An intelligent detection scheme for sonar signal tracking

V.S. KODOGIANNIS
Mechatronics Group
Dept of Computer Science
University of Westminster
London HA1 3TP
UNITED KINGDOM

D. TOMTSIS[+]
[+]TEI of West Macedonia
Koila,
Kozani GR-50100

GREECE

Abstract: - A novel version of the adaptive sonar signal thresholding scheme using RBF neural networks is proposed in this paper. Intensity thresholding has proven to be an effective technique to eliminate the low energy noise and to reduce the computational load in an underwater target tracking system. The adopted technique yields unbiased estimates under a non-homogenous sea environment, because the false alarm rate is maintained at a constant level while the threshold changes with different sea environments. In addition, the threshold for different range cells can be adaptively estimated, since the noise under estimation is strictly local so that the distance the sonar signals travelled does not affect the received intensities of noise and targets. Finally, the computational requirements are greatly reduces through the introduction of a recursive proposed scheme.

Keywords: - Radial basis function networks, sonar, signal thresholding.

1 Introduction

Intensity thresholding has been proven to be an effective technique for eliminating the low energy noise and reducing the computational burden in an active target-tracking algorithm for an underwater sonar system. The significance of sonar signal thresholding lies in that measurements gathered from sonar devices are generally corrupted by various noise signals such as the ambient and reverberation noise and they contribute to a high false alarm rate. The performance of a tracking system is dependent on the rate of false alarm and the probability of detection, which in turn are determined by the detection threshold.

Most of the active sonar tracking systems in the beginning exploited a simple thresholding scheme; *i.e.*, a pre-selected constant threshold [1-2]. The constant false alarm rate approach has been thoroughly investigated for the selections of radar signal detection threshold where similar problems are encountered. In this scheme, the detection threshold is adjusted according to the probability density of the background noise to keep the false alarm rate within an acceptable level which is predefined as a constant, and hence the name Constant False Alarm Rate (CFAR) approach.

Although studies have been conducted to determine the distribution of various types noise signals in the sea and some theoretical results have been reported in the literature [3-5], the realistic situation vary considerably and hence the most reliable results are still yielded from the actual measured data. Taking into account the spatial and temporal environmental variations of an underwater active sonar, a recursive version of the an adaptive CFAR sonar signal thresholding scheme using radial basis function neural networks is proposed in this research. The proposed thresholding scheme is applied to active sonar tracking system exercise against the realistic sea environment. Experimental results show that the advantages of the proposed neural network-based detection thresholding scheme are the following:

- It yields acceptable perform in a non-homogenous sea environment, because the false alarm rate is kept constant while the threshold changes with different sea environments.
- In addition, it can adaptively estimate the threshold for different range cells because the noise signal under estimation is strictly local so that the distance the signals travelled does not affect the received intensities of noise and targets.
- Finally, the computational burden has been greatly reduced through the introduction of the recursive scheme.

2 The adaptive CFAR sonar signal detection thresholding system

The block diagram of the proposed adaptive CFAR sonar signal detection thresholding system is illustrated in Fig. 1.

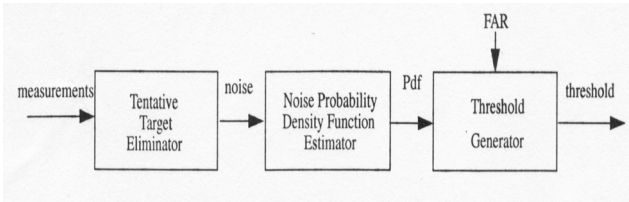


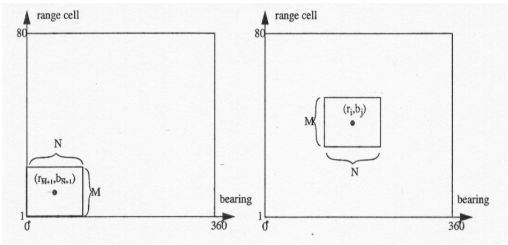
Fig. 1: The Adaptive CFAR Sonar Signal Detection Thresholding

The algorithm involves three functional blocks:

- The tentative target eliminator (TTE) is responsible for removing possible target measurements from the input array.

- The noise probability density function estimator (NPDFE) provides the estimated noise probability density
- The threshold generator (TG) determines the detection threshold from the noise probability density function according to the given FAR.

While the first two blocks are constructed using two RBF networks, the last functional block is simply an integrator. Taking into accounts the time and spatial variations of the environment, only the local information is utilised to estimate the detection threshold of a data point with range r_i, and bearing b_j. The intensities of the points within a reference range-bearing window of size $M \times N$ surrounding the cell under consideration are taken as the inputs to the proposed algorithm. Fig. 2 illustrates a range-bearing window for an omni transmission case. Assume that the cell under consideration is located at (r_i, b_j), and that the reference range-bearing window is of size $M \times N$, where M and N are the window lengths extended along the range and the bearing directions, respectively, and are generally chosen as odd numbers for the purpose of symmetry.



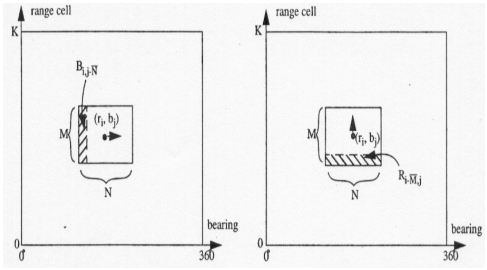
(a) Initialisation Phase (b) Recursion Phase

Fig. 2: Range-bearing Window

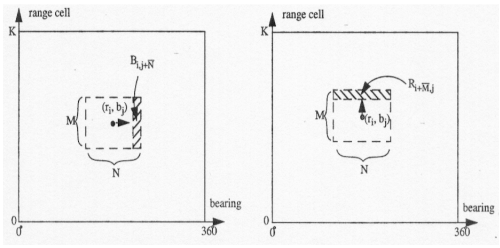
For notational convenience, we usually use the following notation:

$$\overline{M} = (M-1)/2,$$
$$\overline{N} = (N-1)/2$$

The proposed adaptive CFAR sonar signal detection scheme, as shown in Fig.1, functions in one of the following two phases: ***the initialisation phase and the recursion phase***. For each ping of transmission, the estimation of the detection threshold for the first cell considered goes through the initialisation phase, because there is no existing cell with estimated noise probability density function from its neighbouring cells it can build upon. Theoretically, any point can be taken as the initialisation cell. The point $(r_{\overline{M}+1}, b_{\overline{N}+1})$ is chosen as the initialisation cell since the data within its reference window is available first. This situation is illustrated in Fig. 2a. For the rest of the cells, the thresholds can be estimated point-by-point recursively. Fig.3 illustrates the range-bearing window for the above example sliding along both range and bearing directions. Figs. 3a and 3b show the reference windows before and after sliding from (r_i, b_j) to (r_i, b_{j+1}), which is a step of 5 degrees along the bearing.



(a) Reference Window before Slides along Bearing *(c) Reference Window before Slides along Range*



(b) Reference Window after Slides along Bearing *(d) Reference Window after Slides along Range*

Fig. 3: Range-bearing Window Sliding

In the proposed algorithm, instead of taking all the $M \times N$ data points in the new window centred at (r_i, b_{j+1}), as input to estimate the new threshold, only M new data points from the bearing vector $B_{i,j+\overline{N}+1}$ need to be considered, while the effects of M old data points from the bearing vector $B_{i,j-\overline{N}}$ are eliminated. A bearing vector is a column vector of size $M = 2\overline{M}+1$. Figs. 3c and 3d illustrate the case where the reference window slides one range cell along the direction of range. Again, the only new information available is from N points of the range vector $R_{i+\overline{M}+1,j}$ and the old N data points to be deleted are from the range vector $R_{i-\overline{M},j}$. Similarly, a range vector is a row vector of size $N = 2\overline{N}+1$. The computational requirements to process one data point are reduced from $M \times N$ operations to either $2M$ or $2N$ operations depending on the sliding direction, thus decreasing the computational burden considerably.

3 The tentative target eliminator

3.1 TTE-RBF structure

The objective of the tentative target elimination (TTE) is to remove the tentative targets from the measurements. The structure of the TTE is illustrated in Fig. 4. As in a standard RBF network, the TTE has three layers: an input layer, an output layer and a hidden layer. While there is only one node in the input layer that accepts input from the measurement sequence $\{m_s\}$, there are two nodes in the output layer that group the noise sequence as well as the tentative targets.

The hidden layer is composed of two nodes: one is used for the classification of noise *(NODE_MIN)* and the other *(NODE_MAX)* for the tentative target(s) (if there exists any). The kernel RBF is chosen as the commonly used Gaussian function [6]

$$\phi(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}(\frac{w(x-c)}{\sigma})^2} \qquad (1)$$

where, $c = c_{min}$ or c_{max} is the centre, $\sigma = \sigma_{min}$ or σ_{max} is the variance of the node *NODE_MIN* or *NODE_MAX*, respectively.

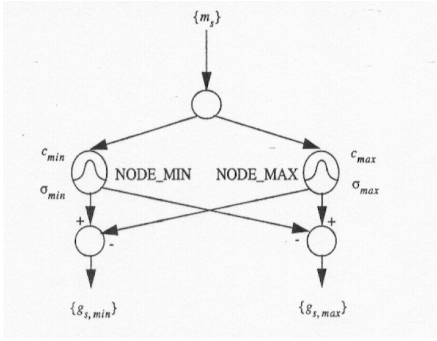


Fig. 4: The Tentative Target Eliminator

The operation of the TTE can be briefly explained as follows. All the sonar measurements within the reference window are first ordered by increasing intensity before feeding to the TTE as an input sequence. The input datum with the lowest intensity other than zero and the one with the highest intensity are used as the initial values for the two node centres. Then the ordered intensities are fed to the input layer of the TTE and are subsequently grouped into lower and higher energy classes. If the higher energy class satisfies the criterion for tentative targets, then the intensity of the individual member of that group is deleted from the measurements. This operates recursively until there are no more tentative targets.

3.2 Linked–List: An I/O Data Structure

The first level in the TTE process is the data structure that is utilised to store ordered data sequences, *i.e*. the linked-list. The structure of a linked-list is depicted in Fig. 5. As shown in Fig. 5a, an element in the linked–list has three fields: A pointer field that points to the next element in the linked–list, an address field that stores the address of input data, and a classification field that records one of the three types of data being classified: *UNCLASSIFIED, NOISE* and *TENTATIVE NOISE*. The linked-list is addressed via a head-pointer that resides on a separate head node that has two fields or records. In addition to the pointer or the address field, the head node also possesses a *z* field. Zero-intensity elements are not included in the linked-list, but the number of zero intensity elements is recorded in the *z* field of the head node. Thus, the first element that the linked-list points to has the lowest nonzero intensity and the last element that the linked-list points to has the highest intensity within their corresponding window. Therefore, the length of the linked-list $L \le (M \times N)$ indicates the number of nonzero elements. These are illustrated in Fig. 5b.

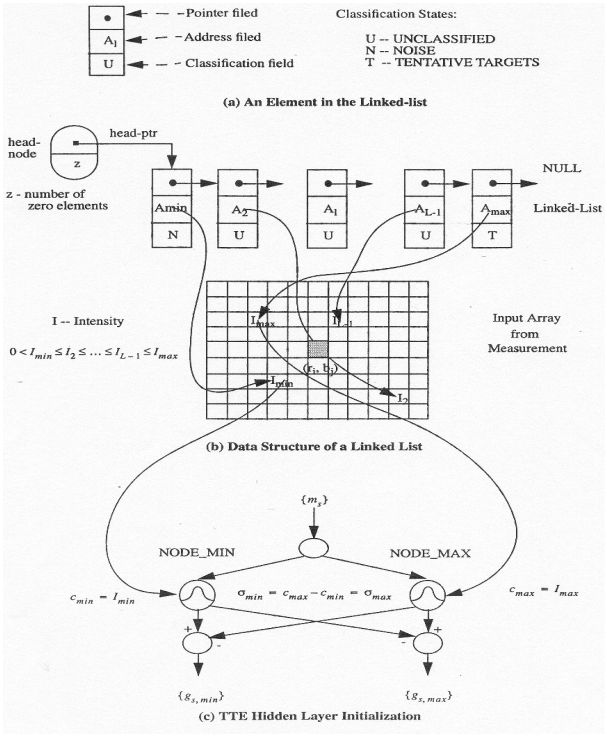


Fig. 5: Linked-List: a Structure for Ordered Data Sequence

3.3 TTE Adaptive Learning Procedures

The TTE operates in two phases: the initialisation phase and the recursion phase. The detailed procedure for each phase is given below:

3.3.1 Initialisation Phase

Step 1: The measurement data from the reference window, which are saved in an $M \times N$ input array, are first arranged in the increasing order of intensity and saved in a linked-list as described above.

Step2: The TTE is initialised as follows: The content I_{min} addressed by the first element in the linked –list is set to be the centre c_{min} of the node *NODE_MIN* (c_{min}=I_{min}). The corresponding classification is set to *NOISE*. The highest intensity I_{max} that is addressed by the last element in the linked-list is set to be the centre c_{max} of the node *NODE_MAX* (c_{max}=I_{max}). Its corresponding classification is set as *TENTATIVE TARGETS*. The variances of both nodes are set as the difference between the centres of the two nodes: $\sigma_{max} = \sigma_{min} = c_{max} - c_{min}$. The initial weights are set to unity: $w_{max} = w_{min} = 1$. Fig. 5c shows the hidden layer-state of the TTE after initialisation.

Step 3: Starting from the beginning of the linked-list, and selecting those elements that were classified as "*UNCLASSIFIED*" to be the inputs, the response of the systems to each input, denoted as m_s, is given by

$$\begin{aligned} g_{s,\max} &= \phi_{\max}(m_s) - \phi_{\min}(m_s) \\ g_{s,\min} &= \phi_{\min}(m_s) - \phi_{\max}(m_s) \end{aligned} \qquad (2)$$

If $g_{s,\max} > g_{s,\min}$, set the input element as tentative target, update the centre c_{max}, the variance σ_{max}, and the weight w_{max} of the node *NODE_MAX* according to

$$c_{\max} = \nabla(w_{\max}) \cdot (w_{\max} c_{\max} + m_s) \quad (3)$$

$$\sigma_{\max} = \frac{1}{w_{\max}} (c_{\max} - c_{\min}) \quad (4)$$

$$w_{\max} = \frac{1}{\nabla(w_{\max})} \quad (5)$$

where $\nabla(w_{\max}) = \frac{1}{w_{\max} + 1}$ is defined as the weight adaptor for w_{max}. Otherwise, set the input element as *NOISE*, update the centre c_{min}, the variance σ_{min}, and the weight w_{min} corresponding to the node NODE_MIN according to

$$c_{\min} = \nabla(w_{\min}) \cdot (w_{\min} c_{\min} + m_s) \quad (6)$$

$$\sigma_{\min} = \frac{1}{w_{\min}} (c_{\max} - c_{\min}) \quad (7)$$

$$w_{\min} = \frac{1}{\nabla(w_{\min})} \quad (8)$$

where the weight adaptor for w_{min} is defined as $\nabla(w_{\min}) = \frac{1}{w_{\min} + 1}$.This is repeated until the end of the linked-list is reached. Thus, two classes of data are grouped and classified as *NOISE* and *TENTATIVE TARGETS*.

Step 4: If the criterion for tentative target $c_{\max} >> c_{\min}$ is satisfied, then delete these tentative targets from the linked-list, set the classification field of the remaining elements in the linked-list back to *UNCLASSIFIED*, increase the number of zero intensity elements by the number of tentative targets deleted, and return to step 2. This is repeated until there is no more input measurements that can be classified as tentative targets.

Step 5: The remaining elements in the linked-list after step 4 are considered as the noise and their corresponding classification fields are set to *NOISE* before being sent to the next functional block (***NPDFE***) as input. In addition, since the *NODE_MAX* as this point also represents a group of noise signals though with higher energy than the group in *NODE_MIN*, the effect of the *NODE_MIN* needs to combined with that of the *NODE_MAX* such that

$$c_{\min} = \Delta(w) \cdot (w_{\min} c_{\min} + w_{\max} c_{\max}) \quad (9)$$

$$w_{\min} = 1 / \Delta(w) \quad (10)$$

where $\Delta(w) = \frac{1}{w_{\min} + w_{\max}}$ called the weight updating factor, thus setting up the initial state of *NODE_MIN* for the TTE in the recursion phase.

3.3.2 Recursion Phase

Step 1: Before sliding the reference window, step 1 eliminates the effects of the data not included in the new window. There are two cases that should be considered separately. If the window slides along the direction of bearing, as shown in Fig 3a, then group those measurements $\{m_s\}$ that reside within the window of the bearing vector $B_{i,j-\bar{N}}$, eliminate their influence from the resultant noise measurement list by adjusting the centre and the weight of the node *NODE_MIN* according to

$$c_{\min} = \nabla(w_{\min}, N)(c_{\min} - \sum_{s=1}^{N} m_s) \quad (11)$$

$$w_{\min} = 1 \Big/ \nabla(w_{\min}, N) \quad (12)$$

On the other hand, if the window slides along the direction of range, as depicted in Fig. 3b, the consequences of those elements $\{m_s\} \in R_{i-M/2,j}$, which lie outside the new window after sliding, are eliminated by updating the centre and the weight of the nose *NODE_MIN* according to

$$c_{\min} = \nabla(w_{\min}, M)(c_{\min} - \sum_{s=1}^{M} m_s) \quad (13)$$

$$w_{\min} = 1 \Big/ \nabla(w_{\min}, M) \quad (14)$$

In both cases, the modified weight adaptor is defined as

$$\nabla(w_{\min}, K) = \frac{1}{w_{\min} - K}.$$

Step 2: Depending on the directions of sliding, either along the range or the bearing, slide the window by updating the point of interest as $r_i = r_{i+1}$ or $b_j = b_{j+1}$, respectively.

Step 3: Consider the measurement intensities $I(r_i, b_j)$ where $(r_i, b_j) \in R_{i+M/2,j}$ or $(r_i, b_j) \in R_{i,j+N2}$, depending on whether the window slides along the bearing or the range cell. Store the resulting data in a sub-linked –list in the increasing order of intensity.

Step 4: Select the highest intensity as the centre c_{max} of the node *NODE_MAX*, set: $\sigma_{max} = \sigma_{min} = c_{max} - c_{min}$, and repeat steps 3 and 4 described in the initialisation phase until all the tentative targets are deleted from the list.

Step 5: Insert the sub-linked-list in the increasing order of intensity into the linked–list obtained from the previous iteration and sends the resulting data sequence as input to the *NPDFE*, the next functional block.

4 The noise probability density function estimator

4.1 NPDFE-RBF structure

The noise probability density function estimator (NPDFE) is illustrated in Fig. 6. As indicated in Fig. 6a, both the input and the output layers each have one node. The measured noise sequence $\{n_s\}$, which is obtained from the TTE in the previous functional block, is used as the input to the NPDFE. The output is

employed in the training phase to justify the adaptation for the centres and the weights of the hidden layer.

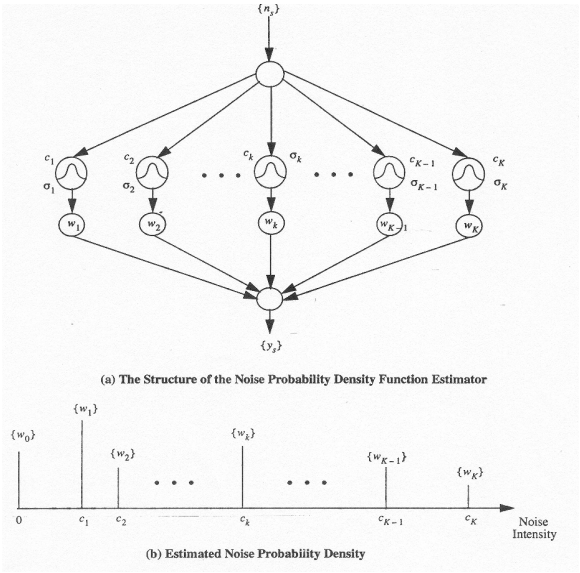


Fig. 6: The Noise Probability Density Function Estimator

In the training phase, the output y_s, as a function of the input $\{n_s\}$ is given as follows:

$$y_s = \sum_k w_k \phi(n_s) - \theta_k \qquad (15)$$

where w_k, θ_k represent the weight and threshold of the k^{th} node, whereas the basis function takes the form of Gaussian density which is typical in RBFs. That is

$$\phi_k(n_s) = \frac{1}{2\pi\sigma_k} e^{-(\frac{n_s - c_k}{\sigma_k})^2} \qquad (16)$$

where c_k and σ_k are the centre and the standard deviation of the Gaussian density function, respectively. The NPDFE learns the noise probability density function adaptively. After learning, the centres of the NPDFE represent the intensities of the noise signals and the weights render their corresponding probability densities.

4.2 NPDFE Adaptive Learning Procedures

As in the TTE, the NPDFE has two learning stages: the initialisation and the recursion stages.

4.2.1 Initialisation Phase

In the initialisation stage, there is no estimated noise probability density function available for the current ping of transmission. The learning process starts all over from the very beginning with zero number of nodes. Contained in the linked-list, the noise process $\{n_s\}$ that was obtained by removing tentative targets from the first available range-bearing window of measurements in a transmission, and then arranged in an increasing order of intensity, is forwarded one by one to the input node of the NPDE.

If the system does not response to an input n_s, *i.e.* its output $y_s=0$, then add a new node centred at

$$c_k = n_s \qquad (17)$$

and set its corresponding weight to:

$$w_k = \frac{1}{\sum_{\forall j \neq k} w_j + 1} \qquad (18)$$

Otherwise if $y_s \neq 0$, update the centre and the weight of the node with the highest response as follows:

$$c_k = \nabla^- w_k (w_k \cdot c_k + n_s), \qquad (19)$$

$$w_k = w_k + \Delta^+ w_k \qquad (20)$$

where $\nabla^- w_k = \frac{1}{w_k + 1}$ and $\Delta^+ w_k = \frac{1}{\sum_{j \neq k} w_j + 1}$ are defined as a weight decrement factor and a weight increment adaptor, respectively. In both cases, adjust the rest of the weights according to

$$w_{l \neq k} = \sum_{\forall j} \Delta^+ w \cdot w_j \cdot w_l \qquad (21)$$

where the modified weight increment adaptor is defined as $\Delta^+ w = \frac{1}{\sum_j w_j + 1}$.

After processing all the noise elements within the window of the initialisation cell, the resulting centres and their corresponding weights give rise to the probability densities of the noise sequence that will be sent as input to the next functional block to calculate the detection threshold.

4.2.2 Recursion Phase

In the recursion phase, the existing structure of the NPDFE from the previous iteration lays the foundation for the current estimation. One can simply eliminate the effect of the input data that lie outside the window of the next iteration by sliding the window, and then accommodate the contribution of new data that has not been included in the last iteration. The learning procedure can be further divided into the following three steps:

Step 1: Depending on the sliding direction, the noise measurements contained in the range–vector window $R_{i-\overline{M},j}$ or the bearing-vector window $B_{i,j-\overline{N}}$ are presented sequentially as inputs to the NPDFE to eliminate their contribution to the noise probability density estimation for the cell before sliding. This is accomplished by simply justifying the centre and the weight related to the node with the highest response as:

$$c_k = \nabla^+ w_k (w_k \cdot c_k - n_s), \qquad (22)$$

$$w_k = w_k - \Delta^- w_k \qquad (23)$$

where $\nabla^+ w_k = \frac{1}{w_k - 1}$ and $\Delta^- w_k = \frac{1}{\sum_{j \neq k} w_j - 1}$ are defined as weight increment factor and weight decrement adaptor, respectively. The weights of the rest of the nodes are updated subsequently as

$$w_{l\neq k} = \sum_{\forall j} \Delta^{-} w \cdot w_j \cdot w_l \qquad (24)$$

where the modified weight decrement adaptor is defined as $\Delta^{-} w = \dfrac{1}{\sum_j w_j - 1}$.

If the weight after adjustment decreases to zero, *i.e.* $w_k=0$, remove the corresponding node from the network.

Step 2: Slide the window either by $r_i=r_i+1$ or by $b_j=b_j+1$ depending on whether the window slides along the range or the bearing direction.

Step 3: Train the system with the new noise measurements from the range-vector window after sliding $R_{i+\overline{M},j}$, provided the window slides along the range direction or from the bearing-vector window $B_{i,j+\overline{N}}$ if the window slides along the bearing direction, to justify the centres and the weights of the NPDFE. The adjustment approaches taken are the same as those in the initialisation phase described above.

After presenting all the input noise signals to the system once, the training process is terminated for that cell. The centres along with their corresponding weights provide the probability density function of the noise measurements and therefore are taken as the output of the NPDFE and sent to the next functional block.

4.3 The Threshold Generator

In the Threshold Generator functional block, the resultant noise probability density is first stored in a reverse ordered linked-list with reference to the centre values in a decreasing order. Then, with the estimated noise probability density, it is simply a matter of integrating the to the desired false alarm rate by

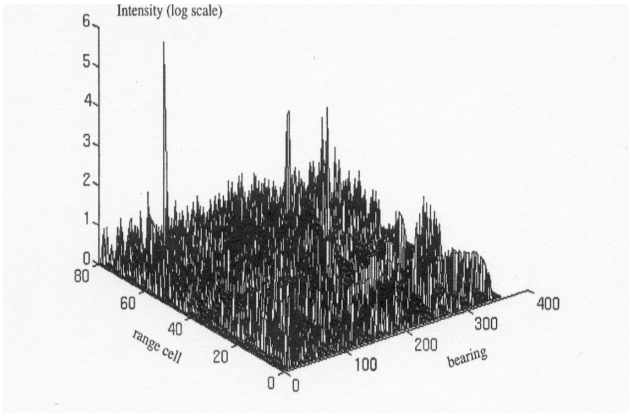
$$\sum_{k=T}^{K} w_k \leq FAR \cap \sum_{k=T-1}^{K} w_k \geq FAR \qquad (25)$$

and the required detection threshold can be obtained as

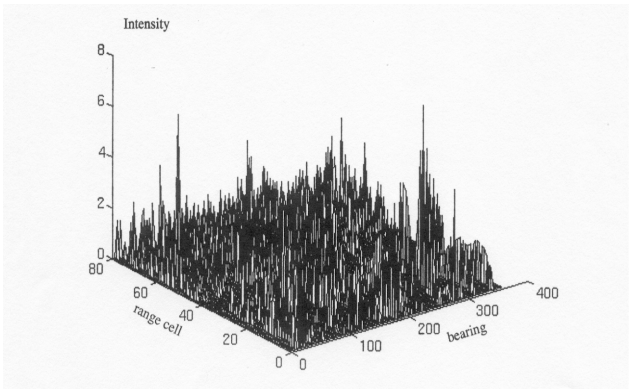
$$DT=c_T \qquad (26)$$

5 Application to active sonar tracking

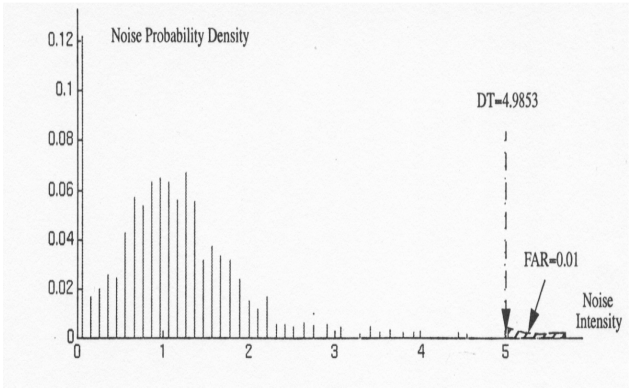
The proposed thresholding scheme has been applied for an active sonar signal tracking system simulator in a realistic sea environment. The above operations are illustrated using an example as shown in Fig. 7. The received measurements from sonar containing both target and noise information is the only viable inputs. Fig. 7a shows a sample of the real sea intensity (in a semilog scale) from underwater active sonar in an omni transmission mode. In this case, the bearing angle varies from 0 to 360 degrees with a step size of 1 degree; whereas the distance or range is discretised into 80 range cells numbered from 1 to 80. The intensity is shown in the log scale because some of the tentative target intensities are much stronger than the average noise intensity. As we have mentioned, the first step is to eliminate the tentative targets from the measurements. The received signals with higher intensity are considered as tentative targets and therefore are deleted form the measurements. Fig. 7b depicts the measurement noise thus obtained.



(a) A sample of the real sea data from underwater sonar



(b) Measurement Noise after Removal of Tentative Targets



(c) Estimated noise probability density function

Fig. 7: Illustration of the proposed scheme

After the removal of the tentative targets, the remaining information is treated as noise, from which the noise probability density function is then estimated. Fig. 7c shows the resulting noise probability density function.

With the estimated noise probability density and a given false alarm rate (FAR=0.01 in this case), shown as the shaded area, the detecting threshold, can be easily obtained as indicated in Fig. 7c.

6 Conclusions

A recursive version of the adaptive CFAR sonar signal thresholding scheme using RBF networks has been proposed in this paper. Both theoretical analysis an experimental results show that the proposed neural network–based recursive thresholding scheme exhibits the following prominent features:

First it yields an acceptable performance under non-homogenous sea environments; the false alarm rate is kept constant while the threshold changes with different, because the false alarm rate is kept constant while the threshold changes with different sea environments. This is due to the fact that the detection threshold is not determined according to a pre-assigned sea noise condition but using the measurements gathered from local information only. Secondly, the proposed scheme can adaptively estimate the threshold for different range cells since the noise signal under estimation is strictly local therefore the received intensities of noise and targets are not affected by the distance travelled by the sonar signals. Finally, the computational requirements has been greatly reduced through the introduction of the recursive scheme due to the fact that the computations required to process one data point reduces from $M \times N$ operations to *2M* or *2N* operations depending on the sliding directions.

References:

[1] Srinivasan, R., Simulation of CFAR detection algorithms for arbitrary clutter distributions, *Radar, Sonar and Navigation, IEE Proceedings,* Vol. 147, No. 1, 2000, pp. 31-40.

[2] Gelfand, S.B., Fortmann, T.E., Bar-Shalom, Y., Adaptive detection threshold optimisation for tracking in clutter, *Aerospace and Electronic Systems, IEEE Transactions on*, Vol. 32, No. 2, 1996, pp. 514-523.

[3] Hennessey, G., Leung, H., Drosopoulos, A., Yip, P.C., Sea-clutter modeling using a radial-basis-function neural network, *Oceanic Engineering, IEEE Journal of*, Vol. 26, No. 3, 2001, 358-372.

[4] Watts, S., The performance of cell-averaging CFAR systems in sea clutter, *Radar Conference, The Record of the IEEE 2000 International*, 2000, pp. 398-403.

[5] Haykin, S., Bhattacharya, T.K., Modular learning strategy for signal detection in a non-stationary environment, *Signal Processing, IEEE Transactions on*, Vol. 45, No. 6, 1997, pp. 1619-1637.

[6] Kodogiannis, V.S., Comparison of advanced learning algorithms for short-term load forecasting,, *JOURNAL OF INTELLIGENT AND FUZZY SYSTEMS*, Vol. 8, No. 4 , 2000, pp. 243-260.