Noisy Speech Processing by Recurrently Adaptive Fuzzy Filters

Chia-Feng Juang, *Member, IEEE,* and Chin-Teng Lin, *Senior Member, IEEE*

***Abstract*—Two noisy speech processing problems—speech enhancement and noisy speech recognition—are dealt with in this paper. The technique we focus on is by using the filtering approach; a novel filter, the recurrently adaptive fuzzy filter (RAFF), is proposed and applied to these two problems. The speech enhancement is based on adaptive noise cancellation with two microphones, where the RAFF is used to eliminate the noise corrupting the desired speech signal in the primary channel. As to the noisy speech recognition, the RAFF is used to filter the noise in the feature domain of speech signals. The RAFF is inherently a recurrent multilayered connectionist network for realizing the basic elements and functions of *dynamic* fuzzy inference, and may be considered to be constructed from a series of dynamic fuzzy rules. As compared to other existing nonlinear filters, three major advantages of the RAFF are observed: 1) *a priori* knowledge can be incorporated into the RAFF, which makes the fusion of numerical data and linguistic information possible; 2) owing to the dynamic property of the RAFF, the exact lagged order of the input variables need not be known in advance; 3) no predetermination, like the number of hidden nodes, must be given since the RAFF can find its optimal structure and parameters automatically. Several examples on adaptive noise cancellation and noisy speech recognition problems using the RAFF are illustrated to demonstrate the performance of the RAFF.**

***Index Terms*—Adaptive noise cancellation, noisy speech recognition, real-time recurrent learning, structure identification.**

I. INTRODUCTION

SIGNALS are usually corrupted by noise in the real world. To reduce the influence of noise, two research topics—the speech enhancement and speech recognition in noisy environments—have arose. For the speech enhancement, the extraction of a signal buried in noise, adaptive noise cancellation (ANC) [1] provides a good solution. In contrast to other enhancement techniques, its great strength lies in the fact that no *a priori* knowledge of signal or noise is required in advance. The advantage is gained with the auxiliary of a secondary input to measure the noise source. The cancellation operation is based on the following principle. Since the desired signal is corrupted by the noise, if the noise can be estimated from the noise source, this estimated noise can then be subtracted from the primary channel resulting in the desired signal. Traditionally, this task is done by linear filtering. In real situations, the corrupting noise is a nonlinear distortion version of the source noise, so a nonlinear filter should be a better choice [2]–[4].

Manuscript received December 27, 1999; revised April 3, 2000.

C.-F. Juang is with the Department of Electrical Engineering, Chung-Chou Institute of Technology, Chang-Hua, 510 Taiwan, R.O.C.

C.-T. Lin is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, 300 Taiwan, R.O.C.

Publisher Item Identifier S 1063-6706(01)01217-6.

About the noisy speech recognition problem, it has been shown in literature that a speech recognizer designed to perform well under noise-free conditions usually shows marked degradation in performance when the background noise is present [5]. Many techniques, like the noise-robust acoustic representations, noise-robust spectral distortion measures, and parameter mapping methods, etc. have been proposed to solve this problem [6]. Among them, the parameter mapping method aims to filter the noise in the feature domain via a noise reduction networ and is the focus of this paper.

Adaptive filtering has achieved widespread applications [1]. Among the various adaptive filters, linear filtering is the most widely used method in this area. However, for applications in the real world such as the ANC and noisy speech recognition by linear filtering, poor performance is usually observed. The development of nonlinear filters is thus necessary. In fact, some kinds of nonlinear adaptive filters have been proposed. Adaptive volterra filter [7] is a linear combination of order stochastics and the combination coefficients are tunable when the signal or noise statistics change. The drawbacks of this filter are that it is constrained to be applied to the class of nonlinear systems that can be represented by the Volterra series expansion and, as the order of the filter increases, much greater complexity in design and filter size occurs. This higher order problem also occurs in another type of filter, the adaptive stack filter [8], which is constrained to be applied to the situations when the threshold levels are small.

Another kind of filter, the neural filter [9], is proposed to take advantage of the universal approximation and powerful learning ability of the neural networks. Applications of this type of filters include channel equalization [12], [13], restoration of images corrupted by impulse noise [9], speech enhancement [3], [10], [11], and noisy speech recognition [14]–[20], etc. In [10], a multilayer perceptron neural network is used as an adaptive filter for noise canceling, where two microphone beamformers are used for suppressing directional background noise. In [11], a recurrent radial basis function network, whose output is fed back as the input variables, is used as an adaptive filter for noise cancellation. As for the noisy speech recognition problem, in [14] and [15], a noise reduction neural network is used to realize a mapping from the set of noisy signals to the set of noise-free signals in the time domain. The results showed even for noisy speech signals that had not been part of the training data, the network successfully produced noise-suppressed output signals. The drawback of the approach is its expensive computation load since every speech sample is propagated through the noise reduction model. To solve this problem, the cepstral noise reduction networks are proposed in [16] and [17], where the networks are performed on the feature domain and the cepstral coefficients are used as features. The results showed that the use of feature vectors as input signals resulted in significantly smaller networks and, hence, in reduction of computation time.

Since the speech signals are temporal signals, in [18] the information about the characteristics of the noise and speech components of the input signals including context information in feature domain is taken into account. The context features together with the current ones are fed to the noise reduction network, a feedforward multilayer perceptron network and a higher recognition rate is achieved with the cost of increasing the network input dimension. In [19], a recurrent neural network is used as a noise reduction network to capture the temporal relationship of the speech signals without increasing the input dimension and determining the context frame number. In [20], a structure different from those above is proposed; a number of multilayer perceptron-based estimators are placed in every filter bank in the feature domain to estimate each filter bank channel output in [20]. Local statistical information of the speech and the noise are estimated and used as inputs to the networks.

In general, the aforementioned neural filters can be trained by numerical data only. When we are constructing information processing systems, like the filters, the available information processing systems usually comes in two forms: numerical form and linguistic form. The inconvenience of incorporating linguistic information expressed by fuzzy IF-THEN rules in design is a shortcoming of the neural filter. Owning to this reason, another type of filter—the adaptive neural fuzzy filter [4] or fuzzy adaptive filter [22]—which makes the combination of numerical information as well as linguistic information possible, are proposed. Besides this combination ability, the learning speed of this type of filter is usually faster than the neural filters. However, all these filters belong to the feedforward filters. To design a feedforward filter, the lagged order of the input variables, which determines the number of filter inputs, must be known in advance. In real situations, the order is usually unknown. For higher order problems, the input number is large, resulting in serious increment in filter size, whereas, if a recurrent fuzzy filter is used, these problems can be solved. If a recurrent filter that can memorize the past history of the input, is used, we only need to feed the current state to the filter and the size of the filter can be reduced considerably. This concept of recurrent fuzzy network is also stated in [23]–[26]. In [23], a quasi-linear fuzzy model (QLFM), a dynamic fuzzy model, is proposed and, in [24]–[26], concepts of dynamic fuzzy reasoning are proposed.

A recurrent filter, the recurrently adaptive fuzzy filter (RAFF), is proposed in this paper. The RAFF expands the basic ability of a neural-fuzzy-type filter to cope with delayed problems via the inclusion of some internal memories in the form of *context elements*. In contrast to other recurrent neural filters whose structures are always opaque to the user, the weight as well as the node in the RAFF has its own meaning and plays a special element in a fuzzy rule. More clearly, with the context elements, the filter performs the following reasoning:

Rule i: IF $x_1(t)$ is A_{i1} and $\cdots$ and $x_n(t)$ is A_{in} and
$h_i(t)$ is G_i
THEN $y_1(t+1)$ is B_{i1} and $h_1(t+1)$ is w_{1i}
and $\cdots$ and $h_m(t+1)$ is w_{mi}

where

x_i	input;
y_1	output;
A_{i1}, A_{in}, G_i and B_{i1}	fuzzy sets;
h_i	internal variable;
w_{1i} and w_{mi}	fuzzy singletons;
n and m	numbers of input and hidden variables, respectively.

The dynamic reasoning implies that the inference output $y_1(t+1)$ is affected by the internal variable $h_i(t)$ and the current internal output $h_i(t+1)$ is a function of previous output value $h_i(t)$, i.e., the internal variable h_i itself forms the dynamic reasoning.

In contrast to other filters, where the filter structure is fixed and should be assigned in advance there are no rules initially in the RAFF; all of them are constructed during on-line learning. Two learning phases, the structure as well as parameter learning phases, are used to accomplish this task. The structure learning phase is responsible for the generation of fuzzy IF-THEN rules as well as the judgement of feedback configuration and the parameter learning phase for the tuning of free parameters of each dynamic rule (like the shapes and positions of membership functions and the singleton values). The RAFF is applied to the ANC and noisy speech recognition problems.

This paper is organized as follows. Section II describes the noisy speech processing techniques, the ANC and the noisy speech recognition, based on filtering approach. The structure of the RAFF and its on-line construction scheme, including structure and parameter learning, under a mean square error (MSE) criterion, is proposed in Section III. In Section IV, the RAFF is applied to the adaptive noise cancellation and noisy speech recognition problems. Finally, conclusions are summarized in the last section.

II. Noisy Speech Processing Problem

The noisy speech processing problem discussed in this section can be divided into two areas, the speech enhancement and noisy speech recognition. The speech enhancement technique, the method based upon adaptive noise cancellation with two microphones, is discussed in Section IIA. Section II-B discusses the other technique—noisy speech recognition by parameter mapping in the feature space of speech signals.

A. Adaptive Noise Cancellation

Adaptive noise cancellation is concerned with the enhancement of noise-corrupted signals and is based upon the availability of a primary input source and an auxiliary (reference) input source located at the noise field, which contains no or little signal as shown in Fig. 1. In Fig. 1, the primary input source contains the desired signal s, which is corrupted by noise n_0 generated from the noise source n. The received signal is thus

$$x(k) = s(k) + n(k). \tag{1}$$

The secondary or auxiliary (reference) input source receives the noise n_1, which is correlated with the corrupting noise n. The principle of the adaptive noise cancellation techniques is

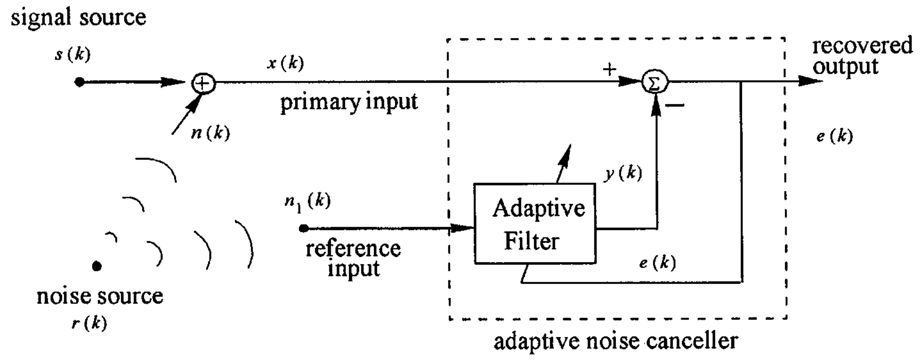


Fig. 1. Adaptive noise cancellation system.

to adaptively process (by adjusting the filter's weights) the reference noise n_1 to generate a replica of n and then subtract the replica of n from the primary input x to recover the desired signal s. We denote the replica of n, i.e., the adaptive filter output, as process y. To show how the system works, we shall follow what is derived in [1]. The assumptions that s, n and n_1 are stationary zero-mean processes, s is uncorrelated with n and n_1, and n and n_1 are correlated are made. Also, the reference input source is situated in such a position that it detects only the noise not the signal s. Here, another constraint that process y is uncorrelated with process s is added owning to the use of nonlinear adaptive filters. From Fig. 1, we have

$$e(k) = s(k) + n(k) - y(k). \tag{2}$$

By squaring and taking expectation on both sides, we can obtain

$$E[e^2(k)] = E[s^2(k)] + E[(n(k) - y(k))^2]. \tag{3}$$

Our objective is to minimize $E[(n(k) - y(k))^2]$. Observing (3), we can see that this objective is equivalent to minimizing $E[e^2(k)]$ and when $E[(n(k)-y(k))^2] = E[(n(k)-F(n_1(k))^2]$ approaches zero, the remaining error $e(k)$ is, in fact, the desired signal $s(k)$, where $F(\cdot)$ represents the function of the nonlinear adaptive filter.

Traditionally, the design of the adaptive filters for the aforementioned noise canceling problem is based upon a linear filter adapted by the least mean square (LMS) or recursive least square (RLS) algorithm. In real situations, the environment between r and n is so complex that n is in fact a nonlinear function of r [2], [3]. Better performance of noise cancellation by using a nonlinear filter can thus be expected. Some nonlinear filters, the polynominal-type filters (e.g., the Volterra filter) [2] and the neural filter [3] have been proposed for adaptive filtering. To use these filters, the order of the input variables needs to be known in advance. Moreover, if the input order is large, increment in network size is serious. If the past history of the input variable can be memorized, however, then the exact order of the input can be unknown and the size of the filter can be reduced. The RAFF proposed in the next section is thus a good choice.

B. Noisy Speech Recognition

A well-performed speech recognition system under noise-free conditions usually show marked degradation in performance when background noise is present. To solve the problem, many approaches have been proposed [6]. The noise removal by noise reduction network is one method trying to remove the noise via a mapping from the noisy input to the clean output in the feature domain. The architecture of the enhancement recognition system is shown in Fig. 2. The system works as follows. In time domain, speech signal $s(k)$ and noise signal $n(k)$ are added and a noise corrupted speech signal $s_n(k) = s(k) + n(k)$ is measured. After receiving the noisy speech signal, the next step is to extract the noisy features from the noisy speech. The function of the noise reduction filter is to map the noisy features to clean features and then feed these filtered features to the recognizer. The general design method of the noise reduction filter is as follows. At first, we generate the noisy speech signal by artificially adding noise to the clean speech. Using the noisy features extracted from the noisy speech as inputs to the filter and the corresponding clean feature as the desired output, the noise reduction filter is trained under MSE criterion. After training, the noise reduction filter maps the noisy feature extracted from the real noisy speech signal to the filtered feature for recognition to obtain a higher recognition rate. The mapping performed by the filter is quite complex; designs by liner transformation are obviously poorer than those by nonlinear transformation [19], which are usually performed by artificial neural networks [14], [19], [18]. Since the speech signal is a temporal signal and information is contained in dynamic form, adjacent information is thus important. In [18], adjacent information (features from adjacent frames) was used to achieve better performance than the mapping without using adjacent information. However, the adjacent information is obtained at the expense of expanding the input dimension. To contain longer temporal information, the input dimension should increase, causing a requirement of much larger network. Hence, a nonlinear recurrent filter with internal memories is a better choice. Although we can use existing recurrent neural networks to do this task [19], the structures of these recurrent neural networks [21], are opaque to the user and the network size has to be decided in advance. As shown below, the RAFF introduced in the next section solves these problems.

III. Structure of the RAFF

In this section, the structure of the RAFF shown in Fig. 3 is introduced. The RAFF consists of nodes, each has some finite

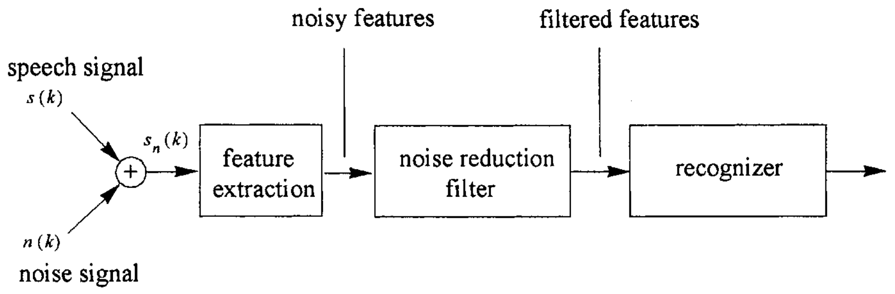


Fig. 2. Structure of the noisy speech recognition system with noise reduction filter.

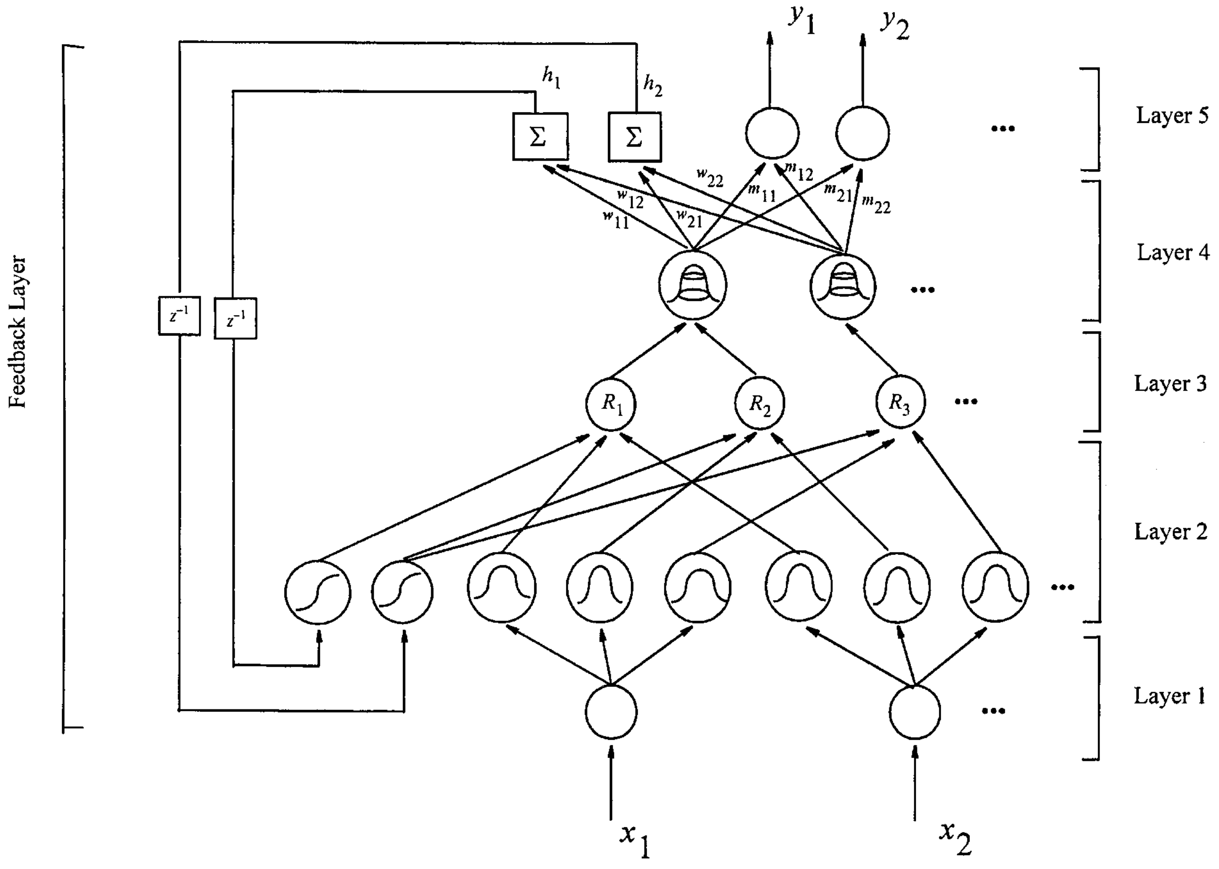


Fig. 3. Structure of the proposed recurrently adaptive fuzzy filter (RAFF).

fan-in of connections represented by weight values from other nodes and some fan-out of connections to other nodes. Basically, it is a five-layered neural fuzzy filter embedded with dynamic feedback connections (the feedback layer in Fig. 3) that brings the temporal processing ability into a feedforward neural fuzzy filter. To give a clear understanding of the filter, the function of the node in each layer is described below. In the following descriptions, the symbol $u_i^{(k)}$ denotes the ith input of a node in the kth layer; correspondingly, the symbol $a^{(k)}$ denotes the node output in layer k.

Layer 1: No computation is done in this layer. Each node in this layer is called an input linguistic node and corresponds to one input variable. The node only transmits input values to the next layer directly; that is

$$a^{(1)} = u_i^{(1)}. \tag{4}$$

Layer 2: Nodes in this layer are called input term nodes, each of which corresponds to one linguistic label (small, large, etc.) of an input variable. Each node in this layer calculates the membership value specifying the degree to which an input value belongs to a fuzzy set. A local membership function, the white Gaussian membership function, is used in this layer because a multidimensional white Gaussian membership function can be easily decomposed into the product of one-dimensional (1-D) membership functions. With this choice, the operation performed in this layer is

$$a^{(2)} = \exp\left\{-\frac{\left(u_i^{(2)} - m_{ij}\right)^2}{\sigma_{ij}^2}\right\} \tag{5}$$

where m_{ij} and σ_{ij} are, respectively, the center (or mean) and the width (or standard deviation) of the Gaussian membership function of the jth term of the ith input variable x_i.

Layer 3: Nodes in this layer are called rule nodes. A rule node represents one fuzzy logic rule and performs precondition matching of a rule. The fan-in of a node comes from two sources: one from layer 2 and the other from the feedback layer. The former represents the rule's spatial firing degree and the latter the rule's temporal firing degree (denoted by $a^{(h)}$). We use the following AND operation on each rule node to integrate these fan-in values

$$a^{(3)} = a^{(h)} \cdot \prod_i u_i^{(3)} = a^{(h)} \cdot e^{-[D_i(\mathbf{x}-\mathbf{m}_i)]^T[D_i(\mathbf{x}-\mathbf{m}_i)]} \tag{6}$$

where $D_i = \text{diag}(1/\sigma_{i1}, 1/\sigma_{i2}, \ldots, 1/\sigma_{in})$ and $\mathbf{m}_i = (m_{i1}, m_{i2}, \ldots, m_{in})^T$. Obviously, the output $a^{(3)}$ of a rule node represents the firing strength of its corresponding rule.

Layer 4: This layer is called the consequent layer and the nodes in this layer are called output term nodes. Each output term node represents a multidimensional fuzzy set (described by a multidimensional Gaussian membership function) obtained during the clustering operation in the structure learning phase. Only the center of each Gaussian membership function is delivered to the next layer for the local mean of maximum (LMOM) defuzzification operation [21], so the width is used for output clustering only. Different nodes in layer 3 may be connected to a same node in this layer, meaning that the same consequent fuzzy set is specified for different rules. The function of each output term node performs the following fuzzy OR operation:

$$a^{(4)} = \sum_i u_i^{(4)} \tag{7}$$

to integrate the fired rules, which have the same consequent part.

Layer 5: Each node in this layer is called an output linguistic node and corresponds to one output linguistic variable. This layer performs the defuzzification operation. The nodes in this layer together with the links attached to them accomplish this task. The function performed in this layer is

$$y_j = a^{(5)} = \frac{\sum_i u_i^{(5)} \hat{m}_{ji}}{\sum_i u_i^{(5)}} \tag{8}$$

where $u_i^{(5)} = a_i^{(4)}$ and $\hat{m}_{ji}$, the link weight, is the center of the membership function of the ith term of the jth output linguistic variable y_j. Here, the membership function is the fuzzy set B_{ij} that will be described at the end of this section, where the whole dynamic fuzzy rule is presented.

Feedback Layer: This layer calculates the value of the internal variable h_i and the firing strength of the internal variable to its corresponding membership function, where the firing strength contributes to the matching degree of a rule node in layer 3. As shown in Fig. 3, two types of nodes are used in this layer, the square node named as *context node* and the circle node named as *feedback term node*, where each context node is associated with a feedback term node. The number of context nodes (and, thus, the number of feedback term nodes) is the same as that of output term nodes in layer 4. Each context node and its associated feedback term node corresponds to one output term node. The inputs to a context node are from all the output term nodes, and the output of its associated feedback term node is fed to the rule nodes whose consequent is the output term node corresponded to this context node. The context node functions as a defuzzifier

$$h_j = \sum_i a_i^{(4)} w_{ji} \tag{9}$$

where the internal variable h_j is interpreted as the inference result of the hidden (internal) rule and w_{ji} is the link weight from the ith node in layer 4 to the jth internal variable. The link weight, w_{ji}, represents a fuzzy singleton in the consequent part of a rule, and also an input term of the internal variable h_j. Due to the hidden and intermediate property of an internal variable, fuzzy singleton instead of fuzzy membership function is used as its input term; a fuzzy term on an internal variable does not make much sense. In (9), the simple weighted sum is calculated [27]. Instead of using the weighted sum of each rule's outputs as the inference result, the conventional average weighted sum $h_j = \sum_i a_i^{(4)} w_{ji} / \sum_i a_i^{(4)}$ can also be used [27].

As to the feedback term node, unlike the case in the space domain, where a local membership function is used, a global membership function is adopted on the universe of discourse of the internal variable to simplify network structure and meet the global property of the temporal history. Here, the global property means that for a cluster in the space domain its history path (memorized by the internal variables) can be anywhere in the space at different time, so a global membership function, which covers the universe of discourse of the internal variable, is used to rank the influence degree each internal variable contributes to a rule. In this paper, the membership function $f(u) = 1/(1+e^{-u})$ is used for each internal variable. With this choice, the feedback term node evaluates the output by

$$a^{(h)} = \frac{1}{1+e^{-h_i}}. \tag{10}$$

This output is connected to the rule nodes in layer 3, which connect to the same output term node in layer 4. The outputs of feedback term nodes memorize the firing history of the fuzzy rules. The feedback module provides the RAFF with the ability to deal with temporal mappings. Without the feedback module, the RAFF functions as a feedforward network which can process static mapping only. To deal with temporal mappings by the feedforward structure, we should increase the input dimension to include context information. This will considerably increase the network size. In addition, it is a problem in deciding how much past input or output information (i.e., the order of the mapping problem) should be fed into the network as the inputs. By including the feedback module in the RAFF, the context information is learned and stored automatically and the temporal problem is solved implicitly.

With the aforementioned node functions in each layer, the RAFF realizes the following dynamic fuzzy reasoning:

Rule i: IF $x_1(t)$ is A_{i1} and $\cdots$ and $x_n(t)$ is A_{in} and
$h_i(t)$ is G_i
THEN $y_1(t+1)$ is B_{i1} and $y_2(t+1)$ is B_{i2} and
$h_1(t+1)$ is w_{1i} and $\cdots$ and
$h_m(t+1)$ is w_{mi}

where

x_i	input;
y_i	output;
$A_{i1}, A_{in}, G_i, B_{i1}$ and B_{i2}	fuzzy sets;
h_i	internal variable;
w_{1i} and w_{mi}	fuzzy singletons;
n and m	numbers of input and internal variables, respectively.

From the above fuzzy rule, we can see the spacial mapping relationship between the input variable $x(k)$ and output variable $y(k)$, the influence of $h_i(t)$ on this spacial mapping, and the influence degree of each rule on other ones as explained as follows. Owning to the monotonic increasing property of the membership function G_i, a higher value of h_i means a higher firing strength (or influence). From the value of $h_i(t+1)$ in the consequent part of each rule, we may see the influence degree of the rule on other ones, or the influence of a spacial cluster on other clusters in the input space. For the cases where *a priori* knowledge of the spacial mapping is clear and known, we may put the *a priori* knowledge in the dynamic rule with w in each rule being set as zero, meaning that the influence degree of h_i to every rule is of the same value, and there is no temporal influence in the beginning. Further temporal relationship, the tuning of w may be learned by the succeeding parameter learning.

IV. Learning Algorithms for the RAFF

In this section, we develop an on-line learning algorithm to find the optimal RAFF under the MSE criterion.

A. Problem Formulation

The problem of the design or adaptation of an optimal fuzzy filter can be phrased as follows. Given a process $\mathbf{x}(m)$ specified in a finite interval length, $m_1 < m < m_2$, we are to design a nonlinear filter in such a way that the estimated value $\hat{s}(k)$ based upon $\mathbf{x}(m)$ is as close as possible to the desired process $s(k)$. Written in mathematical form, we have

$$\hat{s}(k) = F(\mathbf{x}(m)) \tag{11}$$

where k is in the interval $[m_1, m_2]$ and $F(\cdot)$ represents the function of the desired nonlinear filter. The process $\mathbf{x}(m)$ is usually a nonlinear version of $s(k)$ corrupted with noise $n(m)$. The objective is to find the optimal filter $F(\cdot)$ so as to minimize the MSE

$$E[(s(k)-\hat{s}(k))^2] = E[(s(k)-F(\mathbf{x}(m)))^2]. \tag{12}$$

Two steps, the structure learning step and the parameter learning step are used concurrently to achieve this goal and are introduced in the following subsection.

B. Learning Algorithms for the RAFF

Two types of learning—structure and parameter learning—are used concurrently for constructing the RAFF. The structure learning includes the precondition, consequent, and feedback structure identification of a dynamic fuzzy IF-THEN rule. Here, the precondition structure identification corresponds to the input space partitioning. The consequent structure identification is to decide when to generate a new membership function for the output variable based upon clustering. As to the feedback structure identification, the main task is to decide the number of internal variables with its corresponding feedback fuzzy terms and the connection of these terms to each rule. For the parameter learning based upon supervised learning, an ordered derivative learning algorithm is derived to update the free parameters in the RAFF. There are no rules (i.e., no nodes in the network except the input/output linguistic nodes) in the RAFF initially. They are created dynamically as learning proceeds upon receiving incoming training data by performing the following learning processes simultaneously:

a) input/output space partitioning;
b) construction of fuzzy rules;
c) feedback structure identification;
d) parameter identification.

In the above, processes a), b), and c) belong to the structure learning phase and process d) belongs to the parameter learning phase. The details of these learning processes are described in the rest of this section.

1) Input/Output Space Partitioning: The way the input space is partitioned determines the number of rules. Even though the precondition part of a rule in the RAFF includes the external inputs, which represent the spatial information and the internal variable values which represent the temporal information, only the spatial information is used for clustering due to its local mapping property.

Geometrically, a rule corresponds to a cluster in the input space with $\mathbf{m}_i$ and D_i representing the center and variance of that cluster. For each incoming pattern $\mathbf{x}$, the strength a rule is fired can be interpreted as the degree the incoming pattern belongs to the corresponding cluster. For computational efficiency, we can use the spatial firing strength component in (6) directly as this degree measure

$$F^i(\mathbf{x}) = \prod_i u_i^{(3)} = e^{-[D_i(\mathbf{x}-\mathbf{m}_i)]^T[D_i(\mathbf{x}-\mathbf{m}_i)]} \tag{13}$$

where $F^i \in [0,1]$. In the above equation, the term $[D_i(\mathbf{x}-\mathbf{m}_i)]^T[D_i(\mathbf{x}-\mathbf{m}_i)]$ is, in fact, the distance between $\mathbf{x}$ and the center of cluster i. Using this measure, we can obtain the following criterion for the generation of a new fuzzy rule. Let $\mathbf{x}(t)$ be the newly incoming pattern. Find

$$J = \arg \max_{1\le j\le c(t)} F^j(\mathbf{x}) \tag{14}$$

where $c(t)$ is the number of existing rules at time t. If $F^J \le \bar{F}_{\text{in}}(t)$, then a new rule is generated, where $\bar{F}_{\text{in}}(t) \in (0,1)$ is a prespecified threshold that decays during the learning process. Once a new rule is generated, the next step is to assign initial centers and widths of the corresponding membership functions. Since our goal is to minimize an objective function and the centers and widths are all adjustable later in the parameter learning phase, it is of little sense to spend much time on the assignment of the centers and widths for finding a perfect cluster. Hence, we can simply set

$$\mathbf{m}_{(c(t)+1)} = \mathbf{x}, \tag{15}$$

$$D_{(c(t)+1)} = \frac{-1}{\beta}\cdot \operatorname{diag}(1/\ln(F^J)\cdots 1/\ln(F^J)) \tag{16}$$

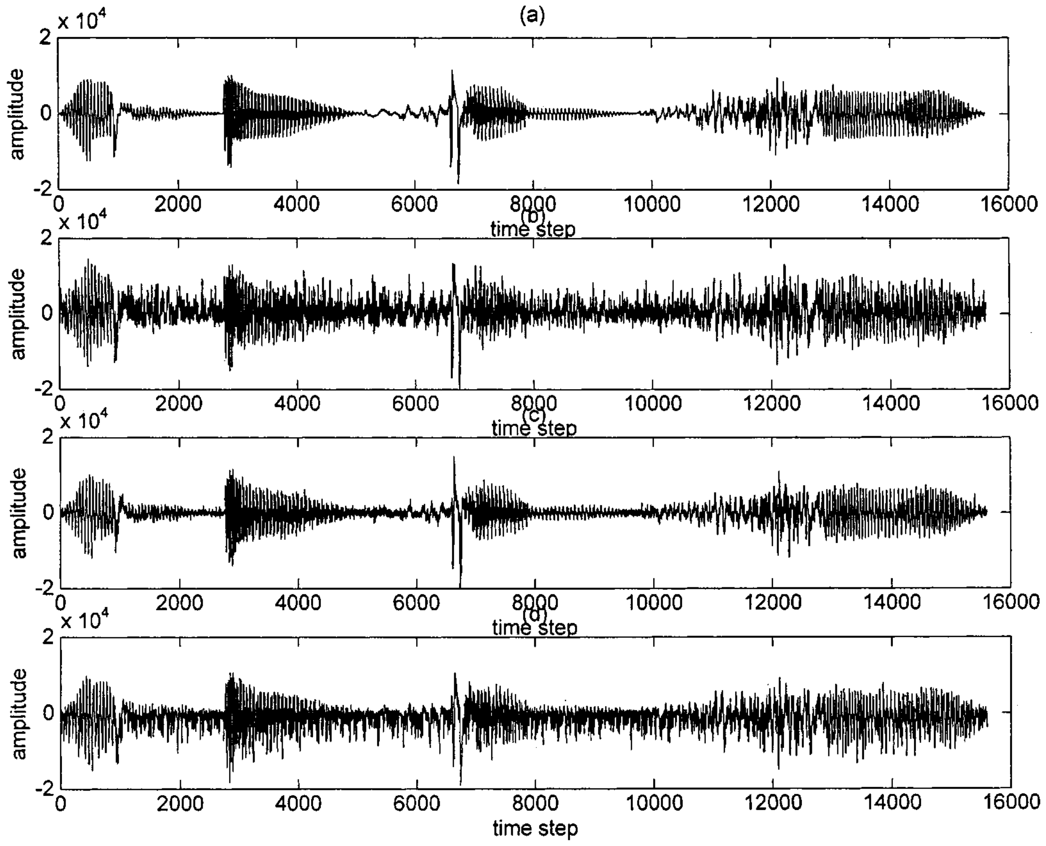


Fig. 4. (a) The original speech signal $s(k)$. (b) The corrupted signal $x(k)$. (c) The recovered signal $e(k)$ by the RAFF. (d) The recovered signal by Elman's recurrent neural network.

according to the first-nearest-neighbor heuristic [28], where $\beta \geq 0$ decides the overlap degree between two clusters. Similar methods are used in [29], [30] for the allocation of a new radial basis unit. However, in [29], the degree measure does not take the width D into consideration. In [30], the width of each unit is kept at a prespecified constant value, so the allocation result is, in fact, the same as that in [29]. In the RAFF, the width is taken into account in the degree measure, so a cluster with larger width (meaning a larger region is covered), will generate fewer rules in its vicinity than a cluster with smaller width. This is a more reasonable result. Another disadvantage of [29] is that another degree measure, the Euclid distance, is required, which increases the computation load.

After a rule is generated, the next step is to decompose the multidimensional membership function formed in (15) and (16) to the corresponding 1-D membership functions for each input variable. For the Gaussian membership function used in the RAFF, the task can be easily done as

$$e^{-[D_i(\mathbf{x}-\mathbf{m}_i)]^T[D_i(\mathbf{x}-\mathbf{m}_i)]} = \prod_j e^{-\frac{(x_j-m_{ij})^2}{\sigma_{ij}^2}} \tag{17}$$

where m_{ij} and σ_{ij} are, respectively, the projected center and width of the membership function in each dimension.

Let $\mu(m_i, \sigma_i)$ represent the Gaussian membership function with center m_i and width σ_i. The whole algorithm for the generation of new fuzzy rules as well as fuzzy sets for each input variable is as follows. Suppose no rules are existent initially.

```
IF $\mathbf{x}$ is the first incoming pattern THEN do
PART 1. { Generate a new rule
    with center $\mathbf{m}_1 = \mathbf{x}$, width $D_1 = \mathrm{diag}(1/\sigma_{\mathrm{init}}, \ldots, 1/\sigma_{\mathrm{init}})$,
    where $\sigma_{\mathrm{init}}$ is a prespecified constant.
  After decomposition, we have $n$ 1-D membership functions,
    with $m_{1i} = x_i$ and $\sigma_{1i} = \sigma_{\mathrm{init}}, i = 1 \ldots n$.
  }
ELSE for each newly incoming pattern $\mathbf{x}$, do
PART 2. { find $J = \arg\max_{1 \leq j \leq c(t)} F^j(\mathbf{x})$,
    IF $F^J \geq \bar{F}_{\mathrm{in}}(t)$
      do nothing
    ELSE
    { $c(t+1) = c(t) + 1$,
    generate a new fuzzy rule, with
      $\mathbf{m}_{c(t+1)} = \mathbf{x}$, $D_{c(t+1)} = -(1/\beta) \cdot \mathrm{diag}(1/\ln(F^J) \cdots 1/\ln(F^J))$.
    After decomposition, we have
      $m_{\mathrm{new}-i} = x_i$, $\sigma_{\mathrm{new}-i} = -\beta \cdot \ln(F^J), i = 1 \ldots n$.
```

2) Construction of Fuzzy Rules: As mentioned in learning process A, the generation of a new input cluster corresponds to the generation of a new fuzzy rule, with its precondition part constructed by the learning algorithm in learning process A and the feedback structure identification scheme to be described below in learning process C. At the same time, we have to decide the consequent part of the generated rule. One simple way is to assign a singleton value to each rule [22]. The drawback of this way is that the number of output singletons will increase in parallel with the number of fuzzy rules. In fact, we may find that some singletons are very close and can be assigned with the same value. The clustering concept is adopted in this process to achieve the purpose. As described in Section V, layer 5, only the center of each output cluster is used during the defuzzification process, and the width of each output cluster is for clustering only. Advantage of this approach is that we can reduce the number of output singletons and avoid the need of keeping other parameters such as the widths of the output clusters. Suppose a new input cluster is formed after the presentation of the current input/output training pair $(\mathbf{x}, \mathbf{d})$. The consequent part is constructed by the following algorithm:

```
IF there are no output clusters,
  do { PART 1 in Process A, with $\mathbf{x}$ replaced by $\mathbf{d}$ }
ELSE
  do {
    find $J = \arg\max_j F^j(\mathbf{x})$,
    IF $F^J \geq \bar{F}_{\text{out}}(t)$
      connect input cluster $c(t+1)$ to the existing output cluster $J$
    ELSE
      generate a new output cluster,
      connect input cluster $c(t+1)$ to the newly generated output cluster.
  }.
```

3) *Feedback Structure Identification:* In learning process B, the number of generated clusters in the consequent part is affected by the problem to be solved. The number of output clusters is large for complex problems and is small for simple ones. Naturally, in the feedback layer, more internal variables are required for more complex problems. Knowing this relationship, for simplicity, we can simply set the number of internal variables equal to the number of output clusters in the consequent part (i.e., the number of output term nodes in layer 4). Hence, during the on-line learning, an internal variable is generated once a output cluster is generated. The fan-in of the context node comes from all the nodes in layer 4, with the link weight assigned with a small random value initially. After an internal variable is generated (meaning a context node is created), the next step is to decide its effect on each rule node. As mentioned in Section II, only a global membership function is assigned to each internal variable and acts as the feedback term node of the corresponding context node. Of course, we can cover the universe of discourse of the internal variable by some local membership functions, but this makes the network structure become larger and complexity increase seriously. When the firing degree of each internal variable to its corresponding membership function is calculated, we should next decide which rules the firing degree is acted on. In other words, we should decide which rule nodes in layer 3 a generated feedback term node should connect to. In general, each rule has its own corresponding internal variable, which is to memorize the history of the rule. But for the rules that have the same consequent part (i.e., connect to the same output term node), the same internal variable is assigned to these rules. With this way, we can effectively reduce the parameter number in the feedback layer.

4) *Parameter Identification:* After the filter structure is adjusted according to the current training pattern, the filter then enters the parameter identification phase to adjust the parameters of the filter based on the same training pattern. Notice that the following parameter learning is performed on the whole filter after structure learning, no matter whether the nodes (links) are newly added or are existent originally. Since the RAFF is a dynamic filter with feedback connections, the real-time recurrent learning (RTRL) algorithm [32] is used. Considering the single-output case for clarity, our goal is to minimize the error function

$$E(t+1) = \frac{1}{2}\left(y_j(t+1) - y_j^d(t+1)\right)^2 \tag{18}$$

where $y_j^d(t+1)$ is the desired output and $y_j(t+1)$ is the current output. For each training pattern, starting at the input nodes, a forward pass is used to compute the activity levels of all the nodes in the network to obtain the current output $y_j(t+1)$. In the followings, for the notation clarity, dependency on time t will be omitted unless emphasis on temporal relationships is required. With the error function defined in (18), we can derive the update rule of $\hat{m}_{ji}$ as

$$\hat{m}_{ji}(t+1) = \hat{m}_{ji}(t) - \Delta\hat{m}_{ji}(t+1) \tag{19}$$

where

$$\Delta\hat{m}_{ji}(t+1) = \eta\left(y_j(t+1) - y_j^d(t+1)\right)\frac{u_i^{(5)}}{\sum_i u_i^{(5)}}. \tag{20}$$

The update value of m_{pq}, the center of the membership function in the precondition part, is

$$\Delta m_{pq}(t+1) = \eta\left(y_j(t+1) - y_j^d(t+1)\right) \times \sum_k \frac{\hat{m}_{jk} - y_j(t+1)}{\sum_k a_k^{(3)}(t)}\frac{\partial a_k^{(3)}}{\partial m_{pq}}(t) \tag{21}$$

where

$$\frac{\partial a_k^{(3)}}{\partial m_{pq}}(t) = \frac{\partial a_k^{(h)}}{\partial m_{pq}} \cdot \mu_k + a_k^{(h)} \cdot \mu_k \cdot 2\frac{x_p(t) - m_{pk}}{\sigma_{pk}^2} \tag{22}$$

and $\mu_k = \exp\{-\sum_i((x_i(t) - m_{ik}/\sigma_{ik}))^2\}$.

The partial derivative $\partial a_k^{(h)}/\partial m_{pq}$ is calculated as

$$\begin{aligned}\frac{\partial a_k^{(h)}}{\partial m_{pq}}(t) &= a_k^{(h)}(t)\left(1 - a_k^{(h)}(t)\right)\left\{\sum_\ell w_{k\ell}\right. \\ &\quad \times \left[\frac{\partial a_\ell^{(h)}}{\partial m_{pq}}(t-1)\mu_\ell(t-1) + \mu_\ell(t-1)a_\ell^{(h)}(t-1)\right. \\ &\quad \left.\left.\cdot\ 2\frac{x_p(t-1) - m_{pq}}{\sigma_{p\ell}^2}\cdot\hat{\delta}\right]\right\}\end{aligned} \tag{23}$$

where

$$\hat{\delta} = \begin{cases} 1, & \text{if rule } q \text{ is connected to node } \ell \text{ in layer 4} \\ 0, & \text{otherwise.}\end{cases}$$

Similarly, the update value of σ_{pq} is

$$\Delta\sigma_{pq}(t+1) = \eta\left(y_j(t+1) - y_j^d(t+1)\right) \times \sum_k \frac{\partial y_j(t+1)}{\partial a_k^{(3)}(t)} \frac{\partial a_k^{(3)}}{\partial \sigma_{pq}}(t). \tag{24}$$

The partial derivative $\partial a_k^{(3)}/\partial\sigma_{pq}$ is

$$\frac{\partial a_k^{(3)}}{\partial\sigma_{pq}} = \frac{\partial a_k^{(h)}}{\partial\sigma_{pq}} \cdot \mu_k + a_k^{(h)} \cdot \mu_k \cdot 2\frac{(x_p(t) - m_{pk})^2}{\sigma_{pk}^3} \tag{25}$$

where the partial derivative $\partial a_k^{(h)}/\partial\sigma_{pq}$ is

$$\frac{\partial a_k^{(h)}}{\partial\sigma_{pq}}(t) = a_k^{(h)}(t)\left(1 - a_k^{(h)}(t)\right)\left\{\sum_\ell w_{k\ell} \times \left[\frac{\partial a_\ell^{(h)}}{\partial m_{pq}}(t-1)\mu_\ell(t-1) + \mu_\ell(t-1)a_\ell^{(h)}(t-1) \cdot 2\frac{(x_p(t-1) - m_{pq})^2}{\sigma_{p\ell}^3} \cdot \hat{\delta}\right]\right\}. \tag{26}$$

The update value of w_{pq} is

$$\Delta w_{pq}(t+1) = \eta_w\left(y_j(t+1) - y_j^d(t+1)\right) \times \sum_k \frac{\partial y_j(t+1)}{\partial a_k^{(3)}(t)} \frac{\partial a_k^{(3)}}{\partial w_{pq}}(t) \tag{27}$$

and

$$\frac{\partial a_k^{(3)}}{\partial w_{pq}} = a_k^{(h)}\left(1 - a_k^{(h)}\right)\frac{\partial h_k}{\partial w_{pq}} \cdot \mu_k \tag{28}$$

where

$$\frac{\partial h_k}{\partial w_{pq}}(t) = a_q^{(4)}(t-1)\delta_{kp} + \sum_\ell w_{k\ell}\frac{\partial a_\ell^{(h)}}{\partial w_{pq}}(t-1)\mu_\ell(t-1). \tag{29}$$

Hence, we have the following recursive form

$$\frac{\partial a_k^{(h)}}{\partial w_{pq}}(t) = a_k^{(h)}(t)\left(1 - a_k^{(h)}(t)\right)a_q^{(4)}(t-1)\delta_{kp} + \sum_\ell w_{k\ell}\frac{\partial a_\ell^{(h)}}{\partial w_{pq}}(t-1)\mu_\ell(t-1). \tag{30}$$

The values $\partial a^{(h)}/\partial m$, $\partial a^{(h)}/\partial\sigma$, and $\partial a^{(h)}/\partial w$ are equal to zero initially and are reset to zero after a period time to avoid the accumulation of too far away errors. Note that two step-size parameters are used in the above equations: η_w for the tuning of memory weight w and η for the remaining parameters. Except the weight parameter w, which is assigned randomly initially, the other parameters all have good initial values assigned during the structure learning. To increase the learning speed of temporal relationship, we may set the learning constant η_w several times larger than η.

The learning algorithm derived above is used in the following examples. Notice that according to the ordered derivative [31], we can also obtain the same parameter learning rules for the RAFF. For some complex problems, where the learned network size is large, this algorithm may not be efficient for real-time application. For this, we can simplify the above learning algorithm by treating the feedback values from layer 4 as outside inputs and the modifiable connections can be regarded to be forward. With this treatment, the parameters can be trained by the conventional backpropagation learning method, and the learning algorithm derived above can be modified as follows. The term $\partial a_k^{(3)}(t)/\partial m_{pq}$ in (21) is simplified as

$$\frac{\partial a_k^{(3)}(t)}{\partial m_{pq}} = a_k^{(h)} \cdot \mu_k \cdot 2\frac{x_p(t) - m_{pk}}{\sigma_{pk}^2}. \tag{31}$$

The term $\partial a_k^{(3)}(t)/\partial\sigma_{pq}$ in (25) is simplified as

$$\frac{\partial a_k^{(3)}}{\partial\sigma_{pq}} = a_k^{(h)} \cdot \mu_k \cdot 2\frac{(x_p(t) - m_{pk})^2}{\sigma_{pk}^3}. \tag{32}$$

The term $\partial h_k(t)/\partial w_{pq}$ in (29) is simplified as

$$\frac{\partial h_k}{\partial w_{pq}}(t) = a_q^{(4)}(t-1)\delta_{kp}. \tag{33}$$

It should be noted that even the RAFF trained by this simplified algorithm might have lower output accuracy, the overall performance of the RAFF is still admirable. This will be verified in the noisy speech recognition problem in the next section. Of course, other existing on-line learning algorithms [33] for tuning the weights of recurrent neural networks can be possibly adopted for tuning the RAFF, too.

V. Experiments

The proposed RAFF is applied to the adaptive noise cancellation and noisy speech recognition problems using the technique introduced in Section II.

A. Adaptive Noise Cancellation

Example 1: The speech to be recovered is a sequence of Mandarin digits and the noise signals are from the NOISEX-92 database [34]. Assume that the relation between noise source $r(k)$ and corrupting noise $n(k)$ is a dynamic nonlinear function

$$n(k) = 0.24r(k-1)n(k-2) + 0.1n(k-2) + 0.25n(k-1) - 0.6r(k-2) + 0.9r(k-1)^2 + 0.9r(k-1) + 1.5r(k). \tag{34}$$

Suppose the noise source r is fed to the filter input directly. The adaptive filter is implemented by the RAFF. Only the currently received noise signal $r(k)$ is used as the input to the RAFF. The noise signal used is the noise on the floor of a car factory. A word

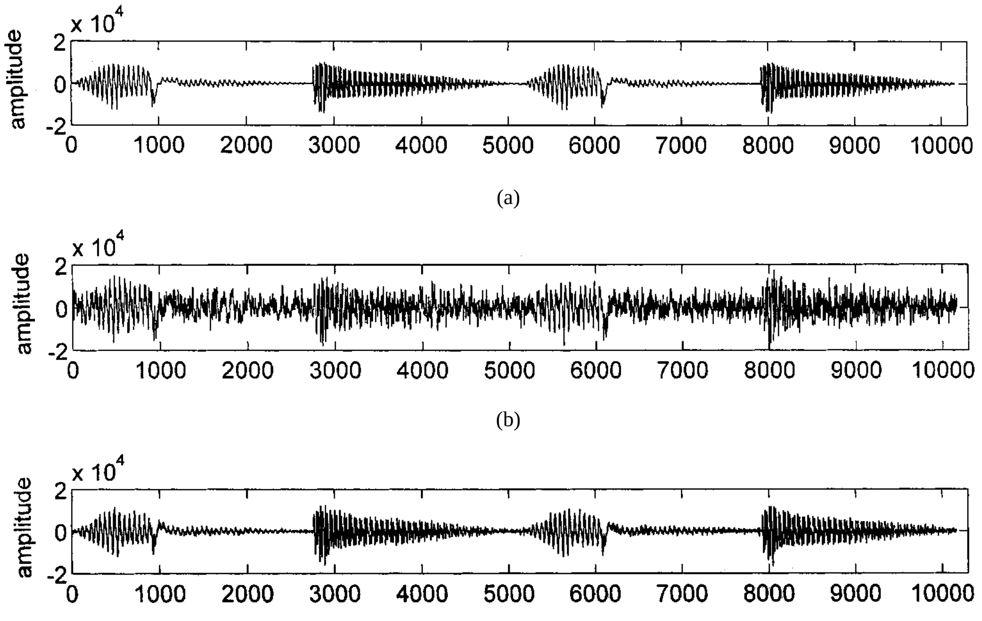


Fig. 5. (a) The original speech signal $s(k)$. (b) The corrupted signal $x(k)$, where the corrupting noise in the first 5100 times steps is the factory noise and that in the succeeding time steps is the F-16 noise. (c) The recovered signal $e(k)$ by the RAFF.

utterance, the digit "0" is recorded and training is performed on this word in advance. The initial parameters of the RAFF are set as $\bar{F}_{\text{in}} = 0.35$, $\bar{F}_{\text{out}} = 0.8$, $\eta = 0.03$, and $\eta_w = 6\eta$. Ten epoches of training are performed, and four input clusters (rules), and four output clusters and internal variables are generated. Afterwards, other speech signals are spoken, and the RAFF is on-line tuned to recover the speech signal. The original speech signal $s(k)$ is shown in Fig. 4(a). The measured noisy speech signal $x(k)$ is shown in Fig. 4(b), where $\text{SNR} \approx -1$ dB. The recovered signal during on-line filtering is shown in Fig. 4(c), where $\text{SNR} \approx 8$ dB. The obtained dynamic fuzzy rules after on-line learning are

Rule 1:IF $r(k)$ is $\mu(-0.74, 0.57)$ and $h_1(k)$ is G
THEN $y(k)$ is $\mu(-0.34, 0.3)$ and
$h_1(k+1)$ is -0.06 and $h_2(k+1)$ is -0.09 and
$h_3(k+1)$ is 0.82 and $h_4(k+1)$ is 0.04.

Rule 2:IF $r(k)$ is $\mu(0.43, 0.30)$ and $h_2(k)$ is G
THEN $y(k)$ is $\mu(0.22, 0.17)$ and
$h_1(k+1)$ is -0.75 and $h_2(k+1)$ is -0.58 and
$h_3(k+1)$ is -0.19 and $h_4(k+1)$ is 0.78.

Rule 3:IF $r(k)$ is $\mu(0.70, 0.21)$ and $h_3(k)$ is G
THEN $y(k)$ is $\mu(0.69, 0.29)$ and
$h_1(k+1)$ is -0.38 and $h_2(k+1)$ is -0.51 and
$h_3(k+1)$ is -0.09 and $h_4(k+1)$ is -0.77.

Rule 4:IF $r(k)$ is $\mu(0.15, 0.56)$ and $h_4(k)$ is G
THEN $y(k)$ is $\mu(0.38, 0.15)$ and
$h_1(k+1)$ is 0.52 and $h_2(k+1)$ is 0.01 and
$h_3(k+1)$ is -0.01 and $h_4(k+1)$ is -0.88.

In the above rules, h_1, h_2, h_3, and h_4 are generated internal variables, $\mu(m_i, \sigma_i)$ represents a Gaussian membership function with center m_i and width σ_i, and G is the global membership function stated previously in Section III. The number of parameters in the learned RAFF is 28.

To see how good the performance of the RAFF is, another recurrent neural network, Elman's network [35], is used for comparison. In this network, the recurrence is obtained by feeding nodes in the hidden layer back to the input layer, functioning as context nodes. So the input layer contains two parts: the true input nodes and the context nodes. Similar structure is also proposed in [36]. Like the training of RAFF, the input training data to the network is $r(k)$ and the desired output is $n(k)$. First, with nearly the same number of network parameters (five hidden nodes, 35 parameters in total) and ten epoches of training as used in training the RAFF, an SNR value of 2.18 of the recovered speech signal is obtained with learning constant value of 0.45. Next, with more network parameters (ten hidden nodes, 120 parameters in total) and 100 training epoches, an SNR value of 2.64 of the recovered speech signal is obtained. This performance is obviously worse than that of the proposed RAFF. The recovered speech sample by the Elman's network is shown in Fig. 4(d).

In the above simulation, the noise source is kept the same during filtering. To see the noise elimination performance for different type of noise during on-line filtering of the RAFF, the F-16 cockpit noise is added for comparison. In Fig. 5(b), the first 5100 speech samples are corrupted by the factory noise, while the succeeding 5100 samples are corrupted by the F-16 noise during on-line filtering. The original speech signal is shown in Fig. 5(a), where the two segments of clean speech corrupted by these two noise sources are the same for comparison purpose. The on-line filtering result is shown in Fig. 5(c). For these two kinds of noise sources, both achieve an enhancement of about 9

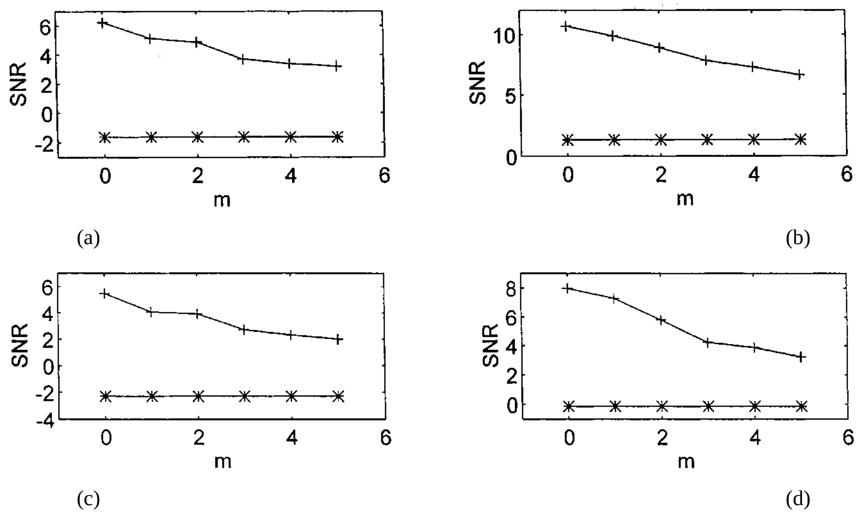


Fig. 6. The original SNR value ("$*$") and the enhanced SNR value ("$+$") using the RAFF for different values of m and different Mandarin digits. (a) "1." (b) "2." (c) "3." (d) "4."

dB, meaning that the cancellation is not affected by the difference of noise sources.

Example 2: In this example, we shall discuss the problem when longer delay exists. Like the function used in Example 1, the dynamic nonlinear function used is

$$\begin{aligned} n(k) = {} & 0.24r(k-1)n(k-2) + 0.1n(k-2) \\ & + 0.25n(k-1) - 0.6r(k-m-2) \\ & + 0.9r(k-m-1)^2 + 0.9r(k-m-1) \\ & + 1.5r(k-m) \end{aligned} \tag{35}$$

where m denotes the delay samples. For example, the value $m = 5$ means that the currently measured noisy speech signal x is affected by the measured noise signal r at least five samples ahead. Different values of m, including $m = 0, \ldots, 5$, are tested. Only one input $r(k)$ is used as the input for the RAFF. For different values of m, the training method is the same as that in Example 1, i.e., about ten epoches of training are performed on the digit "0" and then on-line training and filtering is performed for other words. For different values of m, the filter size is kept the same (four dynamic rules with four output clusters and internal variables) for comparison. The enhanced SNR values with different values of m for different digits (including "1," "2," "3," and "4") are shown in Fig. 6. For comparison, two types of filters are simulated. First, a feedforward filter, the fuzzy adaptive filter [22], with ten rules is applied to the same problem with $m = 4$ in (35). Different number of input values, including $[r(k), r(k-1)]$, $[r(k), r(k-1), r(k-2)]$, and $[r(k), r(k-1), r(k-2), r(k-3)]$, are used as inputs to the fuzzy adaptive filter for comparison. The enhanced SNR by the RAFF with only one input $r(k)$, and that by the fuzzy adaptive filter with different orders of input are shown in Fig. 7. Even with only one input and a smaller filter size, the RAFF performs better than the feedforward fuzzy adaptive filter. By further increasing the input dimension of the fuzzy adaptive filter to cover more delays of the input variables, we can obtain a better result. The resulting network size is, however, quite large and this is not an economic approach. Next, like Example 1, the Elman's recurrent network with ten hidden nods and 100 training epoches is used again for performance comparison in this example. The enhanced result is also shown in Fig. 7 for comparison. This comparison shows the RAFF still outperforms the Elman's network for different Mandarin digits.

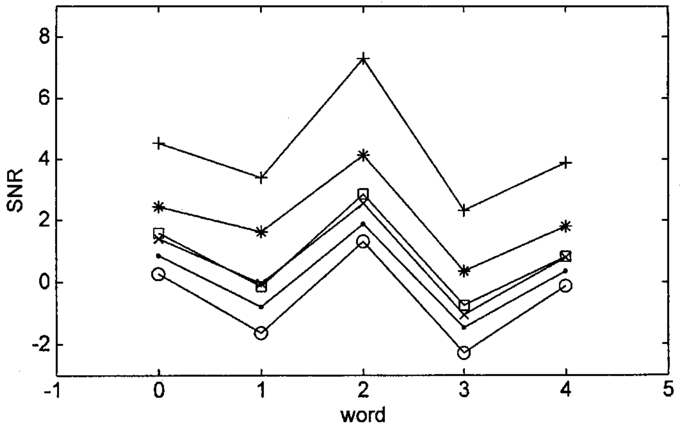


Fig. 7. The original SNR value ("o") and the enhanced SNR value by the RAFF ("$+$"), Elman's network ("□") and the fuzzy adaptive filter with input $[r(k), r(k-1)]$ ("$\cdot$"), $[r(k), r(k-1), r(k-2)]$ ("$\times$"), and $[r(k), r(k-1), r(k-2), r(k-3)]$ ("$*$") at $m = 4$ for different Mandarin digits ("1," "2," "3," and "4").

B. Noisy Speech Recognition

In this subsection, we use the RAFF as a noise reduction filter based upon the architecture in Fig. 2. The database contains ten isolated Mandarin digits "0," . . ., "9." The problems of speaker-dependent and speaker-independent recognition are both tested in the following Examples 1 and 2, respectively.

Example 1—Speaker-Dependent Recognition: The ten digits were spoken by the same speaker, with 20 noise-free repetitions for each word. For these 20 repetitions, ten are used for training, the other ten are used for testing. The time delay neural network (TDNN) is used as a recognizer. The features extracted are cepstral coefficients with order 12 for each frame and 20

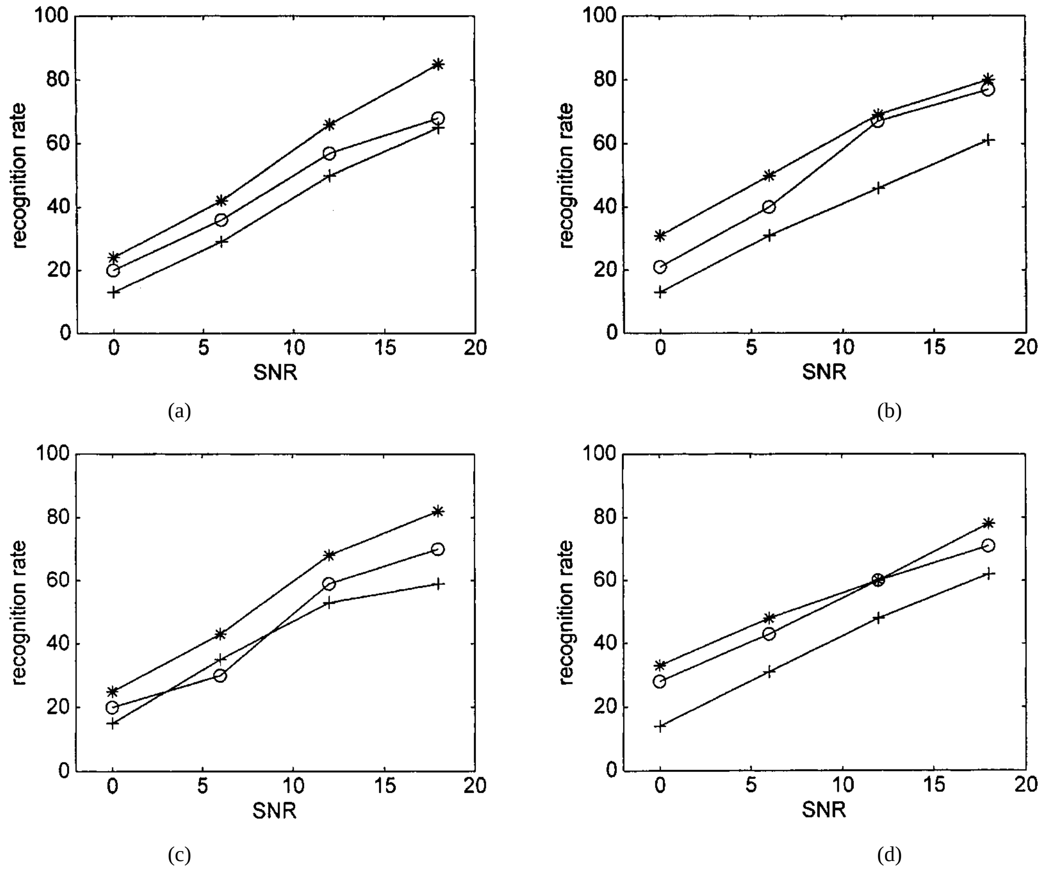


Fig. 8. (a) The recognition rates on testing data for different SNR values by using the RAFF as noise reduction filter ("$*$"), Jordan' recurrent neural network ("$\circ$") and, without using noise reduction filter ("$+$") under different noisy environments. (a) Factory noise. (b) Operation room noise. (c) Buccaneer jet noise. (d) F-16 noise.

constant frames are used for each word to meet the requirement of TDNN. The noise signals are from the NOISE-92 database [34]. Different types of noise including (a) the noise on the floor of a car factory, (b) destroyer operation room noise, (c) buccaneer jet noise, and (d) F-16 cockpit noise are tested. The noisy speech is generated by artificially adding these noise signals to clean speech under a specific SNR value.

Without the noise reduction network, the recognition rate under clean environment is 99%. In training the RAFF as a noise reduction network, the 12 noisy cepstral features of each frame are used as the inputs and the corresponding 12 noise-free cepstral features as the desired outputs. All the 100 words in the training set are used for training under $\mathrm{SNR} = 18$. Each kind of noise has its own corresponding RAFF and the four RAFF's for the four noise types are trained independently. The parameters used for each RAFF are $\bar{F}_{\text{in}} \approx 0.3$, $\bar{F}_{\text{out}} \approx 0.7$, $\eta = 0.005$, $\eta_w = 9\eta$, and 50 epoches of training are performed. The number of rules generated for the four RAFF's is about 22 and that of the output clusters and internal variables are about 17. In total, 1021 parameters are used in the RAFF. The noise reduction effects by the RAFF are tested by the testing database for different SNR values. The original and improved recognition rates for different types of noise at different SNR values are shown in Fig. 8. The results show that higher recognition rates are achieved through the use of the RAFF.

The simplified backpropagation-type learning algorithm derived at the end of Section IV is also applied to train the RAFF in this example. With the same training and testing data and network learning parameters, the obtained recognition results are listed in Table I. Compared with the result in Fig. 8, the simplified learning algorithm reduces only about 3.5% of the recognition rate of the original RAFF in average.

For performance comparison, the Jordan's recurrent neural network [37], [38] is applied to the same problem in this example. Unlike the Elman's recurrent network used in Section V-A, which feeds its hidden-node values back to the input layer, the Jordan's recurrent network feeds its output-node values back to the input layer to form the context nodes. A total of 30 hidden nodes is used in the Jordan's network (1080 parameters in total) and 50 epoches of training are performed. The training method is the same as that of the RAFF with a learning constant value of 0.45 and the strength of self-connections is 0.3. The obtained recognition rates are also shown in Fig. 8 for different types of noise. Fig. 8 obviously shows that the RAFF achieves a higher recognition rate than the Jordan's network.

To see the performance of RAFF on unknown types of noise, the RAFF trained at $\mathrm{SNR} = 18$ for factory noise is tested on the other three types of noise at different SNR values. The resulting recognition rates are listed in Table II. From Table II, we see that even for unknown types of noise, the performance of RAFF is almost the same with that of the one trained and tested on the same noise type shown in Fig. 8; in other words, the performance of RAFF is almost independent of the noise types.

TABLE I
RECOGNITION RATES WITH RAFF TRAINED BY SIMPLIFIED LEARNING ALGORITHM AND WITHOUT RAFF (ORIGINAL) FOR DIFFERENT TYPES OF NOISE AT DIFFERENT SNR VALUES

	Noise Type	Factory	Operation Room	Buccaneer Jet	F - 16
SNR =18	Original	65	61	59	62
	+ RAFF	83	79	77	78
SNR = 12	Original	50	46	53	48
	+ RAFF	67	64	68	72
SNR = 06	Original	29	31	35	31
	+ RAFF	48	44	27	44
SNR = 00	Original	13	13	15	14
	+ RAFF	20	35	23	31

TABLE II
THE RECOGNITION RATES FOR DIFFERENT TYPES OF NOISE WITH THE RAFF TRAINED BY FACTORY NOISE AT $\mathrm{SNR} = 18$

	Noise Type	Factory	Operation Room	Buccaneer Jet	F - 16
SNR =18	Original	65	61	59	62
	+ RAFF	85	85	81	84
SNR = 12	Original	50	46	53	48
	+ RAFF	66	71	59	71
SNR = 06	Original	29	31	35	31
	+ RAFF	42	47	37	43
SNR = 00	Original	13	13	15	14
	+ RAFF	24	27	29	30

Example 2—Speaker-Independent Recognition: The database contains ten isolated Mandarin digits "0," ..., "9." They were spoken by ten different speakers. The training data were spoken by five speakers among the ten, with two noise-free repetitions for each word. The testing data contains two sets. One set is from the original five training speakers with another two noise-free spoken repetitions for each word. The other set is from the other five speakers, with two spoken repetitions for each word. The recognizer and speech features are the same as those in Example 1. The training is performed under $\mathrm{SNR} = 18$ for the factory noise. With the simplified training algorithm and after 70 epoches of training, 27 dynamic fuzzy rules are generated in the RAFF. The recognition rates with and without the RAFF for the aforementioned two different testing data sets under different SNR values of factory noise are shown in Table III. Table III shows that for $\mathrm{SNR} \geq 6$ and, with the RAFF, a recognition rate greater than 70% is achieved for both testing data sets.

From the above two examples, we find that even for unknown types of noise or multispeakers, the improved recognition rates by the RAFF are admirable. Our experimental results also show that the RAFF requires less tuning parameters to achieve the same performance of its compared counter parts. Hence, even the parameter learning method for the RAFF is more complex, its smaller network-size property can compensate this problem and makes the real-time application of the RAFF possible. For a very complex learning problem, where the learned network size is large, we have also derived a new simplified parameter learning method based on the feedforward backpropagation learning algorithm to meet the real-time application. These properties make it more appropriate for real-world applications. One real-world application of the illustrated ten-digit word recognition problem is the vocal phone dialing in noisy environment. Besides, in the above examples, a simple recognizer, the TDNN, is used as the speech recognition kernel to recognize the speech signals after RAFF filtering. In fact, in the real-world speech recognition applications, the RAFF may be combined with other types of recognizers, like the hidden Markov model (HMM), to increase the recognition rate of the recognition system in noisy environment. Furthermore, other robust speech recognition techniques [39], such as the robust feature extraction scheme, can be integrated with the RAFF to further improve the recognition system's performance in noisy environment.

TABLE III
THE RECOGNITION RATES FOR TWO SETS OF TEST DATA, WHERE SET 1 IS THE DATA FROM THE SAME TRAINING SPEAKERS AND SET 2 IS FROM OTHER SPEAKERS

Data Type		SNR=00	SNR=06	SNR=12	SNR=18
Set 1	Original	40	60	83	85
	+ RAFF	54	74	86	87
Set 2	Original	35	59	70	76
	+ RAFF	42	70	74	80

VI. CONCLUSION

Two noisy speech processing techniques, the adaptive noise cancellation for speech enhancement and the noise reduction filter for noisy speech recognition based upon filtering approach are addressed and a novel filter, the RAFF, is proposed. The RAFF owns on-line self-organizing learning ability and is constructed by expanding the general feedforward adaptive neural fuzzy filter to a recurrent one. Using the RAFF, we need not know the exact order of the inputs nor do we have to determine the size of the RAFF in advance. The RAFF can handle these problems by creating and updating recursive fuzzy rules automatically via on-line structure and parameter learning. The RAFF has shown its efficiencies for some noisy speech processing problems. Further works on RAFF include its real application to ANC and extension of the speaker independent recognition problem to including more speakers, e.g., more than 100 speakers. Applications of the RAFF for other temporal filtering problems will also be investigated.

REFERENCES

[1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[2] M. J. Coker and D. J. Simkins, "A nonlinear adaptive noise canceler," in *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Denver, CO, Apr. 1980, pp. 470–473.
[3] W. G. Knecht, M. E. Schenkel, and G. S. Moschytz, "Neural network filters for speech enhancement," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 433–438, Nov. 1995.
[4] C. T. Lin and C. F. Juang, "An adaptive neural fuzzy filter and its applications," *IEEE Trans. Syst., Man, Cybern.*, vol. 27, pp. 635–656, Aug. 1997.
[5] A. Varga and H. J. M. Steeneken, "Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Commun. 12*, pp. 247–251, 1993.

[6] Y. Gong, "Speech recognition in noisy environments: A survey," *Speech Commun.*, vol. 16, pp. 261–291, 1995.

[7] M. Bellanger, *Adaptive Digital Filters and Signal Analysis*. New York: Marcel-Dekker, 1987.

[8] J. H. Lin, T. M. Sellke, and E. J. Coyle, "Adaptive stack filtering under the mean absolute error criterion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 934–954, June 1990.

[9] L. Yin, J. Astola, and Y. Neuvo, "A new class of nonlinear filters—Neural filters," *IEEE Trans. Signal Processing*, vol. 41, pp. 1201–1222, Mar. 1993.

[10] W. G. Knecht, M. E. Schenkel, and G. S. Moschytz, "Neural network filters for speech enhancement," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 433–438, Nov. 1995.

[11] S. A. Billings and C. F. Fung, "Recurrent radial basis function networks for adaptive noise cancellation," *Neural Networks*, vol. 8, no. 2, pp. 273–290, 1995.

[12] G. J. Gibson, S. Siu, and C. F. N. Cowan, "The application of nonlinear structures to the reconstruction of binary signals," *IEEE Trans. Signal Processing*, vol. 39, pp. 1877–1884, Aug. 1991.

[13] K. A. Al-Mashouq and I. S. Reed, "The use of neural nets to combine equalization with decoding for severe intersymbol interference channels," *IEEE Trans. Neural Networks*, vol. 5, pp. 982–988, Nov. 1994.

[14] S. Tamura and A. Waibel, "Noise reduction using connectionist models," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, New York, Apr. 1988, pp. 553–556.

[15] S. Tamura, "An analysis of a noise reduction neural network," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, U.K., May 1989, pp. 2001–2004.

[16] H. B. D. Sorensen, "A cepstral noise reduction multi-layer neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, pp. 933–936.

[17] ——, "Speech recognition in noise using a self-structuring noise reduction model and hidden control models," in *Proc. IJCNN*, 1992, pp. 279–284.

[18] M. Trompf, "Neural network development for noise reduction in robust speech recognition," in *Proc. IJCNN*, vol. 4, 1992, pp. 722–727.

[19] S. Moon and T. N. Hwang, "Coordinated training of noise removing networks," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Minneapolis, MN, Apr. 1993, pp. 573–576.

[20] H. S. Lee and A. C. Tsoi, "Application of multilayer perceptron in estimating speech/noise characteristics for speech recognition in noisy environment," *Speech Commun.*, vol. 17, pp. 59–76, 1995.

[21] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems (With Disk)*. Englewood Cliffs, NJ: Prentice-Hall, May 1996, ch. 13.

[22] L. X. Wang and J. M. Mendel, "Fuzzy adaptive filters with application to nonlinear channel equalization," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 161–170, Aug. 1993.

[23] R. Yager and D. Filev, *Essentials of Fuzzy Modeling and Control*. New York: Wiley, 1994.

[24] V. Gorrini and H. Bersini, "Recurrent fuzzy systems," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Orlando, FL, June 1994, pp. 193–198.

[25] J. Zhang and A. J. Morris, "Recurrent neuro-fuzzy networks for nonlinear process modeling," *IEEE Trans. Neural Networks*, vol. 10, pp. 313–326, Mar. 1999.

[26] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Trans. Neural Networks*, vol. 10, pp. 828–845, July 1999.

[27] J. S. R. Jang and C. T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference system," *IEEE Trans. Neural Networks*, vol. 4, pp. 156–159, Jan. 1993.

[28] C. T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Trans. Comput.*, vol. 40, no. 12, pp. 1320–1336, 1991.

[29] J. Platt, "A resource allocating network for function interpolation," *Neural Comput.*, vol. 3, pp. 213–225, 1991.

[30] J. Nie and D. A. Linkens, "Learning control using fuzzified self-organizing radial basis function network," *IEEE Trans. Fuzzy Syst.*, vol. 40, pp. 280–287, Nov. 1993.

[31] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavior sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, Aug. 1974.

[32] R. J. Williams and D. Zipser, "A learning algorithm for continually running recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[33] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, pp. 1212–1228, Sept. 1995.

[34] A. Varga, H. J. M. Steeneken, M. Tomlinson, and D. Jones, "The NOISEX-92 study on the effect of additive noise automatic speech recognition," in *Description of RSG.10 and Esprit SAM Experiment and Database*, Malvern, U.K.: DRA Speech Res., 1992.

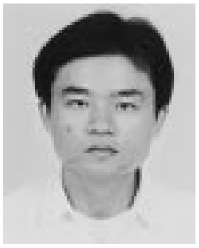
[35] J. L. Elman, "Finding structure in time," *Cognitive Sci.*, vol. 14, pp. 179–211, 1990.

[36] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. Berlin, Germany: Springer-Verlag, 1989.

[37] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Proc. 8th Annu. Conf. Cognitive Science Society*, Amherst, MA, Aug. 1986, pp. 531–546.

[38] ——, "Serial order: A parallel, distributed processing approach"," in *Advances in Connectionist Theory: Speech*. Hillsdale, NJ: Erlbaum, 1989.

[39] Y. Gong, "Speech recognition in noisy environments: A survey," *Speech Commun.*, vol. 16, pp. 261–291, 1995.



Chia-Feng Juang (M'00) received the B.S. and Ph.D. degrees in electrical engineering from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1993 and 1997, respectively.

He is currently an Assistant Professor of Department of Electrical Engineering, Chung-Chou Institute of Technology, Changhua, Taiwan. His current research interests are fuzzy systems, intelligent control, learning systems, noisy speech recognition, and digital signal processing.

Chin-Teng Lin (S'88–M'91–SM'99) received the B.S. degree in control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1986, and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively.

Since 1992, he has been with the College of Electrical Engineering and Computer Science, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., where he is currently a Professor of electrical and control engineering. He has also served as the Deputy Dean of the Research and Development Office of the National Chiao-Tung University since 1998. He is the coauthor of *Neural Fuzzy Systems—A Neuro-Fuzzy Synergism to Intelligent Systems* (Englewood Cliffs, NJ: Prentice-Hall, 1996) and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (Singapore: World Scientific, 1994). He has published approximately 50 journal papers in the areas of neural networks and fuzzy systems. His current research interests are fuzzy systems, neural networks, intelligent control, human-machine interface, and video and audio processing.

Dr. Lin is a member of Tau Beta Pi and Eta Kappa Nu. He is also a member of the IEEE Computer Society, IEEE Robotics and Automation Society, and the IEEE Systems, Man, and Cybernetics Society. He has been the executive council member of the Chinese Fuzzy System Association (CFSA) since 1995 and the supervisor of Chinese Automation Association since 1998. He was the vice chairman of IEEE Robotics and Automation Taipei chapter in 1996 and 1997. He won the Outstanding Research Award granted by National Science Council (NSC), Taiwan, in 1997 and 1999, the Outstanding Electrical Engineering Professor Award granted by the Chinese Institute of Electrical Engineering (CIEE) in 1997, and the Outstanding Engineering Professor Award granted by the Chinese Institute of Engineering in 2000.