A Comparative Analysis of Some Data Preprocessing Techniques for BPNN Tracking Filter

Mimi D. Daneva[1]

Abstract – **In this paper the performances of a BPNN tracking filter depending on the method used for pattern formulation are evaluated and compared. Several input data preprocessing techniques are used for the investigation. The tracking error is compared with this of standard recursive Kalman filter using simulated and real radar database.**

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

I. Introduction

The algorithms for filtering and prediction of the target trajectories, also called tracking filters provide the estimate of the current and future kinematic parameters of the aircraft, used next in the data association process. The accuracy of the estimate, provided by the filter, effects to a great degree to the correct decision for data association, which by it reflects to the safety level in air traffic control [1]. As a part of the total set of algorithms for multiple target tracking, the algorithms for track filtering and prediction can be divided in two groups: probabilistic and heuristic [2]. The tracking filters based on artificial neural networks and the artificial intelligence approach belong to the second group. They differ to the first group filters in two main features [3]. First, they do not need to a priori knowledge about the statistics of the input data, and second, these target state estimators are "model-free" – contrary to the statistical estimators they can estimate input-output functions without a mathematical model of how the output of the dynamic system depends on its input. Using neural approach for data processing the choice of appropriate method of input data representation is an important factor for successful convergence of the training algorithm. In this paper, three methods for data preprocessing and possible combinations of them, used for BPNN tracking filter, are compared by Monte Carlo (MC) experiment with respect to their training performance, the tracking error, the error due to recovering of the original variables after data postprocessing, and the computational costs. The performances of the BPNN filter using different data preprocessing techniques are compared with standard recursive Kalman filter (KF) by 50 MC runs. Simulated input data according to kinematic model of non-maneuvering aircraft in polar coordinates [1], [2] and real radar data records from the plot extractor's output of Monopulse Secondary Surveillance Radar CMSSR-401 [4] are used for the investigation. All experiments are implemented in *MATLAB* environment. Interesting results are obtained.

[1]Mimi D. Daneva is with the Faculty of Communications and Communicational Technologies, 8 Kl. Ohridski str., 1000 Sofia, Bulgaria, e-mail: mimidan@tu-sofia.acad.bg

II. Input Data Representation

The input data representation, pattern formulation or input coding is a process of mapping the input feature space onto a set of input units of the neural network (NN). The choice of the most appropriate input coding effects onto the adequate NN's training performance and the error. For input data representation, normalization coding (NC) is widely used for pattern recognition, system identification or estimation purposes. It represents the input feature accurately, and is distinguished with naturalness and computational simplicity [5]. The simplest form of normalization ranges the input data in $[0;1]$or $[-1;1]$ (NC1). In [6] it is recommendable to distribute the input data in the interval $[0;0.9]$ or $[-0{,}9;0.9]$ (NC2) to avoid the high nonlinearly zones of the neurons' activation functions. The equations of NC1 and NC2 are as follow [6]

$$\mathbf{P}_n = 2\left(\mathbf{P} - P_{\min}\right)/\left(P_{\max} - P_{\min}\right) - 1\ , \qquad (1)$$

$$\mathbf{P}_n = 1{,}8\left(\mathbf{P} - P_{\min}\right)/\left(P_{\max} - P_{\min}\right) - 0.9\ , \qquad (2)$$

where $\mathbf{P}$and $\mathbf{P}_n$ are the original and the normalized pattern vectors, respectively; $P_{\min}$ and $P_{\max}$ are the minimum and maximum values of $\mathbf{P}$.

The input data can also be normalized so that they will have zero mean and unity standard deviation (NC3) as

$$\mathbf{P}_n = \left(\mathbf{P} - m_P\right)/\sigma_P\ , \qquad (3)$$

where m_P and σ_P are the mean and standard deviation of **P**, respectively (one-class normalization). This technique is effective for pattern classification purposes in the case of more one class [5].

Usually, the pattern formulation process is considered as an input space transformation, which transforms the dimension of the input space in more effective features [5]. If this transform is linear, then the function that maps the input space into the output variables for data preprocessing is well defined, and the preprocessing task reduces to determine the coefficients of this linear function with respect to minimize or maximize some optimization criterion. One approach to do it is a discrete cosine transform (DCT), which is widely used for image coding. It originally was developed as an approximation of the optimal Karhunen-Loève transform, but a number of fast algorithms have been developed for its computation [7]. In *MATLAB* environment, one way to compute the DCT is through the form DCT-IV as [8]

$$\mathbf{P}_{\text{DCT}}(k) = w(k)\sum_{n=1}^{N}\mathbf{P}(n)\cos\left(\frac{\pi(2n-1)(k-1)}{2N}\right)\ , \qquad (4)$$

where $w(k) = \begin{cases} 1/\sqrt{N}, & k = 1 \\ \sqrt{2/N}, & 2 \leqslant \mathrm{k} \leqslant N \end{cases}$, N is the length of $\mathbf{P}$, and k is the discrete time. The DCT is a purely real orthogonal transform, which can be used as an additional transformation of the input space before normalization and next processing by the neural network.

The input data for the proposed BPNN tracking filter for non-maneuvering targets are the polar coordinates range ρ in nautical miles (NM), azimuth angle θ in rad, and the altitude h in feet for 6 non-manoeuvering targets. They are preprocessed independently using NC1, NC2 and NC3, and by DCT followed by NC1, NC2 and NC3. The goal is to choose the most appropriate input data preprocessing technique for the BPNN tracking filter, accordingly to the specific high accuracy requirements and acceptable computational costs [2].

The block diagram of data processing with BPNN tracking filter is shown in Fig. 1, where **W** and **B** denote the weight matrix and the bias vector of the BPNN, respectively. The prediction error of BPNN is denoted by $\mathbf{e}_{\text{BPNN}}\left(k,q\right)$. It propagates in backward manner in the neural structure by the training algorithm. The relative error due to reconstruction of the original variables after the data postprocessing is defined by [8]

$$\xi_{rec} = \frac{\parallel \mathbf{P} - \mathbf{P}_{rec} \parallel}{\parallel \mathbf{P} \parallel} \; . \tag{5}$$

For comparison of the results the same input data as for the BPNN filter are processed by standard recursive Kalman filter (KF) [1,2]. The tracking error for a single target is defined by

$$\zeta\left(k+1\right) = \mathbf{Z}\left(k+1\right) - \mathbf{H}\hat{\mathbf{X}}\left(k+1\right) \; , \tag{6}$$

where **H** is the measurement matrix.

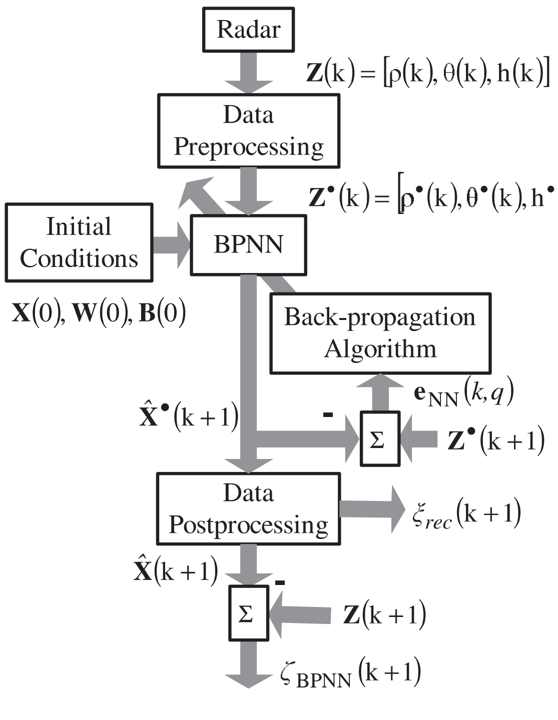
Radar
$\mathbf{Z}(k) = [\rho(k), \theta(k), h(k)]$
Data Preprocessing
$\mathbf{Z}^{\bullet}(k) = [\rho^{\bullet}(k), \theta^{\bullet}(k), h^{\bullet}(k)]$
Initial Conditions
BPNN
$\mathbf{X}(0), \mathbf{W}(0), \mathbf{B}(0)$
Back-propagation Algorithm
$\mathbf{e}_{\text{NN}}(k,q)$
$\hat{\mathbf{X}}^{\bullet}(k+1)$
-
Σ
$\mathbf{Z}^{\bullet}(k+1)$
Data Postprocessing
$\xi_{rec}(k+1)$
$\hat{\mathbf{X}}(k+1)$
-
Σ
$\mathbf{Z}(k+1)$
$\zeta_{\text{BPNN}}(k+1)$

Fig. 1. Block diagram for track filtering and one-stepahead prediction with BPNN

III. Architecture and Training of BPNN Tracking Filter

The BPNN employed is fully connected and has 3 input nodes, a single hidden layer with 12 units, and 3 output units. The input and hidden neurons have bipolar sigmoid functions, the output neurons are linear. This architecture was chosen as optimal in heuristical way [6] by separated experiment. The BPNN can be used as nonlinear predictor of a stationary time series [9]. In our case the elements of the input vector are the past samples of the preprocessed radar measurements, i.e.

$$\mathbf{X}_{inp} = \left[\mathbf{Z}^{\bullet}\left(k-1\right),\; \mathbf{Z}^{\bullet}\left(k-2\right),\; \ldots,\; \mathbf{Z}^{\bullet}\left(k-p\right)\right]^{T} \tag{7}$$

where $\mathbf{Z}^{\bullet}\left(k-1\right)$is the measurement vector after preprocessing procedure, and p=3 is the model (prediction) order. The actual values of the elements of the input vector are used as elements of the desired target vector $\mathbf{T}_{inp}\left(k\right)$. The BPNN's output vector $\mathbf{Y}_{P}\left(k\right)$ produces the estimate $\hat{\mathbf{X}}_{inp}$ of the input vector for one-step-ahead prediction as

$$\mathbf{Y}_{P}\left(k\right) = \hat{\mathbf{X}}_{inp} \; . \tag{8}$$

The Nguyen-Widrow hidden weights initialization procedure [6] was used for network initialization. The Marquardt-Levenberg algorithm, which has the fastest and guaranteed convergence compared with the other BP algorithms [3], [9], was used for BPNN training. The training stop when the global minimum of the network performance function [9]

$$E = \frac{1}{Q}\sum_{q=1}^{Q}\xi_q = \frac{1}{2}\sum_{q=1}^{Q}\sum_{j=1}^{n_{out}}\left(d_{jq} - y_{jq}\right)^2 \tag{9}$$

is reached, where d_{jq} and y_{jq} are the desired target and the actual output signal of the j-th output neuron for the $q-$th training example is found, or the maximum number of epochs or the maximum learning rate. The track filtering and prediction continues till the moment when the track deletion criteria [1], the absence of measurements about this track for three consecutive radar scans, is satisfied.

IV. Experimental Results

Simulated and live data for 6 tracks of non-manoeuvering targets, chosen in random way are used to form the training set for the computer modelling. The real data are recorded from the plot extractor's output of CMSSR-401. The measurement errors are 0.05 NM; 0.07°, and 100 feet for ρ, θ, and h, respectively; the radar sampling time is $T = 10$ s [4]. Different random input sequences are generated for each MC run to simulate the measurement errors. Their normal cumulative distribution function is verificated by chi-square test with significant level $\alpha = 0.05$. The real tracks with length of 140 consecutive scans for the experiments are shown in Fig. 2. They are used as prototypes to model the tracks for the MC simulations, as follow. After polar to Cartesian transform to equalize the dimensions of the coordinates, the measurements from the first and the last radar scan, are connected

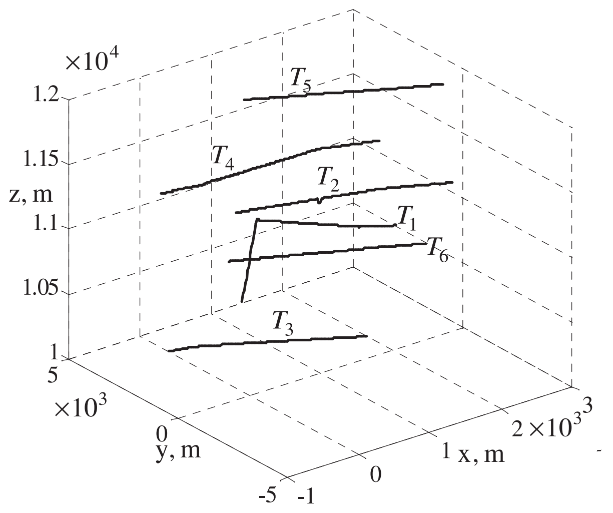

Fig. 2. Live tracks used for the experiments

Table 1. MC simulations results: BPNN and recursive Kalman filter's averaged tracking errors

FILTER / ERRORS	BPNN						KF
	NC1	NC2	NC3	DCT &NC1	DCT &NC2	DCT &NC3	
$\bar{\zeta}_{\rho}^{T_1}$, NM	0,29	0,31	0,24	3,85	4,74	3,40	1,10
$\bar{\zeta}_{\rho}^{T_2}$, NM	0,29	0,32	0,24	3,17	2,96	1,73	1,14
$\bar{\zeta}_{\rho}^{T_3}$, NM	0,36	0,39	0,32	2,09	2,28	1,87	1,50
$\bar{\zeta}_{\rho}^{T_4}$, NM	0,35	0,38	0,32	2,20	2,12	1,86	1,52
$\bar{\zeta}_{\rho}^{T_5}$, NM	0,29	0,32	0,25	1,67	2,00	1,69	1,24
$\bar{\zeta}_{\rho}^{T_6}$, NM	0,30	0,34	0,26	1,98	2,19	1,58	1,23
$\bar{\zeta}_{\theta}^{T_1}$, rad	0,014	0,015	0,013	0,021	0,032	0,021	0,017
$\bar{\zeta}_{\theta}^{T_2}$, rad	0,012	0,013	0,011	0,020	0,015	0,008	0,012
$\bar{\zeta}_{\theta}^{T_3}$, rad	0,014	0,015	0,013	0,011	0,014	0,012	0,008
$\bar{\zeta}_{\theta}^{T_4}$, rad	0,012	0,011	0,007	0,008	0,013	0,010	0,008
$\bar{\zeta}_{\theta}^{T_5}$, rad	0,013	0,014	0,010	0,014	0,015	0,011	0,009
$\bar{\zeta}_{\theta}^{T_6}$, rad	0,012	0,013	0,010	0,015	0,014	0,011	0,017
$\bar{\zeta}_{h}^{T_1}$, ft	24,67	27,27	16,78	783,8	896,8	545,3	39,52
$\bar{\zeta}_{h}^{T_6}$, ft	32,10	34,75	21,98	443,3	414,7	338,2	37,38

with straight line and spaced with scan time T to form the trajectory of the non-manoeuvering target. The track T_1 is modelled using the first and the last point of the section with constant altitude and the first and the last point of the section with varying altitude. Next each track is corrupted with Gaussian noises, added directly to each coordinate to model the measurement errors. Next the tracks are transformed back in polar coordinates and then filtered by the algorithm under the test. The tracking error $\zeta$of each filter is defined by the difference between the measurement vector and the estimated state vector [1]. The experiments include 50 MC runs and track example using real radar data. The BPNN filters with

Table 2. MC simulations results: BPNN training performance and computational costs

CASE	NC1	NC2	NC3	DCT& NC1	DCT& NC2	DCT& NC3
$\bar{k}^T$	1	1	1	1	1	1
$\bar{E}_{final}^{T} \times 10^{-06}$	0,036	0,032	0,080	0,480	0,229	0,196
$\bar{t}_{CPU}$, s	0,69	0,68	0,67	0,50	0,67	0,67
$\bar{n}_{Mflops}$	4,745	4,742	4,755	1,073	1,073	4,694

cases of preprocessing NCs 1 to 3, with and without DCT as an additional data preprocessing are compared with KF. The root mean square (*rms*) values of ζ are averaged over all MC runs to form the *rms* error at each time point. The acceleration's standard deviation for KF is $2g$ m/s^2 [1]. The averaged BPNN's parameters are the averaged number of epochs $\bar{k}$, the averaged net error function at the end of training $\overline{E}_{final}$, and

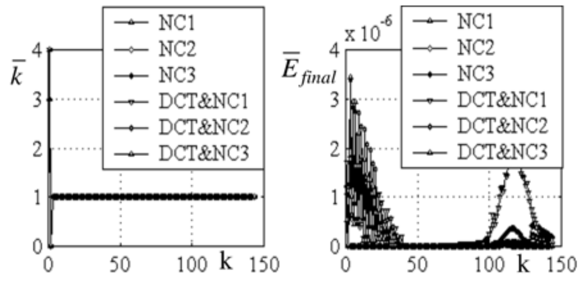

Fig. 3. MC results: BPNN training performance comparison

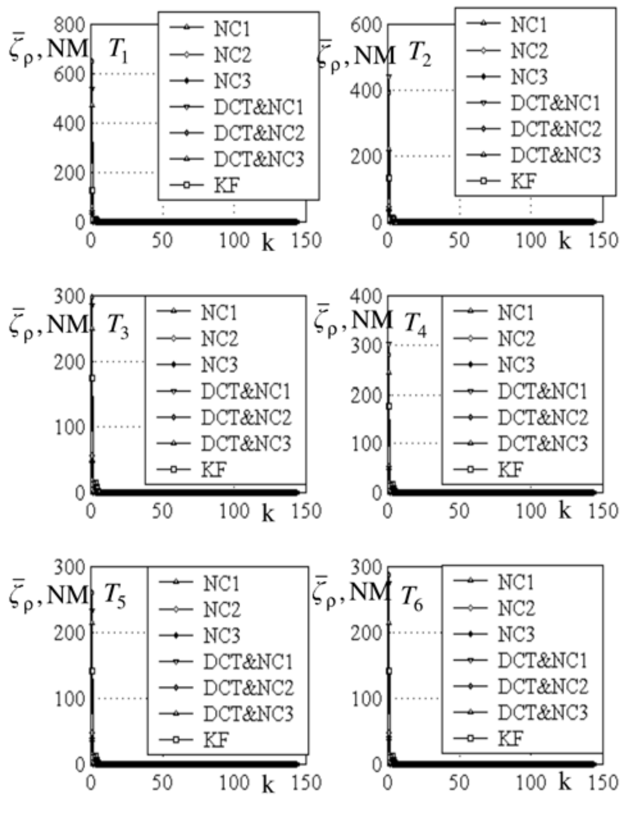

Fig. 4. MC results: BPNN and KF tracking errors for range

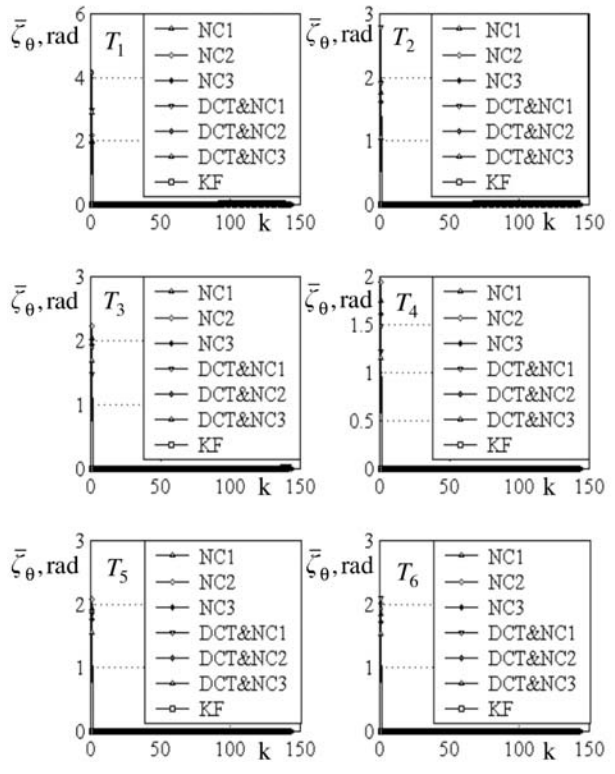

Fig. 5. MC results: BPNN and KF tracking errors for azimuth

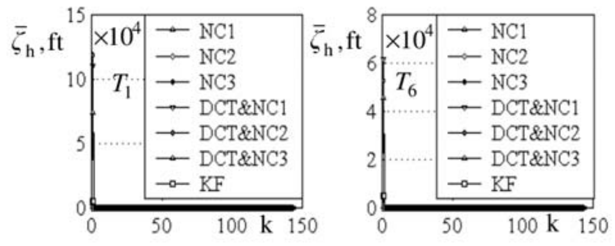

Fig. 6. MC results: BPNN and KF tracking errors for altitude

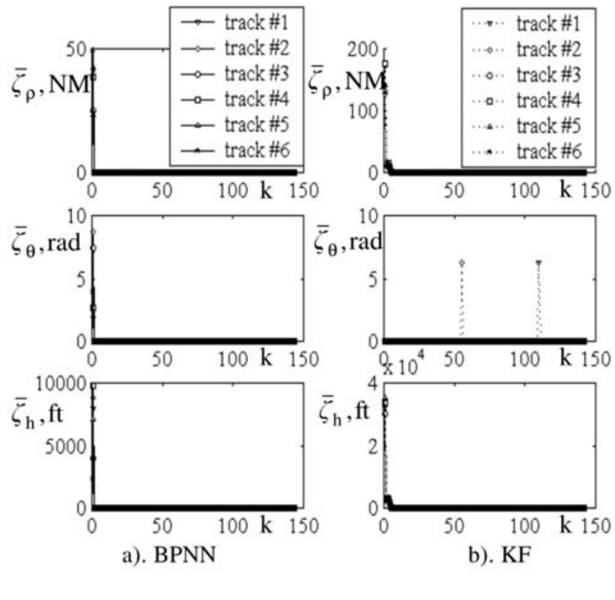

Fig. 7. BPNN and KF's tracking errors for 6 real tracks

Table 3. MC simulations results: averaged relative error due recovering

Case / Error	NC1	NC2	NC3	DCT& NC1	DCT& NC2	DCT& NC3
$\bar{\xi}_{\text{rec}}\times10^{-18}$	0,056	0,015	0,021	37,7	40,4	39,1

the *rms* value of $\bar{\zeta}$ for each coordinate, averaged for the MC runs and for the whole track. The computational complexity of each BPNN filter is estimated by the averaged parameters CPU time and number of Mflops, used for data processing of all training examples (all measurements for all the targets for scan k). All results are obtained by Intel Celeron 500 PPGA with SDRAM 128 MB.

The results from the MC runs are presented in Tables 1 and 2. The BPNN training performances are plotted in Fig. 3, the tracking errors during MC runs for all the coordinate for all tracks are plotted in Figs. 4 to 6. The BPNN filters with NCs 1 to 3 produce smaller tracking errors than the KF. The best BPNN performance is achieved when NC3 is used. That is due to the specific quantities of data preprocessing in this case, which promotes the speed and learning convergence of the BPNN filter. In the cases of DCT&NCs 1 to 3 the BPNN filter needs less training time, epochs, and Mflops, but the learning convergence and the tracking performance degrade. Till this moment all BPNN filters are investigated with one and the same value of the parameter goal $g_{\min} = 0.00001$ during the training. Obviously this value is insufficiently small for the cases of all DCT&NCs. An appropriate correction of the goal will improve the training, but in Figs. 4 and 6 we can see that the BPNN initial errors for these cases are quite high (they are greater than these of KF). That is due to the specific quantities of the DCT, which increases the distances between the classes (i.e. the tracks) in the transformed space. This is suitable for data classification purposes, but in the case of time-series prediction make the initial accuracy worse. In the cases of NCs 1 to 3 the BPNN tracking errors are smaller than the KF. The best results of the BPNN filter are obtained using NC3. This conclusion is confirmed with the example of multiple-target tracking using real radar data record, shown in Fig. 7. The averaged relative error due to re-construction of the original variables after data postprocessing is presented in Table 3. It is due to rounding during the mathematical operations and does not affect to the tracking accuracy of the filters, because the recovering of the data is 100 %.

V. Conclusion

In this paper the results of an experimental comparison among three methods for input data preprocessing and some combinations between them before target track filtering and prediction then has been presented and analyzed by Monte Carlo experiment. The results show that these preprocessing techniques influence onto the BPNN training and it's tracking performance. The smaller tracking errors of the algorithm are obtained when a normalization coding with zero mean and unity standard deviation is used. It improves the accuracy of the tracking algorithm and reduces the computational costs.

References

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

[2] Y. Bar-Shalom (editor), *Multiple-Target Tracking*, vol. I, 1990, and vol. II, 1992, Dedham, Artech House.

[3] B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall International, 1992.

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

[5] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, New York and London, Academic Press, 1972.

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

[7] J. H. McClellan, C. Sidney Burrus, A. V. Oppenheim, T. W. Parks, R. W. Schafer, H. W. Schuessler, *Computer-Based Exer-cises for Signal Processing Using Matlab*® 5, New Jersey, Prentice Hall, 1998.

[8] *Signal Processing Toolbox User's Guide Version 4.2*, The Math Works, Inc., 1998-1999.

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