A SIMPLEX TRAINED NEURAL NETWORK-BASED ARCHITECTURE FOR SENSOR FUSION AND TRACKING OF TARGET MANEUVERS

Yee Chin Wong and Malur Sundareshan
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 85721-0104
Tel: (520) 621-2953; Fax: (520) 621-8076
Email: sundareshan@hermes.ece.arizona.edu

ABSTRACT

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. Validation of these methods comes from several tracking experiments involving targets executing complex maneuvers in noisy and clutter environments. Results from one such experiment is included here for illustration.

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 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. Development of such processing methods aimed at enhancing the tracking performance even in scenarios where a typically noncooperative target is executing complex maneuvers is a particularly challenging task.

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. For the particular case of target tracking, it is rather 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) [1]. The enormous processing complexity 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 feature level fusion architectures that can 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 (such as integration of radar data and image – format data, for instance) 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 Fig. 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 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.

Some of our recent studies [2-4] 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 [5], 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.

NEURAL NETWORK-BASED ARCHITECTURE FOR SENSOR FUSION AND TARGET TRACKING

The basic building blocks of the tracking scheme shown in Fig. 2 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, Fig. 2 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.

Training of the neural network for providing maneuver estimates is implemented with three features extracted from the measured data. Two of these features $\upsilon_1(k)$ and $\upsilon_2(k)$ are obtained from the measurements of the range radar (a TWS radar, for illustration) and the other feature $\upsilon_3(k)$ is obtained from the measurements from the Doppler radar (as shown in Fig. 2). More specifically, the signal $\upsilon_1(k)$ is constructed by normalizing the two components of the innovation data with respect to the covariance as

$$\upsilon_1(k) = \frac{\tilde{z}_x^2(k)}{S_{xx}(k)} + \frac{\tilde{z}_y^2(k)}{S_{yy}(k)}$$

where $\tilde{z}(k) = [\tilde{z}_x(k)\ \ \tilde{z}_y(k)]^T = z(k) - H\hat{x}(k/k-1)$, $z(k)$ being the measurement and $\hat{x}(k/k-1)$ being the state estimate generated by the Kalman filter. $S_{xx}(k)$ and $S_{yy}(k)$ are the diagonal elements of the covariance matrix which is used for Kalman gain computation. Signal $\upsilon_2(k)$ is the change in the heading estimate computed as

$$\upsilon_2(k) = \alpha_{LT}(k) - \alpha_{LT}(k-1)$$

where $\alpha_{LT}(k)$ and $\alpha_{LT}(k-1)$ are the heading estimates computed by the method of least triangles from using three past data points. Finally, the third input feature $\upsilon_3(k)$ is extracted from Doppler radar and is computed as the change in Doppler shift normalized by its variance, i.e.,

$$\upsilon_3(k) = \frac{1}{\sigma_{fd}^2}[f_d(k) - f_d(k-1)] \quad \text{where } f_d(i) = \frac{2}{\lambda}\dot{R}(i)$$

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

Performance evaluation studies conducted earlier [2,3] 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 [3].

OPTIMIZATION OF NEURAL NETWORK ARCHITECTURE AND TRAINING

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. The knowledge acquired by the network during this learning is stored in the set of weights.

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 backpropagation approach is perhaps the most popular approach [5] 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). Also because of the gradient-based search property, it is normally trapped by the first optimal point reached 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 have implemented the Linear least Squares Simplex (LSSIM) algorithm developed by Hsu et. al. [6]. This algorithm employs concepts from simplex optimization and is conducted 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 [6]. 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.

As described by Hsu et. al. , 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.

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 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 [5].

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 Fig. 3. 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 mean square error (MSE) associated with 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 MS 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 than 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 Fig. 3.

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 [8]. 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. In the following we shall briefly present results from only one experiment due to page restrictions.

In this experiment, the target is initially detected at the location (1.5×10^3m, 1.5×10^3m) 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 250m/sec during the first three scans and is radially moving away from the radar. The maneuver consists of a sharp acceleration of 5m/sec^2 performed at the 4th scan (i.e. t=40sec) and lasting for 1 scan period (i.e. 10 sec), after which the constant velocity flight is resumed until the 20th scan.

The tracking performance under these conditions is shown in Figs. 4a-c. The plots of the position errors in the x- and y- directions shown in Fig. 4a 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 (2.18×10^3m, 2.18×10^3m) 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 Fig. 4c. The true prediction of the target acceleration during the maneuver by the neural network deserves a particular note.

REFERENCE

1. D.D. Sworder, "Image enhanced tracking", *IEEE Trans. Aerospace and Electronic Systems*, Vol. AES-25, pp. 701-710, 1989
2. M.K. Sundareshan and F. Amoozegar, "Data fusion and nonlinear tracking filter implementation using multilayer networks", *Proc. of IEEE Int. Conf. On Neural Networks (ICNN' 95)*, Perth, Australia, November, 1995.
3. M.K. Sundareshan and F. Amoozegar, "Neural network fusion capabilities for efficient implementation of tracking algorithms", *Optical Engineering*, Vol. 36, pp. 692-707, March 1997.
4. F. Amoozegar and M.K. Sundareshan, "Constant false alarm rate target detection in clutter: A neural processing algorithm", *Proc. SPIE Int. Symposium on Optical Engineering and Photonics in Aerospace Sensing* (Applications of Artificial Neural Networks), Orlando, FL, April 1994.
5. M.T. Hagan, H.B. Demuth and M. Beale, *Neural Network Design*, PWS Publishing Co., 1996.
6. K.L. Hsu, H.V. Gupta and S. Sorooshian, "Artificial neural network modeling of rainfall-runoff process", *Water Resources Research*, Vol. 31, pp. 2517-2530, 1995.
7. Q. Duan, H.V. Gupta and S. Sorooshian, "Effective and efficient global optimization for conceptual rainfall-runoff models", *Water Resources Research*, Vol. 28, pp 1015-1031, 1992.
8. Y. Bar-Shalom and X.R. Li, *Estimation and Tracking*, Artech House: 1993.

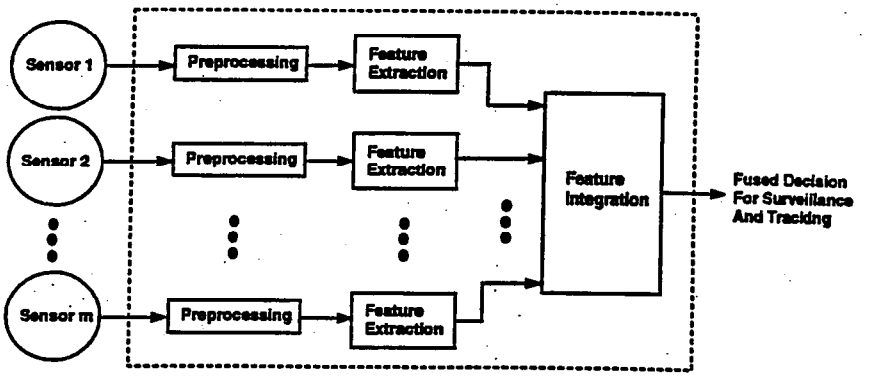

Fig. 1. Backbone processing architecture for intelligent integrated processing of multisensor data in surveillance and tracking

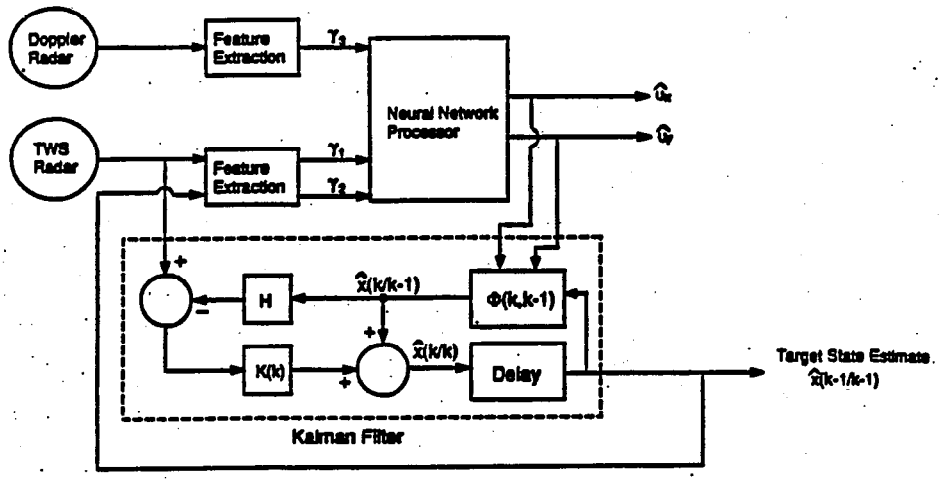

Fig. 2. A neural network-based fusion architecture for tracking target maneuvers

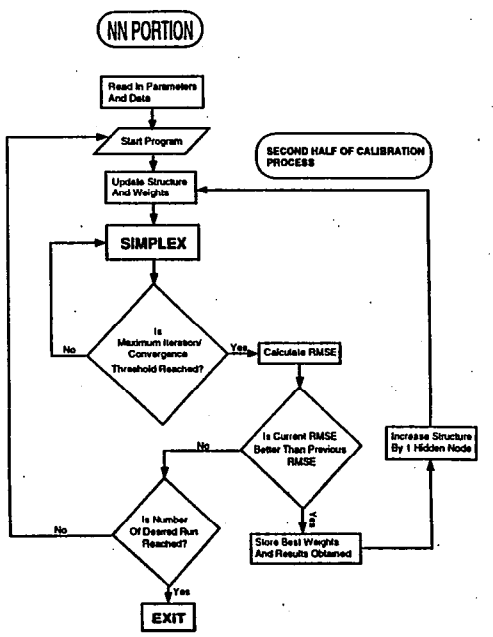

Fig. 3. Flow-chart depicting the training scheme

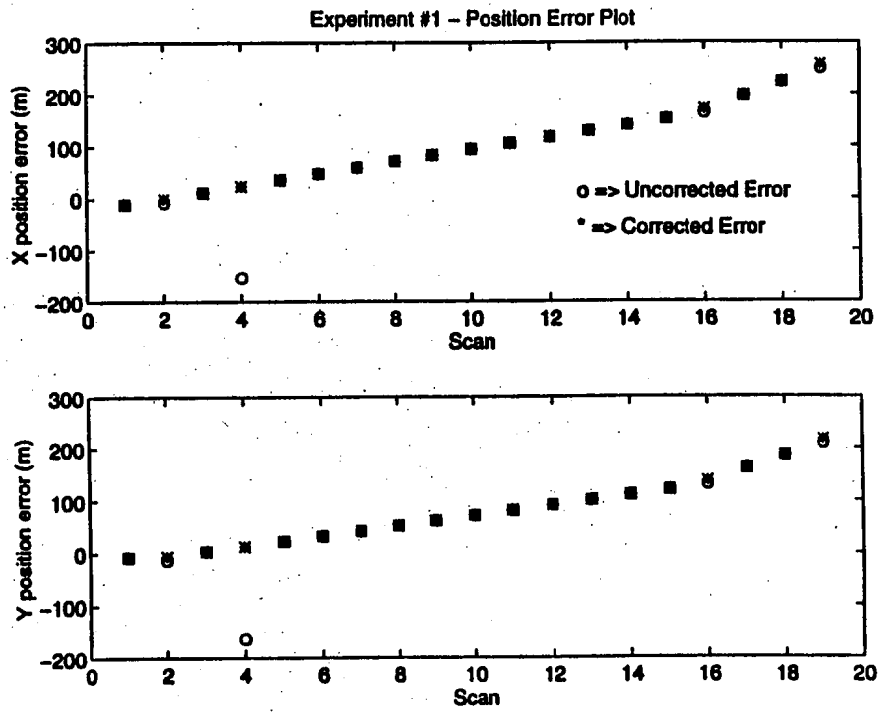

Fig. 4a. Position error plots

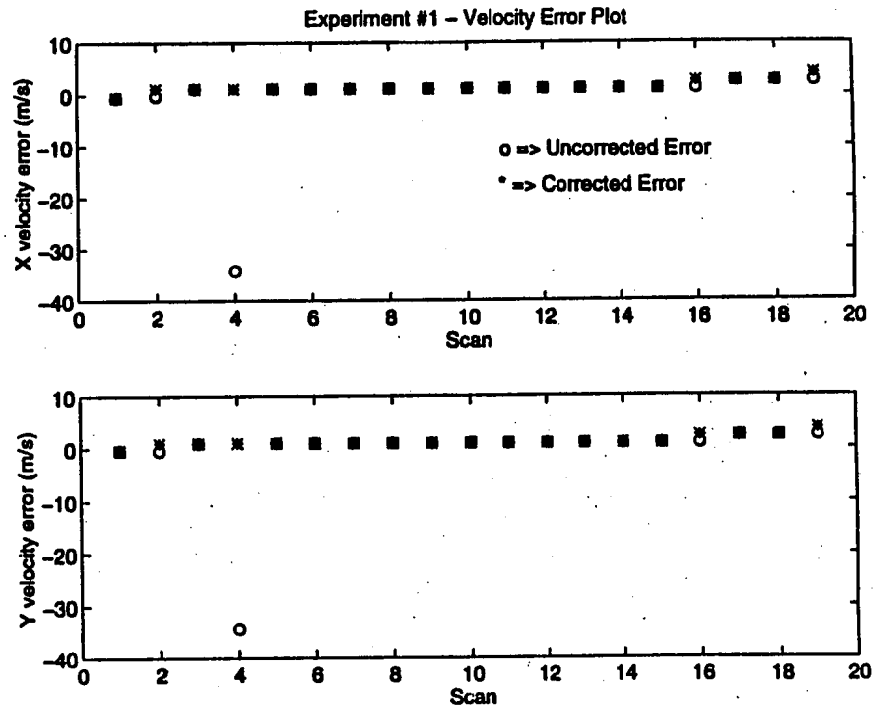

Fig. 4b. Velocity error plots

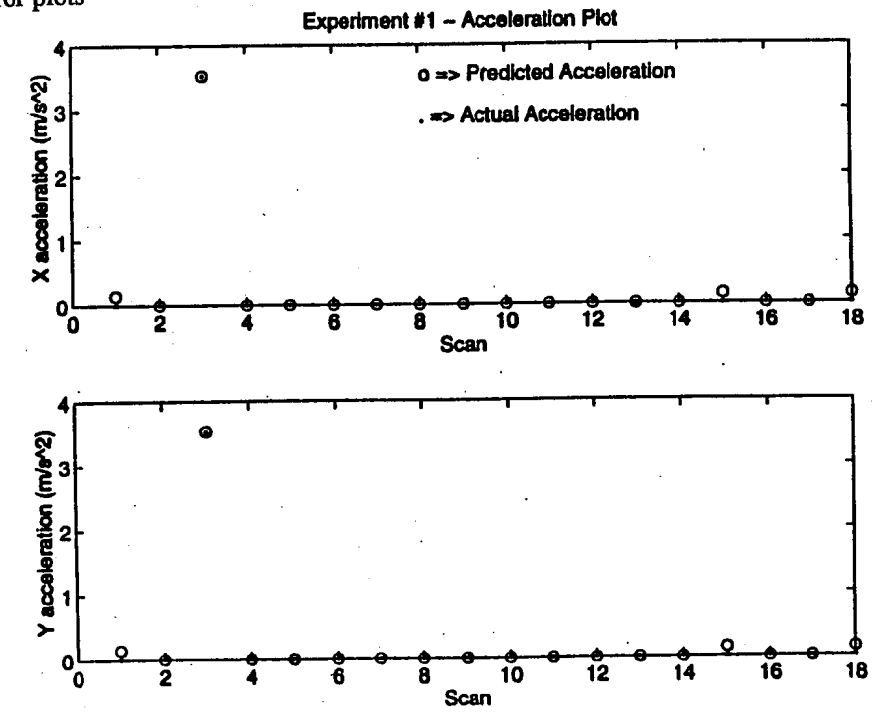

Fig. 4c. Estimated target acceleration