac{2}{1+e^{-\left(\frac{(\chi_i(t)-\tilde{m}_i(t))}{2(\tilde{r}_i(t))^2}\right)}}-1$$

$$=\frac{2}{1+e^{-\left(\frac{\left(\sum_{j=1}^{n_{in}} w_{ij}^{(1)} z_j(t)-m_i\sum_{j=1}^{n_{in}} \tilde{w}_{(i+d_m)j}^{(1)} z_j(t)\right)}{2\left(r_i\sum_{j=1}^{n_{in}} \tilde{w}_{(i-d_r)j}^{(1)} z_j(t)\right)^2}\right)}}-1\in[-1\quad 1], \quad (8)$$

$tf_2(\cdot)$ can be any commonly used activation function such as the purely linear, hyperbolic tangent sigmoid, or logarithmic sigmoid functions [2,11]. As mentioned earlier, the node-to-node links enhance the degree of freedom of the modelled function. In each neuron of the hidden layer, the input from the lower neighbour's output ($\tilde{m}_i(t)$) influences the bias term while the input from the upper

neighbour's output ($\tilde{r}_i(t)$) influences the sharpness of the edges of the hyper-planes in the search space. It can be seen from (8) that the proposed activation function $tf_1(\cdot)$ is characterized by the varying mean ($\tilde{m}_i(t)$) and the varying standard deviation ($\tilde{r}_i(t)$) respectively. Their values will be changed according to changes in the network inputs. Figure 6 shows that the means control the bias while Figure 7 shows that the standard deviations control the sharpness. Referring to Figure 3, when the input data belongs to S1, the corresponding $\chi_i(t)$ will drive the other nodes (with $\tilde{m}_{(i-d_m)}$ and $\tilde{r}_{(i+d_r)}$) to manipulate the characteristics of the S1 data. Similarly, when the input data belongs to S2, the corresponding $\chi_i(t)$ will drive the other nodes to handle it accordingly. Figure 8 explains the operating principle of the proposed neuron. In this figure, P1, P2, and P3 are three sets of input patterns. $\hat{P}_r1$, $\hat{P}_r2$, and $\hat{P}_r3$ are the inputs from the upper neighbour with the corresponding input patterns. Similarly, $\hat{P}_m1$, $\hat{P}_m2$, and $\hat{P}_m3$ are the inputs from the lower neighbour with the corresponding input patterns. When the proposed neuron manipulates the input pattern P1, the shape of the activation function is characterized by $\hat{P}_r1$ and $\hat{P}_m1$, and eventually outputs the pattern P'1. Similarly, when the neuron manipulates the input pattern P2, the shape of the activation function is characterized by $\hat{P}_r2$ and $\hat{P}_m2$. So, the activation function is variable and is dynamically dependent on the input pattern. Hence, the degree of freedom of the modelled function is increased. Comparing with the conventional feed-forward neural network, the N-N-LNN should be able to offer a better performance. In the N-N-LNN, the values of the parameters $w_{ij}^{(1)}$, $w_{ki}^{(2)}$, m_i, r_i, b_k, d_m and d_r are trained by an improved RCGA [1].

3. Network Parameters Tuned by Real-Coded Genetic Algorithm

In this paper, all parameters of the neural networks are trained by the improved RCGA with average-bound crossover and wavelet mutation [1]. The RCGA process is as follows: First, a set of population of chromosomes P = [P_1, P_2, …, P_{pop_size}] is created (where pop_size is the number of chromosomes in the population). Each chromosome **p** contains some genes (variables). Second, the chromosomes are evaluated by a defined fitness function. The better chromosomes will return higher values in this process. The form of the fitness function depends on the application. Third, some of the chromosomes are selected to undergo genetic operations for reproduction by the method of normalized geometric ranking. Fourth,

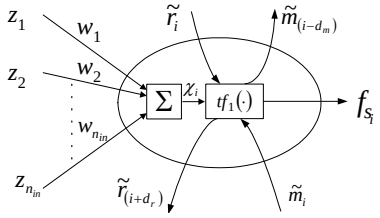


Figure 5. Proposed neuron at node *i* of the hidden layer

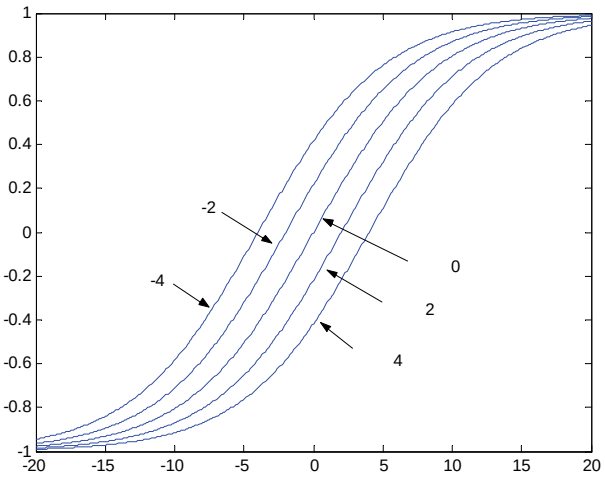


Figure 6. Samples of the activation function $tf_1(\cdot)$ of the proposed neuron with different $\tilde{m}_i$ ($\tilde{r}_i = 0$)

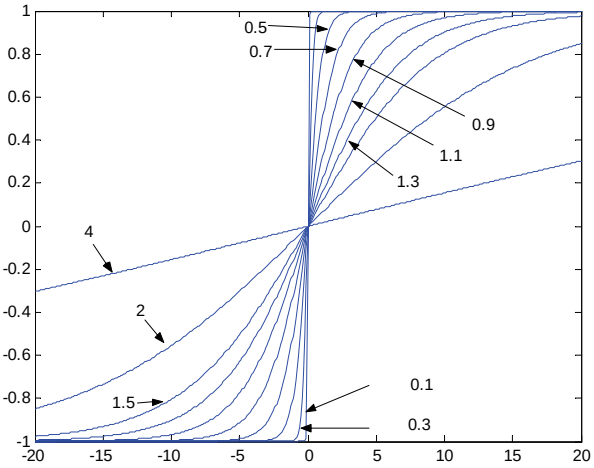


Figure 7. Samples of the activation function $tf_1(\cdot)$ of the proposed neuron with different $\tilde{r}_i$ ($\tilde{m}_i = 0$).

genetic operations of crossover are performed. The crossover operation is mainly for exchanging information between the two parents that are obtained by the selection operation. In the crossover operation, one of the parameters is the probability of crossover μ_c which gives us the expected number of chromosomes that undergo

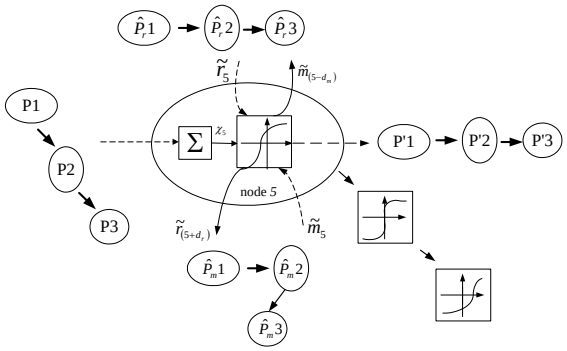


Figure 8. Operating example of the proposed neuron with 3 set of data patterns for hidden node 5

the crossover operation. Crossover operation in [1] is described as follows: 1) four chromosomes (instead of two in the conventional RCGA) will be generated; 2) the best two offspring in terms of the fitness value are selected to replace their parents. The crossover operation is called the average-bound crossover (ABX), which combines the average crossover and bound crossover. The average crossover manipulates the genes of the selected parents, the minimum, and the maximum possible values of the genes. The bound crossover is capable of moving the offspring near the domain boundary. On realizing the ABX operation, the offspring spreads over the domain so that a higher chance of reaching the global optimum can be obtained. After the crossover operation, the mutation operation follows. It operates with the parameter of the probability of mutation (μ_m), which gives an expected number ($\mu_m \times pop\_size \times no\_vars$) of genes that undergo the mutation. The mutation operation is for changing the genes of the chromosomes in the population such that the features inherited from their parents can be changed. The mutation operation is called the wavelet mutation (WM), which applies the wavelet theory to realize the mutation. Wavelet is a tool to model seismic signals by combining dilations and translations of a simple, oscillatory function (mother wavelet) of a finite duration. The wavelet function has two properties: 1) the function integrates to zero, and 2) it is square integrable, or equivalently has finite energy. Thanks to the properties of the wavelet, the convergence and solution stability are improved. After going through the mutation operation, the new offspring will be evaluated using the fitness function. The new population will be reproduced when the new offspring replace the chromosomes with the smallest fitness value. After the operations of selection, crossover and mutation, a new population is generated. This new population will repeat the same process iteratively until a defined condition is met.

One superior characteristic of RCGA is that the detailed information of the nonlinear system to be optimized, e.g. the derivative of the cost function, need not been known. Hence, RCGA is suitable for handling complex optimization problems. In this paper, RCGA is employed to optimize the fitness function characterized by the network parameters of the N-N-LNN. The fitness function is a mathematical expression that quantitatively measures the performance of the RCGA tuning process. A larger fitness value indicates a better tuning performance. By adjusting the values of the network parameters, the fitness function is maximized (the cost value is minimized) based on the RCGA. During the tuning process, offspring with better fitness values evolve. The mutation operation will contract gradually with respect to the iteration number. After the tuning process, the obtained network parameter values will be used by the proposed neural network. As the proposed neural network is a feed-forward one, the outputs are bounded if its inputs are bounded, which happens for most of the real-life applications. Consequently, no convergence problem is present for the neural network itself.

The input-output relationship of the proposed N-N-LNN can be described by,

$$\mathbf{y}^d(t) = g\left(\mathbf{z}^d(t)\right),\ t = 1, 2, \ldots,\ \ n_d\ ., \qquad (9)$$

where

$$\mathbf{z}^d(t) = \left[z_1^d(t) \quad z_2^d(t) \quad \cdots \quad z_{n_{in}}^d(t)\right]$$

and

$$\mathbf{y}^d(t) = \left[y_1^d(t) \quad y_2^d(t) \quad \cdots \quad y_{n_{out}}^d(t)\right]$$

are the given inputs and the desired outputs of an unknown nonlinear function $g(\cdot)$ respectively; n_d denotes the number of input-output data pairs. The fitness function of the RCGA depends on the application. The most common fitness function is given by,

$$fitness = \frac{1}{1+err}, \qquad (10)$$

where *err* is the error.

The objective is to maximize the fitness value of (10) (minimize *err*) using RCGA by setting the chromosome to be $\left[w_{ij}^{(1)} \quad w_{ki}^{(2)} \quad b_k \quad m_i \quad r_i \quad d_m \quad d_r\right]$ for all *i*, *j* and *k*. The range of the fitness value of (10) is (0,1]. After training, the values of these network parameters will be fixed during the operation. The total number of tuned parameters (n_{para}) of the proposed N-N-LNN is the sum of the number of parameters between the input and hidden layers, the number of parameters between the hidden and output layers, and the number of parameters for m_i, r_i, d_m, d_r. Hence,

$$\begin{aligned} n_{para} &= n_{in}n_h + n_{out}\left(n_h + 1\right) + \left(2n_h + 2\right), \\ &= \left(n_{in} + n_{out} + 2\right)n_h + n_{out} + 2 \end{aligned} \qquad (11)$$

4. Industrial Application and Results

In this section, industrial application example will be given to illustrate the merits of the proposed network. The application is on hand-written graffiti recognition.

A hand-written graffiti pattern recognition problem is used to illustrate the superior learning and generalization abilities of the proposed network on a classification problem with a large number of input data sets. In general, the neural network approaches are model-free. Different kinds of neural model applied for hand-written recognition system are reported in [8,10,12,15,16].

4.1 Neural Network Based Hand-Written Graffiti Recognition System

In this example, the digits 0 to 9 and three control characters (*backspace*, *carriage return* and *space*) are recognized by the N-N-LNN. These graffiti are shown in Figure 9. A point in each graffiti is characterized by a number based on the *x*-*y* coordinates on a writing area. The size of the writing area is $x_{\max}$ by $y_{\max}$. The bottom left corner is set as (0, 0). Ten uniformly sampled points of the graffiti will be taken in the following way. First, the input graffiti is divided into 9 uniformly distanced segments characterized by 10 points, including the start and the end points. Each point is labeled as (x_i, y_i), i = 1, 2, …, 10. The first 5 points, (x_i, y_i), i = 1, 3, 5, 7 and 9 taken alternatively are converted to 5 numbers z_i respectively by using the formula $z_i = x_i x_{\max} + y_i$. The other 5 points, (x_i, y_i), i = 2, 4, 6, 8 and 10 are converted to 5 numbers respectively by using the formula $z_i = y_i y_{\max} + x_i$. These ten numbers, z_i, i = 1, 2, …, 10, are used as the inputs of the proposed graffiti recognizer. The hand-written graffiti recognizer as shown in Figure 10 is proposed. Its inputs are defined as follows,

$$\bar{\mathbf{z}}(t) = \frac{\mathbf{z}(t)}{\|\mathbf{z}(t)\|}, \tag{12}$$

where $\bar{\mathbf{z}}(t) = [\bar{z}_1(t) \;\; \bar{z}_2(t) \;\; \cdots \;\; \bar{z}_{10}(t)]$ denotes the normalized input vectors of the proposed graffiti recognizer; $\mathbf{z}(t) = [z_1(t) \;\; z_2(t) \;\; \cdots \;\; z_{10}(t)]$ denotes the ten points in the writing area; $\|\cdot\|$ denotes the l_2 vector norm. The 16 outputs, $y_k(t)$, k = 1, 2, …, 16, indicates the similarity between the input pattern and the 16 standard patterns respectively. The input-output relationship of the training patterns is defined such that the output $y_i(t) = 1$ and others are zero when the input vector belongs to pattern i, i = 1, 2, …, 16. For example, the desired outputs of the pattern recognition system are [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] for the digit "0(a)", [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0] for the digit "0(b)", and [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] for the character "space" respectively. After training, a graffiti determiner is used to determine the output of the graffiti. A larger value of $y_j(t)$ implies that the input pattern matches more closely to the corresponding graffiti pattern. For instance, a large value of $y_0(t)$ implies that the input pattern is recognized as "0".

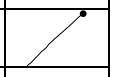
Digits/ Chars	Strokes	Digits/ Chars	Strokes	Digits/ Chas	Strokes
0(a)		5(a)		9	
0(b)		5(b)		Back-space	
1		6		Carriage Return	
2		7		Space	
3		8(a)			
4		8(b)			

Figure 9. Graffiti digits and characters (with the dot indicating the starting point of the graffiti)

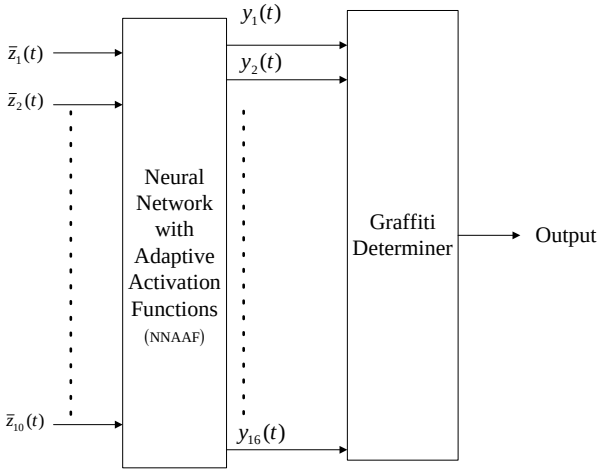


Figure 10. Graffiti digits and characters (with the dot indicating the starting point of the graffiti)

4.2 Results and Analysis

To train the neural network of the hand-written graffiti recognition system, a set of training pattern governing the input-output relationship will be used. 1600 training patterns (100 patterns for each graffiti) will be used in this example. The training patterns consist of the input vectors and its corresponding expected output. The fitness function is given by (10), with

$$err = \frac{\sum_{k=1}^{16} \sum_{t=1}^{100} \left(\frac{y_k(t)}{\|\mathbf{y}(t)\|} - \frac{y_k^d(t)}{\|\mathbf{y}^d(t)\|} \right)^2}{16 \times 100}, \tag{13}$$

where

$$\mathbf{y}^d(t) = \begin{bmatrix} y_1^d(t) & y_2^d(t) & \cdots & y_{16}^d(t) \end{bmatrix}$$

denotes the expected output vector and

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) & y_2(t) & \cdots & y_{16}(t) \end{bmatrix}$$

is the actual network output defined as,

$$y_k(t) = tf_2\left(\sum_{i=1}^{n_h} w_{ki}^{(2)} f_{s_i}(\mathbf{z}(t)) - b_k \right), \; k = 1, 2, \ldots, 16, \quad (14)$$

where

$$f_{s_i}(z(t)) = tf_1\left(\sum_{j=1}^{10} w_{ij}^{(1)} z_j(t), \tilde{m}_i(t), \tilde{r}_i(t) \right), \; i = 1, 2, \ldots, n_h, \quad (15)$$

where $tf_2(\cdot)$ is a pure linear transfer function in this application.

For comparison purposes, a conventional 3-layer fully connected feed-forward neural network (FFCNN) [11], a fixed-structure network with link switches (FSNLS) [6], and a wavelet neural network (WNN) [14, 17–18] (which combines feed-forward neural networks with the wavelet theory, providing a multi-resolution approximation for discriminate functions) trained by the improved RCGA [1] are also used in this example. For all cases, the initial values of the parameters of the neural network are randomly generated. In this application, the lower and upper bounds of the network parameters of the N-N- LNN are $\begin{bmatrix} w_{ij}^{(1)} & w_{ki}^{(2)} & b_k & m_i \end{bmatrix} \in \begin{bmatrix} -4 & 4 \end{bmatrix}$, $r_i \in \begin{bmatrix} 0.5 & 2 \end{bmatrix}$ and $\begin{bmatrix} d_m & d_r \end{bmatrix} \in \begin{bmatrix} 1 & (n_h - 1) \end{bmatrix}$. For the FSNLS, WNN and FFCNN, the network parameters are ranged from –4 to 4. The number of iterations to train the neural networks is 15000. For the RCGA [1], the probability of crossover (μ_c) and the probability of mutation (μ_m) are 0.8 and 0.05 respectively; the weights of the average-bound crossover w_a and w_b are set at 0.5 and 1 respectively; the shape parameter of wavelet mutation ζ is 2, and the population size is 50. All the results are the averaged ones out of 20 runs. In order to test the generalization ability of the proposed neural networks, a set of testing patterns consisting of 480 input patterns (30 patterns for each graffiti) is used.

The average training, best training, average testing and best testing results in terms of mean square error (MSE), and the recognition accuracy rate of all approaches are summarized in Table 1 and Table 2. It can be seen from these two tables that the recognition system implemented by the N-N-LNN outperforms those by the FSNLS, WNN and FFCNN. The best results are achieved when the number of hidden nodes (n_h) is set at 20 for the N-N-LNN, n_h = 22 for the FSNLS, and n_h = 24 for the WNN and FFCNN. In comparison with the FSNLS, WNN and FFCNN, the average training and testing errors of N-N-LNN at n_h = 20 are 0.0157 and 0.0186 respectively. They imply 77.96% improvement over FSNLS at n_h = 22, 96.82% and 76.90% improvement over WNN at n_h = 24, and 129.3% and 101.1% improvement over FFCNN at n_h = 24, respectively. In terms of the average testing recognition accuracy rate, the N-N-LNN (96.96%) gives a better result than the FSNLS (93.21%), WNN (93.92%) and FFCNN (91.25%).

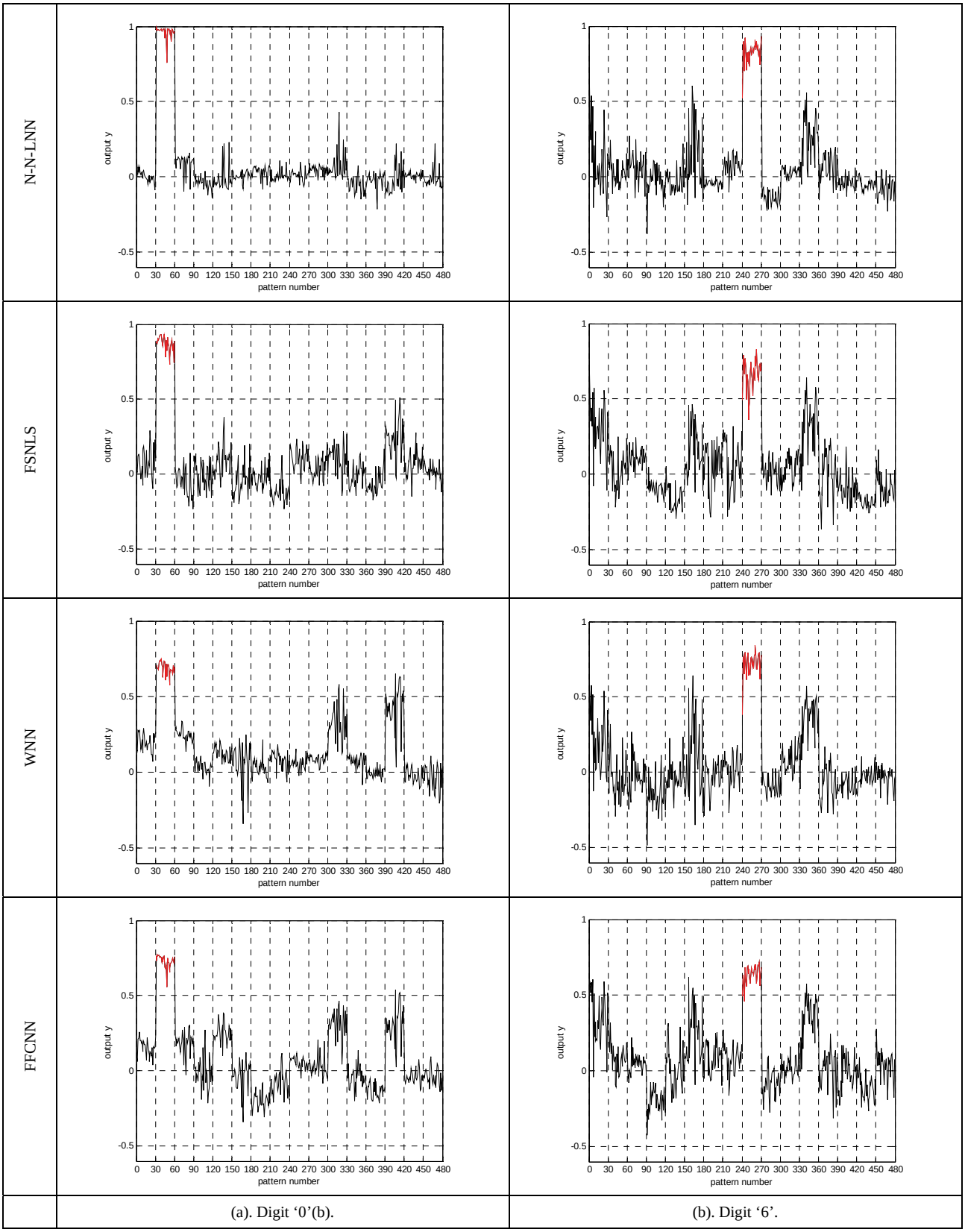
Figure 11 shows the selected output values of the

Table 1. Training results on doing the hand-written graffiti recognition

			$n_h = 18$	$n_h = 20$	$n_h = 22$	$n_h = 24$
N-N-LNN	n_{para}		520	576	632	688
	Ave. training	MSE	0.0185	**0.0157**	0.0169	0.0179
		Acc.	96.50%	**97.38%**	96.85%	96.62%
	Best training	MSE	0.0168	**0.0139**	0.0145	0.0143
		Acc.	96.88%	**98.06%**	97.31%	97.38%
FSNLS	n_{para}		1004	1112	1220	1328
	Ave. training	MSE	0.0337	0.0328	**0.0314**	0.0322
		Acc.	92.46%	92.62%	**93.25%**	93.18%
	Best training	MSE	0.0309	0.0293	**0.0282**	0.0288
		Acc.	93.40%	93.75%	**94.00%**	93.86%
WNN	n_{para}		486	540	594	**648**
	Ave. training	MSE	0.0349	0.0321	0.0316	**0.0309**
		Acc.	92.35%	92.42%	93.10%	**93.23%**
	Best training	MSE	0.0315	0.0292	0.0280	**0.0278**
		Acc.	93.31%	93.81%	94.00%	**94.13%**
FFCNN	n_{para}		502	556	610	**664**
	Ave. training	MSE	0.0393	0.0385	0.0380	**0.0360**
		Acc.	90.17%	90.46%	90.73%	**91.50%**
	Best training	MSE	0.0370	0.0388	0.0361	**0.0326**
		Acc.	90.50%	91.69%	92.56%	**93.06%**

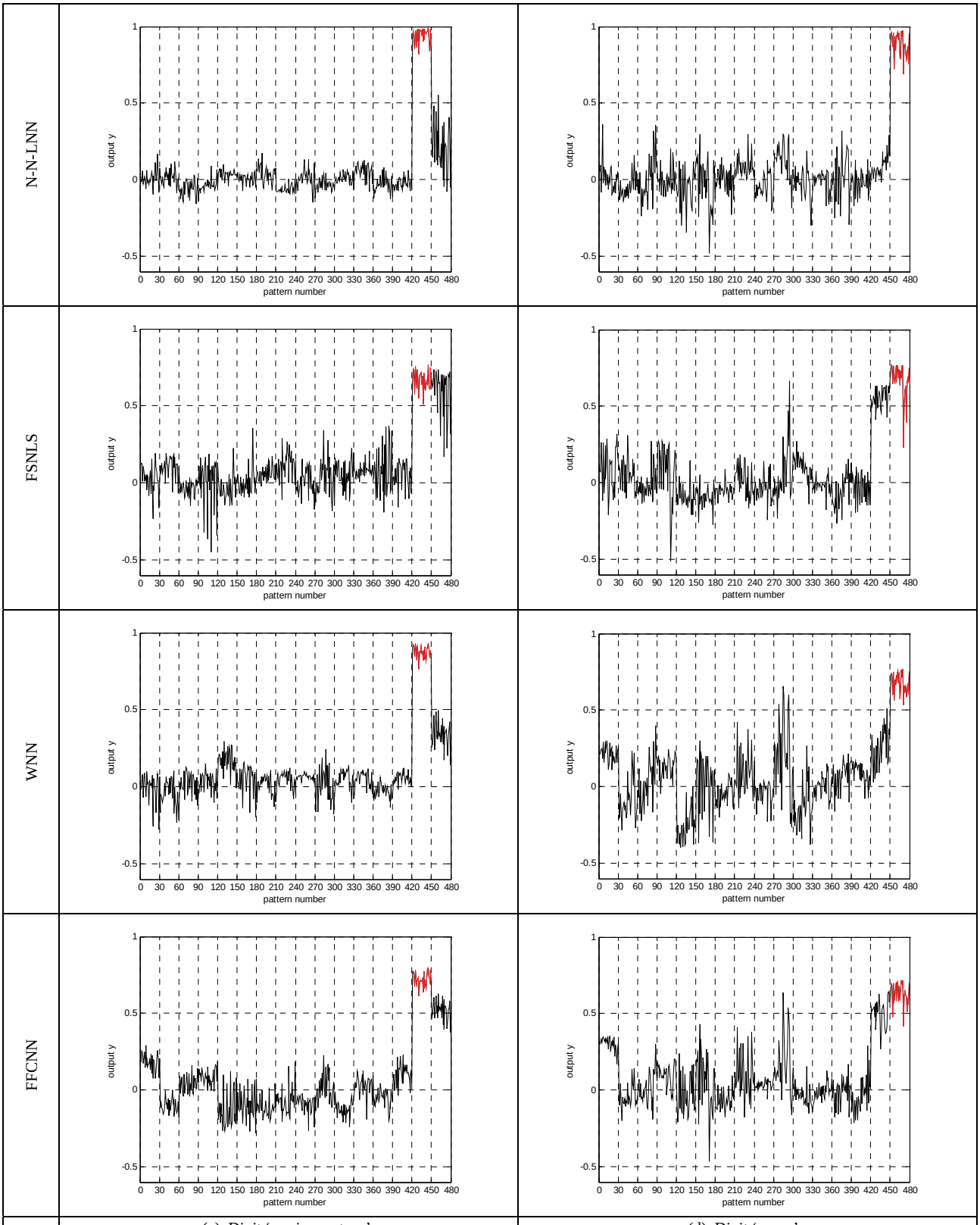
Table 2. Training results on doing the hand-written graffiti recognition

			$n_h = 18$	$n_h = 20$	$n_h = 22$	$n_h = 24$
N-N-LNN	n_{para}		520	576	632	688
	Ave. training	MSE	520	576	632	688
		Acc.	0.0228	**0.0186**	0.0199	0.0211
	Best training	MSE	95.21%	**96.96%**	95.49%	95.40%
		Acc.	0.0204	**0.0171**	0.0185	0.192
FSNLS	n_{para}		95.93%	**97.29%**	96.25%	96.05%
	Ave. training	MSE	1004	1112	1220	1328
		Acc.	0.0363	0.0350	**0.0331**	0.0349
	Best training	MSE	92.28%	92.50%	**93.21%**	93.00%
		Acc.	0.0330	0.0322	**0.0310**	0.0312
WNN	n_{para}		92.92%	93.13%	**93.96%**	93.75%
	Ave. training	MSE	486	540	594	648
		Acc.	0.0365	0.0346	0.0344	**0.0329**
	Best training	MSE	92.08%	92.22%	92.71%	**93.92%**
		Acc.	0.0329	0.0320	0.0322	**0.0308**
FFCNN	n_{para}		92.59%	93.54%	93.75%	**94.38%**
	Ave. training	MSE	502	556	610	664
		Acc.	0.0410	0.0404	0.0393	**0.0374**
	Best training	MSE	90.07%	90.58%	90.68%	**91.25%**
		Acc.	0.0404	0.0393	0.0388	**0.0361**



(a). Digit ‘0’(b). (b). Digit ‘6’.

Figure 11. Output values of the N-N-LNN, FSNLS, WNN, and FFCNN for the 480 (30 for each type) testing graffiti patterns



(c). Digit ‘carriage return’. (d). Digit ‘space’..

Figure 11 (continued). Output values of the N-N-LNN, FSNLS, WNN, and FFCNN for the 480 (30 for each type) testing graffiti patterns

N-N-LNN, FSNLS, WNN and FFCNN for the 480 (30 for each digit/character) testing graffiti. In this figure, the *x*-axis represents the pattern number for corresponding digit/character. The pattern numbers 1 to 30 are for the digit "0(a)", the numbers 31-60 are for the digit "0(b)", and so on. The *y*-axis represents the output y_i. As mentioned before, the input-output relationship of the patterns will drive the output $y_i(t) = 1$ and other outputs are zero when the input vector belongs to pattern i, i = 1, 2, …, 16. For instance, the desired output **y** of the pattern recognition system is [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0] for digit "0(b)". Referring to Figure 11(d), we can see that the output y_{16} of the N-N-LNN for pattern numbers within 451-480 (the character "space") is nearest to 1 and other outputs are nearest to zero. It shows that the recognition accuracy rate achieved by the N-N-LNN is good.

5. Conclusions

A new neural network has been proposed in this paper. The parameters of the proposed neural network are trained by the RCGA. In this topology, the parameters of the activation function in the hidden nodes are changed according to the input to the network and the outputs of other hidden-layer nodes in the network. Thanks to the variable property and the node-to-node links in the hidden layer, the learning and generalization abilities of the proposed network have been increased. Application on hand-written graffiti recognition has been given to illustrate the merits of the proposed N-N-LNN. The proposed network is effectively an adaptive network. By adaptive, we mean the network parameters are variable and depend on the input data. For example, when the proposed neurons of the N-N-LNN manipulate an input pattern, the shapes of the activation functions are characterized by the inputs from the upper and lower neighbour's outputs, which depend on the input pattern itself. In other words, the activation functions, or the parameters of the N-N-LNN, are adaptively varying with respect to the input patterns to produce the outputs. All network parameters of the N-N-LNN depend only on the present state. That means the network is a feed-forward one, causing no stability problem to the network dynamics.

REFERENCES

[1] S. H. Ling and F. H. F Leung, "An improved genetic algorithm with average-bound crossover and wavelet mutation operations," Soft Computing, Vol. 11, No.1, pp. 7–31, January 2007.

[2] F. M. Ham and I. Kostanic, "Principles of neurocomputing for science & engineering," McGraw Hill, 2001.

[3] R. C. Bansal and J. C. Pandey, "Load forecasting using artificial intelligence techniques: A literature survey," International Journal of Computer Applications in Technology, Vol. 22, Nos. 2/3, pp. 109–119, 2005.

[4] S. H. Ling, F. H. F. Leung, H. K. Lam, and P. K. S. Tam, "Short-term electric load forecasting based on a neural fuzzy network," IEEE Transactions on Industrial Electronics, Vol. 50, No. 6, pp. 1305–1316, December 2003.

[5] S. H. Ling, F. H. F. Leung, L. K. Wong, and H. K. Lam, "Computational intelligence techniques for home electric load forecasting and balancing," International Journal Computational Intelligence and Applications, Vol. 5, No. 3, pp. 371–391, 2005.

[6] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of neural network using an improved genetic algorithm," IEEE Transactions on Neural Networks, Vol. 14, No. 1, pp. 79–88, January 2003.

[7] K. F. Leung, F. H. F. Leung, H. K. Lam, and S. H. Ling, "On interpretation of graffiti digits and commands for eBooks: Neural-fuzzy network and genetic algorithm approach," IEEE Transactions on Industrial Electronics, Vol. 51, No. 2, pp. 464–471, April 2004.

[8] D. R. Lovell, T. Downs, and A. C. Tsoi, "An evaluation of the neocognitron," IEEE Transactions on Neural Networks, Vol. 8, No. 5, pp. 1090–1105, September 1997.

[9] Z. Michalewicz, "Genetic algorithm + data structures = evolution programs," 2nd extended edition, Springer-Verlag, 1994.

[10] C. A. Perez, C. A. Salinas, P.A. Estévez, and P. M. Valenzuela, "Genetic design of biologically inspired receptive fields for neural pattern recognition," IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics, Vol. 33, No. 2, pp. 258–270, April 2003.

[11] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline, and backpropagation," Proceedings of the IEEE, Vol. 78, No. 9, pp. 1415–1442, September 1990.

[12] R. Buse, Z. Q. Liu, and J. Bezdek, "Word recognition using fuzzy logic," IEEE Transactions on Fuzzy Systems, Vol. 10, No. 1, February 2002.

[13] L. Davis, "Handbook of genetic algorithms," NY: Van Nostrand Reinhold, 1991.

[14] X. Yao, "Evolving artificial networks," Proceedings of the IEEE, Vol. 87, No. 7, pp. 1423–1447, 1999.

[15] P. D. Gader and M. A. Khabou, "Automatic feature generation for handwritten digit recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 12, pp. 1256–1261, December 1996.

[16] L. Holmström, P. Koistinen, J. Laaksonen, and E. Oja, "Neural and statistical classifiers — Taxonomy and two case studies," IEEE Transactions on Neural Networks, Vol. 8, No. 1, pp. 5–17, January 1997.

[17] J. Zhang, G. G. Walter, Y. Miao, and W. W. N. Lee, "Wavelet neural networks for function learning," IEEE Transactions on Signal Processing, Vol. 43, No. 6, pp. 1485–1497, June 1995.

[18] B. Zhang, M. Fu, H. Yan, and M. A. Jabri, "Handwritten digit recognition by adaptive-subspace self-organizing map (ASSOM)," IEEE Transactions on Neural Networks,

Vol. 10, No. 4, pp. 939–945, July 1997.

[19] C. K. Ho and M. Sasaki, "Brain-wave bio potentials based mobile robot control: Wavelet-neural network pattern recognition approach," in Proceedings IEEE International Conference on System, Man, and Cybernetics, Vol. 1, pp. 322–328, October 2001.

[20] S. Yao, C. J. Wei, and Z. Y. He, "Evolving wavelet neural networks for function approximation," Electron Letter, Vol.32, No.4, pp. 360–361, February 1996.

[21] Q. Zhang and A. Benveniste, "Wavelet networks," IEEE Transactions on Neural Networks, Vol. 3, No. 6, pp. 889–898, November 1992.

Journal of Intelligent Learning Systems and Applications, **2010, 2: 12-18**
doi:10.4236/jilsa.2010.21002 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Evolutive Neural Net Fuzzy Filtering: Basic Description

Juan C. García Infante[1], J. Jesús Medel Juárez[2], Juan C. Sánchez García[1]

[1]Professional School of Mechanical and Electrical Engineering, National Polytechnic Institute, Col. San Francisco Culhuacan Del. Coyoacan, México D.F.; [2]Centre of Computing Research, National Polytechnic Institute, Vallejo, México D. F.
Email: jcnet21@yahoo.com, jjmedelj@yahoo.com.mx, jcsanchezgarcia@gmail.com

Received October 29th, 2009; accepted January 20th, 2010.

ABSTRACT

The paper describes the operation principles of the evolutive neuro fuzzy filtering (*ENFF*) *properties, which based on back propagation fuzzy neural net, this filter adaptively choose and emit a decision according with the reference signal changes of an external reference process, in order to actualize the best correct new conditions updating a process. This neural net fuzzy filter mechanism selects the best parameter values into the knowledge base* (*KB*), *to update the filter weights giving a good enough answers in accordance with the reference signal in natural sense. The filter architecture includes a decision making stage using an inference into its structure to deduce the filter decisions in accordance with the previous and actual filter answer in order to updates the new decision with respect to the new reference system conditions. The filtering process states require that bound into its own time limit as real time system, considering the Nyquist and Shannon criteria. The characterization of the membership functions builds the knowledge base in probabilistic sense with respect to the rules set inference to describe the reference system and deduce the new filter decision, performing the ENFF answers. Moreover, the paper describes schematically the neural net architecture and the decision-making stages in order to integrate them into the filter architecture as intelligent system. The results expressed in formal sense use the concepts into the paper references with a simulation of the ENFF into a Kalman filter structure using the Matlab© tool.*

Keywords*: *Digital Filters, Neuro Fuzzy Systems, Evolutive Systems, Real Time

1. Introduction

The development of new intelligent systems requires mechanism that deduces its own external environment. Having a characterization of its answers in order to discover dynamically the changes of an external process; interpreting the operation levels to select the best response updating its answers weights, in order to follow the best approximation condition in accordance to the external process changes and taking a decision about the actual condition to correct or update a system parameters.

These kinds of systems related to artificial intelligent mechanisms, which have some problems to develop capacities with high evolutive changing processes.

The new systems should use into its architecture evolutive tools in order to get its own perception giving the best decision answers, using this to solve complex problems, actualizing and adapting its perceptions and answers in accordance with a reference dynamical system.

The use of learning techniques based on searching the optimal solutions represents a classical alternative manner to obtain knowledge. The evolutive systems into the neural net fuzzy digital filtering is an option for to obtain different decision answer levels, in accordance with a reference model dynamics, adapting its answers to the possible changes by selecting the best values in order to get the necessary convergence conditions, which should has the best operation each time.

An evolutive system with artificial neural network properties is a computational model imitating natural processes as the biological systems. Constituting in this case the processing elements called neurons, all of them connected forming the neural net structure and a decision stage to loop the process with the best answer [1,2].

The evolutive filtering is constructing with different kind of decision answers, considering a set of characteristics limited by the reference operation region. The goal of this kind of filter is to characterize and infer a system that has uncertainties in its operation, describing the natural process, with a rule set; needing a feedback law in order to follow the basic properties in accordance with a desired input signal, adjusting its parameters to give a

correct solution dynamically. According to this, the filter chooses the best decision instruction to update the system.

The neural net filter stage [3] works as a parallel fuzzy neurons in loop form, which has an iterative searching methodology used for evolutive algorithms and based on the back propagation (*BP*) algorithm since its parameters are updated dynamically ([4–6]) at each iteration by degrees [7]. This process refers to a back propagation parameter adaptation [8], using supervised learning by the knowledge base according with the error *e(k)* ([9,10]) described by the difference between the desired response *y(k)* and the actual signal $\hat{y}(k)$ [11][1].

With this perspective, the paper integrates the *ENFF* concept with its real time restrictions [9], using statistical methods in order to characterize the *Kalman* filter internal structure to give answers with respect to the operation levels in natural way making a specific decision in order to follow the natural reference model ([12,13]).

2. Filtering Stages

The criterion described as the error minimization *e(k)*[3] describes the difference between the desired input *y(k)* and the actual output filter $\hat{y}(k)$, allowing to find the corresponding membership function. Which is the best approximation from the signal $\hat{y}(k)$ to *y(k)* in order to adjust the parameters of the filter and get the correct answer describing the reference system path in order to take the decision into the decision stage. The error value *e(k)* should be close to γ with a limited interval [0, ε] and ε is described as $\inf\{|e(k)| : i,k \in Z_+\}$ (*i* represents an index, with k interval [14,15])[2] Below, there is the stages description of the evolutive filtering, the first three stages are the inference mechanism; the stages d and e are the filtering process and the decision stage is the instruction decision:

- *Input Inference*: In this stage, the natural desired signal *y(k)* from the reference system to the input filter has a description in metric sense, is the first step of the inference mechanism [5].
- *Rule base*: Dynamical rank intervals with respect to the input of the filter use the logical binary connector known as *IF*, representing the second step of the inference mechanism.
- *Output inference*: The expert action with respect to the rule base known as consequence uses the logical binary connector *THEN* to find the correct weight answer as $\hat{a}(k)$ called membership function is the third step of the inference mechanism.
- *Filter algorithm:* This is the filter stage, which takes the digital answer, converted in a natural response. This is the closest distance value to the desired signal, based on the predefined knowledge base.
- *Natural feedback:* the filtering process takes the correct linguistic value and feedbacks the filter parameters, updating its operation according to the natural evolution of the reference system considering the error differences between $y(k)$ and $\hat{y}(k)$ dynamically and using a metric rank of the error functional $J(k)$.[3]
- *Decision stage:* Finally, the filter decision mechanism takes the best answer option according with the values of $\hat{y}(k)$ and $\hat{y}(k-1)$ to actualize, the decision described as $ds(k)$.

Each decision stage is in accordance with the actual and previous values of $\hat{y}(k)$ described as $\hat{y}(k-1)$ doing a comparison between these values in order to select the corresponding instruction answer to the reference system, all the instruction answers are previously defined using the logic connectors *IF-AND-THEN* the filter selects the corresponding instruction to actualize the reference system conditions.

Figure 1, shows the filter architecture interacting with a reference process to describe its operation and take a decision instruction.

3. Filtering Description

The evolutive neural net fuzzy filter (see: Figure 1) requires a knowledge base (*KB*); because, it has all the possible responses in accordance with the reference system or process in which the filter is interacting. The evolutive by neural net fuzzy filter has its state variables bounded by a

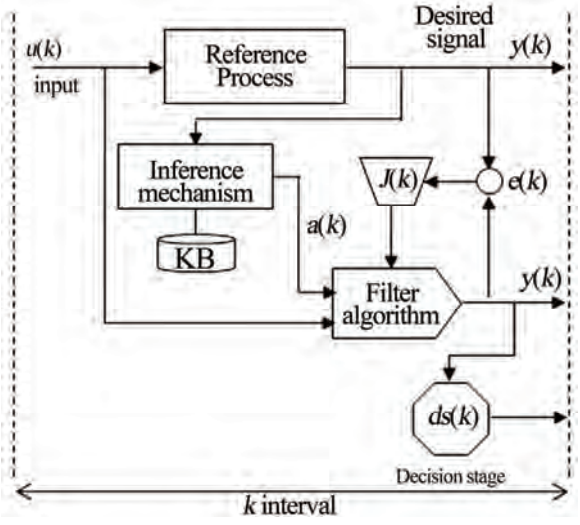


Figure 1. Evolutive neural net fuzzy filter process

[1] $e(k) := y(k) - \hat{y}(k)$, is a fuzzy value.

[2] Inf is the greatest lower bound error value into the error set.

[3] The functional $J(k)$ describes the convergence relation among the real observed and its estimation; symbolically in recursive form: $J(k) = \frac{1}{k}\left[\left((k-1)J(k-1)^2 + e(k)^2\right)\right]^{\frac{1}{2}}$

rank of values described by fuzzy sets (for example, the temperature classified by the fuzzy logic as high, medium or low ([3,8,17,18]).

This kind of filter adaptively modifies its response $\hat{y}(k)$ according to the dynamic changes at the filter desired input $y(k)$ from the reference process alteration (produced by neural excitation). It requires that the error loop inference (comparison between desired input $y(k)$ and actual output $\hat{y}(k)$ changes the filtering responses using the knowledge base (*KB*) with real time conditions (see: [4,6,9]).

The filter classifies its different operation levels by membership functions, based on the error minimum function J_{min} to give a corresponding near specific answer for each kind of desired input $y(k)$ limited by the error distribution function, which is also limited by γ. The classification of the reference system responses realized by the filter is by the error distribution function considering different levels of response (membership functions), each level delimited by specific metrics bounded all into the error distribution region. This classification represents the characterization of the reference process operation in interaction with the filter ([14,19]).

Using this kind of filters avoids the initial instability of the error convergence because it has all information required and limited by an interval into a distribution region as supervised learning technique.

To do a classification of the membership functions the filter uses an error criterion function using metric intervals to delimit its operational levels. The membership function set established inside a distribution function corresponding to the error criterion; this classification with several membership functions is in accordance to each error interval (as operation linguistic levels).

Respect to Figure 1, $u(k)$ is an input natural linguistic value from the reference process interacting with the filter. $T(k)$ is the control area, which include all the filtering processes corresponding to one iteration of a process set, in order to obtain an output response $\hat{y}(k)$. The desired signal described as $y(k)$ is the fuzzy natural reference variable given by the reference system. $e(k)$ is a fuzzy value where the set $\{\gamma_i : \gamma_i > 0, \ \forall i \in Z_+\}$ has $inf\{\gamma_i\} \to |\lambda|, |\lambda| > 0$ and the $sup\{\gamma_i\} \to |\lambda|, |\lambda| < 1$, [5] with γ^* that must be the minimum value between $y(k)$ and $\hat{y}(k)$; both values should be almost equal; in other words, $e(k)$ is the difference between $\hat{y}(k)$, and $y(k)$, being this the criterion distance in order to select the membership function for each case[1].

4. Fuzzy Neural Net Structure

The *ENFF* architecture is develop in order to work as neural net using back propagation properties ([11,19]). This characterizes the operational linguistic levels of the desired variable set $\{y(k)\}$, expressing as a basic estimation $\hat{a}(k)$ result[5], in accordance with the inference rules with respect to the error value criterion selecting the corresponding membership function.

The neural net conditions are [1]:

- The desired input $y(k)$ represents fuzzy labels, and each one has different operation levels.
- The input inference recognizes the variable type (A, B or C).
- The weights required for the internal adaptive adjustments into the filter architecture, to get a correct response described as membership functions, renewing the values of $\hat{a}(k)$ parameter using the knowledge base.
- The connection into the filter according to the error differences[1] has a minimum operation criterion, selecting the membership function based on the minimum cost of the filter response to update its weights.
- It uses supervised learning into the knowledge base, therefore, the neuron has previously all the information included in T_N.
- The decision stage compares the previous output and the actual value of the filter to select a instruction decision [7].

Globally, according to Figure 2, the filter has four neural layers:

The first layer with $p \times n, n, p \in Z_+$ input neurons representing a set of desired inputs $\{y(k)\}$, a single hidden layer with p processing units in which the inference mechanism operates in order to find the respective membership function $\hat{a}(k)$; the output $\{\hat{y}(k)\}$, that is the answers set of the neural process (see: Figure 2). The filter has a decision *stage* layer as evolutive condition in order to give control instructions for an external process according with the previous filter answers.

The neurons architecture are simple processors, which get information from the first input layer from the outside world as desired signal $\{y(k)\}$ to the next neurons into the hidden filter layer.

The neural net structure proceed the result of their previous processing step to nodes in the next higher layers in order to obtain the nearest desired signal value of $y(k)$ at the filter output $\hat{y}(k)$. In consequence the decision stage in the top-layer communicate their decision results to the outside world as $ds(k)$, which its instructions bounded inside the external reference process operation in order to give the best control value.

Subsequently, using rules the fuzzy system performs the classification in both stages: filtering and decision; its

rules are generated automatically by the evolutive neural net fuzzy filtering evaluating features which are difficult to extract or to evaluate directly delimited by the filtering criterion.

The neural net represents a set of services, activating a specific neuron means the use of specific service with different levels of response. The filter requires a variable value according to the inference classification of the desired signal set, according to the changes of the error rank *e(k)* per iteration, one neuron is activating; next iteration the filter could renew the neuron or only change its value by degrees.

The value of *y(k)* defines a neuron to activate or to select, in order to use its own set of operation levels (membership functions by degrees) to give the corresponding value of $\hat{a}(k)$, and gives a correct natural answer [2].

5. Rule Base Strategy

This kind of filtering has operational properties defined by the rules base to learn, recall, associate and compares the new information delimited before according to the error variance limits predefined. The fuzzy rules conditions established as logical connectors (*IF-THEN*) constitute the rules base with respect to the error intervals and its respective response described as membership function.

The rules generated constitute the inference of the desired signal $y(k)$ [5] as the logical connector *IF*, and the logical connector *THEN* selecting automatically the parameter weighs of $\hat{a}(k)$ according to the knowledge base ([18,20]):

The *KB* has an automatic classification of the filter conditions: having the knowledge of the filtering operation levels. In addition, selects the corresponding membership function (value of $\hat{a}(k)$) of the knowledge base, adapting its responses weighs to give a correct answer $\hat{y}(k)$, near to $y(k)$.

A rules base filter characterized by a set of desired signals $\{y(k)\}$ at the filter input, classified by intervals delimited previously into the error difference in order to select the corresponding membership function, which has the corresponding $\hat{a}(k)$ value giving a correct response $\hat{y}(k)$ seeking the closest distance to $y(k)$:

$$T_N = \{(y(k), \hat{y}(k))\}_{k=1}^{N} \quad (1)$$

In addition, mathematically correspondence of $y(k)$ and $\hat{y}(k)$, expressed with respect to the second probability moment, has the infimum value, described as:

$$J_{\min} = \inf_{N} \{\min J(y_0, \hat{y})\}_N \quad (2)$$

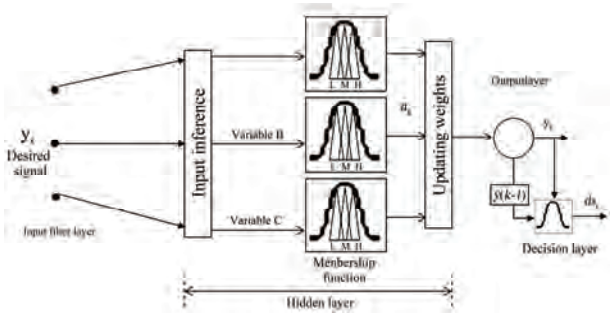


Figure 2. Evolutive neural filter layers (L-low, M-medium, H-high)

The rules set constitute a simple filter operation, based on the error set {e(k)} as limit indicator of the corresponding membership function set to adapt the parameter dynamically.

The decision stage described as selects the best instruction answer to control a process according with the filter answer, and its previous value, with the decision inference mechanism by fuzzy rules.

For the decision stage, which is the finally inference of the evolutive neural net fuzzy filter, using a set of fuzzy rules to get the correct control instruction from its own knowledge base (KB). First, consider the value level and its previous value as the actual environment and select the corresponding condition or strategy from the knowledge base to loop an external process as.

To describe the decision stage mechanism, we use fuzzy rules to infer and select the corresponding value with If-then rules; the filtering system selects and gives a specific instruction answer as in order to update and improve the control instruction performance minimizing the error process, in accordance with the fuzzy rule structure will have:

If value is____ and value is____ Then value is ____.

This filter (ENFF) has two knowledge bases, one for to select the parameter (as in Figure 1) and the second used for the decision stage to select the instruction answer (as in Figure 3).

The Figure 3 illustrates the decision stage mechanism as fuzzy rules as:

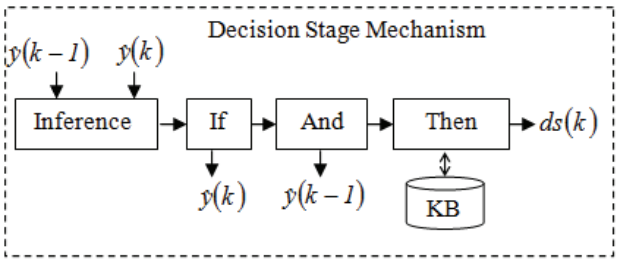


Figure 3. Decision stage inference

6. Real Time Scheme

DEFINITION 1 The real time properties of an evolutive neural net fuzzy digital filter, which is an adaptive filter performed according to ([6] and [10]):

- To take and to emit signals with fuzzy information into intervals limited in accordance to the reference process response according to the stability criteria ([5,6]).
- To take and to emit information through semi-open time intervals, synchronized with the process evolution time, described in a relative way by semi-open time intervals, considering the criteria into references [15,21,22].
- A membership functions group forms a control area, according to the properties considered in a) and b) points, respectively ([18,20,23]) .
- A set of fuzzy rules builds the knowledge base according to the fuzzy desired signal from the reference model $y(k)$, obtaining a specific filter answer $\hat{y}(k)$ ([16,20]).
- The adaptation algorithm updates the filter coefficients according to the selected membership function respectively to the established error criterion symbolically expressed as γ_*.
- The decision stage mechanism selects the best instruction answer as $ds(k)$, in order to perform the external process operation at each time.

The knowledge base has all information that the filter requires to adjust its gains in optimal form and gives an answer accomplishing the convergence range, inside of the time interval (indexed with $k \in \mathbf{Z}_+$) in agreement with the Nyquist sense, without loss of the stability properties [6,20,23]):

- $y(k)$ is a measurable variable classified in metric ranks as degrees in linguistic sense (described into a state space variable bounded symbolically into linguistic natural expressions as high, medium or low values),
- $T(k)$ is the control area described in pairs formed by $\hat{y}(k)$ and $y(k)$ limited in time interval (has a velocity change bounded in the sense exposed by [11]),
- $e(k)$ is the fuzzy value defined by the difference among $\hat{y}(k)$ and $y(k)$, which is bounded by the set $\{\gamma_i : \gamma_i > 0, \ \forall i \in Z_+\}$, with $\inf\{\gamma_i\} \to |\lambda_*|$, such that $|\lambda_*| > 0$, $\sup\{\gamma_i\} \to |\lambda^*|$, $|\lambda^*| < 1$, means that $\hat{y}(k)$ is approximately equal to $y(k)$ metrically speaking as linguistic variables, both are the same natural value.
- $ds(k)$ is the instruction answer of the filter according with the output of the filter $\hat{y}(k)$ and its previous value in fuzzy sense, to selects the corresponding candidate function classified in probabilistic sense into a knowledge base to update an external process into a new condition.

6.1 Local and Global Description

DEFINITION 2 An ENFF in local and global temporal sense has quality of response according to the convergence criterion[1] with respect to the real time conditions [15].

Global characteristics: The convergence intervals defined by $[0, \varepsilon \pm \alpha)$ with measures up to zero through error functional *J*(*k*) considering and the convergence relation[1], temporally parameterized to the membership function according to the linguistic variables values ([3,24]), without loss that /e(k)/<1 in agreement to [6].

According to the evolutive neural net fuzzy concepts, the global characteristics specified in stochastic sense according to [15], where $J(\tau_m) = inf\{min\{\tilde{J}_k\}\} \le \varepsilon$ (see: (2)), with $\{\tilde{J}_k\} \subseteq \{J_k\}$ and

$$P(\tilde{J}_k \le \varepsilon \pm \sigma) = 1, \ \sigma << \varepsilon$$

without loss that its natural evolution described by [22]:

$$\tau_{min} = 0.5 f_{max}^{-1} \tag{3}$$

7. Simulations

For the simulation of the *ENFF*, in this case we use the Kalman filter structure to integrate the filter stages, with a transition matrix described by the knowledge base in accordance to the error functional criterion [5]. The evolution times into a soft system integrated in a *PC* with *AMD* Sempron processor 3100+ with *k* intervals, having a mean evolution time of 0.004 sec ± 0.0002 sec.

The basic reference system in discrete state space expressed by the first order difference, as:

$$x(k+1) = a(k)x(k) + w(k) \tag{4}$$

In accordance with the system (4) the output is:

$$x(k), w(k), v(k) \in R^{n \times 1}$$

$$y(k) = x(k) + v(k) \ : \tag{5}$$

where: {*x*(*k*)}= is the set of internal states, {*a*(*k*)}= is the parameters sequence, {*w*(*k*)}= is the noise set system perturbation, {*y*(*k*)}= is the set of desired signal from the system reference, *v*(*k*)= is the output noises.

[4]Sup is the least upper bound of a partially ordered set

[5]The desired signal commonly has the basic and explicit description as $y(k) = a(k)y(k-1) + \omega(k)$, where a_k is known as stability parameter (see: [12] and [10]), $\omega(k)$ is the perturbation output noise, $y(k)$ is the desired reference signal.

There are different operation levels described in probability sense in order to match with the functional error[1] limit, in accordance with the desired signal *y(k)* and the *Kalman* filter answer $\hat{y}(k)$. The filter operation bounds into the second probability moment, establishing the filter classification as linguistic natural variables expressed as low, medium and high levels.

According to the parameter $\hat{a}(k)$ selected by the rule strategy, the Figure 4 shows the output answer $\hat{y}(k)$ of the filter with respect to the desire signal *y(k)* at the input filter:

Figure 5 shows the decision stage process to obtain the instruction answer as $ds(k)$ according with the $\hat{y}(k)$ values and its previous stage described as $\hat{y}(k-1)$:

The evolution time is less than the reference process proposed as 0.092 sec., satisfying the condition described in (3).

The Figure 6 shows the decision stage as linguistic levels according with the filtering process in order to give different decision answers level described as DA#:

The global convergence time of the filter is 0.08 sec, which is less than the evolution condition of the system, known as system maximum evolution time, oscillating around 0.092 sec.

Figure 7 shows the functional described as *J(k)* with respect to the filter:

8. Conclusions

The paper was about the analysis of the evolutive neural net fuzzy filtering and its real time conditions, in order show the applicability conditions into dynamical systems. The paper describes the adaptive inference mechanism that classifies and deduces the filter answers by the error value as limit, in order to search the adaptive weights and update its parameters to give a correct response dynamically as a natural linguistic answer.

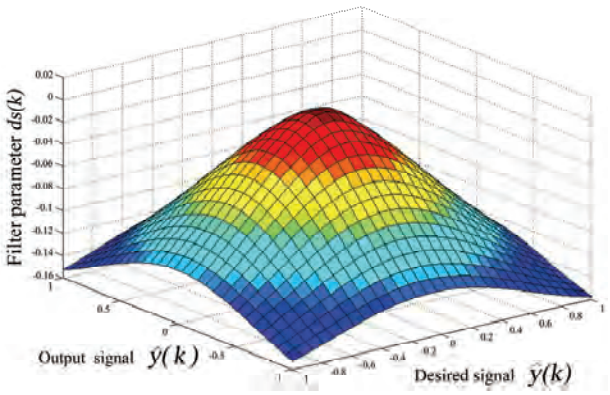


Figure 4. $\hat{y}(k)$ **Estimation in accordance with** $\hat{a}(k)$ **value and** $y(k)$ **reference**

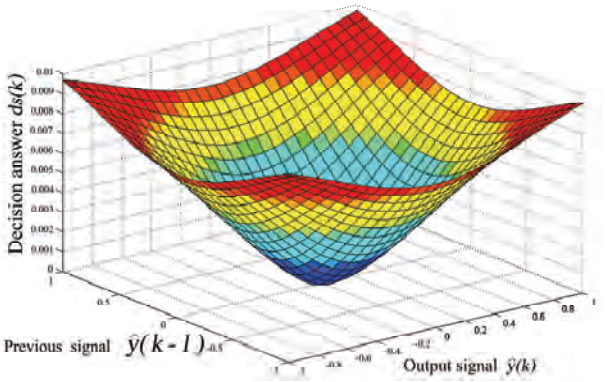


Figure 5. Decision answer $ds(k)$ **with respect to** $\hat{y}(k)$ **and** $\hat{y}(k-1)$

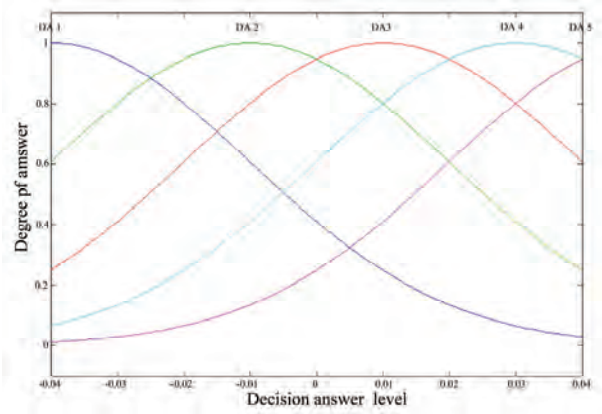


Figure 6. Decision answer as linguistic levels (DA)

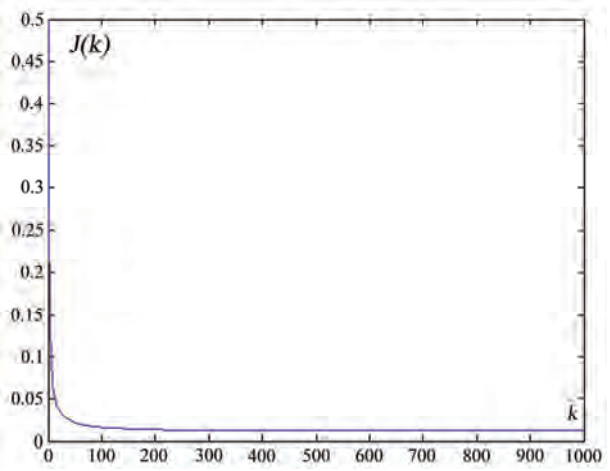


Figure 7. Parameter of convergence γ^* **illustratively by the functional *J(k)***

The paper establishes how to construct and characterize the membership functions of the knowledge base in a probabilistic manner with the decision rules set, making a

description of the real time conditions that the evolutive neural net fuzzy filter (*ENFF*) has to perform, which architecture works as a neural net. The filter has a final decision stage in accordance with the filter operation making a fuzzy inference between the actual and previous filter answers in order to select the corresponding decision answer value.

The results are in formal sense and described using the definitions considered in the papers referenced here. Finally, this work showed a simulation of the *ENFF* operation using the Matlab tool and the *Kalman* filter structureto integrate the evolutive properties, considering five decision answers into the decision stage, having an accurate filtering time response with respect to the reference system in accordance with the real time properties proposed in this work.

9. Acknowledgments

The authors thank the sponsors for the support of this work: Conacyt and National Polytechnic Institute (*IPN*).

REFERENCES

[1] C. Piotr, "Evolution fuzzy rule based system with parameterized consequences," International Journal of Applied Mathematics and Computer Science, Vol. 16, No. 3, pp. 373–385, 2006.

[2] F. Yamakawa, "Fuzzy neurons and fuzzy neural networks," 1989.

[3] S. Abdul, "Fuzzy logic and its uses," http://www.doc.ic.ac.uk.

[4] H. S. Ali, "Fundamentals of adaptive filters," Complex Systems, 2003.

[5] R. Ash, "Real analysis and probability," Ed. Academic Press, USA, 1970.

[6] S. Haykin, "Adaptive filtering," Prentice Hall, 2001.

[7] G. Huang, K. Zhu, and C. Siew, "Real-time learning capability of neural networks," IEEE Transactions on Neural Networks, Vol. 17, pp. 863–878, 2006.

[8] B. Rajen and M. Gopal, "Neuro-fuzzy decision trees," International Journal of Neural Filters, Vol. 16, pp. 63–68, 2006.

[9] J. García, J. Medel, and J. Sánchez, "Real-time neuro-fuzzy digital filtering: Approach," Computer and Simulation in Modern Science, WSEAS press selected papers, Vol. 1, 2008.

[10] J. Medel and P. Guevara, "Caracterización de filtros digitales en tiempo-real para computadoras digitales," Computacióny Sistemas, Vol. 7, No. 3, 2004.

[11] J. Medel, J. García, and J. Sánchez, "Real-time neuro-fuzzy digital filtering: Basic concepts," WSEAS Transactions on Systems and Control, Vol. 8, No. 16, 2008.

[12] J. García, J.Medel, and L. Guevara, "RTFDF description for ARMA systems," WSEAS International Conferences, Canada, 2007.

[13] D. Marcek, "Stock price forecasting: Statistical, classical and fuzzy neural networks," Modeling Decisions for Artificial Intelligence, Springer Verlag, pp. 41–48, 2004.

[14] M. Margaliot and G. Langholz, "New approaches to fuzzy modeling and control design and analysis," World Scientific, 2000.

[15] G. Morales, "Introducción a la lógica difusa," Cinvestav-IPN, 2002.

[16] T. Amble, "Logic programming and knowledge engineering," Addison Wesley, 1987.

[17] J. Smith and A. Eiben, "Introduction to evolutionary computing," Springer, 2003.

[18] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control," IEEE Transactions and Systems, Man, and Cybernetics, Vol. 15, pp. 116–132, 1986.

[19] G. Huang, K. Zhu, and C. Siew, "Real-time learning capability of neural networks," IEEE Transactions on Neural Networks, Vol. 17, pp. 863–878, 2006.

[20] L. Zadeh, "Fuzzy sets," Information and Control, Vol. 8, pp. 338–353, 1965.

[21] K. M. Passino, "Fuzzy control," Addison Wesley, 1998.

[22] M. Shannon, "A mathematical theory of communication," Bell Systems Technical Journal, Vol. 27, pp. 379–423, 623–656, 1948.

[23] E. Mamdani, "Applications of fuzzy algorithms for control of simple dynamic plant," Proceedings IEEE, Vol. 121, pp. 1585–1588, 1974.

[24] M. Montejo, "Lógica difusay control difuso," 2006, http://www.redeya.com.

Journal of Intelligent Learning Systems and Applications, 2010, 2: 19-27
doi:10.4236/jilsa.2010.21003 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Continuous Arabic Sign Language Recognition in User Dependent Mode

K. Assaleh[1], T. Shanableh[2], M. Fanaswala[1], F. Amin[1], H. Bajaj[1]

[1]Department of Electrical Engineering, American University of Sharjah, Sharjah, UAE; [2]Department of Computer Science and Engineering; American University of Sharjah, Sharjah, UAE.
Email: kassaleh@aus.edu

Received October 4, 2009; accepted January 13, 2010.

ABSTRACT

Arabic Sign Language recognition is an emerging field of research. Previous attempts at automatic vision-based recognition of Arabic Sign Language mainly focused on finger spelling and recognizing isolated gestures. In this paper we report the first continuous Arabic Sign Language by building on existing research in feature extraction and pattern recognition. The development of the presented work required collecting a continuous Arabic Sign Language database which we designed and recorded in cooperation with a sign language expert. We intend to make the collected database available for the research community. Our system which we based on spatio-temporal feature extraction and hidden Markov models has resulted in an average word recognition rate of 94%, keeping in the mind the use of a high perplexity vocabulary and unrestrictive grammar. We compare our proposed work against existing sign language techniques based on accumulated image difference and motion estimation. The experimental results section shows that the proposed work outperforms existing solutions in terms of recognition accuracy.

Keywords: *Pattern Recognition, Motion Analysis, Image/ Video Processing and Sign Language*

1. Introduction

The growing popularity of vision-based systems has led to a revolution in gesture recognition technology. Vision-based gesture recognition systems are primed for applications such as virtual reality, multimedia gaming and hands-free interaction with computers. Another popular application is sign language recognition, which is the focus of this paper.

There are two main directions in sign language recognition. Glove-based systems use motion sensors to capture gesture data [1–3]. While this data is more attractive to work with, the gloves are expensive and cumbersome devices which detract from the naturalness of the human-computer interaction. Vision-based systems, on the other hand, provide a more natural environment within which to capture the gesture data. The flipside of this method is that working with images requires intelligent feature extraction techniques in addition to image processing techniques like segmentation which may add to the computational complexity of the system.

Note that while respectable results have been obtained in the domains of isolated gesture recognition and finger spelling, research on continuous Arabic sign language recognition is non-existent.

The work in [4] developed a recognition system for ArSL alphabets using a collection of Adaptive Neuro-Fuzzy Inference Systems (ANFIS), a form of supervised learning. They used images of bare hands instead of colored gloves to allow the user to interact with the system conveniently. The used feature set comprised lengths of vectors that were selected to span the fingertips' region and training was accomplished by use of a hybrid learning algorithm, resulting in a recognition accuracy of 93.55%. Likewise [5] reported classification results of Arabic sign language alphabets using Polynomial classifiers. The work reported superior results when compared with their previous work using ANFIS on the same dataset and feature extraction techniques. Marked advantages of polynomial classifiers include their computational scalability, less storage requirements and absence of the need for iterative training. This work required the participants to wear gloves with colored tips while performing the gestures to simplify the image segmentation stage. They extracted 30 features involving the relative position and orientation of the fingertips with respect to the wrist and each other. The resulting system achieved 98.4% recognition accuracy on training data and 93.41% on test data.

Sign language recognition of words/gestures as opposed to alphabets depends on analyzing a sequence of still images with temporal dependencies. Hence HMMs are a natural choice for model training and recognition as reported in [6]. Nonetheless, the work in [7] presented an alternative technique for feature extraction of sequential data. Working with isolated ArSL gestures, they eliminate the temporal dependency of data by accumulating successive prediction errors into one image that represents the motion information. This removal of temporal dependency allows for simple classification methods, with less computational and storage requirements. Experimental results using k-Nearest Neighbors and Bayesian classifiers resulted in 97 to 100% isolated gesture recognition. Variations of the work in [7] include the use of block-based motion estimation in the feature extraction process. The resultant motion vectors are used to represent the intensity and directionally of the gestures' motion as reported in [8].

Other sign languages such as American Sign Language have been researched and documented more thoroughly. A common approach in ASLR (American Sign Language Recognition) of continuous gestures is to use Hidden Markov Models as classifier models. Hidden Markov Models are an ideal choice because they allow modeling of the temporal evolution of the gesture. In part, the success of HMMs in speech recognition has made it an obvious choice for gesture recognition. Research by Starner and Pentland [9] uses HMMs to recognize continuous sentences in American Sign Language, achieving a word accuracy of 99.2%. Users were required to wear colored gloves and an 8-element feature set, comprising hands' positions, angle of axis of least inertia, and eccentricity of bounding ellipse, was extracted. Lastly, linguistic rules and grammar were used to reduce the number of misclassifications.

Another research study by Starner and Pentland [10] dealt with developing a Real-time ASLR system using a camera to detect bare hands and recognize continuous sentence-level sign language. Experimentation involved two systems: first, using a desk mounted camera to acquire video, that attained 92% recognition and second, mounting the camera in the user's cap, which achieved an accuracy of 98%. This work was based on limited vocabulary data, employing a 40-word lexicon. The authors do not present sentence recognition rates for comparison. Only word recognition and accuracy rates are reported.

This paper is organized as follows. Section 2 describes the Arabic sign language database constructed and used in the work. The methodology followed is enumerated in Section 3. The classification tool used is discussed in Section 4. The results are discussed in Section 5. Concluding remarks along with a primer on future work in presented in Section 6.

2. The Dataset

Arabic Sign Language is the language of choice amongst the hearing and speech impaired community in the Middle East and most Arabic speaking countries. This work involves two different databases; one for isolated gesture recognition and another for continuous sentence recognition. Both datasets are collected in collaboration with Sharjah City for Humanitarian Services (SCHS) [11], no restriction on clothing or background was imposed. The first database was compiled for isolated gesture recognition as reported in [7]. The dataset consists of 3 signers acting 23 gestures. Each signer was asked to repeat each gesture a total of 50 times over 3 different sessions resulting in a total of 150 repetitions of each gesture. The gestures are chosen from the greeting section of the Arabic sign language.

The second database is of a relatively high perplexity consisting of an 80-word lexicon from which 40 sentences were created. No restrictions are imposed on grammar or sentence length. The sentences and words pertain to common situations in which handicapped people might find themselves in. The dataset itself consists of 19 repetitions of each of the 40 sentences performed by only one user. The frame rate was set to 25Hz with a spatial resolution of 720×528. The list of sentences is given in Table 1. Note that this database is the first fully labeled and segmented dataset for continuous Arabic Sign Language. The entire database can be made available on request.

A required step in all supervised learning problems is the labeling stage where the classes are explicitly marked for the classifier training stage. For continuous sentence recognition, not only do the sentences have to be labeled but the individual boundaries of the gestures that make up that sentence have to be explicitly demarcated. This is a time-consuming and repetitive task. Conventionally, a portion of the data is labeled and used as 'bootstrap' data for the classifier which can then learn the remaining boundaries. For the purposes of creating a usable database, a segmented and fully labeled dataset was created in the Georgia Tech Gesture Recognition toolbox (GT^2K) format [12]. The output of this stage is a single master label file (MLF) that can be used with the GT^2K and HTK Toolkits.

3. Feature Extraction

In this section we introduce a feature extraction technique suitable for continuous signing. We also examine some of the existing techniques and adapt them to our application for comparison reasons.

Table 1. Table type styles list of Arabic sentences with English translation used in the recognition system

NO.	Arabic Sentence & English translation
1	ذهبت الى نادي كرة القدم I went to the soccer club
2	انا احب سباق السيارات I love car racing
3	اشتريت كرة ثمينة I bought an expensive ball
4	يوم السبت عندي مباراة كرة قدم On Saturday I have a soccer match
5	في النادي ملعب كرة قدم There is a soccer field in the club
6	غدا سيكون هناك سباق دراجات There will be a bike racing tomorrow
7	وجدت كرة جديدة في الملعب I found a new ball in the field
8	كم عمر اخيك؟ How old is your brother?
9	اليوم ولدت امي بنتا My mom had a baby girl today
10	اخي لا يزال رضيعا My brother is still breast feeding
11	ان جدي في بيتنا My grandfather is at our home
12	اشترى ابني كرة رخيصة My kid bought an inexpensive ball
13	قرأت اختي كتابا My sister read a book
14	ذهبت امي الى السوق في الصباح My mother went to the market this morning
15	هل اخوك في البيت؟ Is your brother home?
16	بيت عمي كبير My brother's house is big
17	سيتزوج اخي بعد شهر In one month my brother will get married
18	سيطلق اخي بعد شهرين In two months my brother will get divorced
19	اين يعمل صديقك؟ Where does your friend work?
20	اخي يلعب كرة سلة My brother plays basketball
21	عندي أخوين I have two brothers
22	ما اسم ابيك؟ What is your father's name?
23	كان جدي مريضا في الامس Yesterday my grandfather was sick
24	مات ابي في الامس Yesterday my father died
25	رأيت بنتا جميلة I saw a beautiful girl
26	صديقي طويل My friend is tall
27	انا لا اكل قبل النوم I do not eat close to bedtime
28	اكلت طعاما لذيذا في المطعم I ate delicious food at the restaurant
29	انا احب شرب الماء I like drinking water
30	انا احب شرب الحليب في المساء I like drinking milk in the evening
31	انا احب اكل اللحم اكثر من الدجاج I like eating meat more than chicken
32	اكلت جبنة مع عصير I ate cheese and drank juice
33	يوم الاحد القادم سيرتفع سعر الحليب Next Sunday the price of milk will go up
34	أكلت زيتونا صباح الامس Yesterday morning I ate olives
35	ساشتري سيارة جديدة بعد شهر I will buy a new car in a month
36	هو توضأ ليصلي الصبح He washed for morning prayer
37	ذهبت الى صلاة الجمعة عند الساعة العاشرة I went to Friday prayer at 10:00 o'clock
38	كبيرا بالتلفاز شاهدت بيتا I saw a big house on TV
39	في الامس نمت عند الساعة العاشرة Yesterday I went to sleep at 10:00 o'clock
40	ذهبت الى العمل في الصباح بسيارتي I went to work this morning in my car

3.1 Proposed Feature Extraction

The most crucial stage of any recognition task is the selection of good features. Features that are representative of the individual gestures are desired. Shanableh *et al.* [7] demonstrated in their earlier work on isolated gesture recognition that the two-tier spatial-temporal feature extraction scheme results in a high word recognition rate close to 98%. Similar extraction techniques are used in our continuous recognition solution.

First, to represent the motion that takes place as the expert signs a given sentence, pixel-based differences of successive images are computed.

It can be justified that the difference between two images of similar background results in an image that only preserves the motion between the two images. These image differences are then converted into binary images by applying an appropriate threshold. A threshold value of $\mu+x\sigma$ is used where μ is the mean pixel intensity of the image difference, σ is the corresponding standard deviation and x is a weighting parameter which was empirically determined based on subjective evaluation whose criteria was to retain enough motion information and discarding the noisy data.

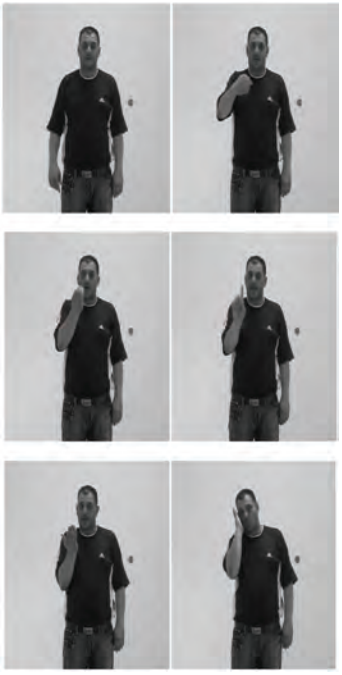
Figure 1 shows an example sentence with thresholded image differences. Notice that the example sentence is temporally downsampled for illustration purposes.

Next, a frequency domain transformation such as the Discrete Cosine Transform (DCT) is performed on the binary image differences.

The 2-D Discrete Cosine Transformation (DCT) given by [13]:

$$F(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f(i,j)\cos\left(\frac{\pi u}{2M}\cdot(2i+1)\right)\cos\left(\frac{\pi v}{2N}\cdot(2j+1)\right) \quad (1)$$

where NxM are the dimensions of the input image 'f' and F(u,v) is the DCT coefficient at row u and column v of the DCT matrix. C(u) is a normalization factor equal to $1/\sqrt{2}$ for u=0 and 1 otherwise.



(a) An image sequence denoting the sentence 'I do not eat close to bed time'.

(b) Thresholded image differences of the image sequence in part (a)

Figure 1. An example sentence and its motion representation

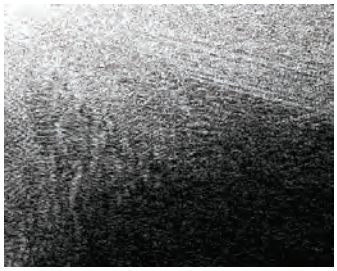


Figure 2. Discrete cosine transform coefficients of a thresholded image difference

In Figure 2, it is apparent that the DCT transformation of a thresholded image difference results in energy compaction where most of the image information is represented in the top left corner of the transformed image. Subsequently, zig-zag scanning is used to select only a required number of frequency coefficients. This process is also known as zonal coding. The number of coefficients to retain or the DCT cutoff is elaborated upon in the experimental results section.

These coefficients obtained in a zig-zag manner make up the feature vector that is used in training the classifier.

3.2 Adapted Feature Extraction Solutions

For completeness, we compare our feature extraction solution to existing work on Arabic sign language recognition. Noteworthy are the Accumulated Differences (ADs) and Motion Estimation (ME) approaches to feature extraction as reported in [8,16]. In this section we provide a brief review of each of mentioned solutions and explain who it can be adapted to our problem of continuous Arabic sign language recognition

3.2.1 Accumulated Differences Solution

The motion information of an isolated sign gesture can be computed from the temporal domain of its image sequence through successive image differencing. Let $I_{g,i}^{(j)}$ denote image index j of the i^{th} repetition of a gesture at index g. The Accumulated Differences (ADs) image can be computed by:

$$AD_{g,j} = \sum_{j=1}^{n-1} \partial_j \left(\left| I_{g,j}^{(j)} - I_{g,i}^{(j-1)} \right| \right) \qquad (2)$$

where n is the total number of images in the i^{th} repetition of a sign at index g. ∂_j is a binary threshold function of the j^{th} frame.

Note that the ADs solution cannot be directly applied to continuous sentences (as opposed to isolated sign gestures). This is so because the gesture boundaries in a sentence are unknown, thus one solution is to use an overlapping sliding window approach in which a given number of video frame differences are accumulated into one image regardless of gesture boundaries. The window

is shift by one video frame at a time. In the experimental results section we experiment with various window sizes. Examples of such accumulated differences are shown in Figure 3 with a window size of 8 video frames. Notice that the ADs capture the frame difference between the current and previous video frames and it also accumulates the frame differences from the current window as well.

Once the ADs image is computed it is then transformed into the DCT domain as described previously. The DCT coefficients are Zonal coded to generate the feature vector.

3.2.2 Motion Estimation solution

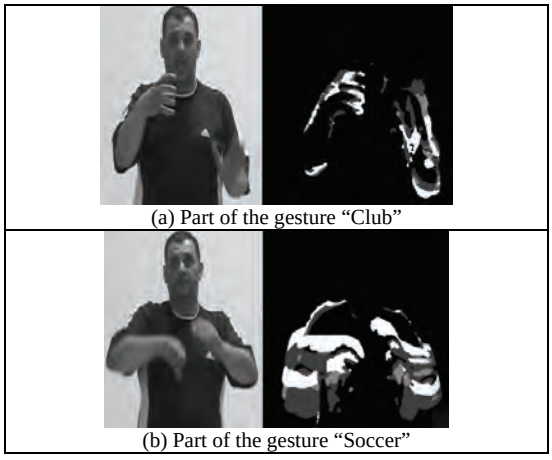
The motion of a video-based sign gesture can also be tracked by means of Motion Estimation (ME). One well known example is the block-based ME in which the input video frames are divided into non-overlapping blocks of pixels. For each block of pixels, the motion estimation process will search through the previous video frame for the "best match" area within a given search range. The displacement, in terms of pixels, between the current block and its best match area in the previous video frame is represented by a motion vector

Formally, let *C* denote a block in the current video frame with *bxb* pixels at coordinates *(m,n).* Assuming that the maximum motion displacement is *w* pixel per video frame then the the motion estimation process will find the best match area *P* within the *(b+2w)(b+2w)* distinct overlapping *bxb* blocks of the previous video frame. An area in the previous video frame that minimizes a certain distortion measure is selected as the best match area. A common distortion measure is the mean absolute difference given by:

$$M(\Delta x, \Delta y) = \frac{1}{b^2} \sum_{m=1}^{b} \sum_{n=1}^{b} \left| C_{m,n} - P_{m+\Delta x, n+\Delta y} \right|, \; -w \le \Delta x, \Delta y \le w \quad (3)$$

where $\Delta x, \Delta y$ refer to the spatial displacement in between the pixel coordinates of C and the matching area in the previous image. Other distortion measures can be used such as mean squared error, cross correlation functions and so forth. Further details on motion estimation can be found in [17] and references within.

The motion vectors can then be used to represent the motion the occurred between two video frames. These vectors are used instead of the thresholded frame differences. In [8] it was proposed to rearrange the *x* and y components of the motion vectors into two intensity images. The two images are then concatenated to generate one representation of the motion that occurred between two video frames. Again, once the concatenated image is computed it is then transformed into the DCT domain as described previously. The DCT coefficients are Zonal coded to generate the feature vector.



(a) Part of the gesture "Club"

(b) Part of the gesture "Soccer"

Figure 3. Example accumulated differences images using an overlapping sliding window of size 8

4. Classification

For conventional data, naïve Bayes classification provides the upper bound for the best classification rates. Since sign language varies in both spatial and temporal domains, the extracted feature vectors are sequential in nature and hence simple classifiers might not suffice. There are two main approaches to dealing with sequential data. The first method aims to combine the sequential feature vectors using a suitable operation into a single feature vector. A detailed account of such procedures is outlined in [14]. One such method involves concatenating sequential feature vectors using a sliding window of optimal length to create a single feature vector. Subsequently, classical supervised learning techniques such as maximum-likelihood estimation (MLE), linear discriminants or neural networks can be used. The second approach makes explicit use of classifiers that can deal with sequential data without concatenation or accumulation, such an approach is used in this paper. While the field of gesture recognition is relatively young, the related field of speech recognition is well established and documented. Hidden Markov Models are the classifier of choice for continuous speech recognition and lend themselves suitably for continuous sign language recognition too. As mentioned in [15], a HMM is a finite-state automaton characterized by stochastic transitions in which the sequence of states is a Markov chain. Each output of an HMM corresponds to a probability density function. Such a generative model can be used to represent sign language units (words, sub-words etc).

The implementation of an HMM framework was carried out using the Georgia Tech Gesture Recognition Toolkit (GT^2K) which serves as a wrapper for the more general Hidden Markov Model Toolkit (HTK). The GT^2K version used was a UNIX based package. HTK is the de-facto standard in speech recognition application

using HMM's.

5. Experimental Results

A logical step in proceeding from isolated gesture recognition would be connected gesture recognition. This can be simulated by concatenating individual gestures into artificial sentences. Intuitively, one would expect better results for connected gesture recognition as opposed to continuous gesturing. This is because concatenated gestures do not suffer from the altered spatial gesturing that occurs as gestures are signed continuously without pauses. The first database consisting of isolated gestures was used to create concatenated sentences of varying length. These sentences were created without any consideration of whether the constructed sentence held any meaning or grammatical structure. This concatenated data was divided into a training set and a testing set comprised of 70% and 30% of the total data respectively. The GT^{2}K Toolkit was then used to perform recognition based on individual words as the basic unit of Arabic sign language. While concatenation is not the aim of this work, the results obtained provide a valuable benchmark for subsequent experiments with continuous sentence signing. An average of 96% sentence recognition and 98% word recognition was obtained on the concatenated testing dataset. The word recognition rate is comparable to previous work in ArSL [7] using similar feature extraction schemes. It would be prudent to note that due to the nature of concatenation, the boundary between gestures is prominent and this might account for the high sentence recognition rate.

The second database was then used to perform continuous sentence recognition. This was also performed with the help of GT^{2}K. This data is also divided into a training (70%) and testing set (30%). An average of 75% sentence recognition and 94% word recognition was obtained on the testing set. A detailed analysis of the various associated parameters is given in the following discussion.

There are several parameters that affect the recognition rates in continuous sign language recognition. Namely, the sections below discuss the effect of varying the number of hidden states, number of guassian mixtures, length of feature vectors and the threshold used for binarizing the image differences. Unless otherwise stated, the length of the feature vectors used throughout the experiments is 100 DCT coefficients.

The following results are based on the word and sentence recognition rates. The former is computed through the following equation:

$$Accuracy_{word} = 1 - \frac{D + S + I}{N} \quad (4)$$

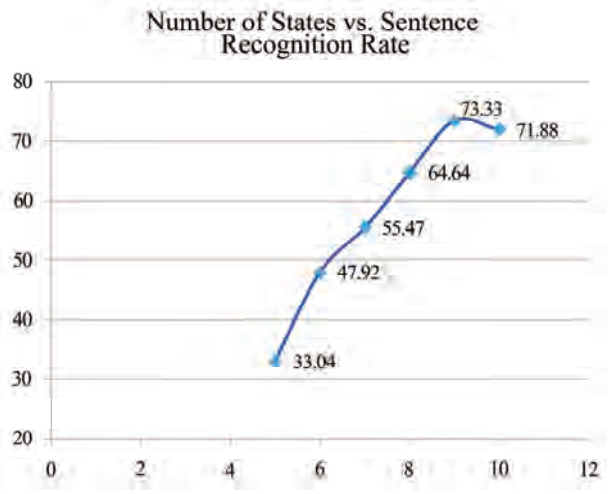


Figure 4. The effect of number of states on the sentence recognition rate (3 Gaussian mixtures are used)

Where D is the number of deletions, S is the number of substitutions, I is the number of insertions, and N is the total number of words. On the other hand, sentence recognition rate is the ratio of the correctly recognized sentences to the total number of sentences. Correctness in this case entails correct recognition of all the words constituting the sentence without any insertions, substitutions, or deletions.

5.1 Number of Hidden States

In Figure 4, the effect of increasing the number of hidden states in the HMM topology on sentence and word recognition rates is examined.

An increasing trend with the recognition rates is observed as the number of states is increased to a certain number and then the classification rates saturates with a subsequent drop. As the number of states in the Hidden Markov Model is increased, we are in effect increasing the degrees of freedom allowed in modeling the data. Working with video data sampled at 25 frames per second, the classification rate increased to a maximum at nine states.

The saturation in recognition accuracy is attributed to the fact that certain gestures do not extend for a long time duration and are only represented by few frame differences. The increase in number of states only serves to increase computation time while adding redundant data that does not contribute to the classification rate.

5.2 Length of the Feature Vector

Figure 5 shows the recognition rates for increasing the number of DCT coefficients within the feature vector.

It is expected that the increase in feature vector size be accompanied by a corresponding increase in recognition rates. This is due to the fact that each DCT coefficient is

uncorrelated with other coefficients and hence no redundant information is present in increasing coefficients. Experimental results shown in Figure 5 show a general increase in recognition rates as the number of DCT coefficients is increased.

The trend shows that any increase in recognition accuracy beyond 100 coefficients is only slightly significant. The increase in computation time is however a limiting factor in increasing the length of the feature vector indiscriminately.

5.3 Number of Gaussian Mixtures

The effect of increasing the number of Gaussian mixtures is shown in Figures 6 and 7.

Gaussian mixtures are used to model the emission probability densities of each state of a continuous Hidden Markov Model.

For multi-dimensional data like the long feature vectors used in this work it is desirable to have a number of Gaussian mixtures so that any emission pdf can be effectively fit. Increasing the number of Gaussian mixture shows substantial improvement in recognition rates. The results depict a general increase in recognition rates as the mixtures are increased. However, the authors feel that the limitation in collecting large amounts of data does not allow the use of more mixtures.

5.4 Choice of Threshold

In the feature extraction process, the image differences are thresholded into binary images based on a threshold of $\mu+x\sigma$, where μ is the mean pixel intensity of the image difference, σ is the corresponding standard deviation and x is a weight parameter. Results are shown in Figure 8 and 9 for different values of the weight parameter.

The recognition accuracy peaks at a weighting parameter value between 1 and 1.25. A subjective comparison of the thresholded image differences shows that these parameter values retain most of the motion information whilst discarding spurious information such as small stray shifts in clothing, illumination and the like.

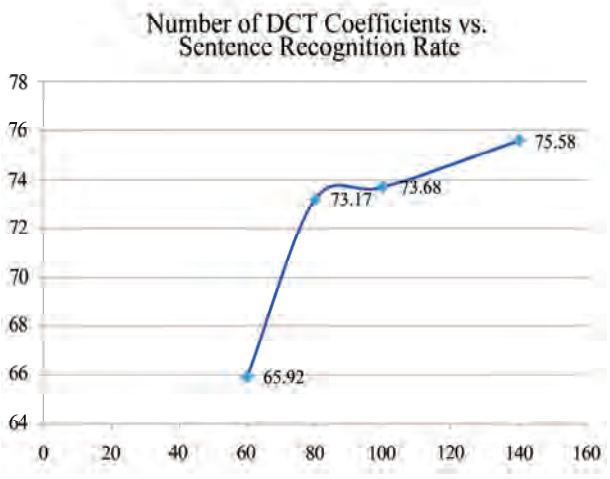


Figure 5. The effect of length of the feature vector on sentence recognition rates (3 Gaussian mixtures are used with a HMM topology using 9 states)

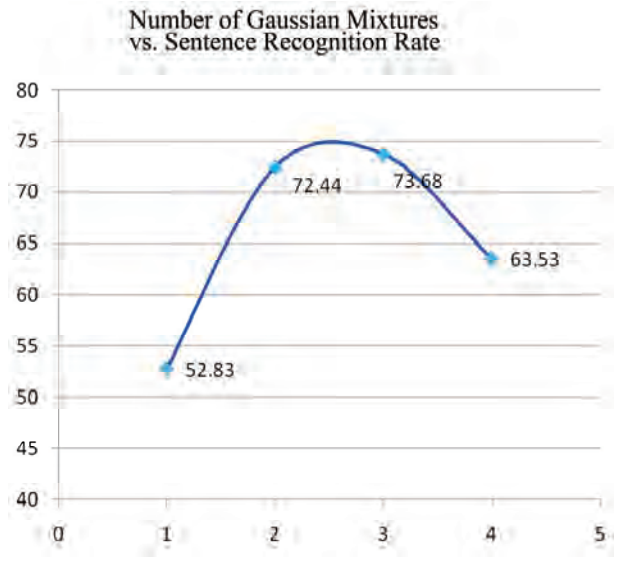


Figure 6. The effect of the number of gaussian mixtures on the sentence recognition rate (HMM topology using 9 states)

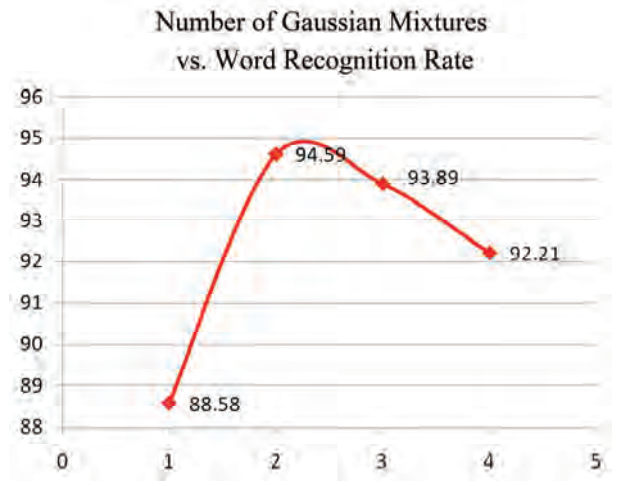


Figure 7. The effect of the number of gaussian mixtures on the word recognition rate (HMM topology using 9 states)

Lastly as mentioned in Section 3.2 above, we compare our solution against existing work on Arabic sign language recognition. Namely we consider both the Ads and ME approaches to feature extraction. Table 2 summaries the sentence and word recognition rates using various feature extraction solutions. The experimental parameters are similar to those used in Figure 9.

The recognition results presented in the table indicate that the proposed solution provides the highest sentence

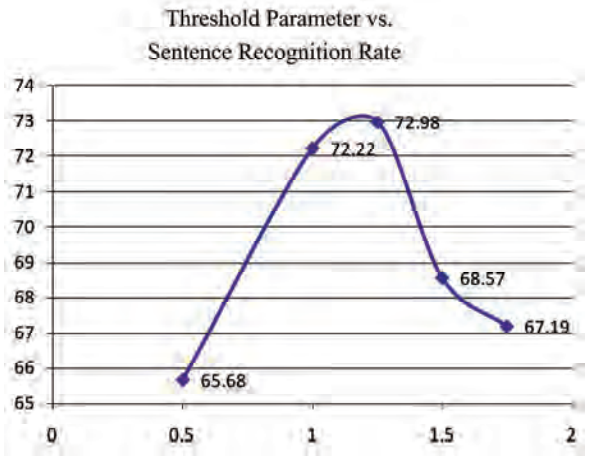


Figure 8. The effect of the weighting factor on the sentence recognition rate (3 Gaussian mixtures are used with a HMM topology using 9 states)

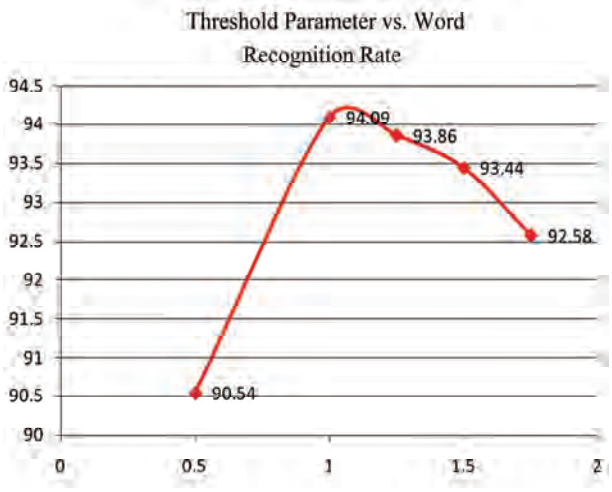


Figure 9. Effect of the weighting factor on the word recognition rate (3 Gaussian mixtures are used with a HMM topology using 9 states)

and word recognition rates. The ADs with the overlapping sliding window approach was not advantageous. Intuitively the ADs image puts the difference between 2 video frames into context by accumulating future frame differences to it. However in HMMs temporal information is preserved and therefore extracting feature vectors from video frame differences without accumulating them will suffice. It is also worth mentioning that increasing the window size beyond 10 frames did not further enhance the recognition rate.

In the ME approach, the image block size and the search range are set to 8x8 pixels which is a typicalsetting in video processing. The resultant recognition rates are comparable to the ADs approach. Note that ME tech-niques do not entirely capture the true motion of a video sequence. For instance with block-based search techniques object rotations are not captured as good as translational motion. Therefore the recognition results are inferior to the proposed solution.

Table 2. Comparisons with existing feature extraction solutions

Feature extraction approach	Sentence recognition rate	Word recognition rate
ADs with an overlapping Sliding window of size 4.	64.1%	91.0%
ADs with an overlapping Sliding window of size 7.	65.2%	90.6%
ADs with an overlapping Sliding window of size 10.	68%	93.71%
Motion estimation	67.9%	92.9%
Proposed solution	73.3%	94.39%

6. Conclusions and Future Work

The work outlined in this paper is an important step in this domain as it represents the first attempt to recognize continuous Arabic sign language. The work entailed compiling the first fully labeled and segmented dataset for continuous Arabic Sign Language which we intend to make public for the research community. The average sentence recognition rate of 75% and word recognition-rate of 94% are obtained using a natural vision-based system with no restrictions on signing such as the use of gloves. Furthermore, no grammar is imposed on the sentence structure which makes the recognition task more challenging. The use of grammatical structure can significantly improve the recognition rate by alleviating some types of substitution and insertion errors. In the course of training, the dataset was plagued by an unusually large occurrence of insertion errors. This problem was mitigated by applying a detrimental weight for every insertion error which was incorporated into the training stage. As a final comment, the perplexity of the dataset is large compared to other work in related fields. Future work in this area aiming to secure higher recognition rates might require a sub-gesture based recognition system. Such a system would also serve to alleviate the motion-epenthesis effect which is similar to the co-articulation effect in speech recognition.

Finally, the frequency domain transform coefficients used as features perform well in concatenated gesture recognition. The average word recognition rate is also sufficiently high with an average of 94%. The authors feel that geometric features might be used in addition to the existing feature to create an optimum feature set.

7. Acknowledgements

The authors acknowledge Emirates Foundation for their support. They also acknowledge the Sharjah City for Humanitarian Services for availing Mr. Salah Odeh to

help in the construction of the databases.

REFERENCES

[1] J. S. Kim, W. Jang, and Z. Bien, "A dynamic gesture recognition system for the Korean sign language (KSL)," IEEE Transations on Systerms, Man, Cybern. Part B, Vol. 26, pp. 354–359, April 1996.

[2] H. Matsuo, S. Igi, S. Lu, Y. Nagashima, Y. Takata, and T. Teshima, "The recognition algorithm with noncontact for Japanese sign language using morphological analysis," Proceedings of Gesture Workshop, pp. 273–284, 1997.

[3] S. S. Fels and G. E. Hinton, "Glove-talk: A neural network interface between a data-glove and a speech synthesizer," IEEE Transactions on Neural Networks, Vol. 4, pp. 2–8, January 1993.

[4] O. Al-Jarrah and A. Halawani, "Recognition of gestures in Arabic sign language using neuro-fuzzy systems," Artificial Intelligence, Vol. 133, No. 1–2, pp. 117–138, December, 2001.

[5] K. Assaleh and M. Al-Rousan, "Recognition of Arabic Sign language alphabet using polynomial classifiers," EURASIP Journal on Applied Signal Processing, Vol. 2005, No. 13, pp. 2136–2146, 2005.

[6] M. AL-Rousan, K. Assaleh, and A. Tala'a, "Video-based signer-independent Arabic sign language recognition using hidden Markov models," Applied Soft Computing, Vol. 9, No. 3, June 2009.

[7] T. Shanableh, K. Assaleh, and M. Al-Rousan, "Spatio-temporal feature-extraction techniques for isolated gesture recognition in Arabic sign language," IEEE Transactions on Systems, Man and Cybernetics Part B, Vol. 37, No. 3, pp. 641–650, 2007.

[8] T. Shanableh and K. Assaleh, "Telescopic vector composition and polar accumulated motion residuals for feature extraction in Arabic sign language recognition," EURASIP Journal on Image and Video Processing, vol. 2007, Article ID 87929, 10 pages, 2007. doi:10.1155/2007/ 87929.

[9] T. Starner, J. Weaver, and A. Pentland. "Real-time American sign language recognition using desk and wearable computer based video," IEEE Transations on Pattern Analysis and Machine Intelligence, Vol. 20, No. 12, pp. 1371–1375, 1998.

[10] T. Starner, J. Weaver and A. Pentland. "A wearable computer-based American sign language recogniser," in personal and ubiquitous computing, 1997.

[11] Sharjah City for Humanitarian Services (SCHS), website: http://www.sharjah-welcome.com/schs/about/.

[12] T. Westeyn, H. Brashear and T. Starner, "Georgia tech gesture toolkit: Supporting experiments in gesture recognition," Proceedings of International Conference on Perceptive and Multimodal User Interfaces, Vancouver, B.C., November 2003.

[13] K. R. Rao and P. Yip, "Discrete cosine transform: Algorithms, advantages, applications," Academic Press, ISBN 012580203X, August 1990.

[14] T. Dietterich, "Machine learning for sequential data: A review," In T. Caelli (Ed.) Structural, Syntactic, and Statistical Pattern Recognition; Lecture Notes in Computer Science, Vol. 2396, pp. 15–30, Springer-Verlag, 2002.

[15] L. Rabiner, B. Juang, "Fundamentals of speech recognition," New Jersey: Prentice-Hall Inc., 1993.

[16] F. S. Chen, C. M. Fu, and C. L. Huang, "Hand gesture recognition using a real-time tracking method and hidden Markov models," Image and Vision Computing, Vol. 21, No. 8, pp. 745–758, 2003.

[17] M. Ghanbari, "Video coding: An introduction to standard codecs," Second Edition, The Institution of Engineering and Technology, 2003.

Journal of Intelligent Learning Systems and Applications, **2010, 2: 28-35**
doi:10.4236/jemaa.2010.21004 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Determination of Optimal Manufacturing Parameters for Injection Mold by Inverse Model Basing on MANFIS

Chung-Neng Huang[1], Chong-Ching Chang[2]

Graduate Institute of Mechatronic System Engineering, National University of Tainan, Tainan, Taiwan, China.
Email: kosono@mail.nutn.edu.tw, jeff0718@mail.nutn.edu.tw

Received October 29th, 2009; accepted January 11th, 2010.

ABSTRACT

Since plastic products are with the features as light, anticorrosive and low cost etc., that are generally used in several of tools or components. Consequently, the requirements on the quality and effectiveness in production are increasingly serious. However, there are many factors affecting the yield rate of injection products such as material characteristic, mold design, and manufacturing parameters etc. involved with injection machine and the whole manufacturing process. Traditionally, these factors can only be designed and adjusted by many times of trial-and-error tests. It is not only waste of time and resource, but also lack of methodology for referring. Although there are some methods as Taguchi method or neural network etc. proposed for serving and optimizing this problem, they are still insufficient for the needs. For the reasons, a method for determining the optimal parameters by the inverse model of manufacturing platform is proposed in this paper. Through the integration of inverse model basing on MANFIS and Taguchi method, inversely, the optimal manufacturing parameters can be found by using the product requirements. The effectiveness and feasibility of this proposal is confirmed through numerical studies on a real case example.

Keywords*: *Optimal Manufacturing Parameter, Injection Mold, Multiple Adaptive Network Based Fuzzy Inference System (Manfis), Taguchi Method

1. Introduction

Recently for the surge in the prices of fuel and raw materials like steel or iron, plastic goods used in industries and everyday life are taking the place of metal ones. Generally, since those products combined by pieces of parts required higher precision and smoothness, the demands on quality and efficiency of production become higher than before. In order to level up the yield rate of made-up articles, the manufacturing process should be improved for the required of different goods [1]. Nowadays, for coping with the diversifying demands of present markets, developed countries in industry have been introducing the technologies of computer-integrated manufacture (CIM) as CAE/CAD/CAM to get competitive advantages [2–3]. That is, for the manufacturing process of an industrial product with completed design, first, its prototype is designed by the original concept. Next, through computer-aided design (CAD) tool complete the initial design. Third, by the analysis technology of computer-aided engineering (CAE) to test and modify the design. Finally, depending on the better design, automotive production can be done by computer-aided manufacture (CAM).

Before concurrent engineering attracting much attention, the technologies of computer-aided engineering analysis were seldom used to estimate designing faults by manufacturers in advance. Where, mold design and manufacturing process should be modified through many times of trial-and-error tests [4–6]. It not only wastes time and cost but also makes such experiences became more difficult in teaching or accumulating. Besides, under the situation of different product required or new materials, the awkward problems as one more times of teaching experience and molding can not be avoided. Sometimes part of business chances may be losing for it.

The most helpful function of CAE is to carry out simulation analysis of prototype design by computers [6]. By which, all possible problems and faults occurring in manufacturing and design stages can be found in advance. It is convenient to diagnose and modify designed before product manufacture for reducing cost and time, and lev-

eling up quality. However, even though those modern computer-aided technologies as mentioned above play an important role in manufacture, the subject of how to determine optimal manufacturing parameters for extremely matching product required still exists [7–24]. Although there are a lot of methods such as statistical regression calculation, neural network model and genetic algorithm, grey relational analysis, and fuzzy theory etc. proposed for optimizing parameters [25–30], lacking of methodology and integration. For it, a concept for building the inverse model of manufacturing platforms by multiple adaptive network fuzzy inference system (MANFIS) is proposed. Through data self-organized and deductive reasoning mechanisms of MANFIS, the optimal manufacturing parameters corresponding to product required can be found. In this paper, the blade of a small-scale wind power generator is selected as a real case studying on injection mold. Through the simulation results by computer-aided analysis software Moldex3D, the appropriateness and effectiveness of the proposal can be confirmed.

2. Solution Design and Problem Statement

2.1 Solution Description

The main purpose of this study is to determine the optimal manufacturing parameters for injection mold. According to the literatures mentioned above know that the manufacturing parameters of injection mold are highly interdependent. That is, the whole system should be considered while part of parameters is undertaken to modify. Here, a method for finding out the optimal parameters is proposed. Figure 1 shows the executing flow of the method. First, since there are always a lot of manufacturing parameters as well as controllable factors existing, in order to realize which ones are the key factors and reduce time and flows in computation, a less number of important factors with more controllability can be extracted through the calculation of Taguchi method. Next, instead of all possible experimental combinations to simulator, the orthogonal arrays basing on those important factors are developed. In addition, for the results found by Taguchi method are unique, and possibly trapping in local optimum, a decimal-fraction random matrix as the numerical stirring is introduced into the orthogonal arrays for wider-range simulation. Finally, by using the simulated results such as warpage displacement or volumetric shrinkage etc. along with the corresponded orthogonal arrays, the proposed inverse model can be built through MANFIS.

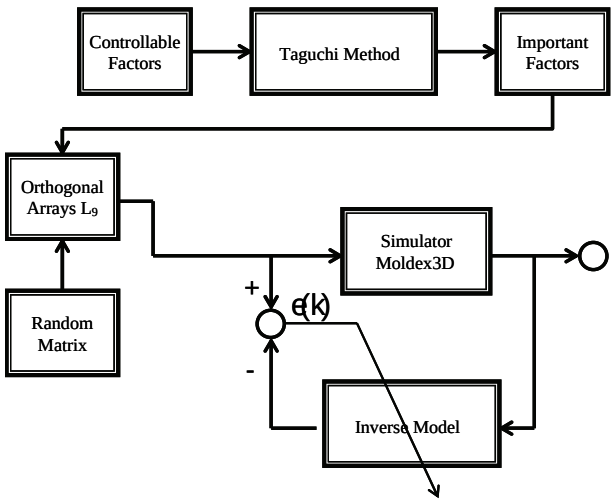


Figure 1. Flow of proposed method

2.2 Real Case Selection

The real case selected for confirming the proposed method is the manufacturing design of a blade for a small-scale wind power generator. Since the blades are the key part of such generators for their generation efficiency and cost, the weight, smoothness, surface friction, physical stress, and twisting angles etc. of them are required seriously in manufacture. In addition, instead of FRP which is denounced by its environmental pollution, the material ABS_NovodurP2GHV_1 is adopted to study. Here, through the analysis of momentum theory and blade element model, the geometric data of the blade is determined as shown in Figure 2. Moreover, the hot and cooling distributions by one-point injection and four groups of cooling runners are shown in Figure 3.

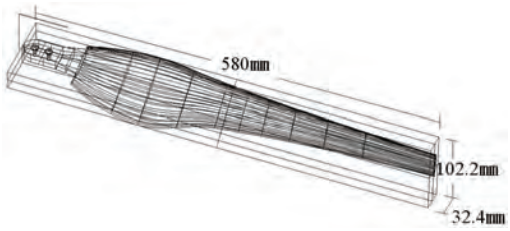


Figure 2. Studying case designed by 3D's flow

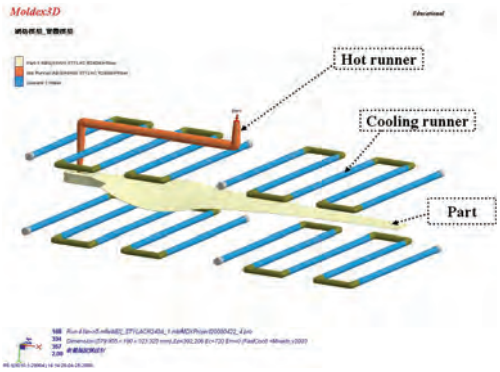


Figure 3. Distribution of hot runner and cooling runner

3. Numerical Studies

Since the initial controllable factors are always selected by the product required. Through the analysis of fish bone diagram shown in Figure 4 and the consideration of required product strength, there are eight factors selected such as fiber percentage of material, material temperature, injection pressure, holding time, holding pressure, mold temperature, cooling time, and filling time.

3.1 Selection of Important Factors by Taguchi Method

Two major tools used in the Taguchi method are the orthogonal arrays and the signal-to-noise ratio. Additional details and application of Taguchi method can be found in the books presented by Phadke [31], Montgomery [32], and Park [33]. In this paper, three- level orthogonal arrays are used. The design parameters and the levels chosen for the Taguchi experiments are listed in Table1. Continuously, a L_{18} (3^8) orthogonal arrays with eight columns and eighteen rows can be developed as shown in Table 2. Each design parameter has three levels assigned to each column of the arrays. The eighteen rows represent the eighteen experiments to be conducted.

Since the assessing indices are the warpage displacements and volumetric shrinkages in three dimensions as x, y, and z axes, respectively, through the computations of simulator Moldex3D all indices corresponding to all experimental combinations in L_{18} (3^8) orthogonal arrays can be found. By substituting these indices into Equations 1 to 3, the important factors and optimal combination for Taguchi method can be extracted and found by assessing the quality characteristic (in Figure 5) and signal-to-noise ratio (in Figure 6). By above results realized that the factors with more controllability as mold temperature, material temperature, injection pressure, and holding time are selected as the important factors.

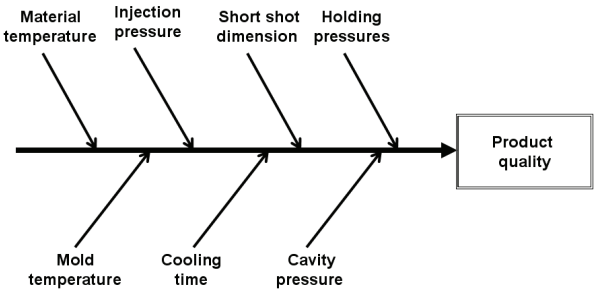


Figure 4. Fish bone diagram for factor analysis

Table 1. Eight controllable factors with three levels

factor \ level	1	2	3
A. (%) percentage of fiber contents	20	16	
B. (℃) material temperature	210	225	240
C.(MPa) injection pressure	90	105	120
D. (S) holding time	2	4	6
E. (MPa) holding pressure	63	73.5	84
F. (℃) mold temperature	50	70	87
G. (S) cooling time	10	20	30
H. (S) filling time	2.3	3.65	5

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{1}$$

$$S = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2}{n-1}} \tag{2}$$

$$S/N = -10 \quad log \frac{S^2}{\bar{y}^2} \tag{3}$$

3.2 Collection of Training Data Sets

For training the inverse model to be with more comprehensively deductive reasoning, all possible combinations basing on the changes of four important factors should be collected in general. However, it would be a cumbersome task for experiment or computation. For the main advantages of orthogonal arrays including experimental plan simplification and feasibility of studying interaction effects among the different parameters, a L_9 (3^4) orthogonal arrays developing with three levels (Table 3) from above four important factors is built as shown in Table 4. Moreover, for more detailed numerical data, a random matrix (Table 5) as well as a stirring is introduced into the L_9 (3^4) orthogonal arrays. Table 6 shows the input-output training data sets through the computation of simulator Moldex3D.

Table 2. L_{18} (3^8) orthogonal arrays

Exp	A	B	C	D	E	F	G	H
1	1	1	1	1	1	1	1	1
2	1	1	2	2	2	2	2	2
3	1	1	3	3	3	3	3	3
4	1	2	1	1	2	2	3	3
5	1	2	2	2	3	3	1	1
6	1	2	3	3	1	1	2	2
7	1	3	1	2	1	3	2	3
8	1	3	2	3	2	1	3	1
9	1	3	3	1	3	2	1	2
10	2	1	1	3	3	2	2	1
11	2	1	2	1	1	3	3	2
12	2	1	3	2	2	1	1	3
13	2	2	1	2	3	1	3	2
14	2	2	2	3	1	2	1	3
15	2	2	3	1	2	3	2	1
16	2	3	1	3	2	3	1	2
17	2	3	2	1	3	1	2	3
18	2	3	3	2	1	2	3	1

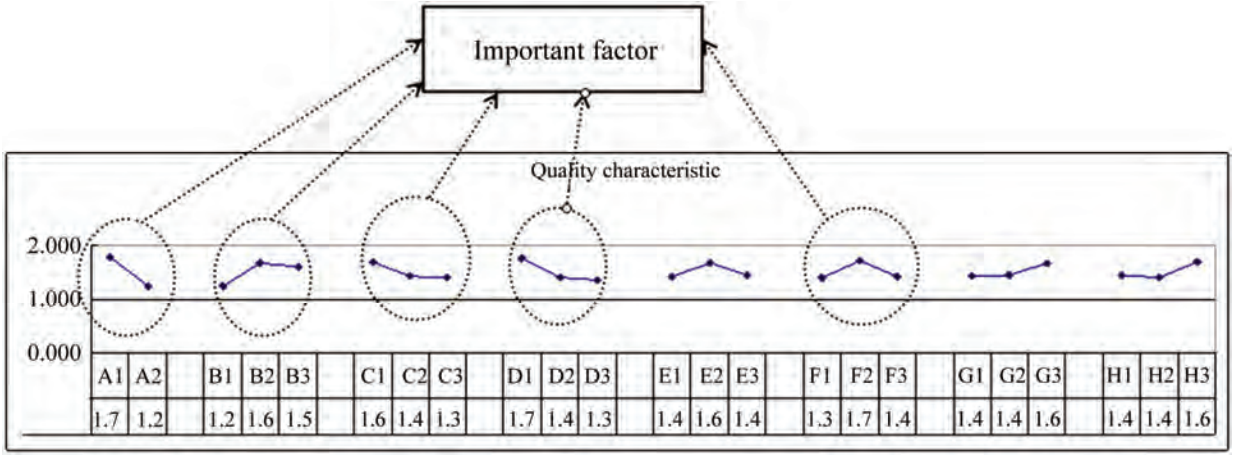


Figure 5. Quality characteristic (2、B1、C3、D3、E1、F1、G1、H2)

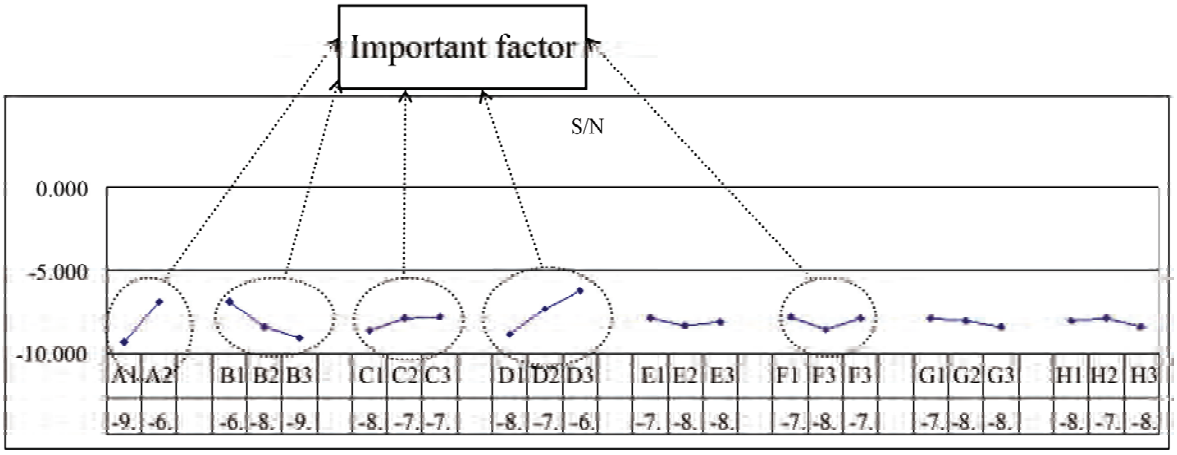


Figure 6. S/N ratio (A2、B1、C3、D3、E1、F1、G1、H2)

Table 3. Four important factors with three levels

factor \ level	1	2	3
B. (℃) material temperature	220	240	260
C.(MPa) injection pressure	110	120	130
D. (S) holding time	4	6	8
F. (℃) mold temperature	45	50	55

Table 4. $L_9(3^4)$ orthogonal arrays

Exp	B	C	D	F
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Table 5. Random matrix example

0.95	0.2311	0.6068	0.486
0.891	0.7621	0.4565	0.0185
0.821	0.4447	0.6154	0.7919
0.922	0.7382	0.1763	0.4057
0.935	0.9169	0.4103	0.8937
0.058	0.3529	0.8132	0.0099
0.139	0.2028	0.1987	0.6038
0.272	0.1988	0.0153	0.7468
0.445	0.9318	0.466	0.4187

3.3 Inverse Model

The proposed inverse model for finding out the optimal manufacturing parameters corresponding to product required is built by MANFIS (in Figure 7), which is an extension of ANFIS to produce multiple real responses of the target system. The number of ANFIS is equal to the number *n* of responses. ANFIS is a fuzzy inference

Table 6. Training data sets for MANFIS

exp	B	C	D	F	X	Y	Z	V
1	220.9	110.2	4.6	45.4	0.402	0.053	-0.129	4.14
2	220.8	120.7	6.4	50.0	0.581	0.058	-0.109	3.962
3	220.8	130.4	8.6	55.7	0.302	0.132	-0.038	3.774
4	240.9	110.7	6.1	55.4	0.326	0.017	-0.083	4.774
5	240.9	120.9	8.4	45.8	0.358	0.004	-0.106	4.568
6	240.0	130.3	4.8	50.0	0.577	-0.041	-0.137	4.888
7	260.1	110.2	8.1	50.6	0.305	0.028	-0.166	5.32
8	260.2	120.2	4.0	55.7	0.483	-0.068	-0.238	5.796
9	260.4	130.9	6.4	45.4	0.69	0.096	-0.131	5.515

X：warpage displacement in x axis
Y：warpage displacement in y axis
Z：warpage displacement in z axis
V：warpage-volumetric shrinkage

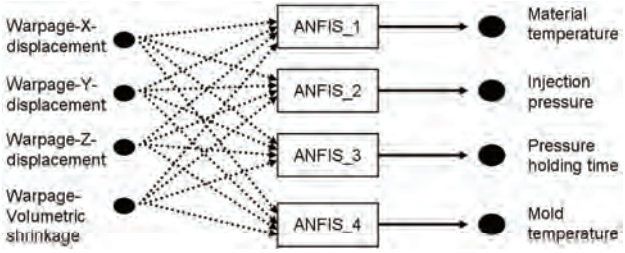


Figure 7. The structure of MANFIS

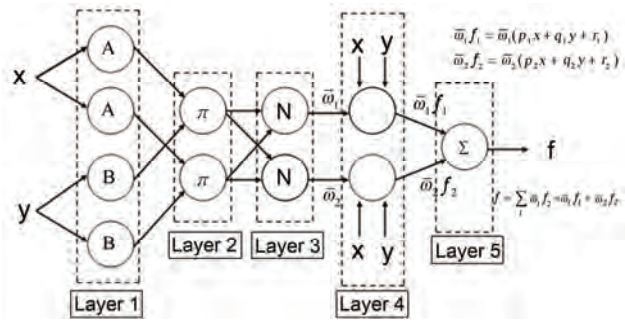


Figure 8. Five-layer structure of ANFIS

system (FIS) implemented in the framework of an adaptive fuzzy neural network. FIS is the process of formulating the mapping from a given input to an output using fuzzy logic.

ANFIS is based on Tagaki-Sugeno FIS. ANFIS generally has two inputs, one output and its rule base contains two fuzzy if-then rules:

Rule l: If x is A_1 and y is B_2 then $f_1 = p_1 + q_1 + r_1$.

Rule 2: If x is A_2 and y is B_2 then $f_2 = p_2 + q_2 + r_2$.

The five-layered structure of this ANFIS is depicted in Figure 8. The detailed description about it can refer to the studies by R. Jang *et al.* [34–37]. Here, by using the training data sets in Table 6 and through thirty times of training, the errors of the unknown constants in each node of MANFIS have been convergent.

4. Analysis and Discussion

For a complete-trained inverse model that is charac- terized with the inverse function of the simulator Moldex3D as well as manufacturing platform. That is, the manufacturing parameters can be found by the product required inversely. Here, the correlations between two kinds of product required and one manufacturing parameter are shown by 3D mesh diagrams in Figure 9. In addition to identify the reasonable areas for product required, the limits to the four important factors in the real case are set as following; mold temperature: over $40^{\circ}C$, material temperature: over $210^{\circ}C$, injection pre- ssure: over 90Mpa, holding time: the smaller warpage the better.

Moreover, for the convenience in observation, the reasonable intervals of each product required are summarized in Table 7.

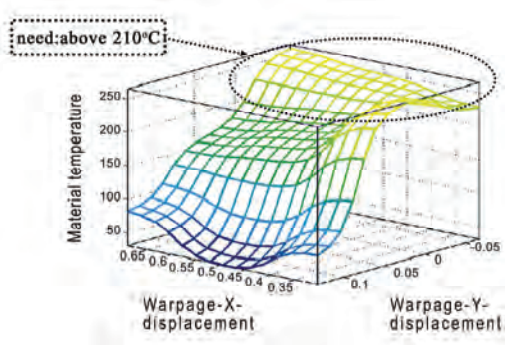
In order to confirm the reliability and preciseness of inverse model, two groups of numerical comparisons are made as shown in Table 8. Here, by comparing with the inputs of inverse model with the outputs of simulator which are corresponding to the outputs of inverse model, the differences between them are tolerably small. This appropriate performance of inverse model also can be observed in Figure 10.

Table 7. Reasonable intervals for product requirements

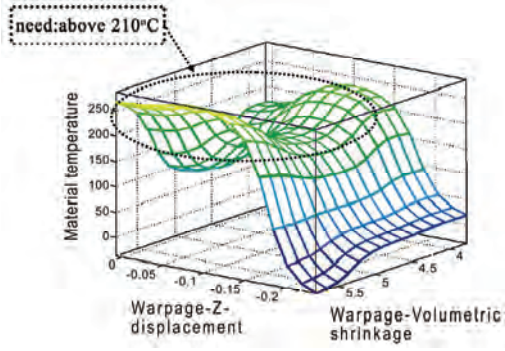
Product required \ factor	B	C	D	F
X(㎜)	0.302~0.65	0.302~0.65	0.302~0.4	0.302~0.5
Y(㎜)	-0.05~0.05	-0.025~0.075	-0.05~0.05	-0.05~0.1
Z(㎜)	-0.15~0.008	-0.175~0.008	-0.1~0.008	-0.15~0.008
V(%)	4	4	4	4

Table 8. Reliability performance of inverse model

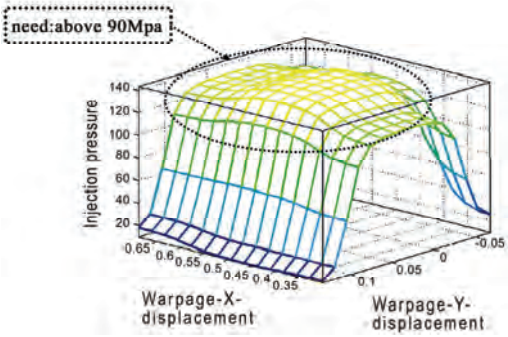
group	Inverse Model								Simulator Moldex3D			
	Input				Output				Output			
	X	Y	Z	V	B	C	D	F	X	Y	Z	V
1	0.302	0.09	0.008	4	220	126	7.26	43.4	0.504	0.012	-0.102	3.851
2	0.302	0.09	-0.07	4	215	108	6.28	57.9	0.346	0.016	-0.028	3.738



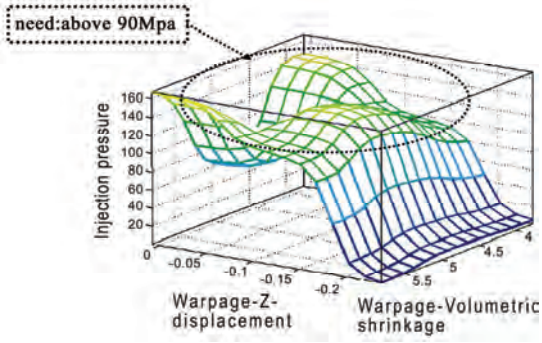
a. warpage-x-displacement and warpagey-displacement to material temperature.



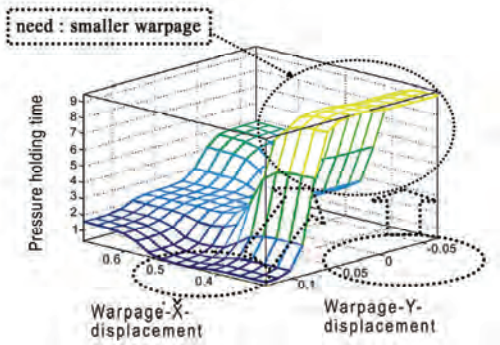
b. warpage-z-displacement and warpage-volumetric-shrinkage to material temperature.



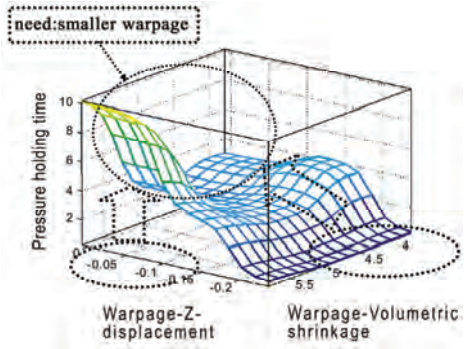
c. warpage-x-displacement and warpage-y-displacement to injection pressure.



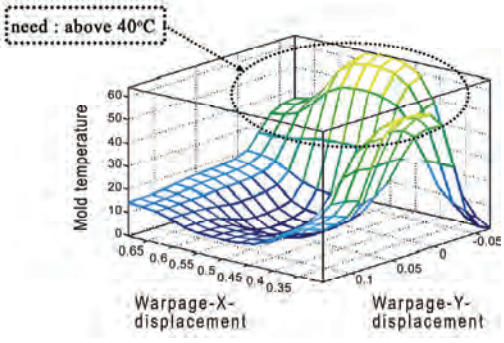
d. warpage-z-displacement and warpage-volumetric-shrinkage to injection pressure.



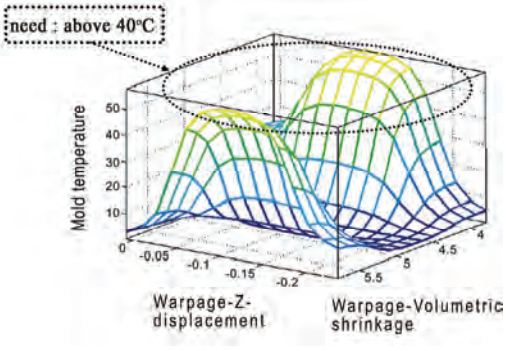
e. warpage-x-displacement and warpage-y-displacement to pressure holding time.



f. warpage-z-displacement and warpage-volumetric-shrinkage to pressure holding time.



g. warpage-x-displacement and warpage-y-displacement to mold temperature.

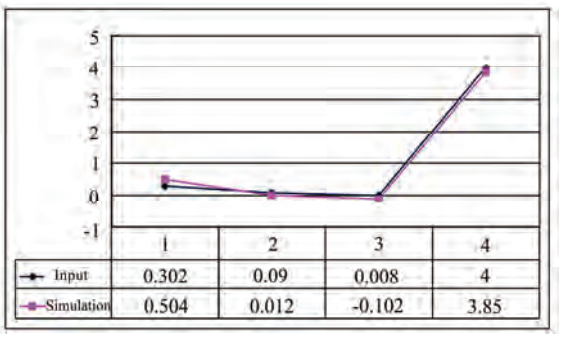


h. warpage-z-displacement and warpage-volumetric-shrinkage to mold temperature.

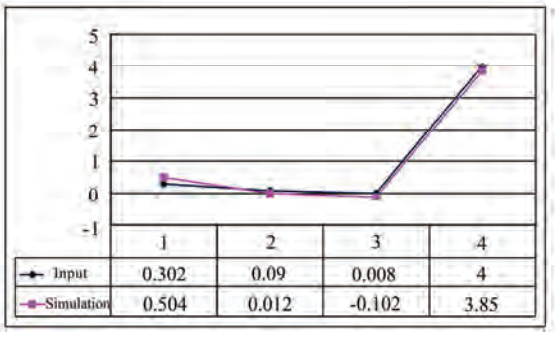
Figure 9. Correlations between two product requirements and one manufacturing parameter in 3D mesh diagrams

Table 9. Optimized comparisons between Taguchi method and proposed method

factor / method	B	C	D	F
Taguchi method	220℃	130Mpa	8s	55℃
Proposal	215℃ ℃	108Mpa	6.28s	57.9℃
requirement / method	X	Y	Z	V
Taguchi method	0.46mm	0.09mm	0.008mm	3.79%
Proposal	0.346	0.016	-0.028	3.74%



a. Group 1



b. Group 2

Figure 10. Reliability performance of inverse model by two groups of data

As mentioned above, although it is easy to find out the optimal manufacturing conditions subjected to single quality required by Taguchi method, in the situation of requiring multiple qualities simultaneously, it is difficult to cope with the problem. Besides, for the changing levels of each controllable factor are ambiguous, it is possible to trap the solution in local optimum. The results are shown in Table 9 just can respond above problem. Where, by examining the manufacturing factors and product required found by Taguchi method and the proposed method, respectively, it can be found that the performance of proposed method is better than that done by Taguchi method.

5. Conclusions

For solving the optimal problem in manufacturing design of injection mold, the method basing on the concept of inverse model is proposed in this paper. Through the method, the optimal manufacturing parameters can be found by using the product required inversely. In addition, the effectiveness and appropriateness of the proposal are confirmed by the numerical studies on the real case. Yet the studied results show that the proposed method not only can improve the insufficiencies of Taguchi method but also offers a more précising and concise approach for the optimization of manufacturing design.

6. Acknowledgments

The authors would like to thank Prof. F.B. Hsiao, who is with the Department of Aeronautics and Astronautics at National Cheng Kung University, for his providing valuable real data for this study.

REFERENCES

[1] S. N. Huang; K. K. Tan and T. H. Lee, “Neural-network-based predictive learning control of ram velocity in injection molding,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 34, Issue 3, pp. 363–368, August 2004.

[2] R. C. Luo and J. H. Tzou, “The development of direct metallic rapid tooling system,” IEEE Transactions on Automation Science and Engineering, Vol. 4, Issue 1, pp. 1–10, January 2007.

[3] R. C. Luo, and Y. L. Pan, , “Rapid manufacturing of intelligent mold with embedded microsensors,” IEEE/ ASME Transactions on Mechatronics, Vol. 12, Issue 2 , pp. 190–197, April 2007.

[4] Y. K. Shen and H. W. Chien, “Optimization of the micro-injection molding process using grey relational analysis and moldflow analysis,” Journal of Reinforced Plastics Composites, Vol. 23, pp. 1799–1814, 2004.

[5] H. Qiao, “A systematic computer-aided approach to cooling system optimal design in plastic injection molding,” International Journal of Mechanical Science, Vol. 48, pp. 430–439, 2006.

[6] J. M. Castro, M. Cabrera-R´ıos, and C. A. Mount-Campbell, “Modelling and simulation in reactive polymer processing,” Modelling and Simulation in Materials Science and Engineering, Vol. 12, pp. S121–S149, 2004.

[7] H. S. Yan, and D. Xu, “An approach to estimating product design time based on fuzzy ν-support vector machine,” IEEE Transactions on Neural Networks, Vol. 18, Issue 3, pp.721–731, May 2007.

[8] B. Ribeiro, “Support vector machines for quality monitoring in a plastic injection molding process,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 35, Issue 3, pp. 401–410, August 2005.

[9] M. Cabrera-R´ıos, J. M. Castro, and C. A. Mount-Campbell, “Multiple quality criteria optimization in in-mold

coating (IMC) with a data envelopment analysis approach," Journal of Polymer Engineering, Vol. 22, No. 5, pp. 305–340, 2002.

[10] M. Cabrera-R´ıos, J. M. Castro, and C. A. Mount-Campbell, "Multiple quality criteria optimization in reactive in-mold coating with a data envelopment analysis approach: II. A case with more than three performance measures," Journal of Polymer Engineering, Vol. 24, No. 4, pp. 435–450, 2004.

[11] C. E. Castro, M. Cabrera-R´ıos, B. Lilly, J. M. Castro, and C. A. Mount-Campbell, "Identifying the best compromises between multiple performance measures in injection molding (IM) using data envelopment analysis (DEA)," Journal of Integrated Design and Process Science, Vol. 7, No. 1, pp. 78–86, 2003.

[12] D. E. Smith, D. A. Tortorelli, and C. L. Tucker, "Analysis and sensitivity analysis for polymer injection and compression molding," Computer Methods in Applied Mechanics and Engineering, Vol. 167, pp. 325–344, 1998.

[13] A. Gokce, K. T. Hsiao, and S. G. Advani, "Branch and bound search optimization injection gate locations in liquid composite molding processes," Composites A, Vol. 33, pp. 1263–1272, 2002.

[14] D. E. Smith, "Design sensitivity analysis and optimization for polymer sheet extrusion and mold filling processes," International Journal for Numerical Methods in Engineering, Vol. 57, pp. 1381–1411, 2003.

[15] J. K. L. Ho, K. F. Chu, and C. K. Mok, "Minimizing manufacturing costs for thin injection molded plastic components," International Journal of Advanced Manufacturing Technology, Vol. 26, pp. 517–526, 2005.

[16] N. R. Subramanian, L. Tingyu, and Y. A. Seng, "Optimizing warpage analysis for an optical housing," Mechatronics, Vol. 15, pp. 111–127, 2005.

[17] G. Courbebaisse and D. Garcia, "Shape analysis and injection molding optimization," Computational Materials Science., Vol. 25, pp. 547–553, 2002.

[18] K. Alam and M. R. Kamal, "A robust optimization of injection molding runner balancing," Computers and Chemical Engineering, Vol. 29, pp. 1934–1944, 2005.

[19] S. Dowlatshahi, "An application of design of experiments for optimization of plastic injection molding processes," Journal of Manufacturing Technology Management, Vol. 15, pp. 445–454, 2004.

[20] S. J. Liu and Y. S. Chen, "The manufacturing of thermoplastic composite parts by water-assisted injection-molding technology," Composites A, Vol. 35, pp. 171–180, 2004.(water-assisted)

[21] C. P. Fung, "Multi-response optimization of impact performances in fiber-reinforced poly (butylene terephthalate)," Journal of Thermoplastic Composite Materials, Vol. 19, pp. 191–205, 2006. (fiber)

[22] C. P. Fung and P. C. Kang, "Multi-response optimization in friction properties of PBT composites using Taguchi method and principal component analysis," Journal of Materials Processing Technology., Vol. 170, pp. 602–610, 2005.(**T**)

[23] C. H. Wu and Y. L. Su, "Optimization of wedge-shaped parts for injection molding and injection compression molding," International Communications in Heat and Mass Transfer, Vol. 30, pp. 215–224, 2003.

[24] S. J. Liu, C. H. Hsu, and C. Y. Chang, "Parametric characterization of the thin-wall injection molding of thermoplastic composites," Journal of Reinforced Plastics Composites, Vol. 21, pp. 1027–1041, 2002.

[25] H. P. Heim, "The statistical regression calculation in plastics processing—Process analysis, optimization and monitoring," Macromolecular Materials and Engineering, Vol. 287, pp. 773–783, 2002. (R)

[26] H. Kurtaran and T. Erzurumlu, "Efficient warpage optimization of thin shell plastic parts using response surface methodology and genetic algorithm," International Journal of Advanced Manufacturing Technology, Vol. 27, pp. 468–472, 2006.

[27] B. Ozcelik and T. Erzurumlu, "Comparison of the warpage optimization in the plastic injection molding using ANOVA, neural network model and genetic algorithm," Journal of Materials Processing Technology, Vol. 171, pp. 437–445, 2006.(N)

[28] C. P. Fung, "The study on the optimization of injection molding process parameters with gray relational analysis," Journal of Reinforced Plastics Composites, Vol. 22, pp. 51–66, 2003.(G)

[29] S. H. Chang, J. R. Hwang, and J. L. Doong, "Optimization of the injection molding process of short glass fiber reinforced polycarbonate composites using grey relational analysis," Journal of Materials Processing Technology, Vol. 97, pp. 186–193, 2000.(G glass fiber)

[30] G. A. Vagelatos, G. G. Regatos, and S. G. Tzafestas, "Incremental fuzzy supervisory controller design for optimizing the injection molding process," Expert Systems with Applications, Vol. 20, pp. 207–216, 2001.(F)

[31] M. S. Phadke, "Quality Engineering Using Robust Design," Englewood Cliffs, NJ: Prentice-Hall, 1989.

[32] D. C. Montgomery, "Design and Analysis of Experiments," New York: Wiley, 1991.

[33] S. H. Park, Robust Design and Analysis for Quality Engineering. London, U.K.: Chapman & Hall, 1996.

[34] R. Jag. , "Neuro-fuzzy modeling architectures, analysis and applications," PhD Thesis University of California, Bcrkley, July 1992.

[35] Zhang, J.; Shu-Hung Chung, H.; Wai-Lun Lo, "Chaotic time series prediction using a neuro-fuzzy system with time-delay coordinates," IEEE Transactions on HKnowledge and Data EngineeringH, Vo. 20, HIssue 7H, pp. 956–964, July 2008.

[36] Hinojosa, J.; Domenech-Asensi, G., "Multiple adaptive neuro-fuzzy inference systems for accurate microwave CAD applications," HCircuit Theory and Design, ECCTD'07. 18th European Conference onH, pp.767–770, August 27th–30th, 2007.

[37] G. Domenech-Asensi, J. Hinojosa, R. Ruiz, J.A, Diaz-Madrid, "Accurate and reusable macromodeling technique using a fuzzy-logic approach," IEEE International Symposium on Circuits and Systems, ISCAS'08, pp.508–511, May 18th–21st, 2008.

Journal of Intelligent Learning Systems and Applications, **2010, 2: 36-42**
doi:10.4236/jilsa.2010.21005 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Synthesis of Nonlinear Control of Switching Topologies of Buck-Boost Converter Using Fuzzy Logic on Field Programmable Gate Array (FPGA)

Johnson A. Asumadu, Vaidhyanathan Jagannathan, Arkhom Chachavalnanont

Electrical and Computer Engineering, Western Michigan University, Kalamazoo, USA.
Email: johnson.asumadu@wmich.edu

Received September 25th, 2009; accepted January 11th, 2010.

ABSTRACT

An intelligent fuzzy logic inference pipeline for the control of a dc-dc buck-boost converter was designed and built using a semi-custom VLSI chip. The fuzzy linguistics describing the switching topologies of the converter was mapped into a look-up table that was synthesized into a set of Boolean equations. A VLSI chip–a field programmable gate array (FPGA) was used to implement the Boolean equations. Features include the size of RAM chip independent of number of rules in the knowledge base, on-chip fuzzification and defuzzification, faster response with speeds over giga fuzzy logic inferences per sec (FLIPS), and an inexpensive VLSI chip. The key application areas are: 1) *on-chip integrated controllers; and* 2) *on-chip co-integration for entire system of sensors, circuits, controllers, and detectors for building complete instrument systems.*

***Keywords*:** *Multi-Fuzzy Logic Controller* (*MFLC*), *Field Programmable Gate Array* (*FPGA*), *Buck-Boost Converter, Boolean Look-Up Table, Co-Integration*

1. Introduction

The design of control laws for power converters and inverters has been based mainly on linear control theory. However, most power electronics switching topologies have variable structures which are non-linear, characterized by discontinuities, and are therefore more difficult to model. The three most popular control methods used are state-space averaging technique [1], variable structure control (VSC) [2], and sliding mode control [3]. The state-space technique may lead to instability, the VSC control has hysteresis as drawback, and the sliding-mode control is very complicated. The implementation of the above-mentioned control laws are complicated, high cost and almost not practical.

Fuzzy logic controllers (FLCs) are very suitable for variable structures but current applications of FLCs use software for implementation. However, hardware implementation will be cheaper and faster. The proposed method combines hybrid linguistic models used in FLCs and computational paradigms of Boolean algebra. The knowledge base of the converter switching topologies is used to form a look-up table which is described by Boolean equations which are easily implemented using FPGA. Results showed that the size of the FPGA is independent of the number inference rules in the knowledge base, speeds in giga FLIPS are achieved because it is hardware based, and very fast control response.

2. Proposed Method

2.1 Single-Input-Single-Output (SISO) Fuzzy Controller

Consider the generalized buck-boost converter shown in Figure 1. This is a single-input-single-output (SISO) system. The input variable is the capacitor voltage (v_C) and the output variable is the duty ratio D. It can be shown that the average model has the following equations:

$$\frac{di_L(t)}{dt} = \frac{(1-D)v_C(t)}{L} + \frac{DE}{L} \quad (1)$$

$$\frac{dv_C(t)}{dt} = \frac{(1-D)i_L(t)}{C} - \frac{v_C(t)}{CR} \quad (2)$$

The membership functions for v_C and D are shown in Figure 2. The SISO system of Figure 1 has a single fuzzy input V (v_C) and a single fuzzy output D.

The fuzzy membership sets V and D are represented by five linguistic qualifiers L (low), ML (medium low), OK (okay), MH (medium high), and H (high). Hence there

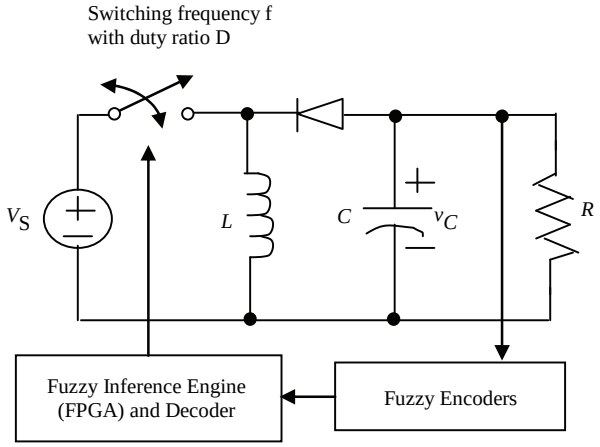


Figure 1. Buck-boost switching converter with a fuzzy controller

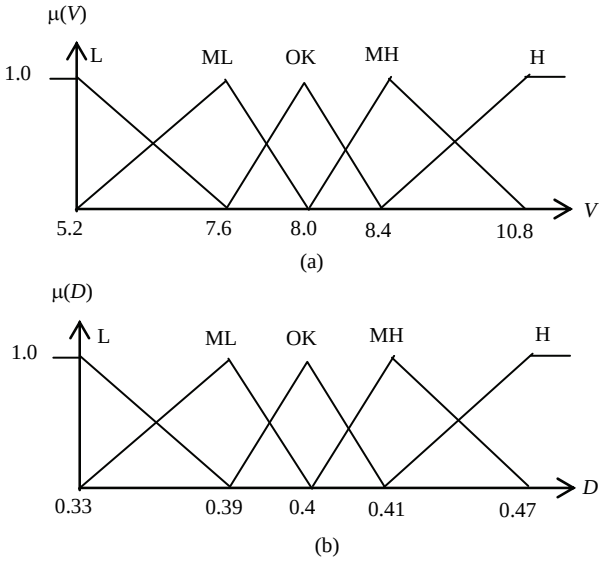


Figure 2. Membership functions for (a) voltage and (b) duty ratio

are 5x5=25 possible combinations that can generate 25 possible rules. However, the following 5 rules are sufficient to describe the membership values of Figure 2.

RULE 1: IF *V* is *H*, THEN *D* is *L*
RULE 2: IF *V* is *MH*, THEN *D* is *ML*
RULE 3: IF *V* is *OK*, THEN *D* is *OK* (3)
RULE 4: IF *V* is *ML*, THEN *D* is *MH*
RULE 5: IF *V* is *L*, THEN *D* is *H*

2.2 Logic Synthesis of Fuzzy Controllers

Consider a SISO fuzzy controller with input universe of discourse A^1 and output universe of discourse B^1. Manzoul [4] showed that the number of unique fuzzy inputs (ufi) is equal to the dimension of the input universe of discourse. That is:

$$\text{ufi}=\dim[A^1]=q_1 \quad (4)$$

In the fuzzification process, each input is mapped into only one value (one element) in the input universe of discourse and zero value to all other values (elements). The objective here is to use a look-up table. Therefore, the number of rows in the look-up table is equal q_1. It was also shown in [4] that the total least integer number of binary bits *X* for all the inputs is given by

$$X=\log_2\ q_1 \quad (5)$$

Each fuzzy input has only one output value and therefore the maximum number of distinct output values is also equal to q_1. In the defuzzification process only one element is selected in the output universe of discourse. It is therefore, easier and economical to use the defuzzified

outputs in the look-up table. If the there are p elements ($p \leq q_1$) in the output universe of discourse, the dimension of the output universe of discourse is given as

$$\dim[B^1]=p \quad (6)$$

The total least integer number of binary bits Y for all the outputs is given by

$$Y=\log_2 p \quad (7)$$

3. Buck-Boost Converter Fuzzy Controller

3.1 Fuzzy Controller

Consider a SISO controller to control the output voltage of the buck-boost converter of Figure 1. The input variable is the voltage and output variable is the duty ratio. The duty ratio maintains the voltage at a selected value. The ranges of capacitor voltage (v_C) and duty ratio D are (5.2 to 10.8) V and (0.33 to 0.47) respectively. The membership values are in the interval [0, 1], where 0 denotes no membership and 1 denotes full membership. Assume that

$$\text{im}[V^1]=\dim[D^1]=29 \quad (8)$$

From (5) the least number of binary bits to represent the input values is given as

$$X=\log_2 \quad 29=5$$

Similarly, using (6) the least number of binary bits to represent output values is given as

$$Y=\log_2 \quad 29=5$$

In Appendix I the 5 rules of (3) are expressed numerically. The fuzzy relation obtained is too big to be shown here because of the size (29x29 matrix). Table 1 of Appendix I shows the summary of the complete computations of the controller.

3.2 Look-Up Table

The look-up table representing the fuzzy controller is given in Table 2. The input universe of discourse has dimension of 29 and the output universe has dimension of 29. The look-up can be described by 7-variable Boolean equations. That is,

$$\begin{aligned}
f0&=\sum(5,6,8,9,10,12,16,17,1923,26,27,28)\\
f1&=\sum(2,4,5,7,8,10,11,14,17,18,21,24,27)\\
f2&=\sum(2,3,5,6,8,9,14,15,16,21,22,24,25,27,28,29)\\
f3&=\sum(1,5,6,7,14,16,17,18,19,20,24,25,26) \quad (9)\\
f4&=\sum(1,2,3,4,14,24,25,26,27,28,29)\\
f5&=\sum(1,2,3,4,5,6,7,8,9,10,11,12,13,16,17,18,19,20,21,22,23)\\
f6&=\sum(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)
\end{aligned}$$

Table 1. Summary of the controller computations

Input Voltage	Fuzzified Input	Fuzzy Output	Output Duty Ratio (X)
0.0 - 5.20	1.0, 0.0, 0.0, ., ., ., 0.0	*0,0,.08,.16,.25,.33,.41,.5,.58,.66,.75,.83,.91,1.0	120
5.21 - 5.40	0.0, 1.0, 0.0, ., ., ., 0.0	*,.08,.08,.08,.16,.25,.33,.41,.5,.58,.66,.75,.83,.91,.91	118
5.41 - 5.60	0.0, ., ., 1.0, ., ., ., 0.0	*,.16,.16,.16,.16,.25,.33,.41,.5,.58,.66,.75,.83,.83,.83	116
5.61 - 5.80	0.0, ., ., 1.0, ., ., ., 0.0	*,.25,.25, ,.25, ,.25, ,.25,.33,.41,.58,.66,.75,.75,.75,.75	114
5.81 - 6.00	0.0, ., ., 1.0, ., ., ., 0.0	*,.33,.33,.33,.33,.33,.33,.41,.5,.58,.66,.66,.66,.66,.66	111
6.01 - 6.20	0.0, ., ., 1.0, ., ., ., 0.0	*,.41,.41,.41,.41,.41,.41,.41,.5,.58,.58,.58,.58,.58,.58	109
6.21 - 6.40	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5	106
6.41 - 6.60	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,.58,.58,.58,.58,.58,.58,.5,.41,.41,.41,.41,.41,.41,.41	103
6.61 - 6.80	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,.66,.66,.66,.66,.66,.58,.5,.41,.33,.33,.33,.33,.33	101
6.81 - 7.00	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,.75,.75,.75,.75,.66,.58,.5,.41,.33,.25, ,.25, ,.25, ,.25	99
7.01 - 7.20	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,.83,.83,.83,.75,.66,.58,.5,.41,.33,.25,.16,.16,.16	98
7.21 - 7.40	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,.91,.91,.83,.75,.66,.58,.5,.41,.33,.25,.16,.08,.08	97
7.41 - 7.60	0.0, ., ., 1.0, ., ., ., 0.0	*,.5,1.0,.91,.83,.75,.66,.58,.5,.41,.33,.25,.16,.08,.0	96
7.61 - 7.80	0.0, ., ., 1.0, ., ., ., 0.0	**,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.41,.33,.25,.16,.08,0.0	94
7.81 - 8.00	0.0, ., ., 1.0, ., ., ., 0.0	**,.5,1.0,.5,**	70
8.01 - 8.20	0.0, ., ., 1.0, ., ., ., 0.0	0,.08,.16,.25,.33,.41,.5,.5,.5,.5,.5,.5,.5,.5,.5,**	45
8.21 - 8.40	0.0, ., ., 1.0, ., ., ., 0.0	0,.08,.16,.25,.33,.41,.5,.58,.66,.75,.83,.91,1.0,.5,*	43
8.41 - 8.60	0.0, ., ., 1.0, ., ., ., 0.0	.08,.08,.16,.25,.33,.41,.5,.58,.66,.75,.83,.91,.91,.5,*	42
8.61 - 8.80	0.0, ., ., 1.0, ., ., ., 0.0	.16,.16,.16,.25,.33,.41,.58,.66,.75,.75,.75,.75,.5,*	41
8.81 - 9.00	0.0, ., ., 1.0, ., ., ., 0.0	.25, ,.25, ,.25, ,.25,.33,.41,.58,.66,.75,.75,.75,.75,.5,*	40
9.01 - 9.20	0.0, ., ., 1.0, ., ., ., 0.0	.33,.33,.33,.33,.33,.41,.5,.58,.66,.66,.66,.66,.66,.5,*	38
9.21 - 9.40	0.0, ., ., 1.0, ., ., ., 0.0	.41,.41,.41,.41,.41,.41,.5,.58,.58,.58,.58,.58,.58,.5,*	36
9.41 - 9.60	0.0, ., ., 1.0, ., ., ., 0.0	.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,.5,*	33
9.61 - 9.80	0.0, ., ., 1.0, ., ., ., 0.0	.58,.58,.58,.58,.58,.58,.5,.41,.41,.41,.41,.41,.41,.41,*	30
9.81 - 10.00	0.0, ., ., 1.0, ., ., ., 0.0	.66,.66,.66,.66,.66,.58,.5,.41,.33,.33,.33,.33,.33,.33,*	28
10.01 - 10.20	0.0, ., ., 1.0, ., ., ., 0.0	.75,.75,.75,.75,.66,.58,.5,.41,.33,.25, ,.25, ,.25, ,.25,.25,*	25
10.21 - 10.40	0.0, ., ., 1.0, 0.0, 0.0	.83,.83,.83,.75,.66,.58,.5,.41,.33,.25,.16,.16,.16,.16,*	23
10.41 - 10.60	0.0, ., ., 0.0, 1.0, 0.0	.91,.91,.83,.75,.66,.58,.5,.41,.33,.25,.16,.08,.08,.08,*	21
10.61 - 10.8	0.0, ., ., 0.0, 0.0, 1.0	1.0,.91,.83,.75,.66,.58,.5,.41,.33,.25,.16,.08,.0,.0,*	20

NOTE: *=0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 **=0,0,0,0,0,0,0,0,0,0,0,0,0,0

Duty ratio D=-0.0011051*X+0.4

Table 2. Look-up table for fuzzy controller

	INPUT						OUTPUT						
#	BINARY					#	BINARY						
	y_4	y_3	y_2	y_1	y_0		f_6	f_5	f_4	f_3	f_2	f_1	f_0
1	0	0	0	0	1	120	1	1	1	1	0	0	0
2	0	0	0	1	0	118	1	1	1	0	1	1	0
3	0	0	0	1	1	116	1	1	1	0	1	0	0
4	0	0	1	0	0	114	1	1	1	0	0	1	0
5	0	0	1	0	1	111	1	1	0	1	1	1	1
6	0	0	1	1	0	109	1	1	0	1	1	0	1
7	0	0	1	1	1	106	1	1	0	1	0	1	0
8	0	1	0	0	0	103	1	1	0	0	1	1	1
9	0	1	0	0	1	101	1	1	0	0	1	0	1
10	0	1	0	1	0	99	1	1	0	0	0	1	1
11	0	1	0	1	1	98	1	1	0	0	0	1	0
12	0	1	1	0	0	97	1	1	0	0	0	0	1
13	0	1	1	0	1	96	1	1	0	0	0	0	0
14	0	1	1	1	0	94	1	0	1	1	1	1	0
15	0	1	1	1	1	70	1	0	0	0	1	1	0
16	1	0	0	0	0	45	0	1	0	1	1	0	1
17	1	0	0	0	1	43	0	1	0	1	0	1	1
18	1	0	0	1	0	42	0	1	0	1	0	1	0
19	1	0	0	1	1	41	0	1	0	1	0	0	1
20	1	0	1	0	0	40	0	1	0	1	0	0	0
21	1	0	1	0	1	38	0	1	0	0	1	1	0
22	1	0	1	1	0	36	0	1	0	0	1	0	0
23	1	0	1	1	1	33	0	1	0	0	0	0	1
24	1	1	0	0	0	30	0	0	1	1	1	1	0
25	1	1	0	0	1	28	0	0	1	1	1	0	0
26	1	1	0	1	0	25	0	0	1	1	0	0	1
27	1	1	0	1	1	23	0	0	1	0	1	1	1
28	1	1	1	0	0	21	0	0	1	0	1	0	1
29	1	1	1	0	1	20	0	0	1	0	1	0	0

3.3 Control Instrument Implementation

3.3.1 FPGA Implementation

The FPGA configuration is implemented by using Xilinx's X95 (XC9536-10-PC44 CPLD) and XS40 (XC40 05XL FPGA) boards. Both boards come with 8051 microcontroller with 12 MHz speed. The Xilinx's development system is used to implement the Boolean equations, which also represents the fuzzy controller. Figure 3 shows the semi-custom VLSI chip of the FPGA functional block diagram.

3.3.2 Instrument

Figure 4 shows the diagram of the fuzzy logic controller chip pin outs and output waveforms. The binary inputs of the controller y_0, y_1, . . ., y_4 are the encoded fuzzified crisp output voltage v_C of the dc-dc converter. The binary outputs of the controller f_0, f_1, . . ., f_6 are defuzzified to provide the variable duty ratio D.

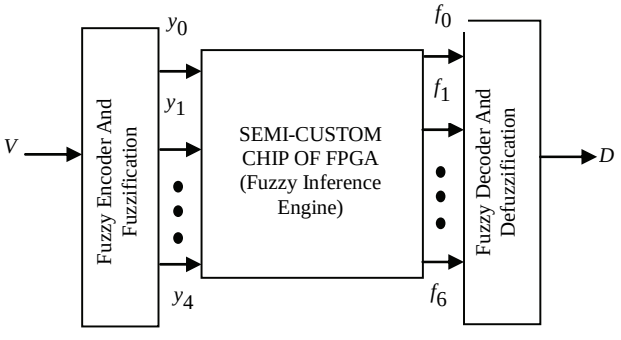
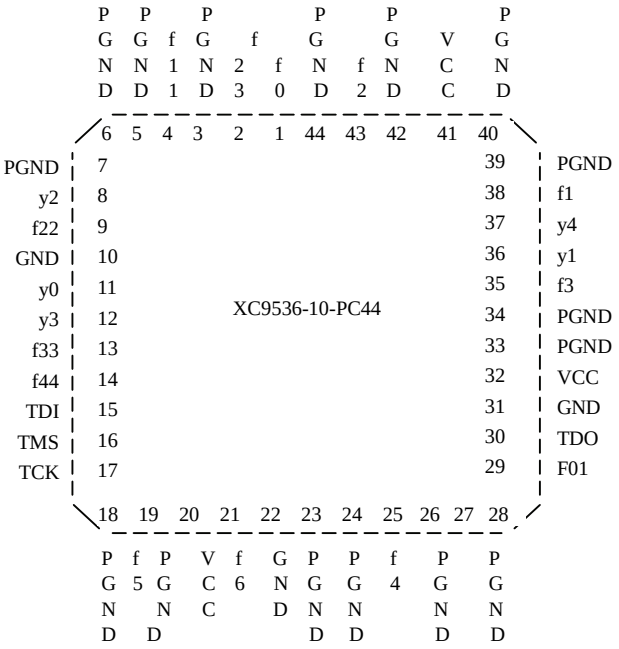


Figure 3. Semi-custom VLSI fuzzy controller chip

3.3.3 On-Chip Cointegration

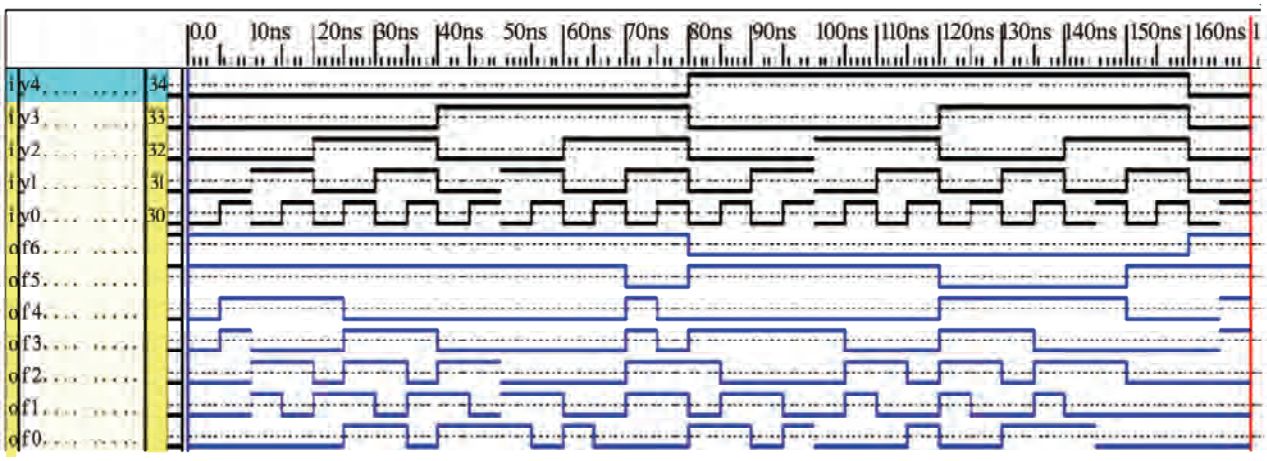
Microsystems technology has made inroads in instrumentation and measurement (I&M) applications. The fuzzy logic FPGA controller chip can be fabricated to have compact size. The FLC chip of this paper provides an interesting fabrication technique that has been identified in I&M applications: cointergration of control/sensors/detectors and circuits for building complete instrument systems.

The on-chip controller mentioned in the paper uses a compact FPGA chip, which can be fabricated as part of an integrated system of sensors, circuits, controllers, and detectors squeezed onto nanometer wafer. Such microchips can be used in a variety of applications in different environments and requirements. Besides mass volume production of these microchips, there are other advantages such as cheaper parts and assembly of instruments, better functionality, lower mass and size, speed of control, lower cost, and higher efficiency. Some companies are now using multi fuzzy logic controller-based (MFLC) chips in some of their new dryers, dishwashers, and washing machines.



Legend : PGND = Tie pin to GND for additional ground path or leave unconnected
VCC = Dedicated Power Pin
GND = Dedicated Ground Pin
TDI = Test Data In, JTAG pin
TDO = Test Data Out, JTAG pin
TCK = Test Clock, JTAG pin
TMS = Test Mode Select, JTAG pin
PROHIBITED = User reserved pin

(a)



(b)

Figure 4. (a) FPGA pin out; (b) input and output waveforms of FLC chip

4. Results

The output voltage of Figure 1 is regulated at about 8.0 V through a feedback loop using the fuzzy controller on FPGA chip for adjusting the duty ratio *D*. The input voltage V_s=12 V and the circuit parameters are *L*=100 μH, *C*=330 μF. The steady-state voltage is about 8.0 V when the load R is changed from 10 Ω to 5 Ω. Figure 5 shows the test results of a fuzzy controller chip.

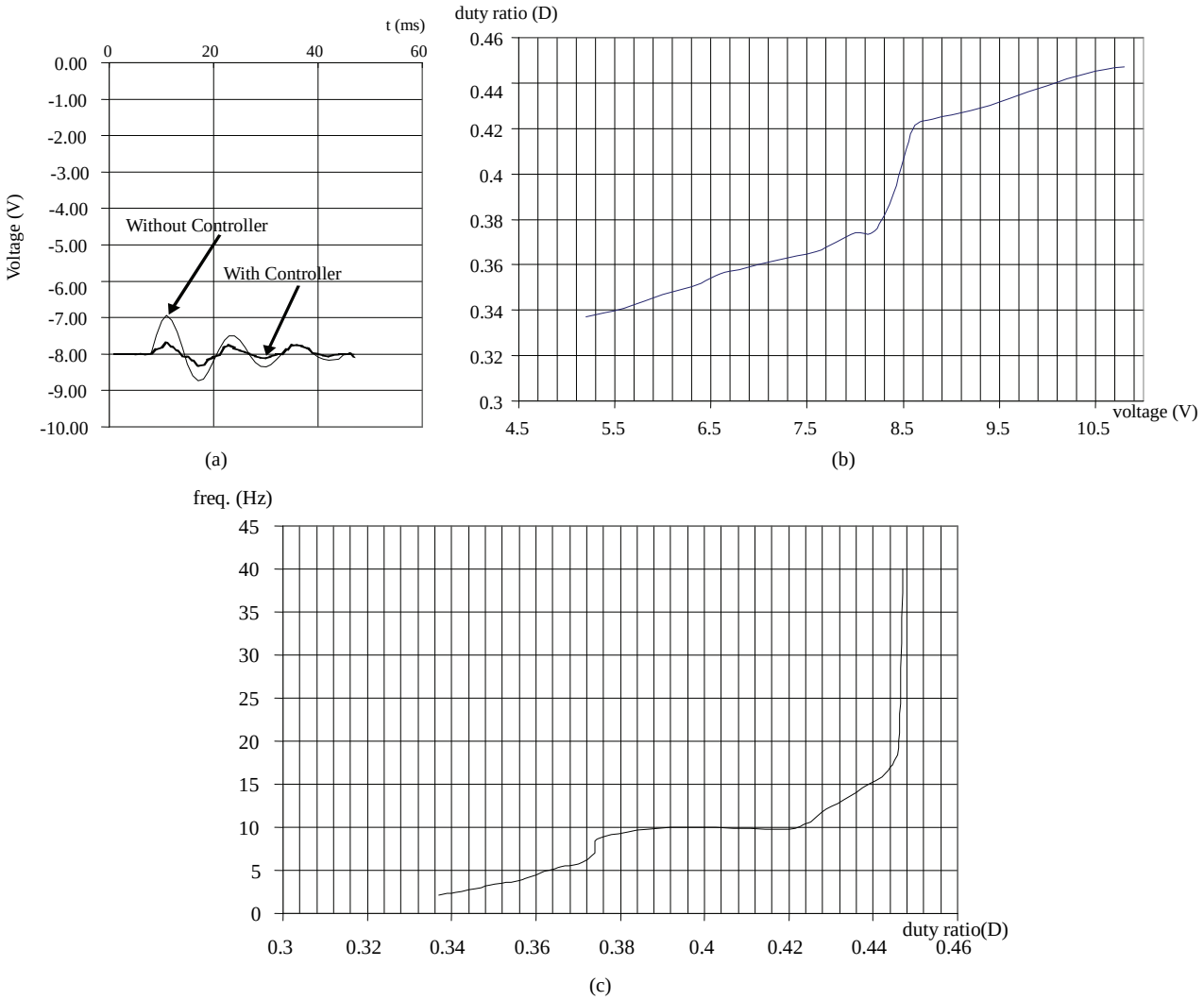


Figure 5. (a) Buck-boost converter results; (b) graph between voltage & duty; (c) relationship of frequency & duty ratio

Figure 5(a) shows the effect on the output voltage when changing the load resistance. Figure 5 shows that the output voltage $v_c(t)$ can respond instantaneously to changes in the duty ratio D. The hardware implementation is cheaper and faster than using software.

5. Conclusions

The fuzzy controller implemented on an FPGA was used to control a dc-dc buck-boost switching converter. The size of the semi-custom VLSI was independent of the number of rules in the knowledge base and therefore no memory space was required to store the large of rules. The important aspect of the controller is that it was hardware-based and consequently speeds over giga FLIPS can be achieved. The on-chip controller is a compact FPGA which may be integrated with sensors and circuits to provide a complete co-integration system.

REFERENCES

[1] R. Erickson, S. Cuk, and R. D. Middlebrook, "Large-signal modeling and analysis of switching regulators," IEEE PESC, pp. 240–250, 1982.

[2] W. Gao and J. C. Hung, "Variable structure control of non-linear systems," IEEE Transactions on Industrial Electronics, Vol. 40, No. 1, pp. 45–55, February 1993.

[3] H. Sira-Ramirez and M. Rios Bolivar, "Sliding-mode control of DC-DC power converters via extended-linearization," IEEE Transactions on Circuits and Systems I, Vol. 41, No. 10, 1994.

[4] M. A. Manzoul, "Fuzzy controllers on semi-custom VLSI chips," Fuzzy Control Systems, CRC Press, Inc., Boca Raton, Florida, pp. 551–560, 1994.

APPENDIX I

The 5 Rules of Equation 3 Expressed Numerically

RULE 1:
IF
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.08,0.16,0.25,0.33,0.41
,0.5,0.58,0.66,0.75,0.83,0.91,1.0]
THEN
[1.0,0.91,0.83,0.75,0.66,0.58,0.5,0.41,0.33,0.25,
0.16,0.08, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

RULE 2:
IF
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.5,1.0,0.91,0.83,0.75,0.66,
0.58,0.5,0.41,0.33,0.25,0.16,0.08,0]
THEN
[0.08,0.16,0.25,0.33,0.41,0.5,0.58,0.66,0.75,0.83,
0.91,1.0,0.5, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

RULE 3:
IF
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.5,1.0,0.50,0,0,0,0,0,0,0,0,0,0,0,0,0]
THEN
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.5,1.0,0.50,0,0,0,0,0,0,
0,0,0,0,0,0]

RULE 4:
IF [0.08,0.16,0.25,0.33,0.41,0.5,0.58,0.66,0.75,0.83,
0.91,1.0,0.5, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
THEN [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.5,1.0,0.91,0.83,
0.75,0.66,0.58,0.5,0.41,0.33,0.25,0.16,0.08,0]

RULE 5:
IF [1.0,0.91,0.83,0.75,0.66,0.58,0.5,0.41,0.33,0.25,
0.16,0.08, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
THEN [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.08,0.16,0.25,
0.33,0.41,0.5,0.58,0.66,0.75,0.83,0.91,1.0]

Journal of Intelligent Learning Systems and Applications, **2010, 2: 43-48**
doi:10.4236/jemaa.2010.21006 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Parameters Estimation of an Electric Fan Using ANN

Himanshu Vijay, D. K. Chaturvedi

Department of Electrical Engineering, Dayalbagh Educational Institute, Deemed University, Agra, India.
Email: dkc.foe@gmail.com

Received November 10th, 2009; accepted January 8th, 2010.

ABSTRACT

Electric Fans are very commonly used in the industries, domestic applications and in tunnels for cooling and ventilation purposes. Fan parameters estimation is an important task as far as the reliable operation of a fan system is concerned. Basically, a fan is mainly consisting of a single phase induction motor and therefore fan system parameters are essentially the electrical parameters e.g. resistances, reactances and some load parameters (*fan blades*).*These parameters often change under varying operating conditions and the knowledge of these parameters is necessary to have optimum and efficient operation of the system. Therefore, fan system parameters are required to be estimated. Further, fan system parameters estimation is required to ensure the smooth system operation and to avoid any malfunctioning of the system during abnormal working conditions. In this paper, Artificial Neural Networks* (*ANN*) *approach has been used for parameter estimation of a fan system. The simulated and experimental results are compared.*

Keywords*: *Artificial Neural Networks, Fan System, Mathematical Modeling, Parameters Estimation

1. Introduction

A fan system is basically meant for getting air to people occupying a building, office, residential complex, shops or public places etc. and therefore directly impacting the human comfort. Fan circulates the air and provides the pressure required to push it. For many applications like shop ventilation, material handling, boiler usage etc., fans become crucial for process support and human health. In the manufacturing sector, fans use about 78.7 billion kWh of energy every year. This consumption is approximately 15% of the electricity used by motors [1–3].

In manufacturing, fan reliability is critical to plant operation. For example, where fans serve material handling applications, fan failure will immediately create a process stoppage. In industrial ventilation applications, fan failure will often force a process to be shut down. Even in heating and cooling applications, fan operation is essential to maintain a productive work environment. Fan failure leads to conditions in which worker productivity and product quality declines. This is especially true for some production applications e.g. electronic component manufacturing and plastics injection molding.

There are mainly two types of fans namely centrifugal and axial [14]. These types are characterized by the path of the airflow through the fan. Centrifugal fans use a rotating impeller to increase the velocity of an air stream to gain kinetic energy. Centrifugal fans are capable of generating relatively high pressures. These fans are generally used in “dirty” airstreams (high moisture and particulate content), in material handling applications, and in systems at higher temperatures.

Axial fans move an air stream along the axis of the fan. The air is pressurized by the aerodynamic lift generated by the fan blades. Axial fans are commonly used in “clean air,” low-pressure, high-volume applications.

The components of a fan system must function well in order to ensure efficient operation. The cost-effective operation and maintenance of a fan system requires attention not only to the needs of the individual pieces of equipment, but also to the system as a whole. In this concern, fan system parameters estimation plays a prominent role which addresses the following issues:

1) Establishing current conditions and operating parameters.

2) Assessing energy consumption with respect to performance.

3) Continuing to monitor and optimize the system.

4) Continuing to operate and maintain the system for peak performance.

In this paper the method used for determining the fan

parameters is based on on-line methods with the application of artificial neural network algorithm. For the purposed of simulating the fan system and the test conditions, the software Matlab version 7.1 and LabView version 8.0 have been used.

2. Mathematical Modeling of Fan System

Basically, a fan is a single phase induction motor having stator, rotor working on the principle of electromagnetic induction. Stator having two winding and a capacitor to make it self starting. Rotor rotates with fan body. Hence a fan system mainly consists of a single phase induction motor, two to six blades usually made of wood, metal, or plastic; which mount under, on top of, or on the side of the motor. The majority of fans have either four or five blades, while most industrial fans have three. Metal arms, called blade irons (alternately blade brackets, blade arms, blade holders, or flanges), which connect the blades to the motor. Here, we are mainly concerned with the modeling of single phase induction motor which is the main component of a fan system.

The induction motor has only one stator winding (main winding) and operates with a single-phase power supply. In all single-phase induction motors, the rotor is the squirrel cage type [2]. Equivalent Circuit of a Single-Phase Induction Motor is shown in the Figure 1.

There is only a single mmf established by the excited stator coil and, thus, only a single pulsating flux exists. However, one-half of the mmf, hence one-half of the turns, are associated with each of the forward and backward mmf components. A set of slips can be defined for both the forward-revolving and backward-revolving fields as

$$s_f = \frac{\omega_{sf} - \omega_m}{\omega_{sf}} \tag{1}$$

$$s_b = \frac{\omega_{sb} - \omega_m}{\omega_{sb}} \tag{2}$$

Equations (1) and (2) can be solved for ω_m and the resulting expressions equated to yield

$$s_b = 2 - s_f \tag{3}$$

The developed torque can be calculated directly from the equivalent circuit as the power delivered to the energy conversion resistance divided by mechanical speed giving

$$T_d = T_{df} + T_{db}$$

$$T_d = \frac{\frac{1}{2} I_f^2 R_r \frac{(1-s_f)}{s_f}}{\omega_m} - \frac{\frac{1}{2} I_b^2 R_r \frac{(1-s_f)}{(2-s_f)}}{\omega_m} \tag{4}$$

The first term on the right-hand side of Equation (4) is the torque (T_{df}) produced by the forward-revolving field while the second term is the torque (T_{db}) resulting from the backward-revolving field. The developed torque can alternately be found as the sum of the power across the air gap divided by the associated synchronous speed.

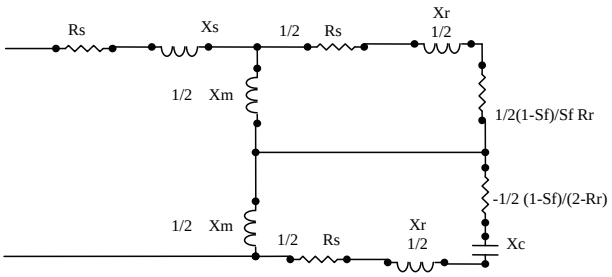


Figure 1. Single-phase induction motor equivalent circuits

$$T_d = T_{df} + T_{db}$$

$$T_d = \frac{\frac{1}{2} I_f^2 R_r \frac{1}{s_f}}{\omega_{sf}} + \frac{\frac{1}{2} I_b^2 R_r \frac{1}{(2-s_f)}}{\omega_{sb}} \tag{5}$$

For both cases the second term (T_{db}) is a negative quantity reflecting the fact that the backward-revolving field results in a torque that acts against the direction of rotation.

The impedance of the forward running rotor is

$$Z_f = j\frac{X_m}{2} \left\| \left(\frac{R_r}{2s_f} + \frac{jX_r}{2} \right) \right. \tag{6}$$

The impedance of the backward running rotor is

$$Z_b = j\frac{X_m}{2} \left\| \left(\frac{R_r}{2(2-s_f)} + \frac{jX_r}{2} \right) \right. \tag{7}$$

$$Z_s = R_s + jX_s \tag{8}$$

$$Z_{in} = Z_s + Z_f + Z_b \tag{9}$$

$$I_1 = \frac{V_1}{Z_{in}} \tag{1 0}$$

By current division, the forward Current is

$$I_f = \frac{\frac{1}{2} jX_m}{Z_f + jX_m} I_1 \tag{11}$$

And backward current is

$$I_b = \frac{\frac{1}{2} jX_m}{Z_b + jX_m} I_1 \tag{12}$$

With I_f and I_b determined, the developed torque

T_d can be readily calculated by Equation (5)

The Load Torque is

$$T_L = k_1 + K\omega^2 \quad (13)$$

The accelerating torque is

$$T_{acc} = T - T_L \quad (14)$$

Here $T_{acc} = J\dot{\omega} + B\omega$ and

$$\dot{\omega} = \frac{(T_{acc} - B\omega)}{J}$$

where J=moment of Inertia

B=Damping Coefficient

From the above discussion we find six basic model parameters namely stator resistance (R_s), stator reactance (X_s), rotor resistance (R_r), rotor reactance (X_r), magnetizing reactance (X_m) and capacitive reactance (X_c) and these parameters are required to be estimated for analyzing the overall performance of the system. To determine these parameters Artificial Neural Network (ANN) is used [9].

3. Fan System Parameters Estimation Using ANN

There are three approaches for modeling the fan system: white-box modeling [11], grey-box modeling [10] and black-box modeling [4,7,12]. In the white-box modeling, one assumes a known structure for the system and finds the parameters of the assumed structure using offline tests. In the grey-box modeling, one assumes a known structure for the system and uses the online measurements to estimate the physical parameters. In the black-box modeling, the structure of the model is not assumed to be known a priori. The only concern is to map the input data set to the output data set. Among the three approaches for modeling the fan system, the grey-box modeling of papers assumes a known structure for the fan system, and tries to estimate the physical parameters from online measurements. The main advantage of this category is that it yields the physical parameters. Each parameter has its physical meaning, which sounds good especially for system engineers.

The ANN is used in this paper for estimating the system parameters of a fan manufactured by Crompton Greaves Ltd. India. The ANN structure is suitably selected for this purpose. The block diagram of ANN parameter estimator is shown in Figure 2. The feed- forward back-propagation ANN program is written in Matlab version 7.1 for parameter estimation.

The input vector for ANN is consisting of angular speed (rpm), voltage (V), and current (A) at different delay time. The output vector is consisting of all system parameters at different operating conditions. The ANN model consisting of four layers namely, one input layer, two hidden layers and one output layer. The number of neurons at input layer is equal to the number of inputs and the number of neurons at output layer is equal to the number of system parameters. The number of neurons for both hidden layers is 8. Although it may be changed but 8 neurons are giving good results. Once, the ANN model is trained off line with these simulated data using Levenberg – Morquate algorithm (the training performance of ANN is shown in Figure 3), then this trained ANN model is used to predict the system parameters for on-line data acquired from the experimental set up. The manufacturer data is shown in Table 1.

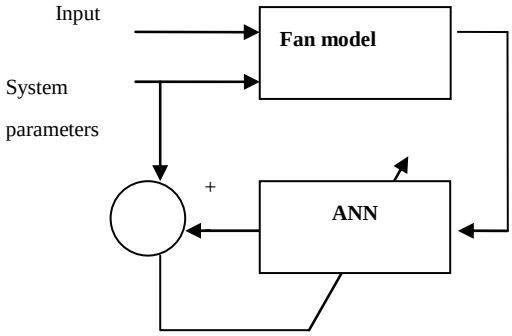


Figure 2. ANN model development for system parameters estimation from simulation results

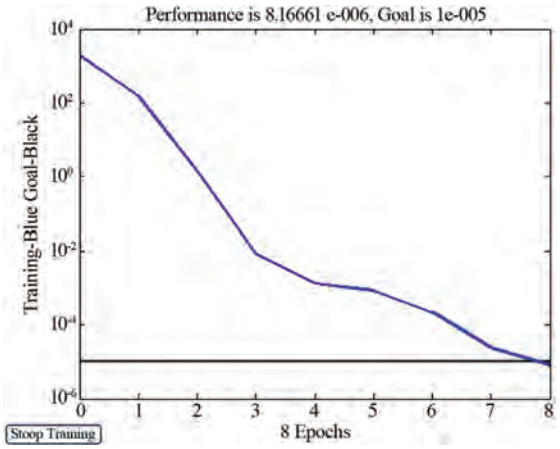


Figure 3. ANN training graph

Table 1. Manufacturer data for fan system

S.No.	System Parameter	Value (Ohms)
1.	Rs	2.02
2.	Rr	4.12
3.	Xs	2.79
4.	Xm	106.8
5.	Xr	2.12
6.	Xc	7.0

4. Experimental Setup

The experimentation is done in Electrical Engineering Lab at Dayalbagh Educational Institute, (Deemed Univ.), Agra, India. The theoretical results are further validated on a physical model. The physical model is consisting of a ceiling fan of Crompton Greaves with the specifications of 1200 mm long blades, 230V, 50Hz, and 60W.

The laboratory model is consisting of a ceiling fan, voltage controller, data acquisition (DAQ) board, and man-machine interface. The real time data acquired with the help of National Instrument Lab-view software and the parameter estimation algorithm is implemented on the real time Matlab software (version7.1) with a 50 ms step size on digital signal processing (DSP) board. The DAQ and DSP boards are installed in a personal computer with the corresponding development software. The analog to digital input channel of the DSP board receives the input signal such as fan speed, supply voltage and current. These input variables are used to calculate the system parameters on-line [9] with the help of ANN as shown in Figure 4. The estimated parameters compared with the manufacturer data under normal operating conditions.

5. Results and Discussion

The random noise is incorporated in the training data to increase the generalization capability (fault tolerant capability) of ANN and the results are shown in Table 2.

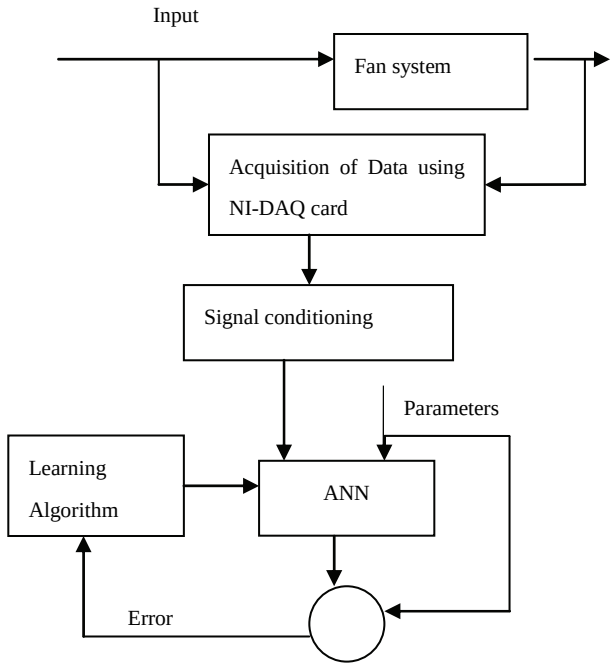


Figure 4. Experimental set up of parameter estimation of fan system using ANN

Further, it is shown that resistances are somewhat increasing while reactances (excluding Xc) are decreasing with increase in supply voltage which is increased from 80V to 230V in the steps of 10 Volts. This is due to the fact that with increase in voltage, the fan temperature gets increased and so does the resistance values while at low voltages, slip being high, the inductive reactance values are high and then with the rise in voltage the speed raises. But these variations get stabilized at near about normal voltage as the fan attains its rated speed.

From the results shown in Figure 5 and Table 2 the

Table 2. Parameter estimation using ANN

Fan Parameters in Ohms	Test data (without noise)	Test data (with noise)	Manufacturer data
Rs	2.0194	2.1835	2.02
Xs	2.7095	2.7975	2.79
Xm	106.7839	104.8351	106.8
Rr	4.1199	3.9647	4.12
Xr	2.1188	1.9660	2.12
Xc	6.9993	6.8436	7.00

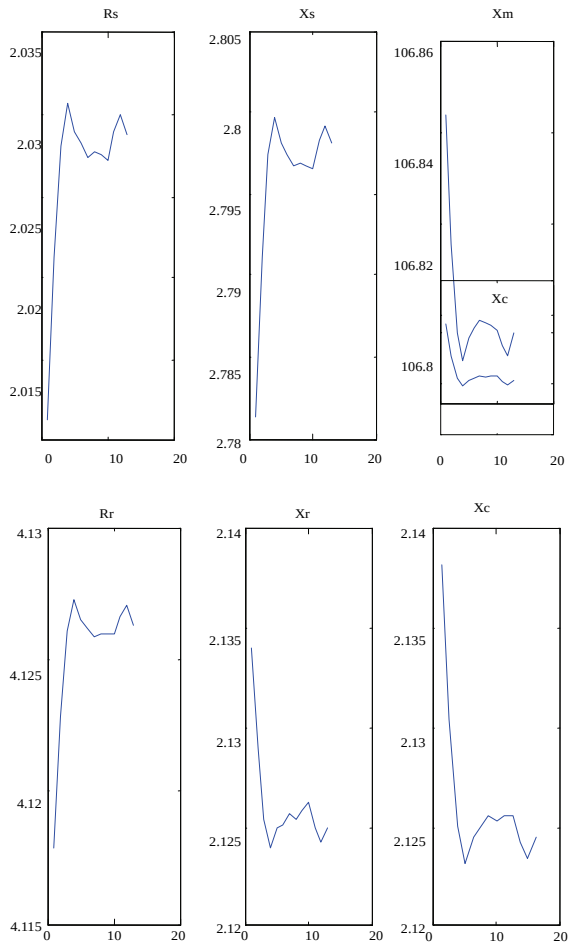


Figure 5. Variation in system parameters

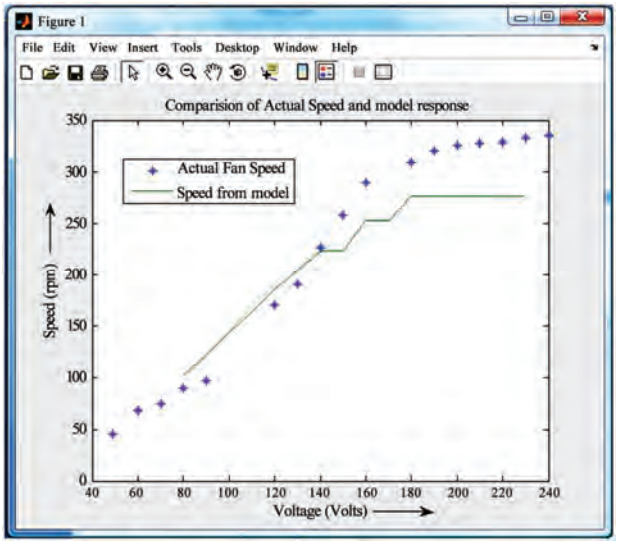


Figure 6. Comparison of simulated and experimental fan speed at different voltages

system parameters are very much dependent upon the operating conditions i.e. the applied voltage, temperature, humidity etc. If there is any abnormality in the system, the parameter changes to a great extent and that abnormality could be recognized and system may be protected from the major fault. The simulation and experimental results for the fan system are compared as shown in Figure 6.

6. Conclusions

In this paper, the mathematical model for fan system is developed and matlab code is written to validate it. Then the experimental data acquired using LABVIEW from fan system in the laboratory and used to estimate the system parameters using ANN approach under different operating voltages and the results have been compared. The results show that the ANN on-line parameter estimation method is fairly good and quite useful for monitoring the system conditions.

It is clear from the experimental results that the performance characteristics obtained from experimental data and those from the simulated data using artificial neural networks (ANN) are in close proximity. So the method used for parameter estimation, performance prediction is almost accurate for all practical purposes.

As the model development is based on the dimensional information, the method is applicable to different types of electric fans (i.e., different frame sizes as well as of different ratings). Even at the design stage the model can be applied to estimate the performance of the electric fans and to check whether the performance deviates from the desired one. The developed technique will be very useful to designers and manufacturers.

The work may be extended to improve the results by incorporating system nonlinearities in the model and pre-processing the experimental data for ANN training. Also suitable control system as well as the protection system may be developed based on on-line parameter variations.

REFERENCES

[1] A. Arredondo, R. Partha, and W. Erin, "Implementing PWM fan speed control within a computer chassis power supply," 20th Annual IEEE Applied Power Electronics Conference and Exposition, pp. 148–151, March 2005.

[2] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of induction motor," IEEE Transactions on IA, Vol. 22, No. 5, pp. 820–827, September/October 1986.

[3] S. F. Wang, R. Y. Chen, X. Fang and J. B. Wang, "Research on adjust speed control system of partial fan," IEEE International Conference on Automation and Logistics, pp. 1053–1057, August 2007.

[4] AMCA Publication 203–90, "Field performance measurement of fan systems," 0203X90A–S Arlington Heights, Ill.: the Air Movement and Control Association International, Inc. AMCA, 1990.

[5] B. K. Bose, "Expert systems, fuzzy logic, and neural network applications in power electronics and motion control," Proceedings of IEEE, Vol. 82, pp. 1303–1323, August 1994.

[6] M. Calvo and O. P. Malik, "Machine parameter estimation as a pattern recognition problem," Proceedings of IEEE Power Engineering Society Summer Meeting, Vancouver, BC, Canada, July 15th–19th, 2001.

[7] D. K. Chaturvedi, "Soft computing techniques and its applications in electrical engineering," Springer, 2008.

[8] G. Daniel "Principles of artificial neural networks," World Scientific Publishing Co. Pte. Ltd.

[9] R. D. Fard, M. Karrari, and O. P. Malik, "Synchronous generator model identification for control application using Volterra series," IEEE Transactions on Energy Conversion., Vol. 20, No. 4, pp. 852–858, December 2005.

[10] K. S. Fu, Ed., "ANN application of pattern recognition," Boca Raton, FL: CRC, 1982.

[11] H. Tsai, A. Keyhani, J. Demcko, and R. G. Farmer, "On-line synchronous machine parameter estimation from small disturbance operating data," IEEE Transactions on Energy Conversion, Vol. EC–10, pp. 25–36, March 1995.

[12] H. B. Karayaka, A. Keyhani, G. T. Heydt, B. L. Agraval, and D. A. Selin, "Synchronous generator model identification and parameter estimation from operating data," IEEE Transactions on Energy Conversion, Vol. 18, No. 1, pp. 121–126, March 2003.

[13] M. Karrari and O. P. Malik, "Identification of physical parameters of a synchronous generator from on-line measurements," IEEE Transactions on Energy Conversion, Vol. 19, No. 2, pp. 407–415, June 2004.

[14] G. Kenne, T. Ahmed-Ali, L. F. Lagarrigue, and H. Nkwawo, "Nonlinear systems parameters estimation using radial basis function network," Control Engineering Practice, Vol. 14, No. 7, pp. 819–832, 2006.

[15] W. S. Meisel, Computer-Oriented Approaches to Pattern Recognition, Academic Press, New York, 1972.

[16] D. Pitis, "Energy efficient single stage axial fan (ENEF)," IEEE Canada Electrical Power Conference, pp. 280–285, October 2007.

[17] D. K. Chaturvedi, "Modeling and simulation of systems using matlab/simulink@," CRC Press, U.K., 2009.

Journal of Intelligent Learning Systems and Applications, **2010, 2: 49-53**
doi:10.4236/jilsa.2010.21007 Published Online February 2010 (http://www.SciRP.org/journal/jilsa)

Particle Filtering Optimized by Swarm Intelligence Algorithm

Wei Jing, Hai Zhao, Chunhe Song, Dan Liu

Institute of Information and Technology, Northeastern University, Shenyang, China.
Email: Jingw@mail.neuera.com, {zhhai, songchh, liud}@mail.neuera.com

Received October 27th, 2009; accepted January 7th, 2010.

ABSTRACT

A new filtering algorithm — PSO-UPF was proposed for nonlinear dynamic systems. Basing on the concept of re-sampling, particles with bigger weights should be re-sampled more time, and in the PSO-UPF, after calculating the weight of particles, some particles will join in the refining process, which means that these particles will move to the region with higher weights. This process can be regarded as one-step predefined PSO process, so the proposed algorithm is named PSO-UPF. Although the PSO process increases the computing load of PSO-UPF, but the refined weights may make the proposed distribution more closed to the poster distribution. The proposed PSO-UPF algorithm was compared with other several filtering algorithms and the simulating results show that means and variances of PSO-UPF are lower than other filtering algorithms.

Keywords*: *Filtering Method, Particle Filtering, Unscented Kalman Filter, Particle Swarm Optimizer

1. Introduction

Sequential signal processing has a wide range of applications in many fields such as statistical signal processing [1], target tracking [2,3], *et al*.. Currently, there are many filtering algorithm such as EKF [4], UKF [5], PF [6], UPF [7], and so on. Particle filtering is a young filtering method. Its advantage over other sequential methods is particularly distinctive in situations where the used models are nonlinear and the involved noise processes are non-Gaussian. An important feature in the implementation of particle filters is that the random measure is recursively updated. With the random measure, one can compute various types of estimates with considerable ease. Particle filtering has three important operations, sampling, weight computation, and re-sampling. With sampling, one generates a set of new particles that represents the support of the random measure, and with weight computation, one calculates the weights of the particles. Re-sampling is an important operation because without which particle filtering will get poor results. With re-sampling one replicates the particles that have large weights and removes the ones with negligible weights.

Eberhart and Kennedy (1995) proposed the Particle Swarm Optimization (PSO) algorithm is motivated from the simulation of birds' social behavior. With many advantages of computing with real number, few parameters to be adjusted, the PSO algorithm is applied in many fields such as NN-training, Optimization, and Fussy Control etc. PSO is an optimization strategy generally employed to find a global best point.

In this paper, a new filtering algorithm — PSO-UPF was proposed for nonlinear dynamic systems. Basing on the concept of re-sampling, particles with bigger weights should be re-sampled more time, and in the PSO-UPF, after calculating the weight of particles, some particles will join in the refining process, which means that these particles will move to the region with higher weights. This process can be regarded as one-step predefined PSO process, so the proposed algorithm is named PSO-UPF. Although the PSO process increases the computing load of PSO-UPF, but the refined weights may make the proposed distribution more closed to the poster distribution. The proposed PSO-UPF algorithm was compared with other several filtering algorithms and the simulating results show that means and variances of PSO-UPF are lower than other filtering algorithms.

The remaining of the paper is organized as follows: in the Section 2, a brief description of GPF is presented. In the Section 3, firstly, the PSO algorithm is introduced, and a new type PSO — one-step predefined PSO process is proposed. Then the details of the new PF this paper proposed — PSO-UPF is presented. In Section 4 the proposed algorithm is compared to other several different

filtering algorithms, and finally, some concluding remarks is given in Section 5.

2. Basic Particle Filter

The problem being addressed here is an estimating problem of the state of a system as a set of observations becomes available on-line, which can be expressed as follows:

$$\begin{aligned} x_t &= f(x_{t-1}) + w_{t-1} \\ y_t &= h(u_t, x_t) + v_t \end{aligned} \tag{1}$$

where $x_t \in \Re^{n_x}$, $y_t \in \Re^{n_y}$, $u_t \in \Re^{n_u}$, $v_t \in \Re^{n_v}$ are the state of the system, the output observations, the input observations, the process noise and the measurement noise. The mappings:

$f : \Re^{n_x} * \Re^{n_x} \mapsto \Re^{n_x}$ and $h : \Re^{n_x} * \Re^{n_v} \mapsto \Re^{n_y}$ represent the deterministic process and measurement models.

In the bayes filtering paradigm, the posterior distribution is updated recursively over the current state x_t given all observations $Z_t = \{z_i\}_{i=1}^{t}$ up to and including time t as follows [6]:

$$p(x_t \mid Y_{t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid Y_{t-1}) dx_k \tag{2}$$

$$p(x_t \mid Y_t) = \frac{p(y_t \mid x_t)\, p(x_t \mid Y_{t-1})}{p(y_t \mid Y_{t-1})} \tag{3}$$

$$p(x_t \mid Y_t) = \frac{p(y_t \mid x_t)\, p(x_t \mid Y_{t-1})}{p(y_t \mid Y_{t-1})} \tag{4}$$

Using Monte Carlo sampling points, particle filter executes the filtering process by generating weighted sampling points of state variances recursively. Generic particle filter algorithm can be predicted as follows [6]:

Initialization

For each particle

draw the states $x_0^{(i)}$ from the prior $p(x_0)$;

End

For each loop

importance sampling step

For each particle

sample $\hat{x}_0^{(i)} \sim q(x_t \mid x_{0:t-1}^{(i)}, y_{1:t})$

End

For each particle

evaluate the importance weights up to a normalizing constant:

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(y_t \mid x_t^{(i)})\, p(\hat{x}_t^{(i)} \mid x_{t-1}^{(i)})}{q(\hat{x}_t^{(i)} \mid \hat{x}_{0:t-1}^{(i)}, y_{1:t})} \tag{5}$$

For each particle, normalize the importance weights:

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} [\sum_{j=1}^{N} w_t^{(j)}]^{-1} \tag{6}$$

// Re-sample

Eliminate the samples with low importance weights and multiply the samples with high importance weights, to obtain N random samples $x_{0:k}^{(i)}$ approximately distributed according to $p(x_{0:k} \mid z_{1:k})$.

Assign each particle an equal weight: $w_t^i = 1/N$.

When executing particle filtering, the choice of the proposal distribution is very important. Usually, the transition prior distribution is chosen to be the proposal distribution:

$$q(x_t \mid X_{t-1}, Y_t) = p(x_t \mid x_{k-1}) \tag{7}$$

But as not considering the recent observation, when using the transition prior distribution as the proposal distribution, the filtering result is usually poor, especially when the noise is heavy. In this paper, a kind of semi-iterative unscented transformation was proposed to address this issue.

3. The PSO-UPF Algorithm

3.1 Particle Swarm Optimizer Algorithm

PSO is an optimization strategy generally employed to find a global best point. At each time step, a particle updates its position and velocity by the following equations:

$$\begin{aligned} v_{ij}(t+1) = w * v_{ij}(t) + c_1 r_{1j}(t)(p_{ij}(t) - x_{ij}(t)) \\ + c_2 r_{2j}(t)(p_{gj}(t) - x_{ij}(t)) \end{aligned} \tag{8}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{9}$$

$$w_t = w_{\max} - (w_{\max} - w_{\min}) \frac{t}{g_{\max}} \tag{10}$$

where $j \in \{1, 2, ..., Dn\}$, $i \in \{1, 2, ..., n\}$, n is the size of the population and Dn is the Dimension of the space searched; w is the inertia weight, generally updated as equation 3 [10], and $g_{\max}$ is the maximal evolution generations. c_1 and c_2 are two positive constants; r_1 and r_2 are two random values into the range $[0, 1]$.

3.2 PSO in the PF Process

As depicted in the Section 2, in the re-sampling process of regular particle filtering, the particles with bigger weights will have more offspring, while the particles with smaller weights will have few or even on offspring. It inspires us to find more particles with big weights and reduce the number of particles with small weights, which will make the proposal distribution more closed to the poster distribution. And it is the aim of using particles swarm optimization in the particles filtering process.

The most important issue of using particles swarm

optimizer is the choice of fitness function. In the proposed algorithm, we want to find more particles with bigger weights, so the fitness can be chosen as the value of weights directly. As usually the aim of PSO algorithms is to minimize the fitness function, so in the PSO-PF, the fitness is the minus value of particle weight as follows:

$$fitness_t^{(i)} = \frac{p(y_t \mid x_t^{(i)})\, p(\hat{x}_t^{(i)} \mid x_{t-1}^{(i)})}{q(\hat{x}_t^{(i)} \mid \hat{x}_{0:t-1}^{(i)}, y_{1:t})} \tag{11}$$

Secondly, as the computing consumption of particle filtering is already very large, so the computing consumption of new introduced PSO process should be reduced. In the PSO-PF algorithm, for every particle, at first, a random number in the range of 0 and 1 will be generated, and only when the number is smaller than a predefined threshold, the PSO process can be conducted. A new type of PSO process — one step predefined PSO is introduced. In this process, only when the new location is better than the original one, the particle will move to the new one, and the location updating process will be conducted only one time in each particle filtering generation.

Finally, the direction of particle updating is determined by the location of best particle in current generation and itself location. And as in the one-step predefined PSO process, particle need not remember its individual best location of history; the updating mode uses the social only mode of PSO algorithm.

3.3 The PSO-UPF Algorithm

In this section, the mentioned one-step predefined PSO process is used in the UPF algorithm. The information of UPF algorithm can be found in [9,10]. And he PSO-UPF algorithm can be decried as follows:

For each particle

draw the states $x_0^{(i)}$ from the prior $p(x_0)$;

set $\overline{x}_0^{(i)} = E[x_0^{(i)}]$

$P_0^{(i)} = E[(x_0^{(i)} - \overline{x}_0^{(i)})(x_0^{(i)} - \overline{x}_0^{(i)})^T]$

$x_0^{(i)a} = E(x_0^{(i)a}) = [(x_0^{(i)})^T \ 0 \ 0]^T$

$P_0^{(i)a} = E[(x_0^{(i)a} - \overline{x}_0^{(i)a})(x_0^{(i)a} - \overline{x}_0^{(i)a})^T]$

For each loop

// Generate proposal distribution and resample

For each particle

// calculate sigma points:

$\chi_{t-1}^{(i)a} = [\overline{\chi}_{t-1}^{(i)a} \ \ \overline{\chi}_{t-1}^{(i)a} \pm \sqrt{(na+\lambda)P_{t-1}^{(i)a}}]$

// update particles into next time:

$\chi_t^{(i)x} = f(\chi_{t-1}^{(i)x}, \chi_{t-1}^{(i)v})$, $\overline{x}_t^{(i)x} = \sum_{j=0}^{2n_a} W_j^{(m)} \chi_{j_{t|t-1}}^{(i)x}$

$P_{t|t-1}^{(i)x} = \sum_{j=0}^{2n_a} W_j^{(m)} [\chi_{j_{t|t-1}}^{(i)x} - \overline{x}_{j_{t|t-1}}^{(i)x}][\chi_{j_{t|t-1}}^{(i)x} - \overline{x}_{j_{t|t-1}}^{(i)x}]^T$

$\gamma_{t|t-1}^{(i)x} = h(\chi_{t|t-1}^{(i)x}, \chi_{t|t-1}^{(i)n}]$, $\overline{y}_{t|t-1}^{(i)x} = \sum_{j=0}^{2n_a} W_j^{(m)} \gamma_{j_{t|t-1}}^{(i)x}$

// Measurement update:

$P_{\tilde{y}_t \tilde{y}_t} = \sum_{j=0}^{2n_a} W_j^{(m)} [\gamma_{j_{t|t-1}}^{(i)x} - \overline{y}_{j_{t|t-1}}^{(i)x}][\gamma_{j_{t|t-1}}^{(i)x} - \overline{y}_{j_{t|t-1}}^{(i)x}]^T$

$P_{x_t y_t} = \sum_{j=0}^{2n_a} W_j^{(m)} [\chi_{j_{t|t-1}}^{(i)x} - \overline{x}_{j_{t|t-1}}^{(i)x}][\gamma_{j_{t|t-1}}^{(i)x} - \overline{y}_{j_{t|t-1}}^{(i)x}]^T$

$K_t = P_{x_t y_t} P_{\tilde{y}_t \tilde{y}_t}^{-1}$

$\overline{x}_t^{(i)} = \overline{x}_{t|t-1}^{(i)} + K_t (y_t - y_{t|t-1}^{(i)})$

$\hat{P}_t^{(i)} = P_{t|t-1}^{(i)} + K_t P_{\tilde{y}_t \tilde{y}_t} K_t^T$

// Sample

$\hat{x}_t^{(i)} \sim q(x_t^{(i)} \mid x_t^{(i)}, y_{1:t}) \triangleq N(\overline{x}_t^{(i)}, \hat{P}_t^{(i)})$

Set $\hat{x}_{0:t}^{(i)} = (x_{0:t}^{(i)}, \hat{x}_t^{(i)})$ and $\hat{P}_{0:t}^{(i)} = (P_{0:t}^{(i)}, \hat{P}_t^{(i)})$

For each particle:

Calculate fitness as equ.11, denoted by F;

Find the particle with best fitness, suppose the location is L*;

For each particle

Generate a random number c in the range of [0 1];

If c<C // C is the predefined threshold.

newLocation* = originalLocation + rand*(L*-orginalLocaion);

calculate the fitness of newLocation* , denoted by F*;

if F*<F

orginalLocation = newLocation;

end if

end if

end for

For each particle

Normalize the importance weights as equ.4

Re-sample

Eliminate the samples with low importance weights and multiply the samples with high importance weights, to obtain N random samples $x_{0:k}^{(i)}$ approximately distributed according to $p(x_{0:k} \mid z_{1:k})$.

Assign each particle an equal weight: $w_t^i = 1/N$.

4. The Simulation Experiments

In this paper, the proposed algorithm was compared to other seven filtering algorithm — EKF, UKF, GPF, GPFMCMC, EKFPF, EKFPFMCMC, UPF, Experimental settings are shown in the Table 1. The settings of

other six filtering algorithms are the same as [4–8].

4.1 Benchmark Function

In this paper, the following benchmark function [10] was chosen to test the proposal algorithm.

Bechmark 1:

$$x_t = 1 + \sin(w\pi(t-1)) + \frac{1}{2}x_{t-1} + u_t$$

$$y_t = \begin{cases} \frac{1}{5}x_t^2 + v_t, & t \le 30 \\ \frac{1}{2}x_t - 2 + v_t & t > 30 \end{cases}$$

Bechmark 2:

$$x_t = 1 + \sin(0.04 * \pi * (t-1)) + 0.5 * x_{t-1} + w_{t-1}$$

$$y_k = 0.2 * x_k^2 + \cos(x_t)/10 + v_k$$

where $w_t \sim \gamma(3,1)$, $v_k \sim N(0,1e-2)$, particle number N=200, sample time T=100, whicle the results were the average of 50 times of experiments. $C = 0.5$, $R = 1e-2$, $Q = 0.75$; $\alpha = 0.5$, $\beta = 0.5$, $k = 1$。

4.2 Experimental Results and Some Remarks

Experimental results are shown in Figure 1–Figure 2 and Table 1~Table 1. All results are the means of 100 runs, as the results of UPF and SIUPF are close and far different with others, an enlargement figure was drawn as Figure 3.

As shown in the experimental results, it is clear that, EKF has the worst results, but has the best running time. While the proposed algorithm has the best results, but has longer running time. In theory, the running time of PSO-PF is between one and two times of UPF, due to the refining step, and the simulation results has proved it.

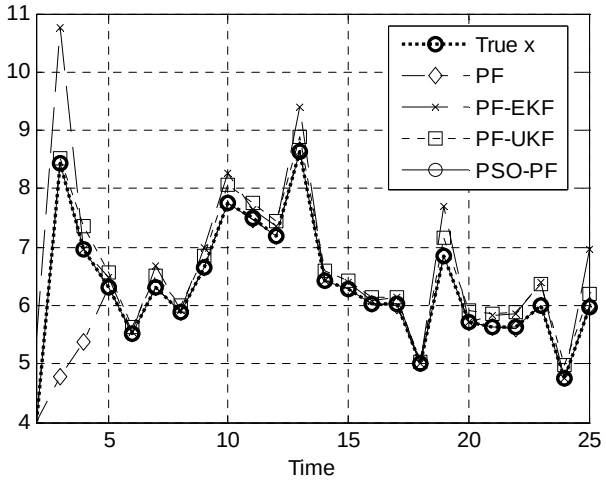


Figure 1. Results on benchmark 1

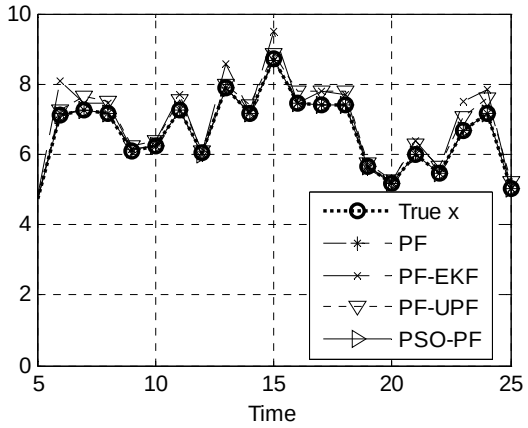


Figure 2. Results on benchmark 2

Table 1. Results on benchmark 1

	MSE		MeanRunTime
	mean	variance	
EKF	0.6975	0.1248	-
UKF	0.3234	0.1297	-
PF	0.18001	0.1650	0.4733
PF-EKF	0.1426	0.2147	4.1292
PF-UKF	0.1239	0.1107	7.2136
PSO-PF	0.0268	0.0359	8.2324

Table 2. Results on benchmark 2

	MSE		MeanRunTime
	mean	variance	
EKF	0.3375	0.1438	-
UKF	0.2347	0.1277	-
PF	0.2301	0.6247	0.3903
PF-EKF	0.3181	0.1147	5.1652
PF-UKF	0.2339	0.1347	7.1336
MPF	0.0968	0.0959	8.1224

5. Conclusions

Basing on the concept of re-sampling, particles with bigger weights should be re-sampled more time, this paper has proposed a new type of particle filtering algorithm — PSO-UPF. In the PSO-UPF, after calculating the weight of particles, some particles will join in the refining process, which means that these particles will move to the region with higher weights. This process can be regarded as one-step predefined PSO process, so the proposed algorithm is named PSO-UPF. Although the PSO process increases the computing load of PSO-UPF, but the refined weights may make the proposed distribution more closed to the poster distribution. In the following experiment, the proposed algorithm has better performances than other several types of filtering methods.

REFERENCES

[1] D. Guo and X. Wang, "Quasi-monte Carlo filtering in nonlinear dynamic systems," IEEE Transactions on Sig-

nal Process, Vol. 54, No.6, pp. 2087–2098, 2006.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, *et al.*, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking [J]," IEEE Transactions on Signal Processing, Vol. 20, No. 2, pp. 174–188, 2002.

[3] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners [J]," IEEE Transactions on Signal Processing, Vol. 50, No. 3, pp. 736–746, 2002.

[4] B. D. Anderson, and J. B. Moore, "Optimal filtering," Prentice–Hall, New Jersey, 1979.

[5] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," Proceedings of AeroSense: The 11th International Sysmpsium on Aerospace/Defence Sensing, Simulation and Controls, Orlando, Florida, Muti Sensor Fusion, Tracking and Resource Management II, pp. 182–193. 1997.

[6] Y. Shi, and R. C. Eberhart, "A modified particle swarm optimizer," in Proceedings of the IEEE International Conference on Evolutionary Computation, Piscataway, NJ: IEEE Press, pp. 69–73, 1998.

[7] J. A Riget, "Diversity-guided particle swarm optimizer —the ARPSO," EVALife Technical Report, Department of Computer Science, University of Arhus, 2002.

[8] D. Guo, X. Wang, and R. Chen, "New sequential monte carlo methods for nonlinear dynamic systems," Statistics and Computing, Vol. 15, No. 2, pp. 135–147, 2005.

[9] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, "Estimation with applications to tracking and navigation: Theory, Algorithm and Software [M]," New York: Wiley, 2001.


9772150840002 02