A BPNN Tracking Filter Using Canonical Transform onto the Kinematic Model

Mimi D. Daneva and Tzanko P. Georgiev

Abstract – **An algorithm for tracking on non-maneuvering targets with back-propagation neural network (BPNN) is presented. The canonical transform and normalization procedure are used for pattern formulation. The algorithm's performance is compared with recursive Kalman filter based on Monte Carlo experiment and an example using real radar data record in MATLAB environment.**

Keywords – **Data processing, neural networks, radar**

I. INTRODUCTION

The accuracy problem in radar data processing in Cartesian coordinates occupies an important place in track fusion, where the data about the target's parameters are obtained from several sensors. The target's kinematic model in this case is coupled between the coordinates. The cause dues to the fact that the target's positions are actually measured in spherical coordinates and their convertion to Cartesian coordinates sometimes is attended by a pseudolinear error coupling. In this case the track filtration and extrapolation using the standard proved in practice methods as recursive Kalman filter (KF) without an additional preprocessing in case of long data set size leads to poor filter's performance. The cause is the interdependency of the components of the Cartesian measurement vector from the polar range between the target and the sensor, which leads to non-stationary measurement covariance matrix and a lack of convergence of the KF. The canonical transform (CT) converts the coupled target dynamic model into a decoupled canonical form that is independent in respect to the time (radar sampling interval) and the space (the range radar-target) [1], [2]. The CT is obtained by simultaneously diagonalizing the noise covariance matrices of the kinematic model, followed by a spatial-temporal normalization procedure. It helps for more efficient implementation of the tracking algorithms.

The track life stages are initiation, confirmation and deletion. Usually, the missed measurements for five sequential radar scans are used as track deletion criteria [2].

The data processing algorithms using neural network [3], [4] do not need to *a priory* knowledge about the statistics of the input data. BPNN initialization set all the synaptic weights and biases to small uniformly distributed random numbers. The Nguyen-Widrow hidden weight initialization procedure is recommended for multiple-layered perceptrons neural networks. It prevents premature saturation at all hidden neurons, which contributes the learning convergence and ensures more efficient training [3]. Different error back-propagation algorithms are well described in [3], [4]. The Levenberg-Marquardt algorithm is required to evaluate only first-order derivatives, always is stable and needs to less CPU time and flops than the other training algorithms.

In this paper an algorithm for track filtration and first order prediction in Cartesian coordinates using back-propagation neural network (BPNN) is presented. The net training performance function (mean squared error, MSE) for BPNN with different number of hidden units is analyzed. The Monte Carlo verification of the results with recursive Kalman filter (KF) [2] for 100 runs is presented using MATLAB package. An illustrative example with real radar data record from Monopulse Secondary Surveillance Radar CMSSR-401 [5] is shown.

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

Tzanko Georgiev is from the Faculty of Automation, Technical University of Sofia, 8 Kl. Ochridski str., 1000 Sofia, Bulgaria, e-mail: tzg@vmei.acad.bg.

II. THE CANONICAL TRANSFORM ONTO THE KINEMATIC MODEL

The model of the non-maneuvering target motion in Cartesian coordinate system is second-ordered (with nearly constant velocity). The associated state equations are [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 (the first derivative of the position) $\dot{\eta}_i$. The symbol i marks the one of the three Cartesian coordinates x, y, and z and is used for notational simplicity. The discrete time interval is noted by k. The matrices w and ν (both of dimension three) are mutually uncorrelated vector-valued stochastic processes, each with zero mean, known variance and uncorrelated with $\mathbf{X}(0)$. The $\mathbf{\omega}(k)$ gives the presence of the velocity's changes. The $\mathbf{\nu}(k)$ models the radar measurement errors. The noise covariance matrices are defined by

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

where the superscript T denoted the transpose operator. 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 letter T signifies the radar sampling time.

The canonical transform (CT) onto the kinematic state model is obtained by simultaneously diagonalizing the noise covariance matrices Q and R followed by a spatial-temporal normalization procedure. The coupling between the

coordinates finds expression in the off-diagonal elements of the covariance matrices only. The CT is required for commutative matrices Q and R, i. e. to exist their orthogonal matrices. In general case Q and R are not commutative, so it is necessary to diagonalize them. The requirement for their diagonalization is both of them must be symmetric matrices, and R – positive definite. So, the transformation matrix M, which simultaneously diagonalizes Q and R, is orthogonalized and normalized matrix with respect to R, and is defined by

$$\mathbf{M} = \left[m_{ij}\right] = \left[\mathbf{e_1} \quad \mathbf{e_2} \quad \mathbf{e_3}\right] \tag{3}$$

where $\mathbf{e_i}, i = 1,2,3$ are a set of R-orthonormal eigenvectors, which correspond to the eigenvalues $\lambda_i^2, i = 1, 2, 3$, obtained by generalized singular value decomposition of the matrix $\left[\mathbf{R}^{-1}\mathbf{Q}\right]$. By using the Gram-Schmidt procedure, all $\mathbf{e_i}, i = 1, 2, 3$ are **R**-orthonormalized, i. e. the following conditions

$$\mathbf{Q}\mathbf{e_i} = \lambda_i \mathbf{R}\mathbf{e_i}, \quad i = 1, 2, 3 \tag{4}$$

$$\mathbf{e_i}^T \mathbf{R} \mathbf{e_j} = \delta_{ij}, \quad i, j = 1, 2, 3 \tag{5}$$

are implemented. In Eq. (5) δ_{ij} is the Kronecker delta function.

After the transformation by the matrix **M**, the covariance matrices of the kinematic model become

$$\widetilde{\mathbf{Q}} = \mathbf{M}^T \mathbf{Q} \mathbf{M} = \mathrm{diag}\left[\lambda_1^2 \quad \ldots \quad \lambda_3^2\right] \tag{6}$$

$$\widetilde{\mathbf{R}} = \mathbf{M}^T \mathbf{R} \mathbf{M} = \mathbf{I} \tag{7}$$

where $\mathbf{I}_{(3x3)}$ denotes $(3x3)$ identity matrix. Then, the space dimensionless form of the dynamic system described by Eqs. (1) and (2) is

$$\widetilde{\mathbf{X}}(k+1) = \mathbf{F}\widetilde{\mathbf{X}}(k) + \mathbf{G}\widetilde{\boldsymbol{\omega}}(k) \tag{8}$$

$$\widetilde{\mathbf{Z}}(k) = \mathbf{H}\widetilde{\mathbf{X}}(k) + \widetilde{\mathbf{v}}(k) \tag{9}$$

where

$$\widetilde{\mathbf{X}} = \mathbf{M}^T \mathbf{X}; \widetilde{\mathbf{Z}} = \mathbf{M}^T \mathbf{Z}; \widetilde{\boldsymbol{\omega}} = \mathbf{M}^T \boldsymbol{\omega}; \widetilde{\mathbf{v}} = \mathbf{M}^T \mathbf{v} \tag{10}$$

The system described by Eqs. (8) and (9) is mutually uncorrelated between the coordinates (x, y, z) in the sense that the state vector's components and their estimations with respect to the detached coordinates are independent each other. The formulas for recovering the original variables are

$$\mathbf{X} = \left[\mathbf{M}^T\right]^{-1} \widetilde{\mathbf{X}}; \mathbf{Z} = \left[\mathbf{M}^T\right]^{-1} \widetilde{\mathbf{Z}}; \tag{11}$$

$$\boldsymbol{\omega} = \left[\mathbf{M}^T\right]^{-1} \widetilde{\boldsymbol{\omega}}; \mathbf{v} = \left[\mathbf{M}^T\right]^{-1} \widetilde{\mathbf{v}} \tag{12}$$

Next the dynamic system must be transformed in relation to the time by setting the radar sample interval T=1. Then the CT of the system with Eqs. (1) and (2) is determined as [1]

$$\mathbf{X}^*(k+1) = \mathbf{F}^*\mathbf{X}^*(k) + \mathbf{G}^*\boldsymbol{\omega}^*(k) \tag{13}$$

$$\mathbf{Z}^*(k) = \mathbf{H}^*\mathbf{X}^*(k) + \mathbf{v}^*(k) \tag{14}$$

where all the vectors and matrices are physically dimensionless and

$$\mathbf{X}^* = \mathbf{A}\widetilde{\mathbf{X}} = \mathbf{A}\mathbf{M}^T\mathbf{X}; \mathbf{Z}^* = \widetilde{\mathbf{Z}} = \mathbf{M}^T\mathbf{Z}; \tag{15}$$

$$\boldsymbol{\omega}^* = T^2\widetilde{\boldsymbol{\omega}} = T^2\mathbf{M}^T\boldsymbol{\omega}; \mathbf{v}^* = \widetilde{\mathbf{v}} = \mathbf{M}^T\mathbf{v}; \tag{16}$$

$$\mathbf{F}^* = \mathbf{F}_{|T=1}; \mathbf{G}^* = \mathbf{G}_{|T=1}; \mathbf{H}^* = \mathbf{H}_{|T=1} = \mathbf{H} \tag{17}$$

where $\mathbf{A} = \mathrm{diag}\left[\mathbf{A_2} \quad \mathbf{A_2}\right]$, $\mathbf{A_2} = \mathrm{diag}\left[1 \quad T\right]$ for the second-order kinematic model.

III. PROPOSED BPNN TRACKING FILTER

The proposed tracking filter uses BPNN for track filtration and one-step-ahead prediction to form the estimate of the current and future kinematic state variables of non-maneuvering aircraft (position and velocity) from position-only radar measurements. The data about the target's motion are presented in 3-dimensional Cartesian coordinate system. The kinematic model is described with Eqs. (1), and (2). The CT is used as additional radar data preprocessing before normalization procedure and the next processing with BPNN. For verification of the BPNN tracking filter's performance the same transformed input data are processed with recursive KF for 100 Monte Carlo runs.

The block diagrams of the data processing with BPNN and KF using CT are shown in Fig. 1.

The optimal BPNN architecture for filtering and first order prediction [3] in this case includes an input layer with $n_{inp} = 3$ neurons, a hidden layer with $n_{hid} = 15$ neurons, and an output layer with $n_{out} = 3$ neurons. The number of the hidden units is chosen according to heuristic rule. The neurons in each layer are full connected 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 $\hat{\mathbf{X}}^*(k+1)$. The prediction error is defined as the difference between the pattern vector $\mathbf{P}(k+1)$ and $\hat{\mathbf{X}}^*(k+1)$ [3]. In general case the prediction with BPNN is described with the equations

$$\begin{aligned}\hat{\mathbf{X}}^*(k+1) = \Re_N\{f_{out}[f_{hid}[f_{inp}[\mathbf{P}(k), \\ \mathbf{w}_{in}(k)], \mathbf{w}_{hid}(k)], \mathbf{w}_{out}(k)\}\end{aligned} \tag{18}$$

$$\mathbf{w}(k) = \Re_L\{e, \mu, \mathbf{w}(k-1)\} \tag{19}$$

where f_{out}, f_{hid}, f_{inp} are the activation functions of the output, hidden, and input layer neurons; $\mathbf{w}_{inp}$, $\mathbf{w}_{hid}$, $\mathbf{w}_{out}$ are their corresponding weight matrices. The relations $\Re_N$ and $\Re_L$ describe the BPNN architecture and learning algorithm. The net error is denoted by e, and μ is the learning rate.

The algorithm for tracking filter with BPNN includes the following steps.

Step 1. Canonical transform using Eq. (13) and Eq. (14).

Step 2. Data normalization procedure to obtain the pattern vector **P** as

$$\mathbf{P} = \frac{1.8\left(\mathbf{Z}^* - Z^*_{min}\right)}{\left(Z^*_{max} - Z^*_{min}\right)} - 0.9 \tag{20}$$

to avoid of the hard non-linearity zones of the neurons' activation functions. It prevents the synaptic weights of the hidden units from premature saturation at the first few iterations.

Step 3. BPNN initialization: The Nguyen-Widrow hidden weights initialization procedure is used.

Step 4. The training data set loading: The training set is formed using the measurements received from the first five ra

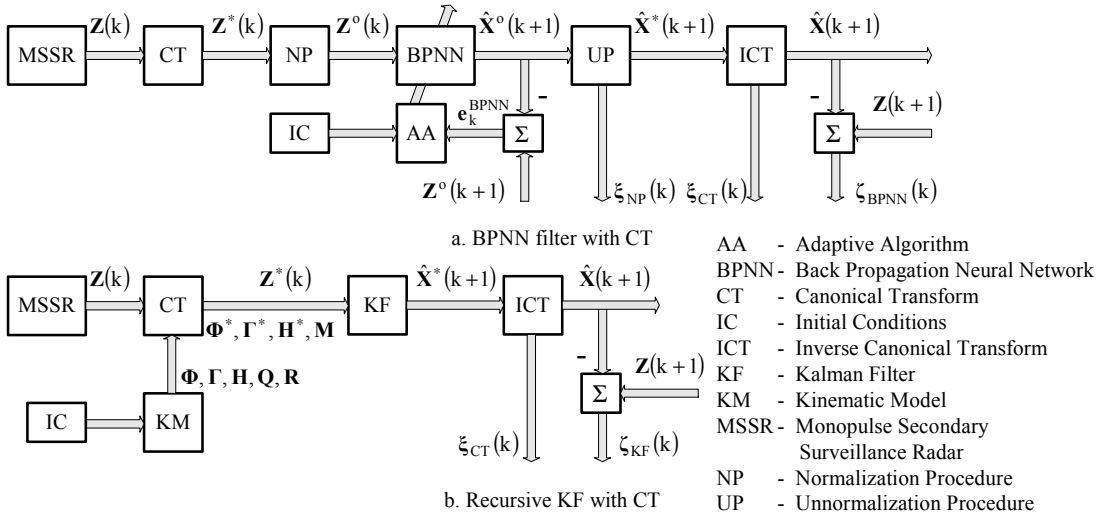

Fig. 1. Block diagram for track filtering and prediction using CT

dar scans because of the track deletion criteria, i. e. the measurement vector $\mathbf{Z}^*(k), k = 1,..,5$ is used.

Step 5. Forward computations: Compute the net internal activity levels and the output signals of all the neurons in the layers according the Levenberg-Marquardt training algorithm.

Step 6. Error back-propagation: Compute the vectors of local error gradients in the output and the hidden layer using the same training algorithm.

Step 7. Iterations till the global error minimum is found or the maximum number of epochs or the maximum learning rate is achieved.

Step 8. Load the next (unknown) data set from the same track with batch size 5 and go back to *Step 5*, *Step 6* and *Step 7*.

Step 9. Repeat cyclically the *Steps 5* to *8* till the track end is found.

Step 10. Data unnormalization procedure using the inverse formula of Eq. (20).

Step 11. Recovering the original variables with inverse canonical transform according to Eq. (11) and Eq. (12).

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 [5]. The modeled noises $\boldsymbol{\omega}(k), \mathbf{v}(k)$ and the dynamic system driving input $\mathbf{u}(k)$ 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 $\sigma_\omega = 2g$ [2] according to the airworthiness for non-maneuvering aircrafts. The $\mathbf{v}(k)$'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 [5]. The corresponding driving input's variance is assumed to be three times larger than the measurement error variance [2]. The original radar measurements are received in spherical coordinates range ρ in nmi, azimuth θ in deg, and altitude h in feet. They are transformed in Cartesian coordinates (x, y, z) each with dimension in meters.

The BPNN training parameters, CPU time and flops for training and prediction phase are shown in Table I. They are averaged (denoted by the symbol " ¯ ") for N runs of the algorithm. Some neural architectures with different number of hidden units are investigated. The same input data for one simulated track chosen in random way are used for N=500.The different data at each run of Monte Carlo experiment with N=100 are used. The parameters are denoted, as follow: n_{hid} - number of hidden neurons; $\bar{n}_{ep}^{(tr)}$ - epochs; $\bar{E}$ - averaged net performance function (MSE) with averaged gradients $\bar{\nabla}E_{x^o}, \bar{\nabla}E_{y^o}, \bar{\nabla}E_{z^o}$. The CPU time and flops required for training and prediction are denoted by $\bar{t}_{CPU}^{(tr)}, \bar{t}_{CPU}^{(pr)}, \bar{n}_{flops}^{(tr)}, \bar{n}_{flops}^{(pr)}$, respectively. Three cases are considered. Case I and Case II consider the BPNN tracking filter with CT and without CT, respectively, using simulated data. Case III represents the BPNN performance with CT when an example real track with length 130 scans is used. The BPNN with $n_{hid} = 15$ is chosen as optimal, because the training is faster and requires less CPU time and flops than the other architectures. On the other hand, the net performance function and gradients are approximately from the same order. The trained net does not need to additional iterations during the processing of the unknown data sets. The performance of the BPNN filter is compared with KF for 100 Monte Carlo runs. The mean and standard deviation of the averaged for all runs absolute values of the tracking errors $\bar{\varsigma}^{BPNN}$ and $\bar{\varsigma}^{KF}$ are shown in Table II. The root mean-squared filter's tracking error is defined by

$$\zeta_{RMS} = \sqrt{(x - \hat{x})^2 + (y - \hat{y})^2 + (z - \hat{z})^2} \qquad (21)$$

Table III contains the Monte Carlo statistics of the root means squared (RMS) positional errors due to recovering the original variables after CT and NP. The net performance functions in Case III in respect to the epoch's number and n_{hid} are plotted on the top Fig. 2. The RMS tracking positional error for BPNN during the training (scans 1,..,5) and prediction phase (unknown data sets processing by the

TABLE I
BPNN PARAMETERS DURING THE TRAINING PHASE

Case			n_{hid}	$\bar{n}_{ep}^{(tr)}$	$\bar{E}$	$\bar{\nabla}E_{x^o}$	$\bar{\nabla}E_{y^o}$	$\bar{\nabla}E_{z^o}$	$\bar{\mu}_{max}$	$\bar{t}_{CPU}^{(tr)}$	$\bar{t}_{CPU}^{(pr)}$	$n_{flops}^{(tr)}$	$n_{flops}^{(pr)}$
							$.10^{-14}$			s	s	$.10^6$	$.10^6$
k=1,..,505	N=500	I.	5	23	$1{,}66.10^{-21}$	0,60	-0,85	0,69	0,0275	3,19	0,45	2,731	0,058
			15	6	$1{,}50.10^{-23}$	-0,21	-0,44	-0,10	0,0024	3,19	0,46	12,517	0.372
			20	5	$2{,}88.10^{-24}$	-0,17	-0,75	0,05	0,0017	5,17	0,48	14,223	0,639
			25	5	$2{,}53.10^{-24}$	0,11	-0,24	0,49	0,0011	5,71	0,47	25,780	0,980
k=1,..,130	N=100	I.	15	5	$1{,}45.10^{-24}$	-0,57	-0,08	-0,65	0,0031	4,23	0,51	6,772	0,372
		II.	15	5	$2{,}60.10^{-24}$	0,64	-0,65	0,49	0,0024	4,72	0,43	12,092	0,364
	III.		15	5	$5{,}33.10^{-26}$	0,92	2,14	2,45	0,0010	3,35	0,40	6,827	0,372

TABLE II
MONTE CARLO RESULTS

Tracking Filter	BPNN			KF
	Case	Training	Prediction	
$m_{\bar{\zeta}_x}$, m	I.	$5{,}68.10^{-14}$	$1{,}11.10^{-12}$	0,0714
	II.	$9{,}24.10^{-15}$	$1{,}48.10^{-12}$	
	III.	$1{,}57.10^{-10}$	$1{,}57.10^{-10}$	0,7075
$m_{\bar{\zeta}_y}$, m	I.	$2{,}13.10^{-14}$	$1{,}21.10^{-12}$	0,0730
	II.	$1{,}28.10^{-14}$	$1{,}40.10^{-12}$	
	III.	$2{,}79.10^{-10}$	$2{,}79.10^{-10}$	1,3534
$m_{\bar{\zeta}_z}$, m	I.	$5{,}68.10^{-14}$	$1{,}22.10^{-12}$	0,1076
	II.	$1{,}43.10^{-13}$	$2{,}41.10^{-12}$	
	III.	$3{,}41.10^{-10}$	$3{,}41.10^{-10}$	0,0940
$\sigma_{\bar{\zeta}_x}$, m	I.	$6{,}34.10^{-29}$	$5{,}50.10^{-12}$	0,0552
	II.	$4{,}75.10^{-30}$	$9{,}09.10^{-12}$	
	III.	$2{,}31.10^{-10}$	$2{,}07.10^{-10}$	0,2301
$\sigma_{\bar{\zeta}_y}$, m	I.	$2{,}85.10^{-29}$	$6{,}41.10^{-12}$	0,0582
	II.	$3{,}01.10^{-29}$	$7{,}46.10^{-12}$	
	III.	$3{,}95.10^{-10}$	$3{,}55.10^{-10}$	0,4401
$\sigma_{\bar{\zeta}_z}$, m	I.	$6{,}34.10^{-29}$	$6{,}37.10^{-12}$	0,0749
	II.	$1{,}27.10^{-28}$	$1{,}36.10^{-11}$	
	III.	$4{,}06.10^{-10}$	$3{,}65.10^{-10}$	0,0344
$\bar{t}_{CPU}$, s	I.	4,23	0,51	0,37
	II.	4,72	0,43	
	III.	3,35	0,40	0,38
$n_{flops}, .10^6$	I.	6,772	0,372	0,670
	II.	12,092	0,364	
	III.	6,827	0,372	0,670

TABLE III
ERRORS DUE TO RECOVERING: STATISTICS

$m_{\bar{\xi}_{RMS}^{CT}}$, m	$\sigma_{\bar{\xi}_{RMS}^{CT}}$, m	$m_{\bar{\xi}_{RMS}^{NP}}$, m	$\sigma_{\bar{\xi}_{RMS}^{NP}}$, m
$5{,}38.10^{-14}$	$1{,}27.10^{-14}$	$0{,}18.10^{-13}$	$4{,}69.10^{-15}$

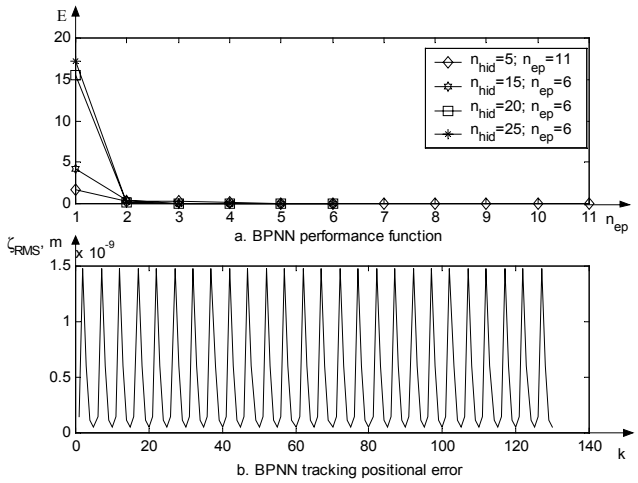

Fig. 2. The algorithm's performances for the real track

trained net from the same track - scans 6,..,130) phases is plotted on the bottom of Fig. 2. All the results are obtained by Intel Celeron 500 PPGA with SDRAM 128 MB.

CONCLUSIONS

This paper has presented the algorithm for BPNN tracking filter using canonical transform. The BPNN learning in this case is faster than the learning without the transform and requires fewer flops. The Monte Carlo simulations and the results with real input data show that this neural filter works efficiently in different environment and produces smaller tracking error than the recursive Kalman filter. The errors due to recovering the original variables do not affect to the accuracy of the algorithm.

REFERENCES

[1] X. Rong Li, "Canonical Transform for Tracking with Kinematic Models", *IEEE Trans., Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1212-1223, 1997.

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

[3] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, Stuttgart, John Wiley & Sons & B. G. Teubner, 1993

[4] *Neural Networks Toolbox – User's Guide*, Mathworks, Inc., 1994.

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