Yee Chin Wong; Malur K. Sundareshan

A simplex trained neural network-based architecture for sensor fusion and tracking of target maneuvers

Kybernetika, Vol. 35 (1999), No. 5, [613]--636

Persistent URL: http://dml.cz/dmlcz/135311

KYBERNETIKA — VOLUME 35 (1999), NUMBER 5, PAGES 613–636

A SIMPLEX TRAINED NEURAL NETWORK–BASED ARCHITECTURE FOR SENSOR FUSION AND TRACKING OF TARGET MANEUVERS[1]

YEE CHIN WONG AND MALUR K. SUNDARESHAN

One of the major applications for which neural network-based methods are being successfully employed is in the design of intelligent integrated processing architectures that efficiently implement sensor fusion operations. In this paper we shall present a novel scheme for developing fused decisions for surveillance and tracking in typical multi-sensor environments characterized by the disparity in the data streams arriving from various sensors. This scheme employs an integration of a multilayer neural network trained with features extracted from the multi-sensor data and a Kalman filter in order to permit reliable tracking of maneuvering targets, and provides an intelligent way of implementing an overall nonlinear tracking filter without any attendant increases in computational complexity. A particular focus is given to optimizing the neural network architecture and the learning strategy which are particularly critical to develop the capabilities required for tracking of target maneuvers. Towards these goals, a network growing scheme and a simplex algorithm that seeks the global minimum of the training error are presented. To provide validation of these methods, results of several tracking experiments involving targets executing complex maneuvers in noisy and clutter environments are presented.

1. INTRODUCTION

A variety of sensing devices ranging from radar systems to lasers and optical imaging systems are presently being developed for surveillance and tracking operations. The limitations of using a single sensor in these operations, such as limited accuracy and lack of robustness, have motivated the trend towards designing surveillance and tracking systems with multiple sensors deployed on the same platform (an airborne or spaceborne reconnaissance platform or a tactical missile seeker, for instance) which can provide large amounts of useful data to detect, track, and identify targets of interest. However, current surveillance and tracking algorithms usually use information from only one sensor (such as Track–While–Scan (TWS) radar) or attempt to combine information from different sensors in an ad hoc manner. While it is intuitive that using additional data available can result in improved detection, classification

[1] A version of this paper was presented at the 5th Mediterranean Conference on Control and Systems held in Paphos (Cyprus) on June 21–23, 1997.

and track maintenance performance, attempting to include this data efficiently will require novel processing methods which need to be carefully tailored due to the disparate forms of data collected. A major limitation precluding the integration of additional data, perhaps of a disparate form from the main data form being used, is the resulting complexity of the needed processing. It is well known that while simple linear processing algorithms employing a Kalman filter for target state estimation can be synthesized for processing radar data, inclusion of a different form of data (image or image-format data, for instance) will require a nonlinear processing method (such as an Extended Kalman filtering algorithm) [18]. The enormous processing complexity this may introduce could render the implementation impractical due to the real-time processing requirements underlying the tracking function and the need to keep up with the rapid target motions during the maneuvers. Consequently, an intelligent architecture that facilitates successful fusion of the diverse data forms to result in improved tracking performance in the face of complex target maneuvers is highly desirable.

Our interest in this work centers on the development of fusion architectures that can integrate features extracted from sensor measurements and assist in efficiently performing target surveillance and tracking, since such architectures will not only permit fusion of data from sensors which could have diverse characteristics but also will present interesting and nontrivial questions to be investigated. The two major steps in the design of such architectures are, (i) feature extraction and (ii) feature integration. In particular, an architecture (as depicted in Figure 1) that subjects the data stream coming from each sensor to a feature extraction operation (perhaps after some preprocessing to align, order and/or reformat the data as desired), which is in turn followed by a feature integration operation to arrive at a fused decision for surveillance and tracking, constitutes the backbone for intelligent integrated processing of multisensor data in these applications.

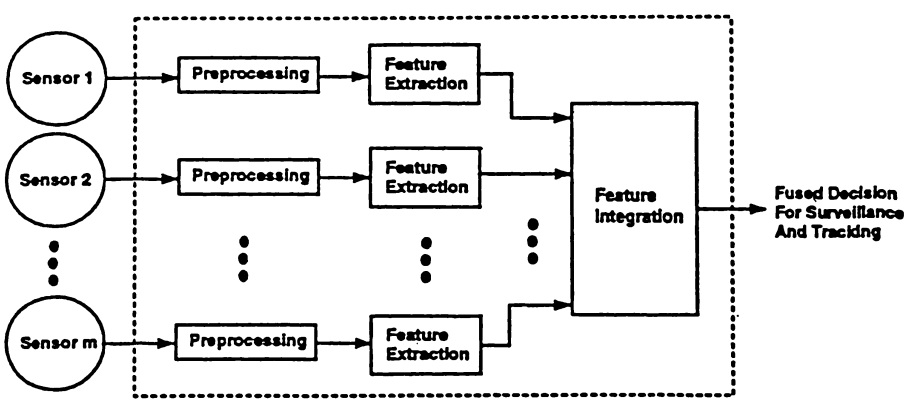

Fig. 1. Backbone processing architecture for intelligent integrated processing of multisensor data in surveillance and tracking.

While there exist a number of mathematical tools to perform feature extraction

(such as vector quantization, Karhunen – Loeve expansion, and wavelet decomposition), for addressing the second issue, viz. integration of features extracted from diverse data forms, an attractive framework is provided by neural networks. The ability to efficiently fuse information of different forms is one of the major capabilities of trained neural networks that is being recognized only in recent times. While development of innovative adaptive control algorithms for nonlinear dynamical plants which attempt to exploit these capabilities [11, 17] seems to be more popular, a corresponding development of nonlinear signal processing algorithms using these approaches, particularly for applications in target surveillance and guidance operations, does not appear to have received a similar level of attention. Principal requirements in these applications are real-time processing capability to deal with the high rate data streams coming in and the ability to rapidly develop appropriate outputs for target detection, recognition and tracking with high degree of accuracy and reliability. Since neural networks with appropriate training can be endowed with the abilities to identify simultaneously multiple correspondences existing between presented data elements and further to utilize these correspondences to produce good global solutions, they seem quite attractive in the development of architectures that meet the requirements stated above. In particular, the primary motivation for employing neural networks in these applications comes from the efficiency with which features extracted from different sensor measurements can be utilized as inputs for developing the needed estimates on target location and target motion. For implementation in maneuver tracking applications, the pattern classification capability of a neural network can hence be used to process the features extracted from different data streams (from diverse sensors), and hence provides a mechanism to simply integrate new and additional data to a tracking filter originally designed to process data from a primary tracking sensor as shown in Figure 2.

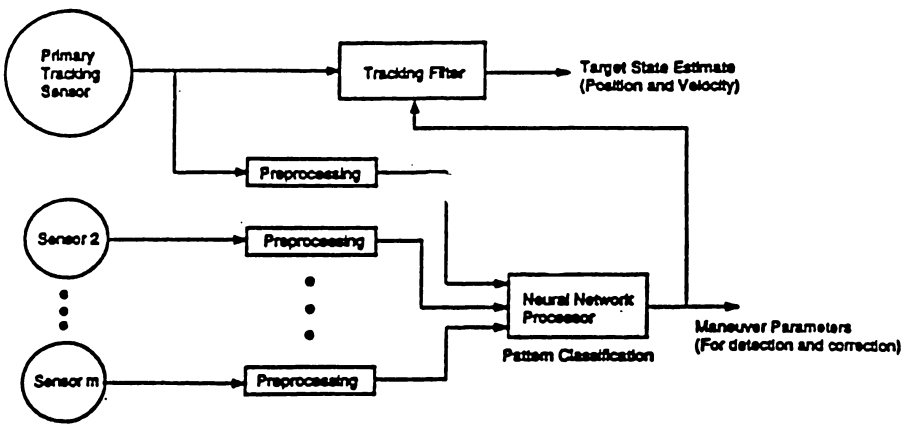

Fig. 2. Feature integration by neural network for sensor fusion in maneuver tracking applications.

Some of our recent studies [1, 15, 16] have helped obtaining an understanding

of the ability of neural networks to fuse information from different sensors to assist in simple implementations of target detection and tracking algorithms. The primary focus in this paper is on developing an optimized neural network architecture and an efficient training scheme that endows the neural network the capability to perform fusion of target measurements in order to reliably track target maneuvers executed in severe clutter and noise environments. Towards these goals, we shall introduce a network growing scheme and implementation of a simplex optimization algorithm for training a multilayer neural network. Unlike the more commonly used approach of error backpropagation [9], the simplex optimization approach enables one to more efficiently seek the global optimum in the training task, and consequently permits the trained network to process a set of features extracted from the sensor measurements in order to rapidly make the necessary association with certain critical parameters representative of the target maneuver. A target tracking system architecture is developed by integrating the trained neural network with a Kalman filter that performs the target state estimation function. An illustration of the performance of the tracking scheme is given by conducting a few simulation experiments involving representative target tracking scenarios with target maneuvers executed in noise and clutter environments. Since at the most fundamental level the operation of a multilayer neural network can be explained in terms of approximating a nonlinear input-output map, the present tracking architecture that comprises of an integration of a Kalman filter with a trained neural network in essence provides an intelligent way of implementing an overall/nonlinear tracking filter without any increases in computational complexities.

2. NEURAL NETWORK-BASED ARCHITECTURE FOR SENSOR FUSION AND TARGET TRACKING

The basic building blocks of the tracking scheme shown in Figure 3 are the neural network and the Kalman filter. The neural network accepts as inputs a set of features extracted from the sensor data and is trained to output estimates of a set of maneuver parameters characterizing the target maneuver that is represented in the feature set. Since features abstracted from the measurements obtained from dissimilar sensors are typically used as inputs to the neural network, the processing of data by the network implements a feature integration process and thus performs sensor fusion. The neural network outputs are fed to a Kalman filter which implements a recursive state estimation algorithm based on a linear model of the target dynamics. For the sake of illustration of specific details regarding the features extracted and the training conducted with these features, Figure 3 depicts a fusion environment comprising of a range radar and a Doppler radar. It is to be emphasized that this is only for simplicity in discussing the details and will not limit the type of sensor that may be brought in to provide target measurements.

The tracking scheme integrates the trained neural network with a Kalman filter designed based on the target dynamical model

$$x(k+1) = Fx(k) + Gu(k) + \nu(k) \tag{1}$$

where $x^T(k) = [x(k)\,\dot{x}(k)\,y(k)\,\dot{y}(k)]$ is the state vector, $u^T(k) = [u_x(k)\,u_y(k)]$ is the input vector consisting of the acceleration components in the x and y directions, and $\nu(k)$ is the process noise. The matrices F and G are given by

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & T \end{bmatrix} \quad \text{and} \quad G = \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix}$$

and where T is the time interval between two consecutive measurements. Assuming that the measurement provides the position of the target along the two coordinates (obtained from using a range radar, for instance), we have the observation sequence

$$z(k) = H\,x(k) + \omega(k) \tag{2}$$

where $z(k)$ is the measurement vector, $\omega(k)$ is the measurement noise and

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Both $\nu(k)$ and $\omega(k)$ are assumed to be white Gaussian noise sequences with zero means and covariances Q and R respectively. The two processes $\nu(k)$ and $\omega(k)$ are assumed to be uncorrelated.

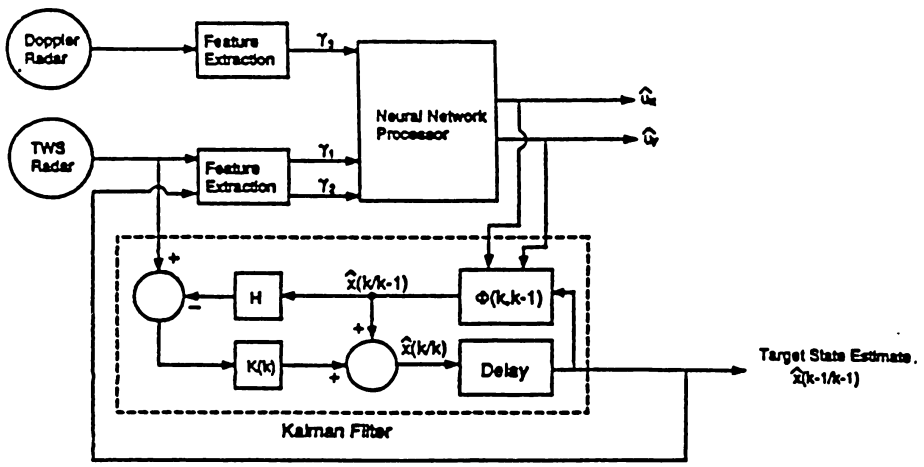

Fig. 3. A neural network-based fusion architecture for tracking target maneuvers.

The received measurements $u(k)$, $k = 1, 2, \ldots$, are processed by a Kalman filter to generate the minimum MSE estimates by implementing the recursive algorithm briefly described in the following steps:

(i) One-step Prediction

$$\hat{x}(k|k-1) = F\hat{x}(k-1|k-1) + Gu(k-1). \tag{3}$$

(ii) Filtering

$$\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)\,[z(k) - H\hat{x}(k|k-1)]. \tag{4}$$

(iii) Gain Computation

$$K(k) = P(k|k-1)\,H^T[H\,P(k|k-1)\,H^T + R]^{-1}. \tag{5}$$

(iv) Covariance Updating

$$P(k|k-1) = FP(k-1|k-1)\,F^T + GQG^T \tag{6}$$

$$P(k|k) = [I - K(k)\,H]\,P(k|k-1). \tag{7}$$

It may be noted from step (ii) that the filter processes the innovation sequence $\{\tilde{z}(1), \tilde{z}(2), \ldots, \tilde{z}(k), \ldots\}$ where

$$\tilde{z}(k) = z(k) - H\,\hat{x}(k|k-1), \quad k = 1, 2, \ldots \tag{8}$$

Thus, with a primary tracking sensor (such as a TWS radar, as shown in Figure 3) providing the required data, the Kalman filter will satisfactorily filter out the noise and permits a reliable tracking of target.

When the target is not maneuvering (i.e., when $u(k) = 0$), the mean of the innovation sequence is zero. However, when the target begins to maneuver (i.e., when $u(k) \neq (0)$, the mean of tildez(k) is no longer zero and can be utilized to detect maneuver. Appropriate features extracted from the sensor data can hence be processed to obtain estimates of the maneuver to facilitate the Kalman filter to continue to track the target reliably even when it is maneuvering. In the architecture depicted in Figure 3, this function is implemented by a multilayer feedforward neural network.

Training of the neural network for providing maneuver estimates is implemented with three features extracted from the measured data. Two of these features $v_1(k)$ and $v_2(k)$ are obtained from the measurements of the range radar (a TWS radar, for illustration) and the other feature $v_3(k)$ is obtained from the measurements from the Doppler radar (as shown in Figure 3). More specifically, the signal is constructed by normalizing the two components of the innovation data with respect to the covariance as

$$v_1(k) = \frac{\tilde{z}_x^2(k)}{S_{xx}(k)} + \frac{\tilde{z}_x^2(k)}{S_{yy}(k)} \tag{9}$$

where $\tilde{z}(k) = [\tilde{z}_x(k)\,\tilde{z}_y(k)]^T$ defined in (8), and $S_{xx}(k)$ and $S_{yy}(k)$ are the diagonal elements of the covariance matrix

$$S(k) = H\,P(k|k-1)\,H^T + R$$

which is used for the Kalman filter gain computation in (5). Since the generation of innovation process requires both the measured data from TWS radar and the state estimate developed from Kalman filter, the box entitled "Feature Extraction" in Figure 3 has two input lines – one from the radar and the other from the output

of the Kalman filter. Signal $v_2(k)$ is the change in the heading estimate computed as

$$v_2(k) + \alpha_{\mathrm{LT}}(k) - \alpha_{\mathrm{LT}}(k-1) \tag{10}$$

where $\alpha_{\mathrm{LT}}(k)$ and $\alpha_{\mathrm{LT}}(k-1)$ are the heading estimates computed by the method of least triangles [8] from using three past data points. Finally, the third input feature $v_3(k)$ is extracted from Doppler radar and is computed as the change is variance, i.e.,

$$v_3(k) = \frac{1}{\sigma^2_{f_d}} [f_d(k) - f_d(k-1)] \tag{11}$$

where $f_d(i) = \frac{2}{\lambda \dot{R}(i)}$ provides a measure of the radial velocity, $\dot{R}(i)$, at instant i (λ denotes the wavelength of the transmitted wave) and $\sigma^2_{f_d}$, variance of the Doppler shift.

Performance evaluation studies conducted earlier [15, 16] provide ample evidence that the three features contain adequate information to train the network to provide reasonably accurate maneuver estimates when the target maneuvers involve longitudinal accelerations of arbitrary magnitudes. This performance has been tested in several tracking scenarios comprising of various degrees of clutter and noise. Furthermore, the resulting performance levels have been shown to compare favorably with some classical maneuver tracking schemes [16].

3. OPTIMIZATION OF NEURAL NETWORK ARCHITECTURE AND TRAINING

3.1. Neural network training by Simplex algorithm

Perhaps the most significant characteristic that enables a neural network to serve as a useful computational device is its learning capability. Implementation of an appropriately tailored learning algorithm, i.e. a rule for adjustment of the network parameters (specifically the interconnection weights) can endow the network the ability to develop the needed structure to result in a corresponding desired computation.

A number of alternate procedures exist for training a neural network with the available data and different training algorithms usually yield different sets of interconnection weights. While the error back-propagation approach is perhaps the most popular approach [9] for training multilayer neural networks, it has a few shortcomings as well. The backpropagation approach, being a gradient-based search algorithm, is sensitive to the initial starting point (i.e. preliminary selection of weights to start the algorithm) and has the tendency to converge to a local minimum. This is generally undesirable since it implies that the knowledge acquired by the network is not optimal. To counter this problem, modified backpropagation algorithms have been developed which include a momentum term that can kick the parameters out of sub-optimal solutions. However, with these algorithms one has to fiddle around with the momentum term and hope that, with the selected starting point, a globally optimal solution can be reached. In general, there is no guarantee of achieving a global optimum.

In our quest to improve the efficiency of the neural network learning, which we believe is critical in equipping the network with the knowledge required for reliably recognizing complex target maneuvers, we employ a training scheme based on classical simplex optimization approach [14]. For achieving some reduction in the overall training complexity, the algorithm is executed by splitting the 3-layer neural network into two portions – a linear portion and a nonlinear portion. The connections between the input layer and the hidden layer form the nonlinear portion, while the connections between the hidden layer and the output layer constitute the linear portion. The simplex optimization method is used to find the optimal weights in the nonlinear portion, while a linear least squares minimization is used to determine the optimal weights in the linear portion of the network [10]. For implementation in the present context, the algorithm can be designed with two distinct stopping criteria. The search for the weights in a specified network structure can be terminated either when the size of the simplex is smaller than a prespecified threshold or the number of iterations performed exceeds a preset threshold.

Figure 4 illustrates how the simplex algorithm converges to an optimal solution. The simplex is initialized by selecting a set of $N + 1$ points in the N-dimensional weight vector of dimension N). This selection is made by randomly assigning all weight values within the bounds $W_{\max}$ and $W_{\min}$. In Figure 4 is shown an illustrative case of 4 simplex points (for a problem with 3 dimensional vectors). The well known simplex evolution strategy [14] is then executed, which involves determining the point where the mean square error (MSE) (for optimization of an error functional formulated as the criterion for training) is the largest and computing the centroid of the remaining simplex points. A new simplex point is then created by an expansion or contraction operation which involves joining the centroid computed to the simplex point with the highest MSE by an invisible line and locating an expansion point or a contraction point on this line as shown in Figure 5. The highest MSE point is then replaced by the newly generated point to form the new simplex on which the set of operations is repeated. This strategy of evolving the simplex is repeated until one of the termination criteria stated earlier is reached.

As described by Hsu et al [10], implementing the simplex algorithm described above with multiple restart operation (i.e. reinitializing the simplex and executing the algorithm on the new simplex points) has global search property and hence prevents the training procedure to be trapped by local minima of the error function. Furthermore, as discussed in [7], multiple restarts of the simplex search each time a convergence to a small cluster is attained, guarantees that the procedure will find a globally optimal solution with probability approaching 1.0.

As noted before, the efficiency of the neural network learning depends critically on the generation of training vectors used as inputs to the neural network. In the present application of training the network to recognize target maneuvers, two of the three input features are developed from the innovation data, which is in turn constructed from the TWS radar measurements and the Kalman filter outputs. Consequently, network training cannot be conducted in isolation and must be performed in conjunction with the Kalman filter in an architectural arrangement shown in Figure 3. A difficulty in the training vector generation however comes from observing

that the neural network needs the target state estimation from the Kalman filter in order to be trained, whereas in order to generate a correct target state estimate the Kalman filter needs the maneuver parameters to be estimated by the neural network. To overcome this problem and to execute an implementable training procedure, an iterative scheme shown in Figure 6a can be used for generation of training data. In this scheme, the Kalman filter first provides a state estimate without including the maneuver made by the target (i.e. by setting $u = 0$ in the target dynamical model given by equation (1)). This state estimate, which is evidently inaccurate, is used to generate the feature vectors needed by the neural network to train. The Kalman filter is next updated with the true maneuver parameters so that an accurate state estimate can be obtained. This process is repeated iteratively until all the training data are generated.

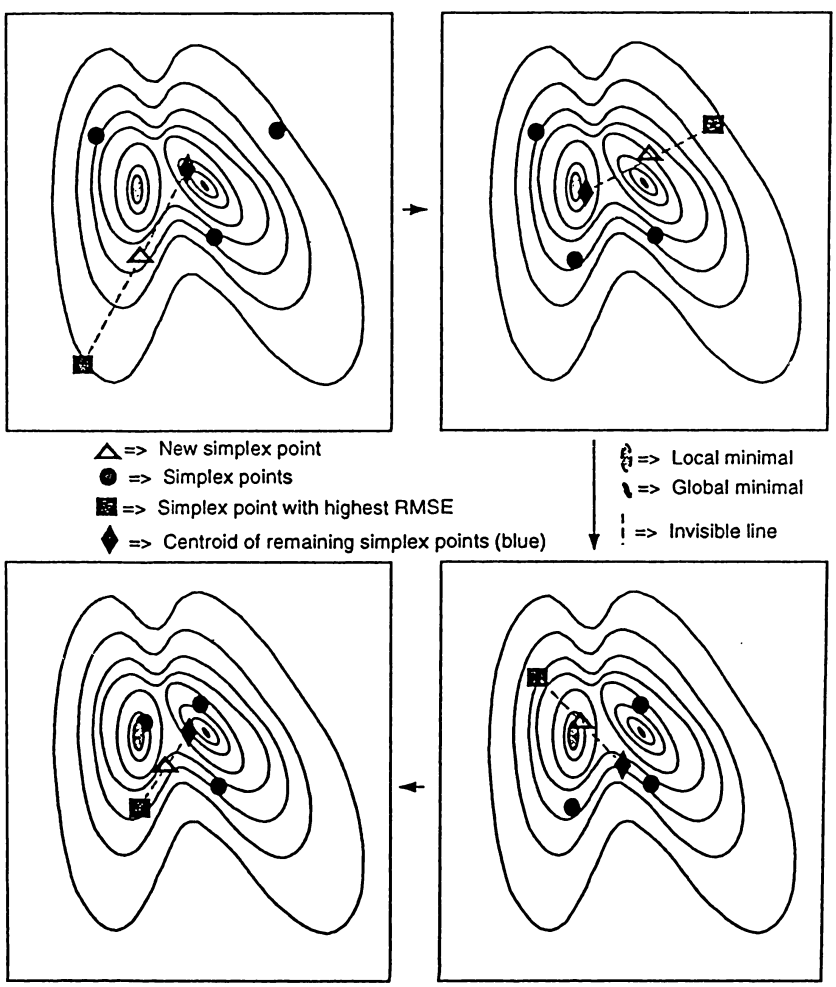

Fig. 4. Convergence of Simplex algorithm to a global solution.

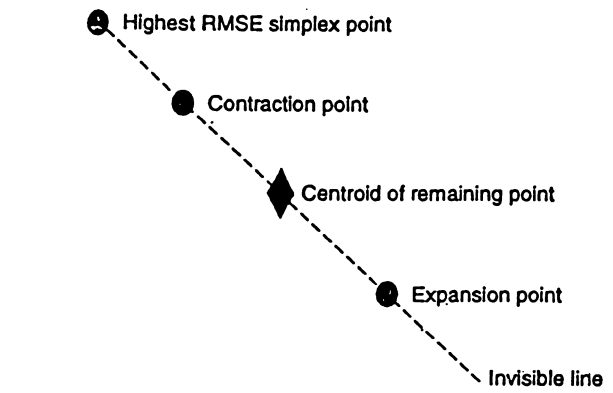

Fig. 5. Illustration of expansion and contraction in the Simplex algorithm.

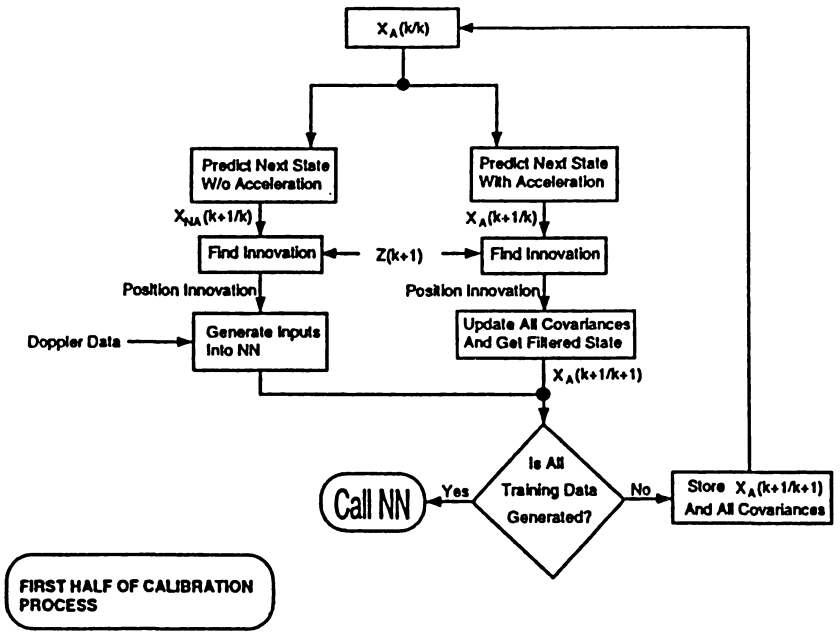

Fig. 6a. Flow-chart depicting the generation of training data.

3.2. Network growing for optimal size

The accuracy with which a neural network models a certain process characterized by an input-output mapping or recognizes a set of input patterns depends on a number factors, the principal one being the size of the hidden layer (or layers in a network which is configured with more than three layers). Only general guidelines however exist for arriving at the optimal architecture to be used in a given application. The more complex the input-output mapping to be approximated, the larger is the set of hidden nodes required, which determines the network size. Employing a larger sized network than necessary has its own drawbacks in that while such a network can learn the input-output mapping presented in the training data, it will attempt to memorize the training patterns used and has poor generalization abilities, i.e. provide the correct functional representation for input-output data not included in the training pattern set [9].

In arriving at a network architecture of optimal size for a given application, two approaches are generally possible. One is to start with a larger number of hidden nodes than necessary and later prune the network by removing redundant nodes. The other is to start with a small sized network initially (with the least number of hidden nodes, for instance) and to progressively grow until a desired degree of accuracy in modeling is achieved. Both of these approaches have been used by various groups of researchers in tailoring an optimal sized network for the specific application at hand.

In our present application in training the network to recognize target maneuvers, we have chosen to use the latter approach for a number of reasons, the principal ones being the following. First of all, the task of training here is a significant one due to the number of feature vectors that may be used for obtaining an adequate representation of complex maneuvers. Consequently, the former approach of starting with a network size larger than required can result in unnecessary increased training complexity, with increased learning times and cost particularly at the initial stages. Secondly, and more importantly, a systematic network growing approach can be built into the overall training algorithm with a convergence condition (stopping rule) being declared when the optimal values of the weights in the correct sized network are obtained.

Such a network growing approach can be integrated with the simplex algorithm described in the last section resulting in an overall training scheme depicted by the flow-chart shown in Figure 6b. For implementation in estimation of maneuver parameters, one starts with the simplest network architecture with one node in the hidden layer, while the input layer comprises of a number of nodes equal to the number of features used for training and the output layer comprises of a number of nodes equal to the number of maneuver parameters to be estimated (which are in turn input to the Kalman filter algorithm). The simplex algorithm is then executed to find the best weights for this structure. Once the weights are found, the MSE for this structure is computed and stored together with its weights. The network is then allowed to grow its hidden layer by adding one node and the simplex algorithm is executed once again with the same training data as before. Once the weights for the new structure are found, the MSE for this structure is computed and compared

to the stored MSE for the previous structure. If the new MSE is smaller than the previously stored MSE by a preset value, the new structure together with its weights and MSE are stored replacing the previous values. The network is then grown by an additional hidden node and the entire process is repeated. If at any stage of this process, the new MSE is worse than the previously stored MSE or is not better by a preset value, then an optimal structure is claimed to have been found and the training is terminated.

A multiple restart of the simplex search can be executed as a part of the overall training process in order to ensure attaining a global minimum of the objective function and hence optimize the training efficiently. The various steps underlying the training process are depicted in the flow-chart given in Figure 6b.

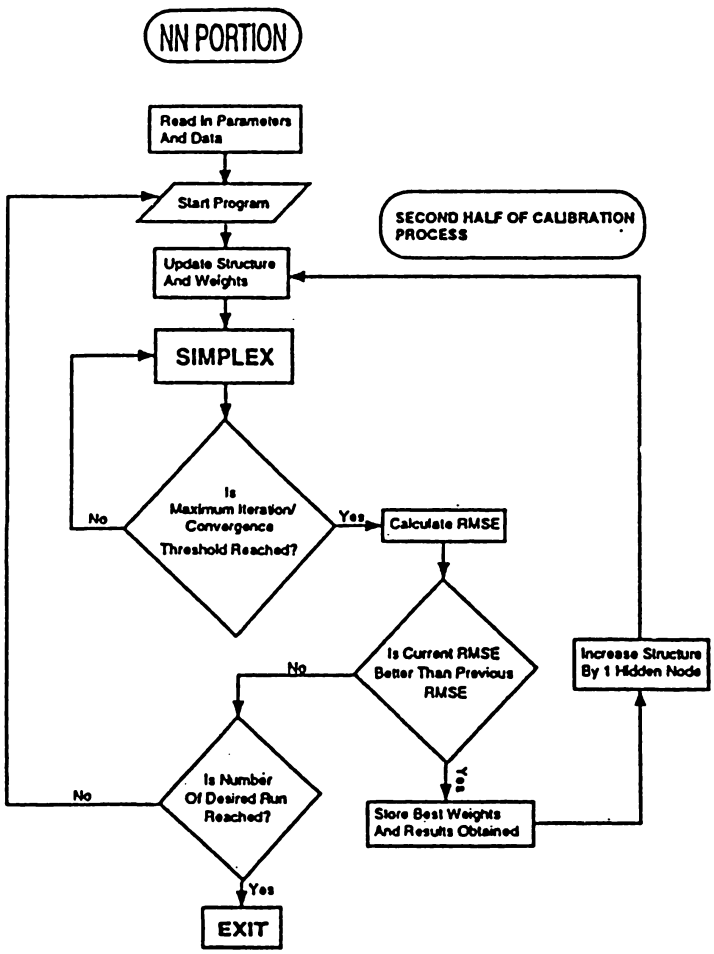

Fig. 6b. Flow-chart depicting the training scheme.

4. TRACKING PERFORMANCE OF NEURAL NETWORK–BASED SCHEME

To perform validation studies that confirm the efficiency of the training scheme used, several tracking experiments were conducted. The following parameter values were employed for the simulations:

(i) Radar scan period = 10 seconds;

(ii) Standard deviation of range = 100 meters;

(iii) Standard deviation of angle = 0.0003 radian;

(iv) Doppler radar wavelength = 8.57×10^{-3} meter.

In order to evaluate any possible degradation in tracking performance due to clutter, simulations of both clutterless and clutter environments were made. The primary difference between the two cases is the use of a standard Kalman filter for the clutterless environment, whereas a Probabilistic Data Association Filter (PDAF) is used to replace this in scenarios that include clutter [2]. For simulations of tracking experiments in clutter environments, the following parameter values were used:

(i) Spatial clutter density = 0.0009;

(ii) Validation gate size = 16 (Chi-square distribution);

(iii) Probability of detection = 0.9;

(iv) Probability of target inside gate = 0.9997.

It may be noted that the choice of the validation gate corresponds to a rather heavy clutter environment.

Experiment 1. In this experiment, the target is initially detected at the location (1.5×10^3 m, 1.5×10^3 m) in cartesian coordinates and its flight path is at an angle 45° with respect to the x-axis. The target travels at a constant velocity of 250 m/sec during the first three scans and is radially moving away from the radar. The maneuver consists of a sharp acceleration of 5 m/sec^2 performed at the 4th scan (i.e. t = 40 sec) and lasting for 1 scan period (i.e. 10 sec), after which esumed until the 20th scan.

The tracking performance under these conditions is shown in Figures 7a–c. The plots of the position errors in the x- and y- directions shown in Figure 7a indicate that the corrected state estimates are fairly accurate and the rather large error at the onset of the maneuver (at the 4th scan) is well corrected. Although the position error plots show a trend for increasing error, it must be noted that the target is moving away form the radar. Thus, although the absolute value of the error appears to increase, the relative error is quite small. For instance, at the end of the trajectory (20th scan) the target is at the location (2.18×10^3 m, 2.18×10^3 m) and the relative error at this instant is only 0.01 %. The apparent divergence in the position is also partly due to the occurrence of a false alarm as can be seen from the acceleration plots in Figure 7c. The true prediction of the target acceleration during the maneuver by the neural network deserves a particular note.

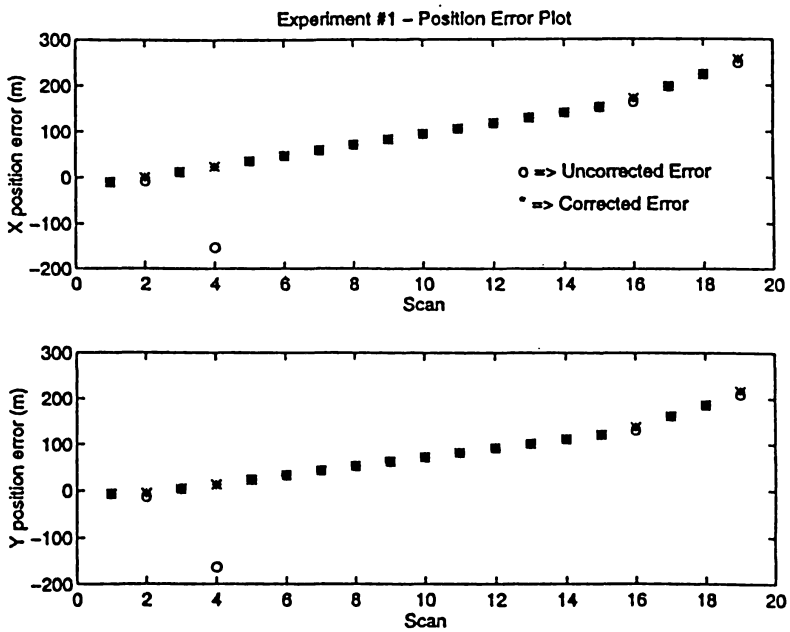

Fig. 7a. Position error plots.

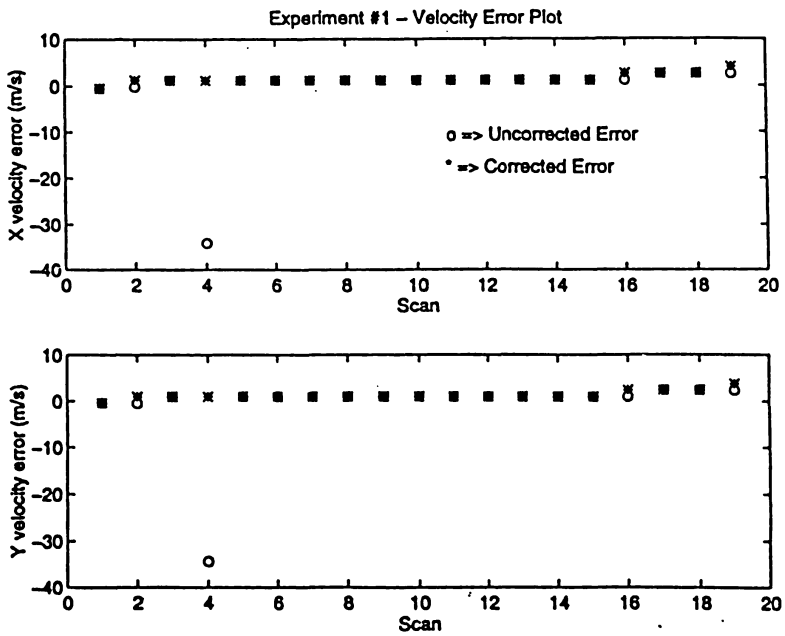

Fig. 7b. Velocity error plots.

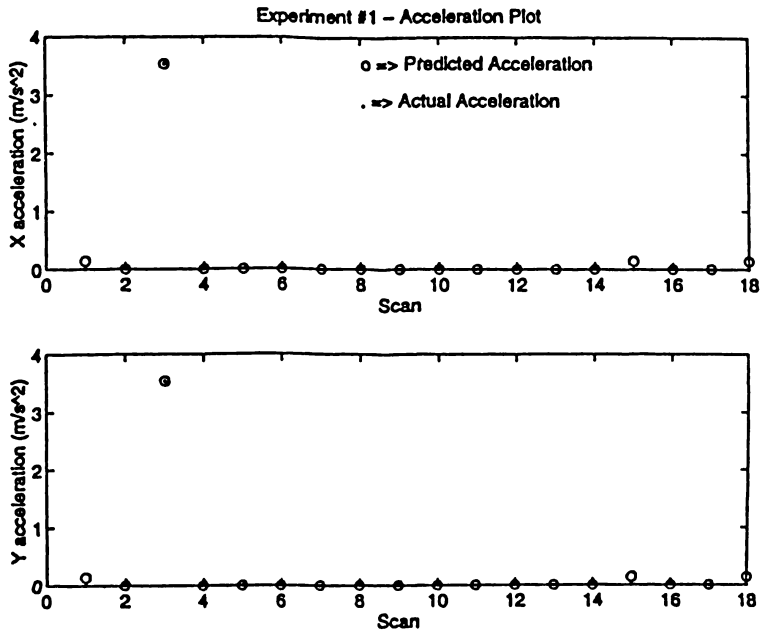

Fig. 7c. Estimated target acceleration in Experiment 1.

Experiment 2. This experiment is similar to Experiment 1 except that the target now moves in a clutter region which extends between the 10th and the 20th scans. The target flying with an initial velocity of 250 m/sec executes at the 4th scan a maneuver comprising of 5 m/sec^2 acceleration for 1 scan period, as before. The tracking performance delivered by the neural network-based tracking scheme is shown in Figures 8a–c. The position error and velocity error, shown in Figures 8a and 8b, indicate a very satisfactory tracking performance and the acceleration plots in Figure 8c indicate that the target maneuver is estimated very accurately.

Experiment 3. The primary goal in this experiment was to evaluate the tracking performance under a combination of maneuvers executed by the target in quick succession. With the target initially located at (1.5×10^3 m, 1.5×10^3 m) and moving at a constant velocity of 250 m/sec while maintaining 45° heading, a combination of two maneuvers were executed. The first maneuver occurs at the 6th scan (i.e. $t = 60$ sec) and consists of a 5 m/sec^2 acceleration which lasts for 1 scan period. The second maneuver consisting of a 10 m/sec^2 acceleration occurs at the 9th scan (i.e. $t = 90$ sec) and lasts for 2 scan periods. After the second maneuver, the target resumes a constant velocity flight. To further provide a challenging envir 09 was added at the on set of the set maneuver and was maintained until the end of the experiment.

The plots depicting the resulting performance are given in Figures 9a–c. It can be seen from Figure 9c that the maneuvers executed by the target are detected and

estimated accurately within one sampling period. The position and velocity error plots in Figures 9a and 9b indicate that the corrected position and velocity estimates are very close to the true target position and velocity, while significant errors can result due to the maneuvers when left uncorrected (see the "uncorrected error" entries in Figures 9a and 9b). It is rather well known that when a combination of different maneuvers as in the present experiment takes place, classical tracking methods (such as those based on an Input Estimation Approach [5] deliver acceptable performance due to the fact that compensation for the first maneuver may not be completed before the second maneuver gets initiated [16]. The reason for this shortcoming is that these methods require one to go back a certain number of sampling periods (depending on the window length selected for the computation of the so called "propagation matrix" [3]) in order to compute the correction amounts to be applied.

Experiment 4. To evaluate the tracking performance when the target executes a combination of different maneuvers with negligible time interval separation between them, in this experiment we let the target perform a $10\,\mathrm{m/sec^2}$ acceleration at the beginning of the 5th scan and lasting only half a scan period. This was then followed by a $20\,\mathrm{m/sec^2}$ acceleration during the 6th scan and lasting for one scan period. The target then resumes a constant velocity flight after the s erformance under these conditions is shown in Figures 10a–c, which demonstrate the capability of the neural network to estimate reliably the different maneuvers even in this exceedingly challenging scenario.

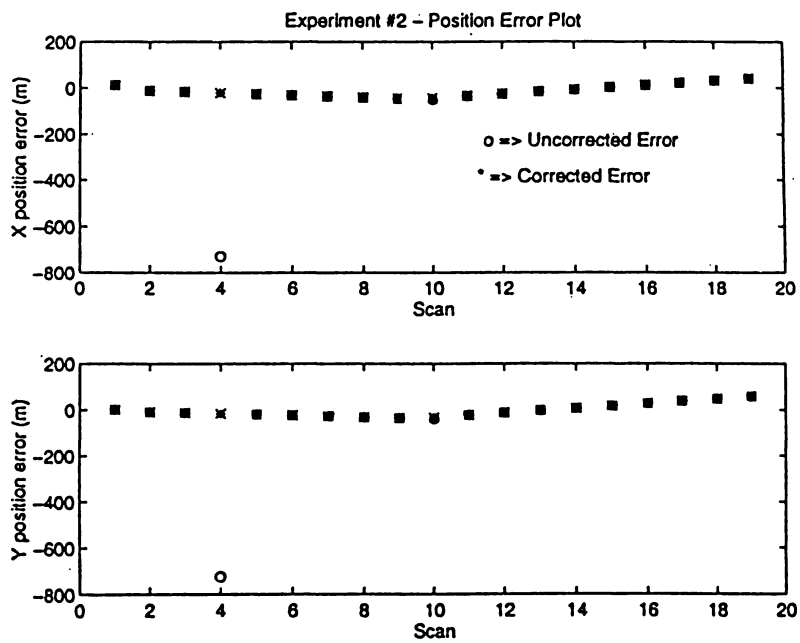

Fig. 8a. Position error plots.

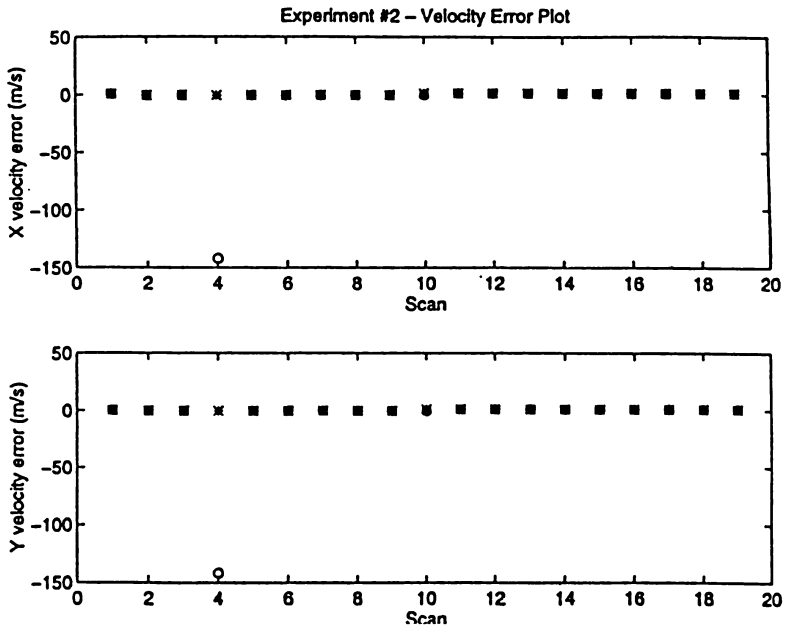

Fig. 8b. Velocity error plots.

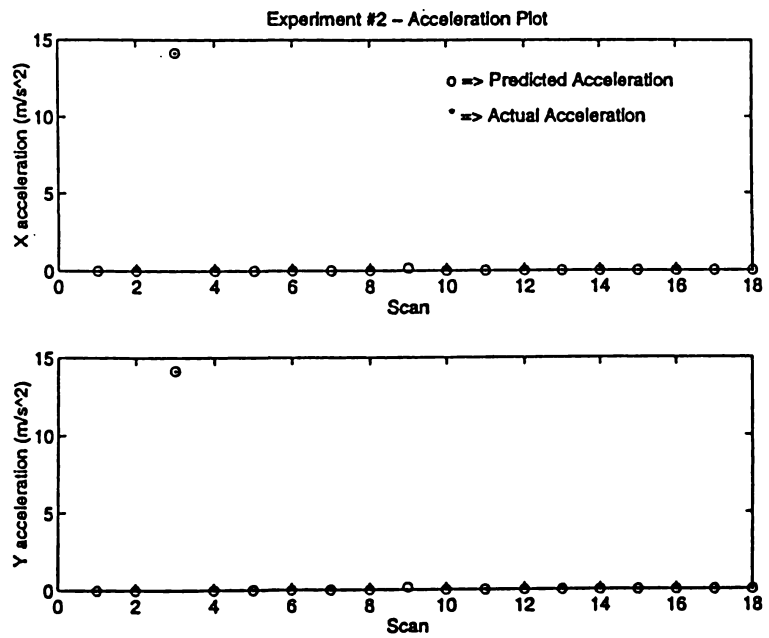

Fig. 8c. Estimated target acceleration in Experiment 2.

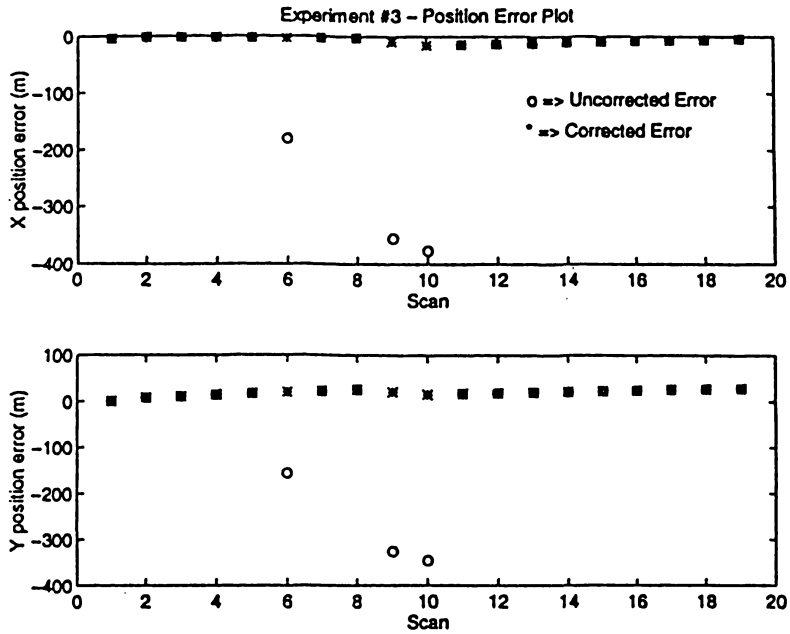

Fig. 9a. Position error plots.

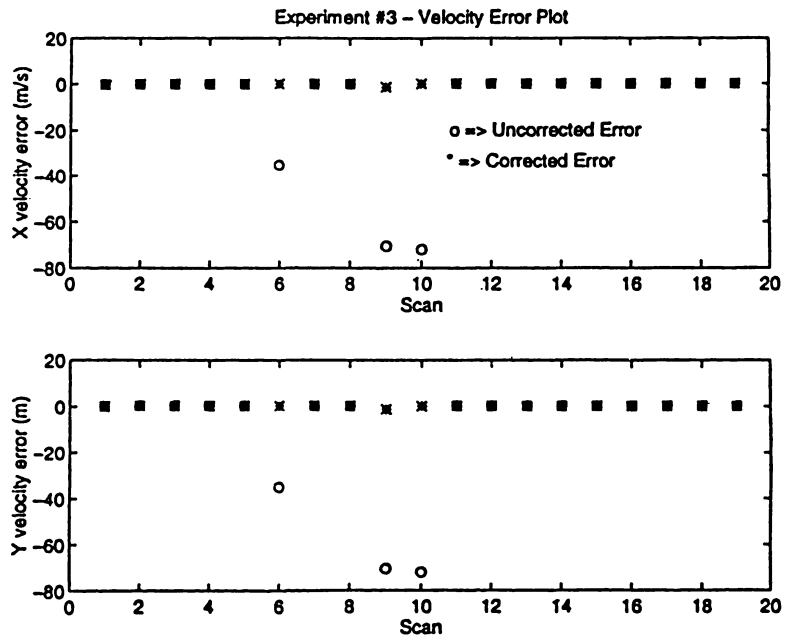

Fig. 9b. Velocity error plots.

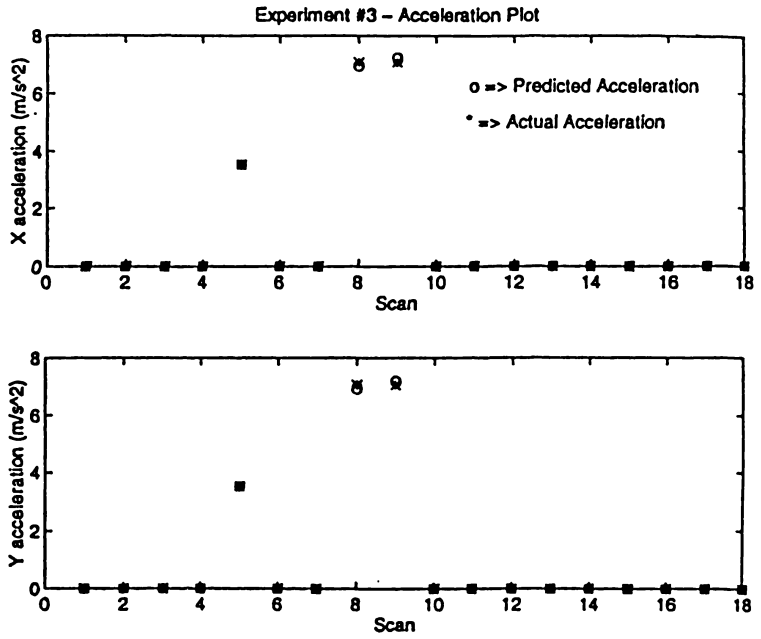

Fig. 9c. Estimated target acceleration in Experiment 3.

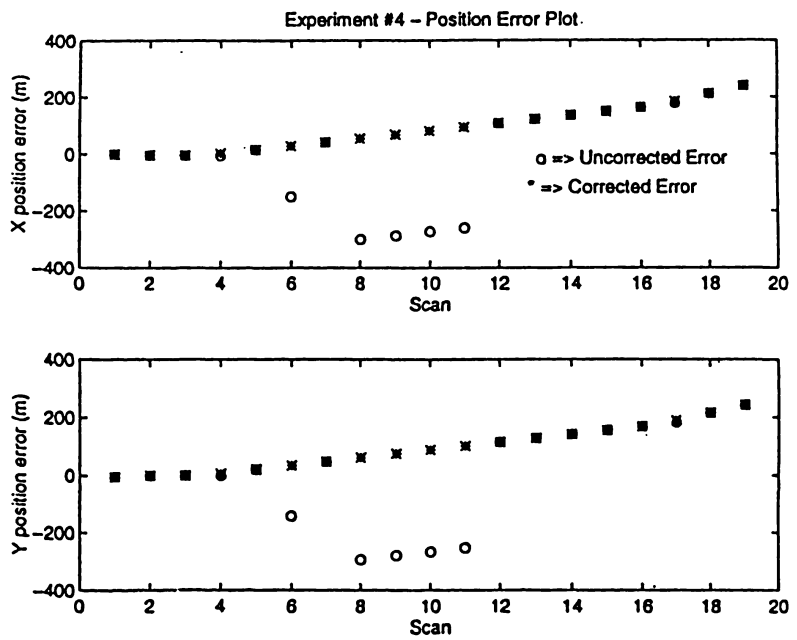

Fig. 10a. Position error plots.

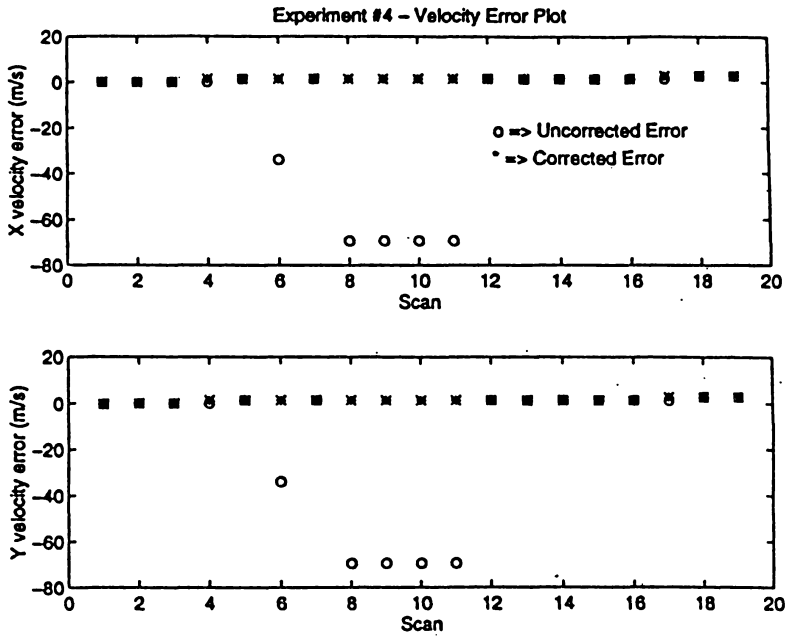

Fig. 10b. Velocity error plots.

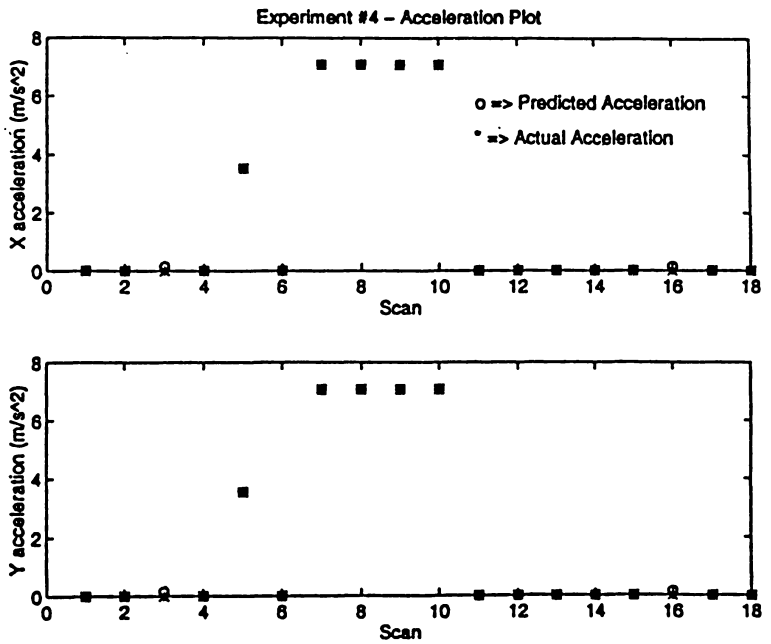

Fig. 10c. Estimated target acceleration in Experiment 4.

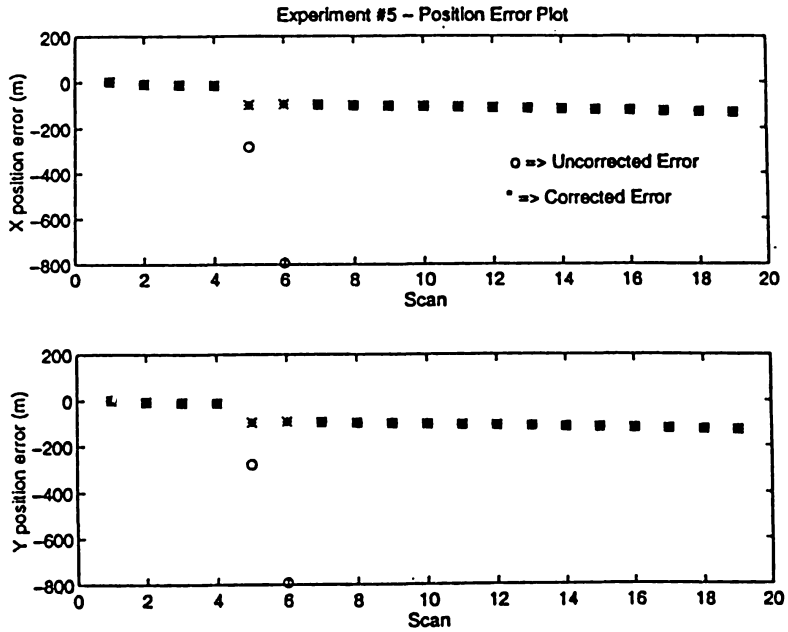

Fig. 11a. Position error plots.

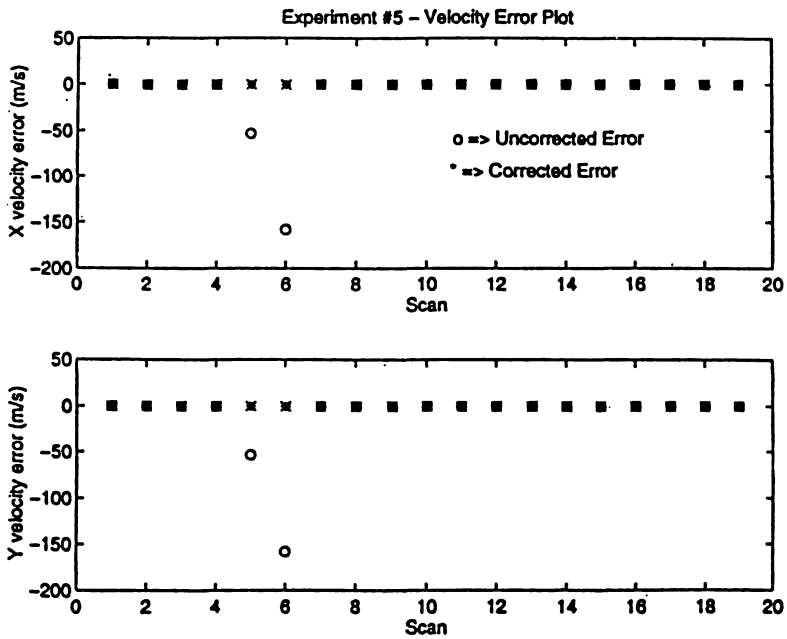

Fig. 11b. Velocity error plots.

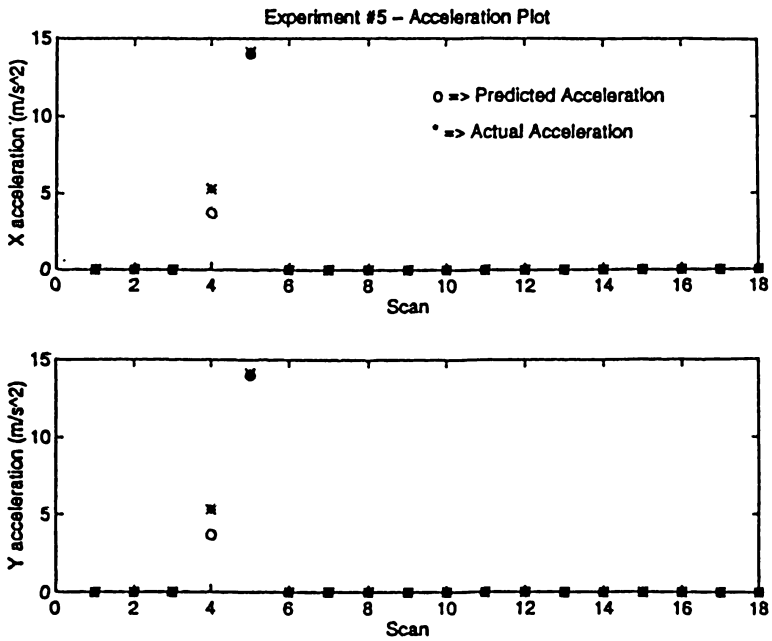

Fig. 11c. Estimated target acceleration in Experiment 5.

5. CONCLUSIONS

The details of a neural network-based tracking architecture, which successfully integrates a neural network trained with feature vectors derived from sensor measurements with a simple Kalman filter algorithm, have been presented in this paper. The performance evaluation studies, some of which are reported here, have indicated that this architecture is capable of yielding specific tracking algorithms that deliver a reliable tracking performance despite the complexity of the environments (complex maneuvers, high degree of noise and clutter, etc.) in which tracking is required. A principal advantage of the present approach that should be underscored is its capability to utilize measurements from a diverse set of sensors for improved tracking. This function of sensor fusion is accomplished by the feature integration performed by the neural network which efficiently combines the different sets of features extracted from the data streams coming from the various sensors.

Since the features extracted from measured data are used for training the neural network to estimate the parameters characterizing the target maneuver, the performance of the target tracking scheme critically depe vectors. If a more complete set of features that efficiently code the relevant information about the maneuver can be abstracted from the measurements, a better learning of the maneuver characteristics can be obtained by the neural network. In particular, a more mathematical

approach for feature extraction can result in a systematic development of the feature vectors from data collected by different types of sensors deployed in the surveillance and tracking environment. Our continuing work is hence focussing on the task of improved feature extraction process.

(Received April 8, 1998.)

REFERENCES

[1] F. Amoozegar and M. K. Sundareshan: Constant false alarm rate target detection in clutter: A neural processing algorithm. In: Proc. SPIE Internat. Symposium on Optical Engineering and Photonics in Aerospace Sensing (Applications of Artificial Neural Networks), Orlando 1994.

[2] Y. Bar-Shalom and X. R. Li: Estimation and Tracking. Artech House, 1993.

[3] P. L. Bogler: Tracking a maneuvering target using input estimation. IEEE Trans. Aerospace Electron. Systems *AES-23* (1987), 298–310.

[4] G. E. P. Box and G. M. Jenkins: Time Series Analysis, Forecasting and Control. Holden Day, San Francisco 1970.

[5] Y. T. Chan, A. Hu and J. B. Plant: A Kalman filter based tracking scheme with input estimation. IEEE Trans. Aerospace Electron. Systems (1979), 237–244.

[6] Y. T. Chien and K. S. Fu: Selection and ordering of feature observations in a pattern recognition system. Inform. and Control *12* (1968), 395–414.

[7] Q. Duan, H. V. Gupta and S. Sorooshian: Effective and efficient global optimization for conceptual rainfall-runoff models. Water Resources Research *28* (1992), 1015–1031.

[8] R. J. Evans, F. Barker and Y. C. Soh: Maximum likelihood estimation of constant heading trajectories. In: Proc. IEEE Internat. Radar Conf., London 1987, pp. 489–493.

[9] M. T. Hagan, H. B. Demuth and M. Beale: Neural Network Design. PWS Publishing Co., 1996.

[10] K. L. Hsu, H. V. Gupta and S. Sorooshian: Artificial neural network modeling of rainfall-runoff process. Water Resources Research *31* (1995), 2517–2530.

[11] A. Karakasoglu, S. I. Sudharsanan and M. K. Sundareshan: Identification and decentralized adaptive control using dynamical neural networks with application to robotic manipulators. IEEE Trans. Neural Networks *2* (1993), 919–931.

[12] X. R. Li and Y. Bar-Shalom: Design of an interacting multiple model algorithm for air traffic control tracking. IEEE Trans. Control Systems Technology *1* (1993), 3, 186–194.

[13] L. Ljung: System Identification: Theory for the User. Prentice-Hall, Englewood Cliffs, N.J. 1987.

[14] A. J. Nelder and R. Mead: A simplex method for function minimization. Comput. J. 7 (1965), 308–313.

[15] M. K. Sundareshan and F. Amoozegar: Data fusion and nonlinear tracking filter implementation using multilayer networks. In: Proc. IEEE Internat. Conf. on Neural Networks (ICNN' 95), Perth 1995.

[16] M. K. Sundareshan and F. Amoozegar: Neural network fusion capabilities for efficient implementation of tracking algorithms. Optical Engineering *36* (1997), 692–707.

[17] S. I. Sudharsanan and M. K. Sundareshan: Supervised training of dynamical neural networks for associative memory design and identification of nonlinear maps. Internat. J. Neural Systems *5* (1994), 165–180.

[18] D. D. Sworder: Image enhanced tracking. IEEE Trans. Aerospace Electron. Systems *AES-25* (1989), 701–710.

Dr. Yee Chin Wong and Prof. Dr. Malur K. Sundareshan, Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721-0104. U. S. A.
e-mails: yeechin@ece.arizon.edu, sundareshan@ece.arizona.edu