An Algorithm for Non-Maneuvering Target's Kinematic Parameters Estimation Using BPNN

Mimi D. Daneva and Roumen K. Kountchev

Abstract – **An algorithm for tracking on non-maneuvering aircrafts in spherical coordinates with back-propagation neural network (BPNN) is presented. The experiments include Monte Carlo verification of the results with standard recursive Kalman filter for 100 runs. An example of the algorithm's performance using recorded real radar data is shown.**

Keywords **– Radar data processing, neural networks**

I. INTRODUCTION

Radar data processing in automatic systems for air traffic control (ATC) obtains information about the aircraft's spatial location, speed and acceleration, heading angle, etc. Multiple target tracking (MTT) procedures use radar information obtained from several radar scans. In this context the hypotheses for aircraft's motion (maneuvering or non-maneuvering target) and the corresponding kinematic models are used under assumption that the current state of the target is subjected to random perturbations (due to unknown system input or disturbances as turbulence, air draughts, etc.) or maneuvers. Automatic systems for ATC use the estimated target trajectories to control the standard distance between two or more aircrafts with application in Traffic Alert and Collision Avoidance Systems and for flight paths control during critic phases of the flight (for example – approach landing). Different methods for track filtration and prediction are used in practice to estimate the target's parameters (position and velocity), such as recursive Kalman filter (KF), α–β filter, non-linear and adaptive filters [1], [2]. In general case they use probabilities and need to *a priori* knowledge about the statistics of the input data.

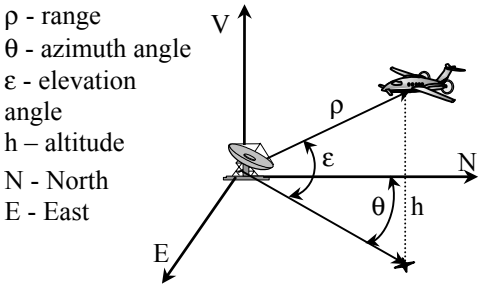


Fig. 1. Tracking in inertial coordinates

The radar data processing algorithms can work in inertial polar or spherical coordinate system (Fig. 1) or in Cartesian coordinates. The choice of the coordinate system depends on the fact whether the information about the target is received from one or more sensors. The multiple-target tracking system includes the following element: sensor data processing and measurement formation; correlation; track initiation, confirmation, and deletion; filtering and prediction; gating. Usually the missed target detection for five consecutive radar scans is used as track deletion criteria [2].

The kinematic model of non-maneuvering target in state space is second-ordered (nearly constant velocity) and is defined by [1], [2]

$$\mathbf{X}(k+1) = \mathbf{\Phi X}(k) + \mathbf{\Gamma \omega}(k) \quad (1)$$

$$\mathbf{Z}(k) = \mathbf{H X}(k) + \mathbf{\nu}(k) \quad (2)$$

where $\mathbf{X}(k) = \left[\mathbf{X_1^T}(k) \quad \mathbf{X_2^T}(k) \quad \mathbf{X_3^T}(k)\right]$ is the state vector of the dynamic system (the aircraft) with components the state vectors $\mathbf{X}_i = [\eta_i \quad \dot{\eta}_i]$ for coordinate i, i=1,2,3. Each vector $\mathbf{X_i}$ contains position η_i and velocity $\dot{\eta}_i$. The symbol i marks the one of the coordinates ρ, θ, and h and is used for notational simplicity. The discrete time interval is noted by k. The vectors $\boldsymbol{\omega}$ and $\boldsymbol{\nu}$ (both of dimension three) are mutually uncorrelated random-valued processes, each with zero mean, known variance and uncorrelated with $\mathbf{X}(0)$. The $\boldsymbol{\omega}(k)$ gives the random velocity's changes. The $\boldsymbol{\nu}(k)$ models the radar measurement errors. The system matrices are defined by

$$\mathbf{\Phi} = \mathrm{diag}[\mathbf{\Phi_2} \quad \mathbf{\Phi_2} \quad \mathbf{\Phi_2}];$$

$$\mathbf{\Gamma} = \mathrm{diag}[\mathbf{\Gamma_2} \quad \mathbf{\Gamma_2} \quad \mathbf{\Gamma_2}];$$

$$\mathbf{H} = \mathrm{diag}[\mathbf{H_2} \quad \mathbf{H_2} \quad \mathbf{H_2}]$$

where $\mathbf{\Phi_2} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}; \mathbf{\Gamma_2} = \begin{bmatrix} T^2/2 \\ T \end{bmatrix}; \mathbf{H_2} = [1 \quad 0]$. The radar sampling time is denoted by T. The noise covariance matrices are

$$\mathbf{Q} = [q_{ij}] = E[\boldsymbol{\omega}(k)\boldsymbol{\omega}(k)^T]; \quad \mathbf{R} = [r_{ij}] = E[\boldsymbol{\nu}(k)\boldsymbol{\nu}^T(k)]$$

where the superscript T denoted the transpose operator.

In this paper an algorithm for non-maneuvering aircraft tracks filtration and first order prediction using multiple-layered perceptrons neural network and error back-propagation learning algorithm is presented. Position-only radar measurements (range, azimuth, and altitude) in inertial coordinate system are used. The specific application of the algorithm is related to high accuracy estimation problem. The performance of the BPNN tracking filter is compared with recursive KF for 100 Monte Carlo runs in MATLAB environment. An illustrative example show the capability of the BPNN algorithm using recorded real input data from Monopulse Secondary Surveillance Radar CMSSR–401 [3].

II. BPNN FOR PREDICTION

Multiple-layered perceptrons NNs with BP training algorithm pertain to the non-recursive class NNs and have a, and classification problems in pattern recognition, theory of

Mimi D. Daneva is from the Faculty of Communications and Communicational Technologies, 8 Kl. Ochridski str., 1000 Sofia, Bulgaria, e-mail: mimidan@vmei.acad.bg

Roumen K. Kountchev is from the Faculty of Communications and Communicational Technologies, Technical University of Sofia, 8 Kl. Ochridski str., 1000 Sofia, Bulgaria, e-mail: rkountch@vmei.acad.bg

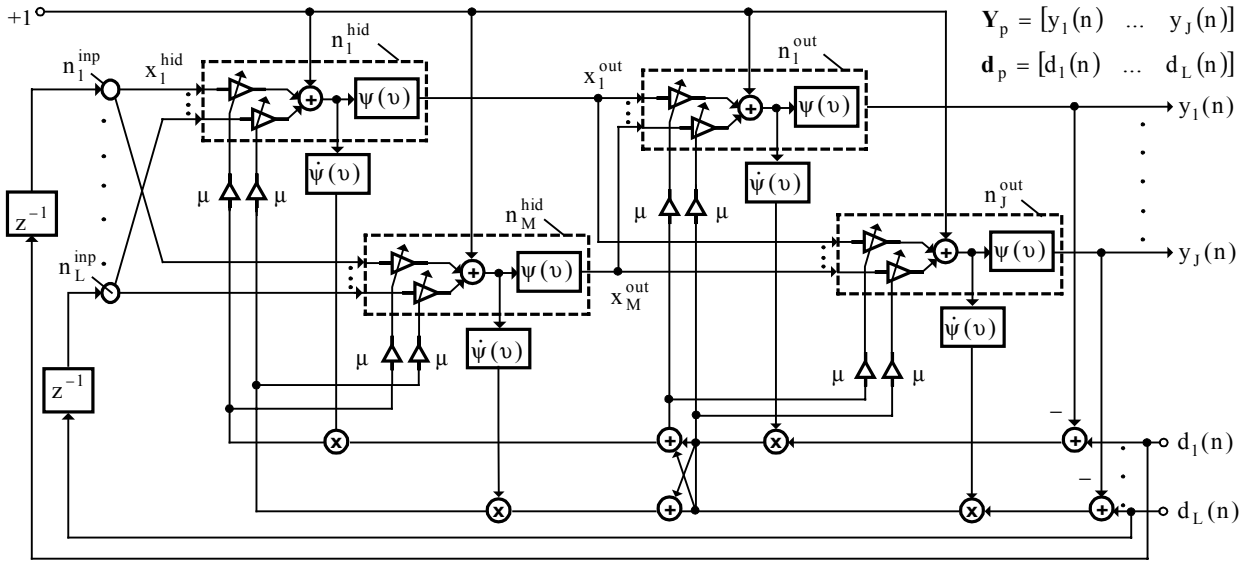


Fig. 2. BPNN for one-step-ahead prediction

automatic control, data processing, etc [4]. The standard architecture of BPNN with one hidden layer [5] for first order prediction is shown in Fig. 2, where z^{-1} denotes the unit delay operator. The input, hidden, and output neurons are denoted by $n_l^{inp}, (l=1,..,L)$, $n_m^{hid}, (m=1,..,M)$, and n_j^{out}, $(j=1,..,J)$, respectively. The internal structure of the input neurons is not shown for simplicity, because they have only sensitive function in the learning process. The activation function for neuron i is sigmoid nonlinearity defined by

$$\psi_i(\upsilon_i) = \alpha \tanh(\gamma \upsilon_i) \tag{3}$$

where α and γ are constants, υ_i is the internal activity level of the neuron i. In some cases is suitable to use linear output neurons.

The error correction learning [5], [6] uses the error back propagated signal (layer by layer) for weights adjustments till the NN output vector $\mathbf{Y}_p(n)$ become closest to the desired response $\mathbf{d}_p(n)$ at iteration n for the learning example p. The local error for neuron n_j^{out} is

$$e_j(n) = d_j(n) - y_j(n) \tag{4}$$

It is propagated in backward manner in the neural structure from the output layer to the hidden layer, and the local gradient for each neuron is computed recursively. The standard BP algorithm uses the steepest-descent gradient approach to minimize the mean-squared error function. The local error function for neuron j for pattern p is defined by

$$\xi_p(n) = \frac{1}{2} e_j^2(n) = \frac{1}{2}\left(d_j(n) - y_j(n)\right)^2 \tag{5}$$

The global error for batch mode learning is obtained after the presentation of all the training examples. It is used as cost function (net performance function) and is defined by

$$E = \frac{1}{2p_{max}} \sum_{p=1}^{p_{max}} \sum_{j=1}^{J} e_j^2(p) \tag{6}$$

where p_{max} is the last training example

The equations of the standard batch mode BP algorithm for the output and the hidden layer are, as follow

$$y_i(p) = \psi(\upsilon_i(p)); \ \upsilon_i(p) = \sum_{s=1}^{s_{max}} w_{is}(p) y_i(p) \tag{7}$$

where s_{max} - total number of inputs (excluding the threshold) applied to neuron i;

$$\Delta \mathbf{w}_{is}(n) = \mu \delta_i(n) y_i(n) \tag{8}$$

$$\delta_i(n) = -\frac{\partial \xi_p(n)}{\partial e_i(n)} \frac{\partial e_i(n)}{\partial y_i(n)} \frac{\partial y_i(n)}{\partial \upsilon_i(n)} \tag{9}$$

where $\Delta \mathbf{w}_{is}(n)$ and $\delta_i(n)$ are the weight changes and local error gradient for neuron i; μ is the learning-rate parameter. The local gradients for the output and the hidden neurons are defined as [5]

$$\delta_j(n) = e_j(n) \dot{\psi}_j(\upsilon_j(n)) \tag{10}$$

$$\delta_m(n) = \dot{\psi}(\upsilon_m(n)) \sum_m \delta_j(n) \mathbf{w}_{mj}(n) \tag{11}$$

When the local approximation of Eq. (5) around the current point $\mathbf{w}(n)$ is used, the minimization is performed at each iteration of the algorithm. The weight change can be obtained as

$$\Delta \mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = -\mu \mathbf{g}(n) \tag{12}$$

where $\mathbf{g} = \nabla_{\mathbf{w}} \xi_p(\mathbf{w})$ is the gradient vector.

The cost function in the neighborhood of point $\mathbf{w}(n)$ using Taylor series is defined by

$$\Delta \xi_p(\mathbf{w}(n)) = \xi_p(\mathbf{w}(n+1)) - \xi_p(\mathbf{w}(n)) \cong \mathbf{g}^T(n) \Delta \mathbf{w}(n) + \frac{1}{2} \Delta \mathbf{w}^T(n) \overline{\mathbf{H}}(n) \Delta \mathbf{w}(n) \tag{13}$$

where $\overline{\mathbf{H}}(n) = \nabla^2 \xi_p(\mathbf{w}(n))$ is the Hessian matrix. As final result of the Eq. (13)'s differentiation $\mathbf{w}(n+1)$ is obtained as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta \mathbf{w}(n) = \mathbf{w}(n) - \overline{\mathbf{H}}^{-1}(n) \mathbf{g}(n) \tag{14}$$

The Levenberg-Marquardt learning algorithm [5] approximates the Hessian matrix by the matrix

$$\left[\nabla^2 \xi_p(\mathbf{w}(n)) + v\mathbf{I}\right], \ v \geq 0, \ \lim_{t \to \infty} v = 0 \tag{15}$$

where **I** - identity matrix.

III. PROPOSED ALGORITHM

The proposed algorithm for tracking on non-maneuvering targets uses BPNN for track filtration and one-step-ahead prediction to form the estimate of the current and future kinematic state variables (position and velocity) from position-only radar measurements. The data about the target's motion are received in cylindrical coordinates (range ρ, azimuth θ, and altitude h. The kinematic model is described by Eqs. (1), and (2). The normalization procedure is used for pattern formulation. It arranges the input data randomly in the interval [-0,9, 0,9] and prevents the hidden weights to get into the zones of high non-linearity of the activation functions [6]. Some BPNNs with different number of hidden nodes are investigated in heuristic way to find the optimal architecture for track filtering and prediction. It includes an input layer with L=3 neurons, a single hidden layer with M=15 neurons, and an output layer with J=3 neurons. The neurons in each layer have full interconnections with the neurons in the previous layer. The input and hidden units have bipolar sigmoidal activation functions with biases. The output units are linear and perform the linear combination of the hidden neurons' reactions to form the estimated value of the state vector $\hat{\mathbf{X}}^*(\mathrm{k}+1)$. The prediction error is defined as the difference between the pattern vector $\mathbf{P}(\mathrm{k}+1)$ and $\hat{\mathbf{X}}^*(\mathrm{k}+1)$ [5].

The algorithm includes the following steps.

Step 1. Data normalization to obtain the pattern vector **P** as

$$\mathbf{P} = \frac{1.8(\mathbf{Z} - \mathrm{Z}_{\min})}{(\mathrm{Z}_{\max} - \mathrm{Z}_{\min})} - 0.9 \qquad (16)$$

Step 2. BPNN initialization: The Nguyen-Widrow hidden weights initialization procedure [6] is used. It prevents all the hidden weights from premature saturation during the first few iterations.

Step 3. The training data set presentation: The training set is formed using the measurements received from the first five radar scans. The training set size is chosen for track deletion criteria [2].

Step 4. Forward computations: Compute the net internal activity levels and the output signals of all the neurons in the layers according the Levenberg-Marquardt learning algorithm.

Step 5. Error back-propagation: Compute the vectors of local error gradients for all the neurons in the output and the hidden layer using the same training algorithm.

Step 6. Iterations till the global error minimum is found or the maximum number of epochs or the maximum learning rate is achieved.

Step 7. Presentation of the next (unknown) data set from the same track with the same size as the training set and go back to *Step 4*, *Step 5* and *Step 6* to perform the prediction phase of the algorithm.

Step 8. Repeat cyclically the *Step 4* to *Step 7* till the track end is found.

Step 9. Recovering the original variables by data unnormalization procedure according to the inverse formula of (16).

IV. EXPERIMENTAL RESULTS

The simulated input data and real radar data record from Monopulse Secondary Surveillance Radar CMSSR-401 are used for the experiments. The radar sample time is T=5 s [3]. The modeled noises $\boldsymbol{\omega}(\mathrm{k})$, $\mathbf{v}(\mathrm{k})$, and the dynamic system's driving input vector $\mathbf{u}(\mathrm{k})$ have Gaussian distribution with zero mean and known variances. The Gaussian cumulative distribution functions of the noises are verificated with χ^2–test and significant level $\alpha = 0.05$. The acceleration standard deviation is $\sigma_\omega = 2\mathrm{g}$ [2] according to the airworthiness for non-maneuvering aircrafts. The $\mathbf{v}(\mathrm{k})$'s standard deviations are $\sigma_{v_\rho} = 0{,}05$ nmi, $\sigma_{v_\theta} = 0{,}07$ deg, and $\sigma_{v_h} = 100$ feet for range, azimuth and altitude, respectively [3]. The corresponding driving input's variance is assumed to be three times larger than the measurement error variance [2].

The BPNN training parameters and the required CPU time and flops for training and prediction phases are shown in Table I. Three cases are considered. The same input data for one random simulated track are used for N=500 runs of the algorithm (Case I) to obtain the optimal NN architecture. The averaged training parameters epochs $\overline{\mathrm{n}}_{\mathrm{ep}}^{(\mathrm{tr})}$, the net performance function $\overline{\mathrm{E}}$ and gradients in respect to each coordinate $\overline{\nabla}\mathrm{E}_\rho, \overline{\nabla}\mathrm{E}_\theta, \overline{\nabla}\mathrm{E}_\mathrm{h}$, the required CPU time and flops for training and prediction phases are compared for different number of hidden nodes. The BPNN with M=15 is chosen as optimal, because the training is faster and requires less CPU time and flops than the other architectures. The net performance function and gradients in this case are approximately from the same order than the cases with the other M. The trained net does not need to additional iterations during the processing of the unknown data sets. The performance of the algorithm is compared using standard recursive Kalman filter with Monte Carlo experiment of N=100 runs (Case II). The same parameters for an example of

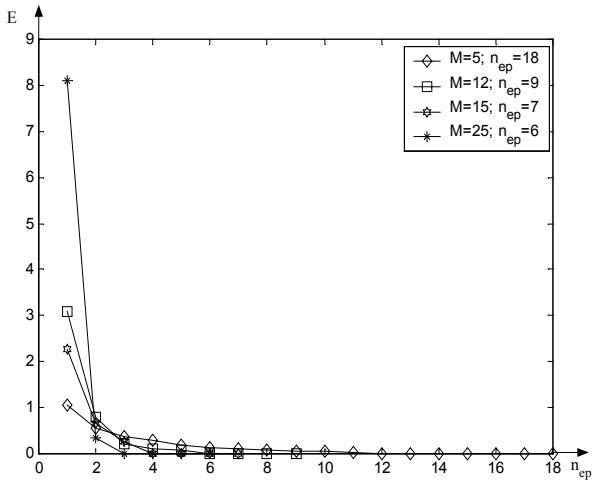
TABLE I
BPNN PARAMETERS DURING THE TRAINING

Case	I.				II.	III.
M	5	12	15	25	15	15
$\overline{\mathrm{n}}_{\mathrm{ep}}^{(\mathrm{tr})}$	15	6	5	5	5	6
$\overline{\mathrm{E}}.10^{-23}$	27,80	1,67	3,90	0,63	0,12	0,10
$\overline{\nabla}\mathrm{E}_\rho.10^{-14}$	-0,61	-0,02	0,49	-0,01	-0,05	-2,38
$\overline{\nabla}\mathrm{E}_\theta.10^{-14}$	-0,22	-0,23	0,02	-0,03	-0,05	-3,95
$\overline{\nabla}\mathrm{E}_\mathrm{h}.10^{-14}$	-0,41	-0,03	0,05	-0,28	0,02	-7,80
$\overline{\mu}_{\max}.10^{-02}$	1,67	0,38	10	0,16	0,14	0,10
$\overline{\mathrm{t}}_{\mathrm{CPU}}^{(\mathrm{tr})}$, s	4,72	3,24	4,01	3,90	3,41	3,02
$\overline{\mathrm{t}}_{\mathrm{CPU}}^{(\mathrm{pr})}$, s	0,42	0,38	0,39	0,45	0,47	0,37
$\mathrm{n}_{\mathrm{flops}}^{(\mathrm{tr})}.10^6$	2,007	4,619	8,059	20,898	6,609	5,627
$\mathrm{n}_{\mathrm{flops}}^{(\mathrm{pr})}.10^6$	0,050	0,239	0,364	0,972	0,364	0,364

TABLE II
MONTE CARLO RESULTS

Tracking Filter	BPNN			KF
	Case	Training	Prediction	
$m_{\bar{\zeta}_\rho}$, nmi	II.	$6{,}75.10^{-15}$	$3{,}37.10^{-13}$	$6{,}11.10^{-07}$
	III.	$1{,}83.10^{-10}$	$1{,}83.10^{-10}$	$1{,}06.10^{-06}$
$m_{\bar{\zeta}_\theta}$, deg	II.	$1{,}99.10^{-14}$	$9{,}03.10^{-13}$	$1{,}07.10^{-06}$
	III.	$2{,}38.10^{-14}$	$2{,}49.10^{-10}$	$1{,}60.10^{-06}$
$m_{\bar{\zeta}_h}$, feet	II.	$6{,}84.10^{-14}$	$9{,}88.10^{-13}$	0,6302
	III.	0	0	0,5439
$\sigma_{\bar{\zeta}_\rho}$, nmi	II.	$1{,}11.10^{-29}$	$5{,}32.10^{-13}$	$2{,}65.10^{-07}$
	III.	$2{,}37.10^{-14}$	$2{,}24.10^{-10}$	$5{,}28.10^{-07}$
$\sigma_{\bar{\zeta}_\theta}$, deg	II.	$5{,}39.10^{-29}$	$4{,}94.10^{-12}$	$4{,}58.10^{-07}$
	III.	$4{,}03.10^{-14}$	$3{,}62.10^{-14}$	$1{,}01.10^{-06}$
$\sigma_{\bar{\zeta}_h}$, feet	II.	$1{,}40.10^{-28}$	$5{,}36.10^{-12}$	0,0400
	III.	0	0	0,4250
$\bar{t}_{CPU}$, s	II.	4,55	0,44	0,11
	III.	3,02	0,37	0,17
$n_{flops}, .10^6$	II.	8,059	0,364	0,405
	III.	5,627	0,364	0,405

real recorded track with constant altitude are presented by Case III. The statistics (mean and standard deviation) of the averaged for all runs absolute values of the tracking errors $\bar{\varsigma}^{BPNN}$ and $\bar{\varsigma}^{KF}$ in Cases II and III are shown in Table II. It is clear that the error statistics for BPNN algorithm in all the cases are smaller than the KF error statistics. The net performance functions in Case III in respect to the epoch's number and the number of hidden units M are plotted in Fig.3. The components of the tracking errors of BPNN algorithm and the KF in relation to each coordinate are plotted in Fig. 4.



The absolute error statistics due to recovering after unnormalization in respect to each coordinate are shown in Table III.

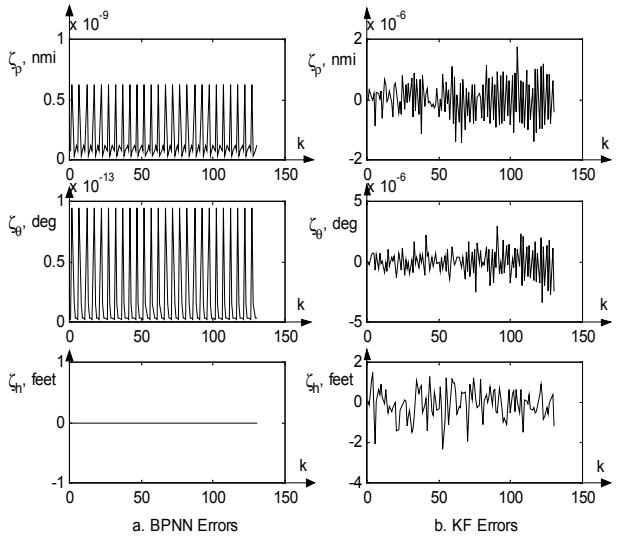


Fig. 3. BPNN performance functions versus the epoch's number

Fig. 4. Tracking errors for Case III

All the results are obtained by Intel Celeron 500 PPGA with SDRAM 128 MB.

TABLE III
ERROR DUE TO RECOVERING: STATISTICS

Coordinate	ρ, nmi	θ, deg	h, feet
$m_{\bar{\xi}^{NP}}.10^{-14}$	0,66	0,61	0,75
$\sigma_{\bar{\xi}^{NP}}.10^{-14}$	0,80	0,81	0,98

V. CONCLUSION

This paper has presented the algorithm for target's kinematic parameters estimation using BPNN. The comparative analysis of the algorithm's performance based on Monte Carlo experiment using recursive Kalman filter is done. The tracking error of the trained net is negligible larger than the tracking error during the training. The error due to recovering after normalization does not affect to the algorithm's accuracy. A parallel hardware implementation of the algorithm using transputer or digital signal processing modules or programmable neural networks modules will reduce the CPU time. It will reflect positively onto the data association, which is the other important problem in MTT and will increase the safety level of ATC.

REFERENCES

[1] A. Farina, F. A. Studer, Radar Data Processing, vol. I, Letchworth, Research Studies Press, 1985.

[2] S. Blackman, Multiple Target Tracking with Radar Applications, Norwood, Artech House, 1986.

[3] *Monopulse Secondary Surveillance Radar System Description*, Technical Report, Cardion Inc., *Report no. 131-162A.*

[4] T. H. Kerr, "Critique of Some Neural Network Architectures and Claims for Control and Estimation", *IEEE Trans., Aerospace and Electronic Systems,* vol. 34, no. 2, pp. 406-418, 1998.

[5] S. Haykin, *Neural Networks*, New York, Macmillan College Publishing Company, 1994.

[6] C. G. Looney, *Pattern Recognition Using Neural Networks*, New York, Oxford University Press, 1997