ELSEVIER

Fuzzy Sets and Systems 110 (2000) 365–377

FUZZY sets and systems

www.elsevier.com/locate/fss

Tracking a near-field moving target using fuzzy neural networks

Ching-Wen Ma*, Ching-Cheng Teng

Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan

Received July 1996; received in revised form October 1997

Abstract

In this paper we explore the problem of tracking a near-field moving target using fuzzy neural networks (FNNs). The moving target radiates narrow band waves that impinge on an array of passive sensors. At a particular time instance, the location of the target is estimated by several judiciously constructed FNN-based angle and distance estimators. When the target is moving, its trajectory can be on-line estimated due to the parallel and real-time computational capability of the FNNs. Computer simulation results illustrate the performance of the FNN-based angle estimator, distance estimator, and the near-field moving target tracker. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Fuzzy logic; Neural network; Fuzzy neural network; Array signal processing

1. Introduction

The applications of combining fuzzy inference systems and artificial neural networks to engineering problems have drawn a lot of attention recently [1,2,9,10,13,17]. In [9], Lin and Lee applied a neural-network-based fuzzy logic system to control and decision systems. After a slight modification of the network, Chen and Teng [4] used a four-layer fuzzy neural network (FNN) to establish a model reference control structure. In [17,13], the neural network has been also successfully applied to estimate the direction-of-arrival (DOA) of narrow-band signals, which are emitted by a far-field target and impinge on an array of passive sensors. In this paper, we will use the FNNs to establish a method to track a near-field moving target. By near-field we mean that the distance from array center to the target is not far enough, so that the wavefronts emitted by the target are spherical rather than planar.

Recently, in the area of array signal processing, there has been considerable interest in developing efficient algorithms for tracking moving targets [6,11]. These algorithms generally assume that the moving target is in the far field, i.e., the signal received by the array has planar wavefront. In this case, only the direction-of-arrival (DOA) of the moving target is estimated. As a result, determining the location of a far-field moving target requires two arrays separated in a sufficient distance. When the source is not far away from the array and the wavefronts are spherical, the location of the near-field moving target can be estimated using only one array. In [7], Huang and Barkat provided a modified version of 2-D MUSIC and a maximum likelihood estimator for the near-field source (target) localization problem. The main disadvantage of these two methods is their high computational

* Corresponding author. Fax: 886-35-715998.
E-mail address: u8112810@cc.nctu.edu.tw (C.-W. Ma)

loading due to a two-dimensional search on the entire solution space. This makes these two algorithms not suitable for tracking a moving target. In [8], Lee et al. introduced a so-called far-field approximation (FFA) approach for the near-field direction finding problem when the array is uniformly spaced and linear. This approach constructs a far-field approximation covariance matrix from the received near-field covariance matrix. It then applies existing far-field direction estimation techniques, like MUSIC [12], to estimate the directions of these near-field sources. The FFA approach, however, is highly biased in some situations as we will see in Section 5 through computer simulations. Moreover, the FFA approach does not provide any information about the distance. Taking view of the success of fuzzy neural networks on engineering problems, we are motivated to solve the near-field moving target tracking problem using FNNs. The fuzzy neural networks are expected to be able to deal with the on-line tracking problems due to its parallel and real-time computational capability.

In this paper we estimate two parameters for locating a target: (1) the angle between array normal direction and the target direction, and (2) the distance from the array center to the target. The proposed FNN-based near-field moving target tracker first estimates the angle by an FNN. According to the value of the estimated angle, different FNN is used to estimate the distance. In other words, the FNN-based tracker involves one FNN for angle estimation and several FNNs for distance estimation. The performance of the FNN-based angle and distance estimators are illustrated by computer simulations. We find that both the angle and distance estimator provide satisfactory estimates. As a result, the proposed FNN-based tracker tracks the moving target with sufficient accuracy.

2. Problem formulation

Consider a near-field moving target which emits narrow-band signals impinging on a passive sensor array composed of p elements. The location of each sensor is arbitrary as shown in Fig. 1. Since the target is located in the near-field, the signal wavefronts are spherical rather than planar. Under the near-field assumption, the observed data collected by the ith sensor at a particular time instance t can be expressed, by

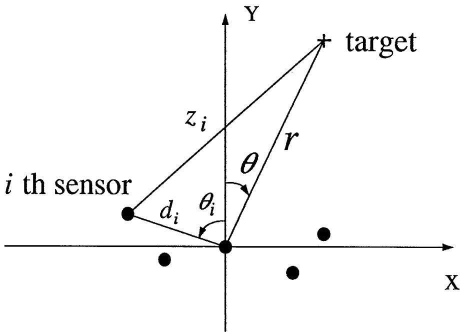

Fig. 1. Array geometry.

its complex envelope, as

$$x_i(t) = a_i(r,\theta)\cdot s(t) + n_i(t), \quad i = 1,\dots,p, \tag{1}$$

where r is the distance from array center to the target, θ is the angle between array normal direction and the target direction, and $s(t)$ and $n_i(t)$ denote the scalar complex waveform of the signal emitted by the target and the additive noise at the ith sensor, respectively. Moreover, $a_i(r,\theta)\cdot s(t)$ presents the complex response at ith sensor, where $a_i(r,\theta)$ is expressed as follows [7]:

$$a_i(r,\theta) = \frac{c_i}{z_i}\exp\left(-\mathrm{j}\frac{2\pi}{\lambda}z_i\right). \tag{2}$$

In (2), λ is the wavelength of the impinging waves, z_i is the distance from the ith sensor to the moving target, and c_i is the gain coefficient of the ith passive sensor. Without loss of generality, we assume that the sensors are omnidirectional, i.e. $c_i = 1$ for $i = 1,2,\dots,p$. As shown in Fig. 1, z_i is computed as follows:

$$z_i^2 = r^2 + d_i^2 - 2rd_i\cos\phi_i, \tag{3}$$

where

$$\phi_i = \theta_i + \theta. \tag{4}$$

Our aim here is to on-line estimate the angle θ and the distance r of the moving target based on the observed data $\{x_i\}_{i=1}^{p}$ using FNNs.

3. Fuzzy neural networks

In this section, we describe a four-layer fuzzy neural network which is a modified version of that proposed

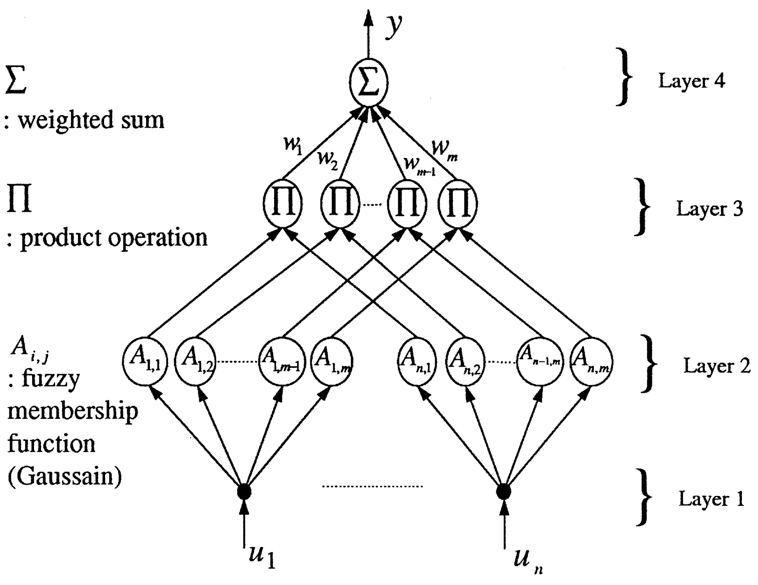

Fig. 2. Structure of a fuzzy neural network.

in [3]. This network will be adopted to estimate the angle θ and the distance r in the next section.

3.1. Structure of the FNN

As shown in Fig. 2, the fuzzy neural network adopted in this paper is an n-input, 1-output, and m-rule fuzzy neural network that maps $\{u_i\}_{i=1}^{n}$ into y. It constructs fuzzy rules *one by one* and adds all rules together. Although it looks like the one proposed in [3], the network uses a different defuzzification operation. Suppose that the jth fuzzy rule reads

IF u_1 IS $A_{1,j}$, u_2 IS $A_{2,j}, \dots, u_n$ IS $A_{n,j}$,

THEN y IS w_j.

Fig. 3 shows the implementation of this rule. Putting all fuzzy rules together, we get the whole fuzzy neural network which is shown in Fig. 2. Layer 1 of such a network is the input layer. It propagates the crisp input u_i to layer 2. Layer 2 is the singleton fuzzification layer, which maps crisp input value u_i into fuzzy set $A_{i,j}$ with membership degree $\mu_{A_{i,j}}(u_i)$. In this paper, $\mu_{A_{i,j}}(\cdot)$ is a Gaussian function, i.e.

$$\mu_{A_{i,j}}(u_i) = \exp\left[-\frac{(u_i - a_{i,j})^2}{2(\sigma_{i,j})^2}\right] \tag{5}$$

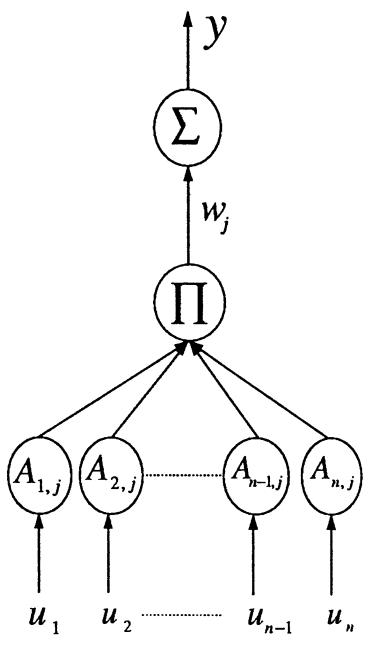

Fig. 3. Implementation of a fuzzy rule.

where $a_{i,j}$ is the mean (or center) and $(\sigma_{i,j})^2$ is the variance (or width) of the Gaussian function. The membership degree $\mu_{A_{i,j}}(u_i)$ is then propagated to layer 3. Layer 3, the fuzzy reasoning layer, performs IF-condition reasoning by a product operation and generates the firing strength α_j of the jth fuzzy rule by

$$\alpha_j = \prod_{i=1}^{n} \mu_{A_{i,j}}(u_i). \tag{6}$$

After the firing strength, α_j, of each rule is computed, the network multiplies α_j by the consequence weight w_j to obtain the contribution of jth rule on the output y. Finally, at layer 4, all the contribution of the m fuzzy rules are summed up to produce the output of the FNN, say

$$y = \sum_{j=1}^{m} w_j \alpha_j. \tag{7}$$

We refer to the defuzzification process (7) as *weighted sum defuzzifer*. This FNN structure is similar to the one reported by Chao [3] except the defuzzification process. It is a universal approximator which is capable of approximating any real continuous function with satisfactory accuracy, provided that sufficient fuzzy rules are used [16].

Remark. Note that instead of using the commonly used *center average defuzzifer*, the proposed network uses the *weighted sum defuzzifer*. In our experience the latter is simpler than, but as efficient as, the former. In [4], Chen also uses the *weighted sum defuzzifer*. However, the network proposed in [4] does not construct fuzzy rules *one by one*. The advantage of constructing fuzzy rules *one by one* is that the number of fuzzy rules can be arbitrarily assigned, regardless of the number of inputs. The fuzzy neural network used in [4] makes fuzzy partitions of the input space. As a result, the number of fuzzy rules increases exponentially with the number of inputs. In the applications of array signal processing, the number of sensors is generally not a small number. If the number of fuzzy rules increases exponentially with the number of inputs (i.e. the number of sensors), the fuzzy neural network will be very large and will be impractical.

3.2. Design procedure

Although the FNN is considered as an universal approximator to any real continuous function, the determination of the membership function $\mu_{A_{i,j}}(\cdot)$ and consequence weight w_j in (6) and (7) plays a crucial role when designing the fuzzy neural network. Proper determination of $\mu_{A_{i,j}}(\cdot)$ and w_j leads to a good function approximator with high accuracy. Otherwise, the FNN will not be a satisfactory function approximator. One way to determine $\mu_{A_{i,j}}(\cdot)$ and w_j is to consult the experts. This way, however, generally provides unsatisfactory accuracy for engineering problems. The advantage of the FNN is that we are able to adjust both w_j and $\mu_{A_{i,j}}(\cdot)$ using the back-propagation algorithm, provided that a lot of input–output training pairs are available. The back-propagation algorithm performs the supervised gradient descent learning. Interested readers may find the details of the learning processes in [4]. In the following we summarize the design procedure.

Step 1: Compute a lot of input–output pairs of the desired function.
Step 2: Construct an initial FNN according to expert knowledge or any other initialization procedure, e.g., the on-line initialization procedure proposed in [3].
Step 3: Train the FNN (i.e., adjust w_j and $\mu_{A_{i,j}}(\cdot)$) by back-propagation algorithm to make the FNN fit the input–output pairs obtained in Step 1.
Step 4: Eliminate redundant rules, which consist of fuzzy sets which look like impulsive functions.

Note that Step 4 aims at confirming the interpolation capability of the FNN. Let $\mathrm{FNN}(u_1, u_2, \dots, u_n)$ denote the output of the FNN when its input is $(u_1, u_2, \dots, u_n)$. After training, we have

$$\|y - \mathrm{FNN}(u_1, u_2, \dots, u_n)\| < \varepsilon \quad \text{for all training pairs.} \tag{8}$$

It, however, does not guarantee

$$\|y - \mathrm{FNN}(u_1, u_2, \dots, u_n)\| < \varepsilon \quad \text{for all } y \text{ and } \{u_i\}_{i=1}^{n}. \tag{9}$$

It is our experience that some of the fuzzy sets look like impulse functions after training. These impulse-like fuzzy sets may produce an impulse at the output. As is well known, the desired function that maps $\{u_i\}_{i=1}^n$ to y is a smooth function. A small change in $\{u_i\}_{i=1}^n$ does not produce large change in y. Suppose that a fuzzy set of the jth rule looks like an impulse and all training data does not fire this rule. Then, the impulse-like fuzzy set may produce an impulse at the output of the FNN, while during the training step, the effect of the impulse-like fuzzy set does not exhibit. Therefore, before putting the FNN into practical usage, the effect of these impulse-like fuzzy sets should be removed. One way to remove this effect is to eliminate fuzzy rules that consist of impulse-like fuzzy sets. The fuzzy rule that consists of the impulse-like fuzzy sets is referred to as a redundant fuzzy rule.

4. FNN-based near-field moving-target tracker

Tracking the near-field moving target requires estimating two parameters: θ and r. The array output vector $\{x_i(t)\}_{i=1}^p$ is used to estimate these two parameters. The purpose of the FNN-based near-field moving-target tracker is to establish a system that maps $\{x_i(t)\}_{i=1}^p$ into θ and r. In this section, we introduce two kinds of FNN-based estimators. One is the FNN-based angle estimator, the other is the FNN-based distance estimator. The FNN-based angle estimator estimates the angle without considering the distance. The FNN-based distance estimator, however, requires information of the estimated angle. According to the value of the estimated angle, different FNN-based distance estimator is used to estimate the distance. Putting the FNN-based angle and distance estimators together, we successfully establish an FNN-based near-field moving target tracker.

4.1. FNN-based angle estimator

In [13], Southall et al. used a neural network to estimate the angle of plan waves emitted by a far-field target. They believed that the amplitude of the received signal is not a strong indicator of the angle. Instead, they thought that the phase is much more important. Using a neural network to learn the relationship between the angle and relative phases between adjacent sensors, they successfully designed a far-field angle estimator. Herein, our problem is more complicated.

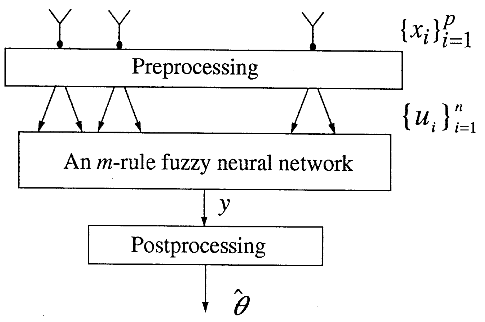

Fig. 4. FNN-based angle estimator.

The target we consider is not located in the far field. Since we consider near-field targets, both magnitude and phase of the array outputs have strong relationship with the angle θ. In other words, both the real and imaginary parts of array responses have strong relationship with θ. Therefore, we use an FNN to establish a function that maps from both the real and imaginary parts of the array responses to the angle θ. Fig. 4 shows the structure of the FNN-based angle estimator. Since the initial phase of the received signal is not known, we preprocess the array output by the following equations:

$$u_{2i-1} = \text{real part of } x_i/x_{i+1}, \tag{10}$$

$$u_{2i} = \text{imaginary part of } x_i/x_{i+1}, \tag{11}$$

$$i = 1, 2, \ldots, p-1. \tag{12}$$

u_{2i-1} and u_{2i} are treated as the inputs of the FNN. The output of the FNN y is postprocessed to produce the angle estimate $\hat{\theta}$. The postprocessing simply shifts and scales y, i.e.,

$$\hat{\theta} = \frac{\theta_{\max} - \theta_{\min}}{2} \times y + \frac{\theta_{\max} + \theta_{\min}}{2}, \tag{13}$$

where $\theta_{\max}$ and $\theta_{\min}$ are the maximum and minimum of the angle that the target may travel to. The reason of the postprocessing is that, in our experiences, the FNN has a fast learning speed if its output y lies in the region $[-1, 1]$.

Remark. We now follow the procedure described in Section 3 to show how to design an FNN-based angle estimator step by step. At Step 1 of the procedure, (1), (10), (11), and (13) are used to compute the input–output pairs, i.e., the pair of $\{u_i\}_{i=1}^n$ and y. One location implies one input–output pair. So, we are able to obtain a lot of input–output pairs. At Step 2, we construct an initial FNN. As mentioned in the previous section, the number of fuzzy rules can be arbitrarily assigned. Our strategy is to assign a large number of fuzzy rules. The fuzzy sets of each fuzzy rule can be designed without considering other fuzzy rules. An important problem is how to choose the initial parameters of the fuzzy sets. Instead of consulting the expert knowledge or using fuzzy-c-means [15], we use the on-line initialization method proposed in [3] to determine the initial parameters, including $a_{i,j}$ and $\sigma_{i,j}$ of $\mu_{A_{i,j}}(\cdot)$ and the consequence weights w_j. The on-line initialization method directly uses the first m training pairs to determine all the initial parameters. Although this method is quite simple, it is efficient and sufficient in practical applications. At Step 3 of the design procedure, the back-propagation algorithm is used to adjust the parameters of the FNN. After training, the FNN approximates the training data with satisfactory accuracy. However, as mentioned in Section 3, the effect of impulse-like fuzzy sets should be eliminated by Step 4. At Step 4, the redundant fuzzy rules, which consist of Gaussian membership functions with small variances, are removed.

4.2. *FNN-based distance estimator*

For angles in different angular section, different FNN-based distance estimator is designed to estimate the distance. Fig. 5 shows the structure of the FNN-based distance estimation for a particular angular section. Note that the output of the postprocessing becomes $\hat{r}^{1/k}$, where k is a factor greater than 1. The reason for using such a factor k is that when the target travels away from the array but the angle is fixed, the farther the target travels, the smaller the array output changes, i.e.,

$$\left.\frac{\mathrm{d}u_i}{\mathrm{d}r}\right|_{r \text{ is small}} \gg \left.\frac{\mathrm{d}u_i}{\mathrm{d}r}\right|_{r \text{ is large}}. \tag{14}$$

This fact makes the back-propagation algorithm hard to *teach* the FNN to map $\{u_i\}_{i=1}^n$ into r. Therefore, we

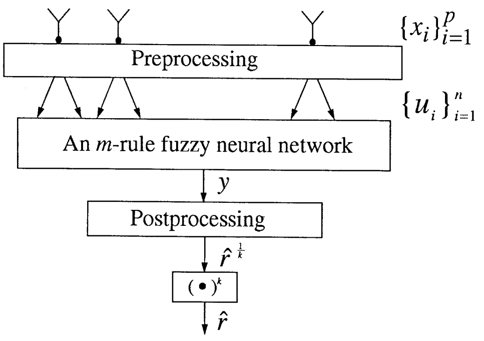

Fig. 5. FNN-based distance estimator.

assume that there exists a constant k which is greater than 1, and the following equation holds:

$$\left.\frac{\mathrm{d}u_i}{\mathrm{d}r^{1/k}}\right|_{r \text{ is small}} \approx \left.\frac{\mathrm{d}u_i}{\mathrm{d}r^{1/k}}\right|_{r \text{ is large}}. \tag{15}$$

The back-propagation algorithm is now able to *teach* the FNN to map $\{u_i\}_{i=1}^n$ into $r^{1/k}$ with sufficient accuracy. Following the procedures described in Section 3, the FNN-based distance estimator can be established.

4.3. *FNN-based near-field moving target tracker*

For different angular section, we design different distance estimator. Using these FNN-based distance estimators and the FNN-based angle estimator, the FNN-based near-field moving target tracker is constructed as shown in Fig. 6. The estimated angle is used to select the proper FNN-based distance estimator for estimating the distance r. The structure is implemented in parallel and is suitable for on-line processing. In other words, it is suitable for tracking a moving target.

5. Computer simulations

In this section, we used various computer simulation results to show the performance of the FNN-based angle and distance estimators and the FNN-based tracker. In Example 1, we presented a detail example of designing an FNN-based angle estimator and compared the performance of the FNN-based angle estimator with the FFA approach proposed in [8]. As we will see, the FFA approach was highly biased when

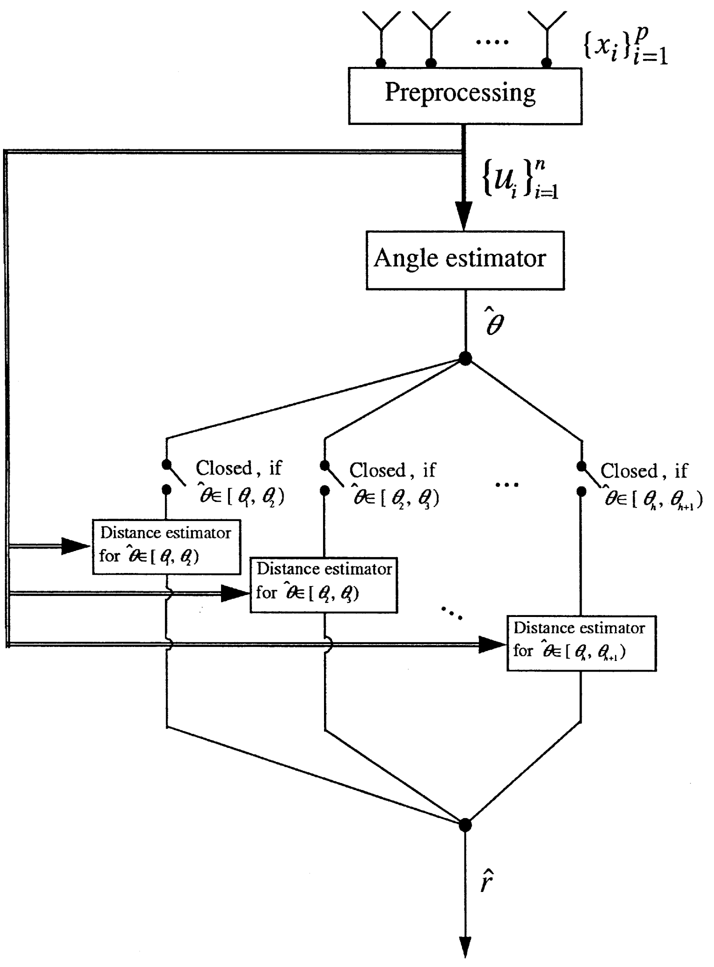
$\{x_i\}_{i=1}^{p}$
Preprocessing
$\{u_i\}_{i=1}^{n}$
Angle estimator
$\hat{\theta}$
Closed, if $\hat{\theta} \in [\theta_1, \theta_2)$
Closed, if $\hat{\theta} \in [\theta_2, \theta_3)$
Closed, if $\hat{\theta} \in [\theta_k, \theta_{k+1})$
Distance estimator for $\hat{\theta} \in [\theta_1, \theta_2)$
Distance estimator for $\hat{\theta} \in [\theta_2, \theta_3)$
Distance estimator for $\hat{\theta} \in [\theta_k, \theta_{k+1})$
$\hat{r}$

Fig. 6. FNN-based near-field moving target tracker.

the angle θ was large and the distance r was short. The FNN-based angle estimator did not have such a shortage. In Example 2, we explored the performance of the FNN-based distance estimator. It turns out that the distance estimates were more accurate when the distance was shorter. Example 3 verified the capability of the FNN-based near-field moving-target tracker. Finally, in Example 4, we compared the computational complexity between the FNN-based tracker and the MLE approach presented in [7].

Example 1. We considered an acoustic signal source (target) located in the near field, emitting 100 Hz-centered narrow-band waves into a linear array with five sonar elements uniformly spaced half a wavelength a part. Since the far-field distance $(p-1)^2(\lambda/2)$ was about 116 m, the target we considered here was located at a distance of 10–80 m. Note that the FNN approach did not require the array to be linear and uniformly spaced. However, in order to compare the performance between the FFA approach and the FNN approach, the array under consideration was linear and uniformly spaced.

For designing an FNN-based angular estimator, we began with computing the training pairs. A certain location of the target implied an input–output training pairs. 820 training pairs computed by (1), (10), (11), and (13) were used to train an FNN consisting of 40 fuzzy rules by back-propagation algorithm. The parameters of the initial FNN were selected by the on-line initialization method proposed in [3]. In other words, 40 input–output pairs out of the 820 training pairs were used to determine the parameters of the 40 fuzzy rules at the beginning. After 300 training steps, the back-propagation algorithm converged. As mentioned in Section 3, we had to eliminate the redundant fuzzy rules. Fuzzy rules consisting of Gaussian membership functions with variances less than 0.1 were considered redundant and were eliminated. There were 16 fuzzy rules that should be eliminated. After eliminating these 16 redundant fuzzy rules, we obtained a 24-rule FNN. Fig. 7 depicts the angle estimation error the 40-rule FNN-based angle estimator, for which the redundant fuzzy rules were not yet eliminated. One impulse appeared in Fig. 7. This means that the 40-rule FNN was not suitable for estimating the angle, because the estimation error might be dramatically large at some situations. Fig. 8 depicts the estimation error of the 24-rule FNN-based angle estimator, for which the redundant fuzzy rules were eliminated. Obviously, the impulse in the error curve had been removed. In other words, the effect of the impulse-like fuzzy sets was avoided.

Regarding the performance, the performance of the FNN-based angle estimator and the FFA approach in noise-free environment were compared. The noise-free environment allowed us to be able to clearly examine the bias of these two methods. Fig. 9 plots the estimation error of these two methods. We observed that the FNN-based angle estimator was better than the FFA approach, especially when the distance was short or the angle was large.

Example 2. After the angle was estimated, we proceeded to estimate the distance by FNN-based distance estimators. The same acoustic signal source (target) and sonar array as those considered in Example 1 were considered here. Assuming that the angle might vary from -40° to 40°, we divided the whole angle region into eight equal sections, i.e., $[-40^\circ, -30^\circ)$, $[-30^\circ, -20^\circ)$, $[-20^\circ, -10^\circ)$, $[-10^\circ, 0^\circ)$, $[0^\circ, 10^\circ)$, $[10^\circ, 20^\circ)$, $[20^\circ, 30^\circ)$, and $[30^\circ, 40^\circ)$. For a particular angular section, we designed an FNN-based direction estimator for it.

In the following, we report how we designed an FNN-based distance estimator for angular section $[0, 10)$. First, a lot of input–output pairs were obtained by setting $\theta = [-2, 0, \ldots, 12]$, $k = 3$ and $r^{1/k} = [10^{1/k}, \ldots, 80^{1/k}]$. Second, the on-line initialization procedure was used to initialize a 40-rule FNN. Then, the back-propagation algorithm was used to adjust the FNN and made the FNN fit the input–output training pairs. Finally, 11 fuzzy rules that consisted of Gaussian membership functions with variances less than 0.05 were considered redundant and were eliminated. Note that, in this example, FNN-based distance estimators for other angular sections had eliminated 8–15 redundant fuzzy rules.

Fig. 10 depicts the estimation errors of these FNN-based direction estimators. As Example 1, data in Fig. 10 were obtained in noise-free environment. We observed that for all the subfigures of Fig. 10, when the distance was less then 40 m, the distance estimation error was very small. When the distance became larger, the estimation error became larger too. Nonetheless, all the estimation errors were bound in $[-0.5, 0.5]$ m. As we will see in the next example, the estimation accuracy was satisfactory for tracking moving targets.

Example 3. The same sonar array was considered in this example. The target, however, was moving now. In order to track the moving target, the FNN-based angle estimator and the FNN-based distance estimators were put together to construct an FNN-based near-field moving target tracker as shown in Fig. 6.

Figs. 11–16 illustrate the tracking performance for different trajectories of the moving target and for different signal-to-noise ratios (SNR). In Fig. 11, the SNR value was 90 dB. We observed that the proposed FNN-based tracker could track the moving target with high accuracy. The high tracking performance resulted from the high accuracy of the FNN-based angle estimator, and the FNN-based distance estimators. Fig. 12 was the result for the same trajectory when the SNR was 80 dB. We found that when the target traveled at a distance about 80 m away from the array center, the tracking error became visible. However, when the target traveled at a distance less than 50 m from the array center, the tracking error was almost not visible. For Fig. 13, the target traveled along the same trajectory

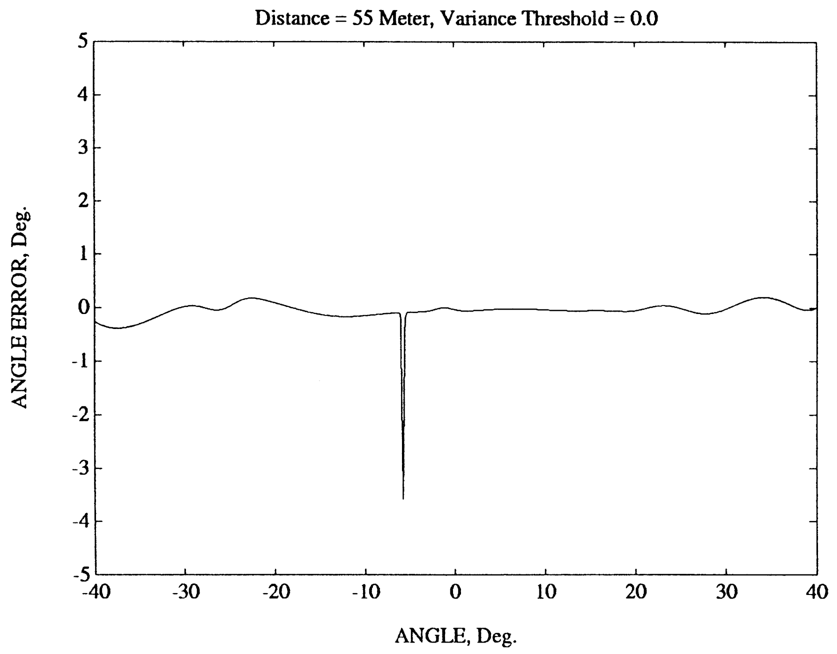

Fig. 7. Angle estimation error before eliminating redundant fuzzy rules.

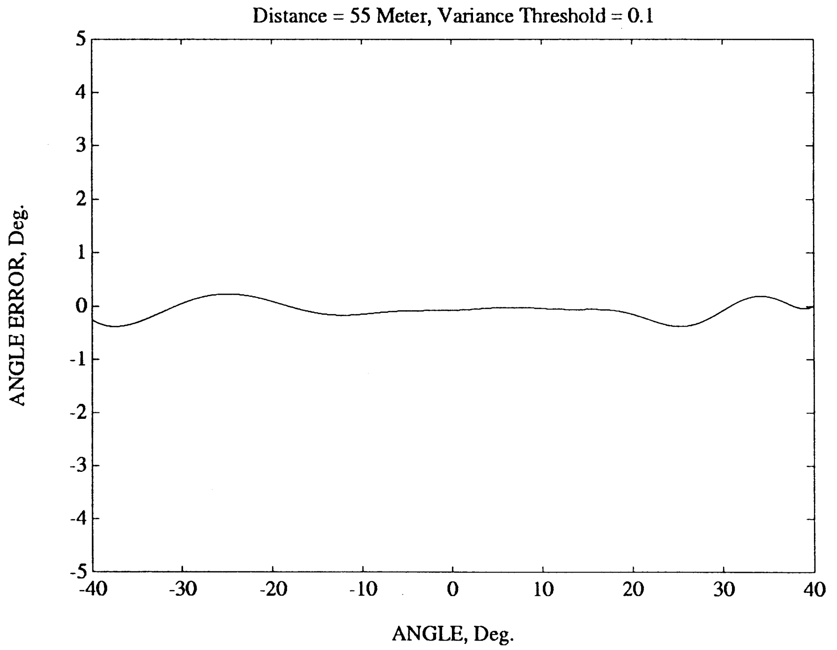

Fig. 8. Angle estimation error after eliminating redundant fuzzy rules.

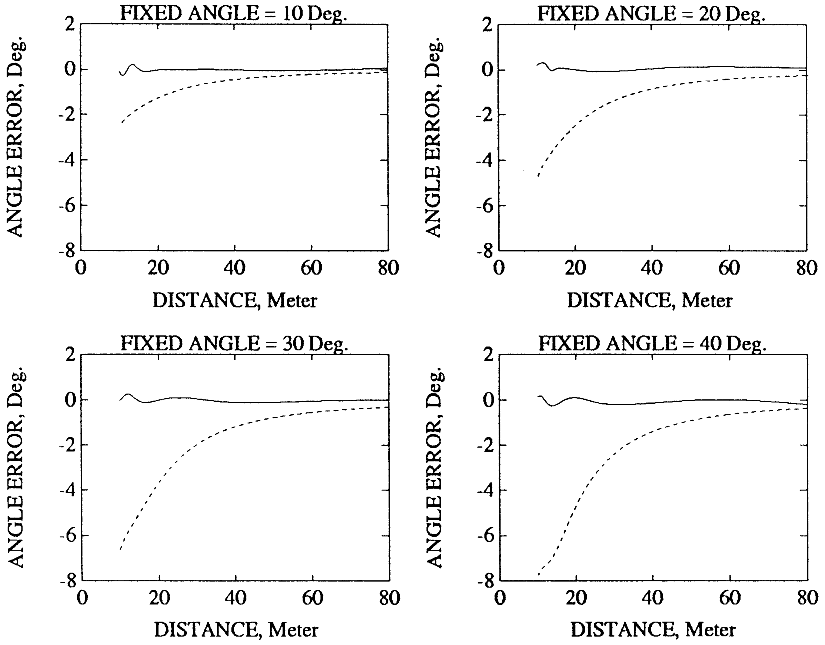

Fig. 9. Angle estimation error vs. distance: solid line (FNN-based angle estimator), dash line (FFA approach).

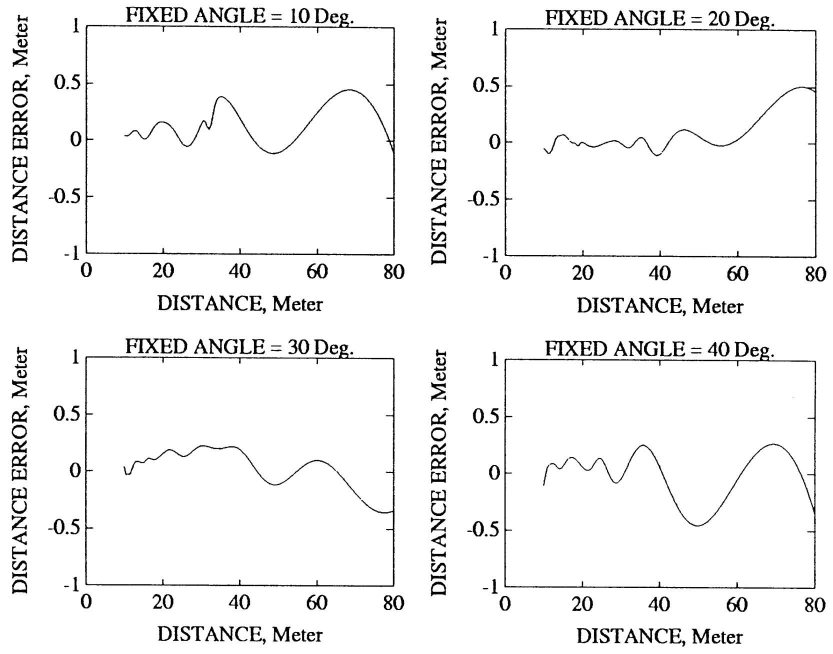

Fig. 10. Distance estimation error vs. distance: FNN-based distance estimator.

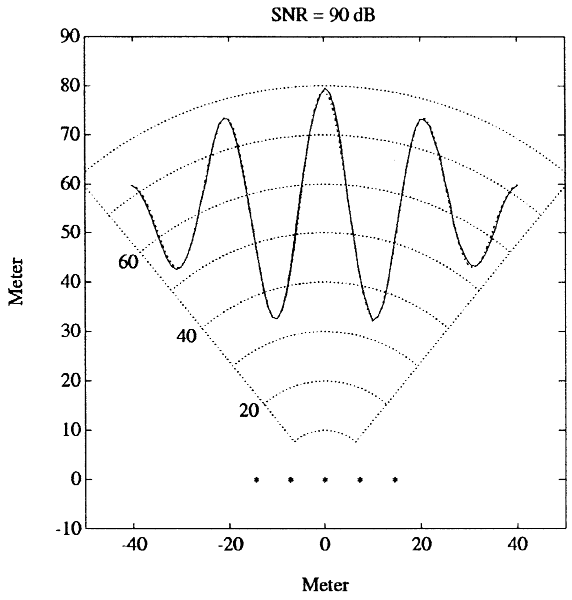

Fig. 11. Tracking performance, smooth trajectory, SNR = 90 dB: solid line (FNN approach), dashdot line (actual trajectory), *-marker (sensor location).

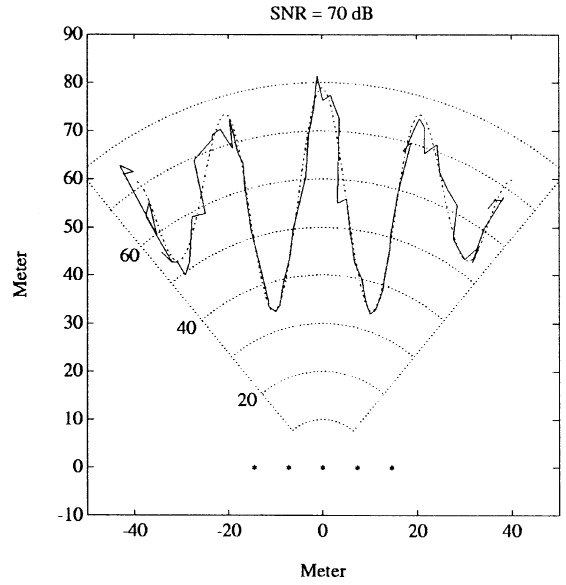

Fig. 13. Tracking performance, smooth trajectory, SNR = 70 dB: solid line (FNN approach), dashdot line (actual trajectory) *-marker (sensor location).

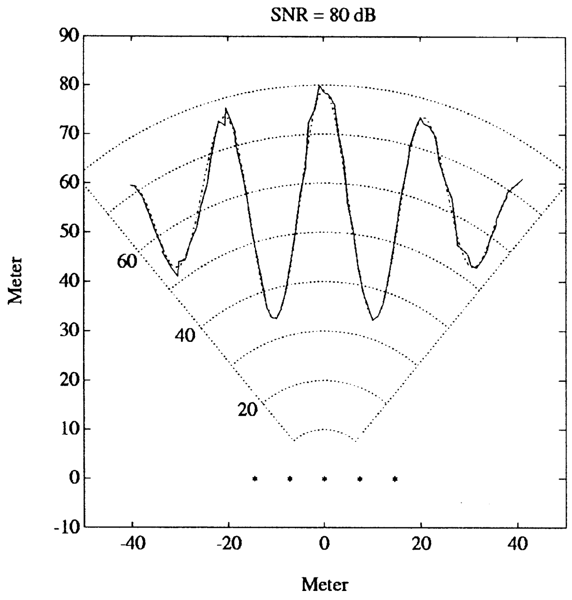

Fig. 12. Tracking performance, smooth trajectory, SNR = 80 dB: solid line (FNN approach), dashdot line (actual trajectory), *-marker (sensor location).

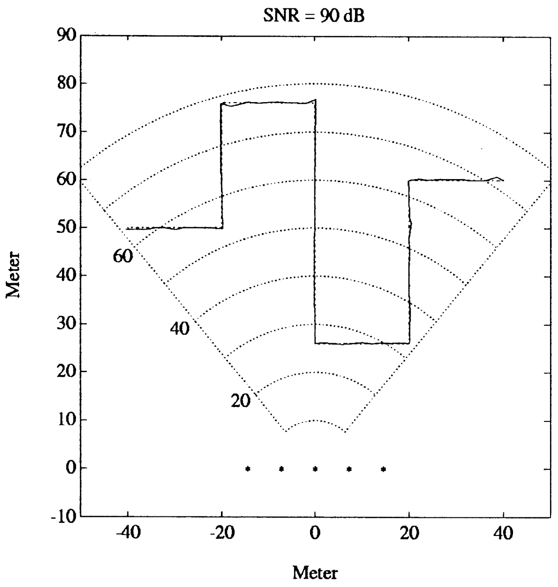

Fig. 14. Tracking performance, non-smooth trajectory, SNR = 90 dB: solid line (FNN approach), dashdot line (actual trajectory), *-marker (sensor location).

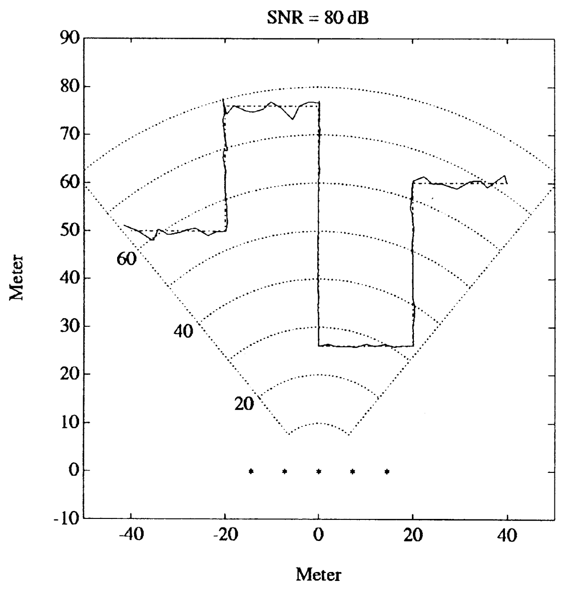

Fig. 15. Tracking performance, non-smooth trajectory, SNR = 80 dB: solid line (FNN approach), dashdot line (actual trajectory), *-marker (sensor location).

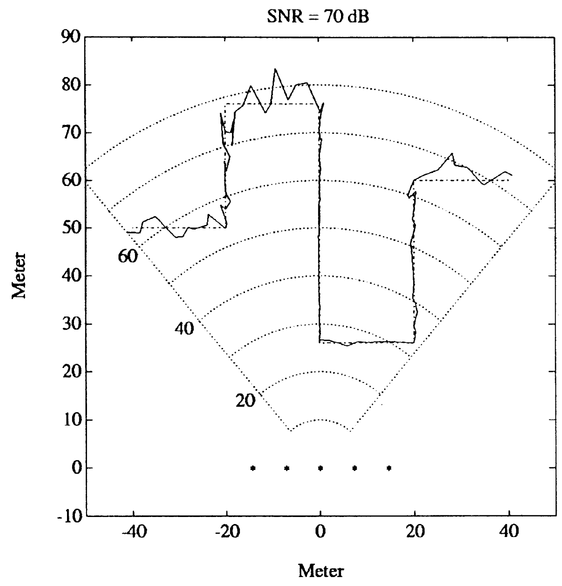

Fig. 16. Tracking performance, non-smooth trajectory, SNR = 70 dB: solid line (FNN approach), dashdot line (actual trajectory), *-marker (sensor location).

and the SNR was 70 dB. The tracking error became more visible. However, when target traveled near the array, the tracking error was also small. Figs. 14–16 show the tracking performance while the target moving along a non smoothing trajectory. We observed that the FNN-based tracker was also efficient for non-smooth trajectories.

Example 4. In this example, we computed the computational complexity of the FNN-based tracker, and compared it with the MLE approach presented in [7]. The FNN-based tracker consisted of an FNN-based angular estimator and many FNN-based distance estimators. At a particular time instance, however, only one FNN-based distance estimator was fired. Therefore, the computational complexity was dominated by an FNN-based angular estimator and an FNN-based distance estimators. In the following, we used the number of floating point operations (flops) defined in MATLAB[1] as an index of the computational complexity. The total flops of an n-input, m-rule, and one-output FNN network is

$$\text{flops at layer 2} + \text{flops at layer 3} + \text{flops at layer 4} \tag{16}$$

$$= n \times 5 \times m + n \times m + 2 \times m. \tag{17}$$

The FNN-based angular estimator in Example 3 consisted of 24 rules and 8 inputs. The distance estimator consisted of about 28 rules and 8 inputs. Totally, the computational complexity of the FNN-based target tracker was about 2600 flops.

Considering the computational complexity of the MLE approach, we started by computing how many points the MLE had to examine in the entire solution space. In Example 3, the angle θ varied from $-40°$ to $40°$; the distance r varied from 10 to 80 m. Examining the MLE cost function at each 1° and each 1 m requires computing the MLE cost function 5600 times. The MLE cost function used in [7] was

$$g(r,\theta) = \text{trace}\{\boldsymbol{P}_{\boldsymbol{a}(r,\theta)} \cdot \hat{\boldsymbol{R}}_x\}, \tag{18}$$

where $\boldsymbol{a}(r,\theta)$ was defined as the complex conjugate transpose of $[a_1(r,\theta), a_2(r,\theta), \ldots, a_p(r,\theta)]$, $\boldsymbol{P}_{\boldsymbol{a}(r,\theta)}$ was the projection operator onto the space spanned by

[1] MATLAB is a trade mark of MathWorks Inc.

$\boldsymbol{a}(r,\theta)$, and $\hat{\boldsymbol{R}}_x$ was the data covariance matrix. For Example 3, MATLAB used 1287 flops to compute Eq. (18). As a result, the computational complexity of the MLE approach was about $5600 \times 1287/2600$ times that of the FNN approach. Note that although the MLE approach might provide accurate estimates even in low SNR situations, the high computational complexity made the MLE approach not suitable for tracking a moving target.

6. Conclusion

We have shown that the FNNs can be used to establish a near-field moving target tracker. This kind of tracker can deal with arbitrary array geometry. Due to its parallel computational capability, it also fulfills the on-line tracking purpose. Regarding the structure, it involves an FNN-based angle estimator and some FNN-based distance estimators. The FNN-based angle estimator estimates the angle without considering the distance. The estimated angle is then used to select a proper FNN-based distance estimator for estimating the distance. Both the angle and distance are estimated on-line. Thus, the moving target is tracked. Computer simulations have validated the tracking capability of the novel tracker. To sum up, we believe that by judiciously applying the fuzzy neural networks, some of the signal processing problems can be well resolved. More applications of the fuzzy neural networks to signal processing problem can be expected in the future. This is also our future research interest.

References

[1] J.F. Baldwin, R.M. Gooch, T.P. Martin, Fuzzy processing of hydrophone sounds, Fuzzy Sets and Systems 77 (1996) 35–47.

[2] J.J. Buckley, Y. Hayashi, Fuzzy neural networks: a survey, Fuzzy Sets and Systems 66 (1994) 1–13.

[3] C.T. Chao, Y.J. Chen, C.C. Teng, Simplification of fuzzy-neural systems using similarity analysis, IEEE Trans. System Man Cybernet. 26 (5) (1996) 344–354.

[4] Y.C. Chen, C.C. Teng, A model reference control structure using a fuzzy neural network, Fuzzy Sets and Systems 73 (1995) 291–312.

[5] Y.M. Chen, J.H. Lee, C.C. Yeh, Two-dimensional angle-of-arrival estimation in presence of finite distance sources, IEEE Trans. Antennas Propagat. 40 (9) (1992) 1011–1021.

[6] A. Eriksson, P. Stoica, T. Söderström, On-line subspace algorithm for tracking moving sources, IEEE Trans. Signal Process. 42 (5) (1994) 2319–2330.

[7] Y.D. Huang, M. Barkat, Near-field multiple source localization by passive sensor array, IEEE Trans. Antennas Propagat. 39 (7) (1991) 968–975.

[8] J.H. Lee, Y.M. Chen, C.C. Yeh, A covariance approximation method for near-field direction-finding using a uniform linear array, IEEE Trans. Signal Process. 43 (5) (1995) 1293–1298.

[9] C.T. Lin, C.S.G. Lee, Neural-network-based fuzzy logic control and decision system, IEEE Trans. Comput. 40 (12) (1991) 1320–1336.

[10] J.M. Mendel, Fuzzy logic systems for engineering: a tutorial, Proc. IEEE 83 (3) (1995) 345–377.

[11] C.R. Rao, C.R. Sastry, B. Zhou, Tracking the direction of arrival of multiple moving targets, IEEE Trans. Signal Process. 42 (5) (1994) 1133–1143.

[12] R.O. Schmidt, Multiple emitter location and signal parameter estimation, IEEE Antennas Propagat. 34 (3) (1986) 276–380.

[13] H.L. Southall, J.A. Simmers, T.H. O'Donnell, Direction finding in phased arrays with a neural network beamformer, IEEE Trans. Antennas Propagat. 43 (12) (1995) 1369–1374.

[14] M. Sugeno, An introductory survey of fuzzy control, Inform. Sci. 36 (1985) 59–83.

[15] M. Sugeno, T. Yasukawa, A fuzzy-logical-based approach to qualitative modeling, IEEE Trans. Fuzzy Systems 1 (1) (1993) 7–31.

[16] L.X. Wang, Adaptive Fuzzy Systems and Control: Design and Stability Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[17] T. Wong, T. Lo, H. Leung, J. Litva, E. Bosse, Low-angle radar tracking using radial basis function neural network, IEE Proc. F 140 (11) (1993) 323–328.