A Neural Extended Kalman Filter Multiple Model Tracker

M. W. Owen, U.S. Navy

SPAWAR Systems Center San Diego
Code 2725, 53560 Hull Street
San Diego, CA, 92152, USA
mark.owen@navy.mil

A. R. Stubberud, University of California

Department of Electrical Engineering and Computer Science
Irvine, CA 92697 USA
arstubbe@uci.edu

***Abstract*-A neural extended Kalman filter algorithm was embedded in an interacting multiple model architecture for target tracking. The neural extended Kalman filter algorithm is used to improve motion model prediction during maneuvers. With a better target motion mode, noise reduction can be achieved through a maneuver. Unlike the interacting multiple model architecture which, uses a high process noise model to hold a target through a maneuver with poor velocity and acceleration estimates, a neural extended Kalman filter is used to predict the correct velocity and acceleration states of a target through a maneuver. The neural extended Kalman filter estimates the weights of a neural network, which in turn is used to modify the state estimate predictions of the filter as measurements are processed. The neural network training is performed on-line as data is processed. In this paper, the results of a neural extended Kalman filter embedded in an interacting multiple model tracking architecture will be shown using a high fidelity model of a phased array radar. Six different targets of varying maneuverability will be tracked. The phased array radar is controlled via Level 4 Data Fusion feedback to the Level 0 radar process. Highly maneuvering threats are a major concern for the Navy and DoD and this technology will help address this issue.**

I. INTRODUCTION

The Robust Tracking with a Neural Extended Kalman Filter (NEKF) project is an Office of Naval Research (ONR) In-House Laboratory Independent Research (ILIR) sponsored effort at SPAWAR Systems Center San Diego. The project's goal is to provide an improved state estimation capability for current U.S. Navy tracking systems. The NEKF provides added capability for real-time modeling of maneuvers and, therefore, enhances the ability of tracking systems to adapt appropriately.

Extended Kalman filters using neural networks have been used in the past in control system technology and for system identification [1, 2]. In this paper, the NEKF will be incorporated into an interacting multiple model tracking architecture to provide robust tracking capabilities that are currently unavailable.

In [3] the second Tracking Benchmark problem was presented to researchers to use as a testing environment for new tracking algorithms. This paper will show preliminary results on this benchmark problem.

II. BACKGROUND

State estimation and tracking of highly maneuvering targets is an extremely difficult task in modern tracking systems. Current state estimation approaches to the tracking problem include alpha-beta filters, Kalman filters, interacting multiple model (IMM) filters, probabilistic data association (PDA) trackers, and joint PDA (JPDA) trackers [4 and 5]. State estimation is the problem of estimating a set of system states that are of interest to a system designer or a decision maker. System states consist of parameters such as position, velocity, frequencies, magnetic moments, and other attributes of interest. A mathematical system model is necessary for the aforementioned filter algorithms to perform state estimation.

2.1 Kalman Filter

A well known state estimation algorithm is the Kalman filter which was developed four decades ago by R. E. Kalman [6]. A Kalman filter consists of the dynamic system to be tracked, a mathematical system model, an observation model, the Kalman gain, a predicted observation, and the system state vector. A problem occurs when the aircraft or system being tracked deviates from the assumed motion model. The filter will tend to lag behind the true state of the target and can even diverge, become unstable, and be unable to estimate the system states. In cases where the motion model and/or the observation model are nonlinear, an extension of the linear Kalman filter must be used. A common nonlinear extension of the Kalman filter is the extended Kalman filter (EKF) [7], which can handle known nonlinearities.

2.2 Interacting Multiple Model Filter

Another well known state of the art tracking technique is the interacting multiple model (IMM) filter [8]. The technique employs multiple models (a bank of Kalman filters) to perform state estimation. Each model may contain a different mathematical system model, observation model, variable dimension state vector, or noise processes. The IMM architecture can also use EKF's.

2.3 Extended Kalman Filter Neural Network Training

If a nonlinear model is unattainable, then a system identification technique might be used to create a model. In the late 80's and early 90's, the technology of using artificial neural networks for identification became popular. An artificial neural network is actually a function approximator, that is, given a set of inputs and a desired set of outputs, a neural network can be trained to approximate a smooth function relating the two. A neural network can be thought of as a nonlinear polynomial in which the coefficients of that polynomial must be found to approximate a desired function. A neural network contains

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **01 SEP 2003**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **A Neural Extended Kalman Filter Multiple Model Tracker**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **SPAWAR Systems Center San Diego Code 2725, 53560 Hull Street San Diego, CA, 92152, USA**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM002146. Oceans 2003 MTS/IEEE Conference, Held in San Diego, California on September 22-26, 2003. U.S. Government or Federal Purpose Rights License, The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**9**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

a set of weights (coefficients) that must be determined in order to approximate a function. To train neural networks, techniques such as backpropagation [9] and the extended Kalman filter [10] have been used. A neural network equation is shown in (2.1).

$$NNm = \sum_{i=1}^{N} wim * \left(fi \left(\sum_{k=1}^{J} Ik * wki \right) \right) \tag{2.1}$$

where $f_i = \dfrac{1}{1+\exp(-x_i)}$ is the output of the ith hidden node, x_i is the dot product sum of the previous input layer's outputs with the connecting weights of the hidden layer, NN_m is the mth output of the neural network, w_{im} is the mth output weight connected to the ith hidden node, w_{ki} is the kth input weight connected to the ith hidden node, and I_k is the kth input feeding the neural network.

2.4 Neural Extended Kalman Filter

The Neural Extended Kalman Filter (NEKF) developed by Stubberud [1] is based on the Singhal and Wu EKF neural network trainer in [10]. The algorithm uses an extended Kalman filter to estimate the states by using a dynamic system model while, at the same time, using the extended Kalman filter to train a neural network to calculate the nonlinearities, mismodeled dynamics, higher order modes, and other unknown facets of a system. Estimation of the system states are performed at once without the necessity of modeling the nonlinearities *a priori* as in the case of the extended Kalman filter. The neural network's function is described below

Given the true target motion model defined by the nonlinear vector equation

$$x_{k+1} = f(x_k, u_k) \tag{2.2}$$

and an estimator's view defined by the "hat" system

$$\hat{x}_{k+1} = \hat{f}(x_k, u_k) \tag{2.3}$$

an NEKF is used to correct the errors in the "hat" system. Ideally this would mean

$$x_{k+1} = f(x_k, u_k) = \hat{f}(x_k, u_k) + NN(x_k, u_k, w_k) \tag{2.4}$$

where NN is the neural network trained on-line as data is processed by the NEKF.

A mathematical system model of the neural network is

$$\overline{x}_{k+1} = \begin{bmatrix} x_{k+1} \\ w_{k+1} \end{bmatrix} = \Phi_{k+1} \begin{bmatrix} x_k \\ w_k \end{bmatrix} = \begin{bmatrix} A + \dfrac{\partial NN}{\partial x_k} & \dfrac{\partial NN}{\partial w_k} \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ w_k \end{bmatrix} \tag{2.5}$$

where

$$x_{k+1} = Ax_k + \left(\frac{\partial NN}{\partial x_k} \right) x_k + \left(\frac{\partial NN}{\partial w_k} \right) w_k \tag{2.6}$$

$$w_{k+1} = w_k \tag{2.7}$$

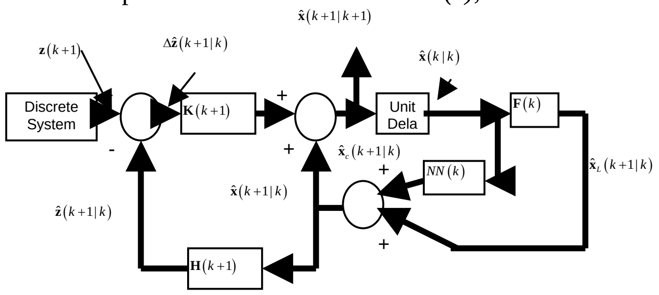
and finally

$$x_{k+1} = A' x_k + \left(\frac{\partial NN}{\partial w_k} \right) w_k \tag{2.8}$$

where

$$A' = A + \frac{\partial NN}{\partial x_k} \tag{2.9}$$

Equations (2.5-2.6) show that the neural network modifies the predicted system state x_{k+1} through the Jacobian of the system transition matrix *A'*. The inputs to the neural network are the updated states of the filter as shown in Fig. 1. The outputs of the neural network *NN(k)*, are the



$\hat{\mathbf{x}}(k+1|k+1)$
$\mathbf{z}(k+1)$
$\Delta\hat{\mathbf{z}}(k+1|k)$
$\hat{\mathbf{x}}(k|k)$
Discrete System
$\mathbf{K}(k+1)$
Unit Dela
$\mathbf{F}(k)$
+
-
+
$\hat{\mathbf{x}}_c(k+1|k)$
+
$NN(k)$
$\hat{\mathbf{x}}_L(k+1|k)$
$\hat{\mathbf{x}}(k+1|k)$
$\hat{\mathbf{z}}(k+1|k)$
$\mathbf{H}(k+1)$
+

Fig. 1. NEKF Block Diagram

corrections to the linear predicted state. The inputs are passed through an input layer, a hidden layer with nonlinear squashing functions, and an output layer as shown in Fig. 2. The outputs of the neural network are nonlinear corrections to the linear predicted state of the underlying Kalman filter.

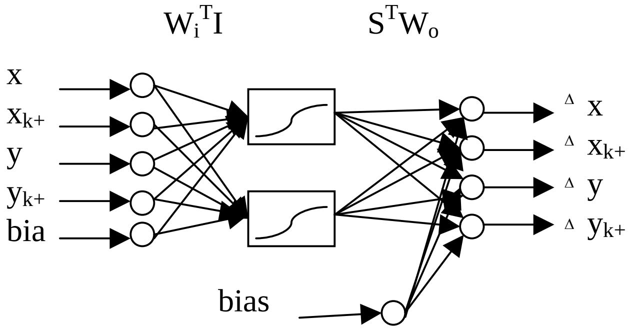


Fig. 2. NEKF Inputs and Outputs

III. NEURAL EXTENDED KALMAN FILTER INTERACTING MULTIPLE MODEL TRACKING ALGORITHM

A new tracking algorithm called the neural extended Kalman filter interacting multiple model (NEKF IMM) algorithm is now discussed from [12]. Combining the NEKF algorithm with the IMM algorithm the authors were able to design a very robust estimator. The NEKF IMM uses 3 models. Two of the models are constant velocity models with a low and high process noise, respectively, and the third model is the NEKF. The algorithm combines the benefits of the IMM soft switching capability between models and the on-line maneuver learning capability of the NEKF. The IMM architecture allows for Kalman filter models of different state dimensions to be mixed together appropriately. What is different and novel in this IMM architecture is that the neural network weights that are not dependent on the dynamic equations are mixed in with the other dynamic models. The state vector mixing equations of the NEKF in a 2 model NEKF IMM architecture are

$$x_{mix} = \begin{bmatrix} x_1 * \mu_{21} + x_2 * \mu_{22} \\ w_k * \mu_{22} \end{bmatrix} \tag{2.10}$$

where *x1* is the system state vector for model 1, *x2* is the system state vector for model 2, *w* is the neural network weight state vector, and μ is the mixing mode probability weight. Equation (2.10) shows that the neural network weight vector is weighted by the mixing mode probability. This is a key point to the architecture's stability. For the covariance mixing

$$P_k mix = \left[\begin{array}{c:c} mix(P_1, P_2) & P_{k|k_j} * \mu_{22} \\ \hdashline P_{k|k_j}^{T} * \mu_{22} & P_{k|k_w} * \mu_{22} \end{array}\right] \tag{10}$$

the upper block covariance mixing is the same as with other IMM dynamic systems, the off diagonal blocks and lower block matrices are due to the neural network weights and are weighted appropriately by the NEKF mixing mode probability μ. With these two modifications to the mixing process of the IMM architecture to accommodate the NEKF neural network weight vector and covariance matrix, the rest of the IMM algorithm is the same.

IV. BENCHMARK TRACKING RESULTS

A set of preliminary results of using the NEKF IMM algorithm on the Benchmark II Problem [3] was published in August, 2003 [12]. These results are only for the 6 targets including false alarms. The metric print out taken from the MATLAB software has been put into Table 1 below. The NEKF IMM algorithm was used to generate the six results along with a heuristic algorithm to pick the waveform type to use at each radar look. The waveform heuristic attempted to keep the SNR returned by the radar above a specified threshold. If the SNR was high above the threshold, i.e. greater than 3dB, the waveform number would be reduced by 1. If the SNR dropped below 6dB above threshold, the waveform number was increased by 1. The waveform heuristic caused the radar to choose waveforms with a long integration time for targets at a longer distance away. The waveform algorithm also chose a short integration time waveform for targets at a short distance away from the radar. Bar-Shalom's, et al, adaptive revisit algorithm was used from [11] to choose the next dwell time. The threshold was set to 8 dB and raised to 12 dB when a reacquisition of the target using a search dwell was required. After the search dwell call, the track dwell mode was reinitiated and the threshold was slowly lowered again to 8 dB. Table 1 shows the results of the NEKF IMM for the Benchmark II scenarios. The control algorithms were designed to minimize the power used by the radar and to maximize the sampling rate of the radar. During all six simulations, the NEKF algorithm was active during target maneuvers. The constant velocity motion model with a low process noise was active during straight line motion. Finally, the constant velocity motion model with a high process noise was active during the quick onset and ending of maneuvers.

TABLE 1. Results for 100 Monte-Carlo Runs.

Tgt #'s	% Lost Tgts	Samp Time (Secs)	Avg # of Samp	Pos RMSE Meter	Speed RMSE Meters/ Second	Dwell Time per Run Millisecs
1	3	2.08	79	110	42	93.3
2	0	1.49	101	84	44	114.3
3	3	1.67	87	110	54	99.5
4	0	1.31	142	36	24	155.1
5	3	1.71	106	142	70	121.0
6	5	1.59	118	86	73	131.2

In Table 1, only Target 6 failed the 4% loss of target metric. The sampling time varied between 1.3 to 2.1 seconds per radar revisit across the 6 targets. The average number of radar revisits varied between 79 and 142 visits. The position RMSE was between 36 and 142 meters. The speed RMSE varied between 24 and 73 meters per second. The dwell time per run was between 93 and 155 milliseconds. The following are more results not shown or discussed in [12].

Fig. 3 shows the trajectory for target 1 in the XY plane. Overlayed on the plot are the 100 Monte-Carlo run estimates for the XY trajectory. In this particular scenario the target stayed at a constant altitude. Since the sampling rate was variable during each Monte-Carlo run some samples were only averaged once.

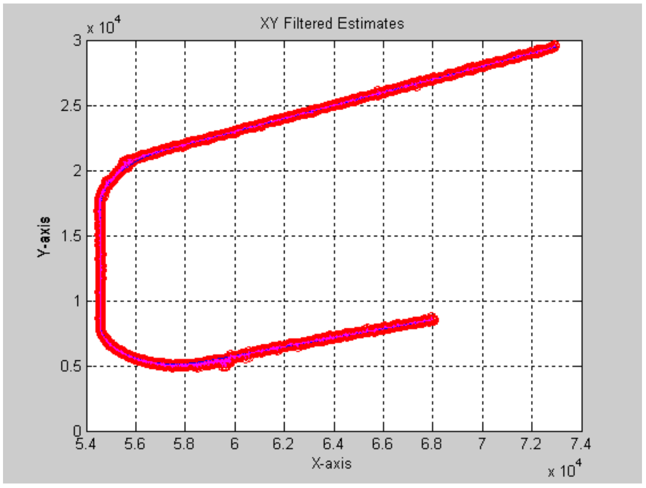


Fig. 3. Target 1 Trajectory

Fig. 4 shows the root mean squared error (RMSE) for the position of target 1 in meters. The RMSE was taken over 100 Monte-Carlo runs. The average position RMSE taken from Table 1 for target 1 is 110 meters. The peak position error during the most severe maneuver was approximately 610 meters around 135 seconds.

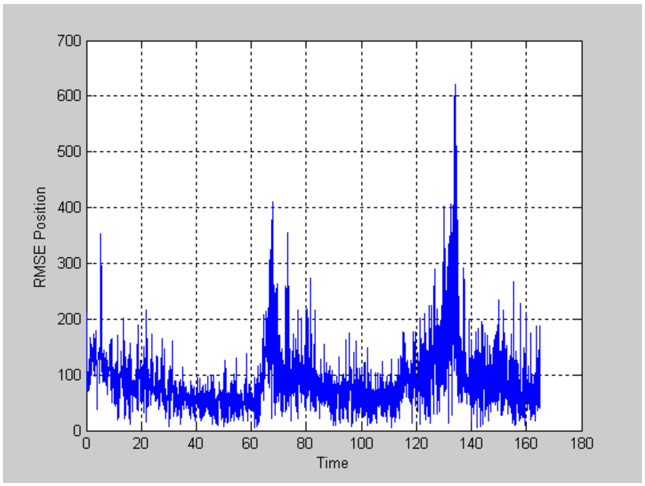


Fig. 4. Target 1 Position RMSE

Fig. 5 shows the RMSE for the velocity of target 1 in meters per second. The RMSE was taken over 100 Monte-Carlo runs as mentioned before. The average RMSE taken from Table 1 for the velocity of target 1 is 42 meters per second. The peak velocity error during the most severe maneuver was approximately 130 meters per second. A peak point at approximately 135 seconds of 180 meters per second was due to a large noise deviation in the Monte-Carlo run and a single point for averaging.

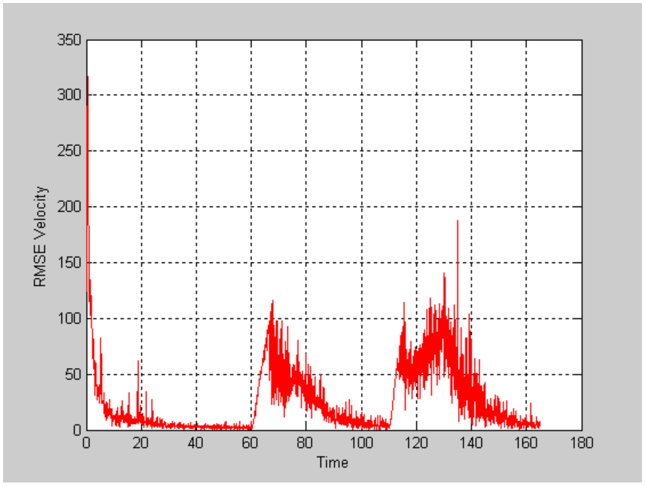


Fig. 5. Target 1 Velocity RMSE

Fig. 6 shows the mode probabilities and the neural network output corrections over time. During the maneuvers the neural network mode probability was approximately 90% or more. During the onset and end of maneuvers the high process noise mode was in effect. During straight line motion the low process noise model was in effect. This figure shows the typical performance of the NEKF IMM across all 6 target scenarios for the Benchmark II. For the results shown in this paper only the velocity states were corrected by the neural network during maneuvers.

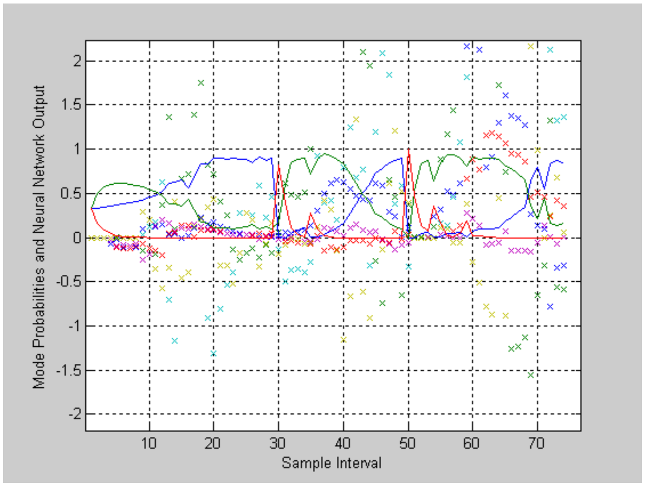


Fig. 6. Target 1 NEKF Outputs and Mode Probabilities

Fig. 7 shows the 3 dimensional target trajectory for target number 2. The three axes in X, Y, and Z are all in meters. The target begins at 4500 meters in altitude and descends to 3000 meters as it moves in towards the sensor located at the origin. There are two 90 degree turns during this scenario in the XY plane. This scenario's target is the closest to the phased array radar sensor. It has the largest SNR returns for the radar, and therefore, can utilize the shortest integration time waveforms for the radar. These waveforms have the most accurate range estimates for the radar.

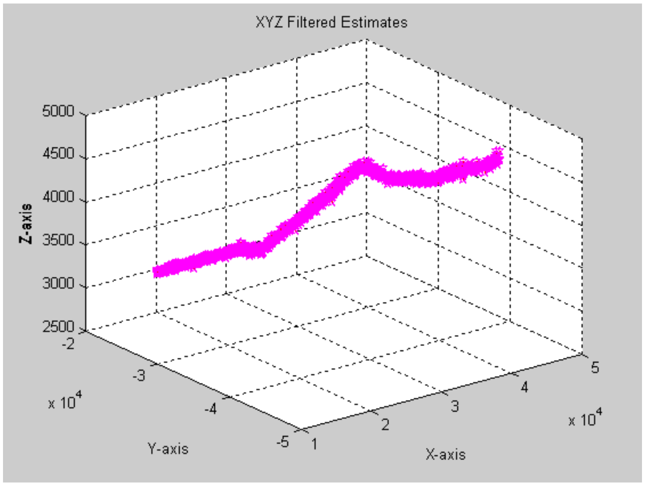


Fig. 7. Target 2 Trajectory

Figs. 8 and 9 show the 2 dimensional plots for the XY and Z trajectories, respectively. Overlayed in the figures are the 100 Monte-Carlo run estimates in both the XY and Z planes, respectively. The two turns in the scenario occur before and after the climbing maneuver, respectively.

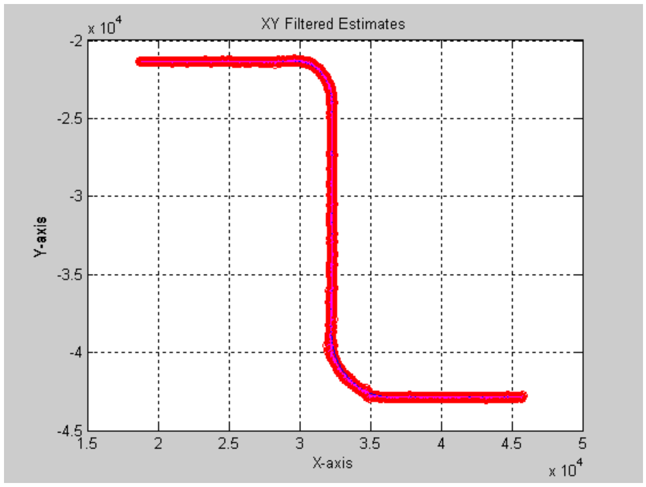


Fig. 8. Target 2 XY Trajectory

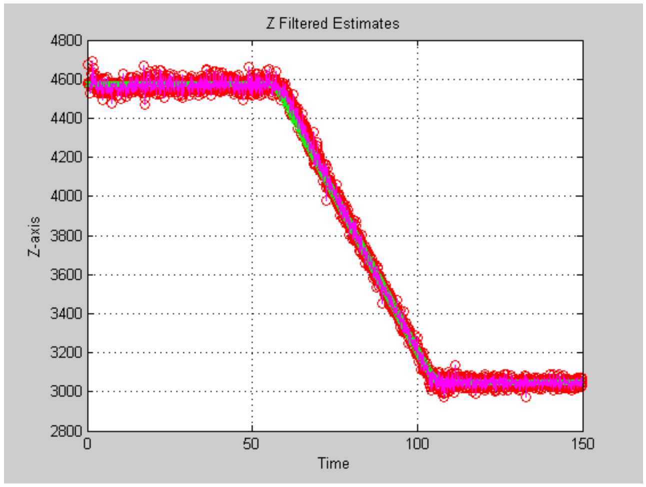


Fig. 9. Target 2 Z Trajectory

Fig. 10 shows the root mean squared error (RMSE) for the position of target 2 in meters. The average position RMSE taken from Table 1 for target 2 is 84 meters. The peak position error during the most severe maneuver was approximately 375 meters.

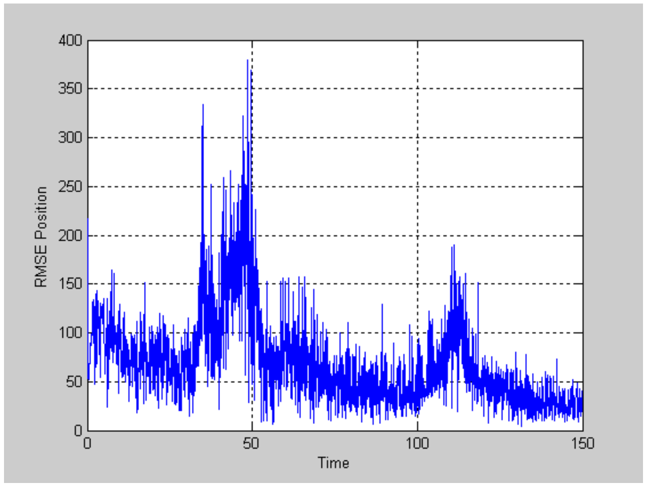


Fig. 10. Target 2 Position RMSE

Fig. 11 shows the RMSE for the velocity of target 2 in meters per second. The average RMSE taken from Table 1 for the velocity of target 2 is 44 meters per second. The peak velocity error during the most severe maneuver was approximately 175 meters per second. An outlier peak point was approximately 275 meters per second. This point was due to lack of averaging in the Monte-Carlo runs.

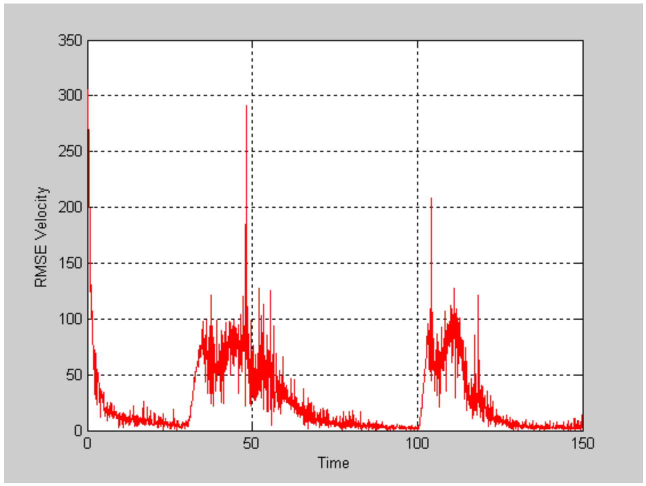


Fig. 11. Target 2 Velocity RMSE

Fig. 12 shows the trajectory for target 3 in the XY plane. Overlayed on the plot are the 100 Monte-Carlo run estimates for the XY trajectory. In this particular scenario the target stayed at a constant altitude. Since the sampling rate was variable during each Monte-Carlo run some samples were only averaged once.

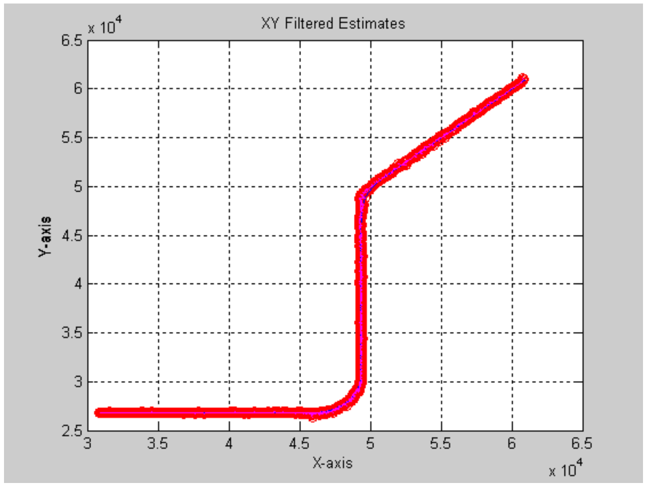


Fig. 12. Target 3 Trajectory

Fig. 13 shows the root mean squared error (RMSE) for the position of target 3 in meters. The average position RMSE taken from Table 1 for target 3 is 110 meters. The peak position error during the most severe maneuver was approximately 500 meters.

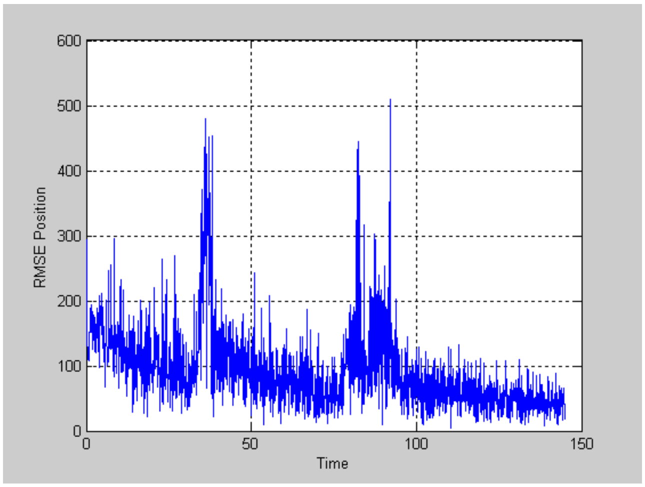


Fig. 13. Target 3 Position RMSE

Fig. 14 shows the RMSE for the velocity of target 3 in meters per second. The average RMSE taken from Table 1 for the velocity of target 3 is 54 meters per second. The peak velocity error during the most severe maneuver was approximately 210 meters per second.

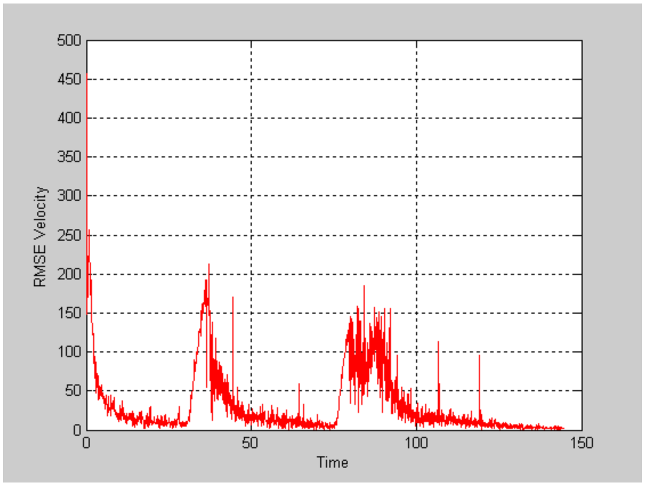


Fig. 14. Target 3 Velocity RMSE

Fig. 15 shows the 3 dimensional target trajectory for target number 4. The target begins at 2300 meters in altitude as it moves towards the sensor located at the origin and then climbs to 4500 meters as it moves away from the sensor. The climb takes 50 seconds to complete at a rate of 45 meters per second in the Z domain. There are two turns during this scenario in the XY plane, a very slow 90 degree turn and then a very quick 90 degree turn right at the end of the first 90 degree turn. Figs. 16 and 17 show the 2 dimensional plots for the XY and Z trajectories, respectively. Notice in Fig. 16 the very gradual first turn followed by the severe turn. After the turns the target moves into the steep climb shown in Fig. 17. Overlayed on the figures are the 100 Monte-Carlo run estimates in both the XY and Z planes.

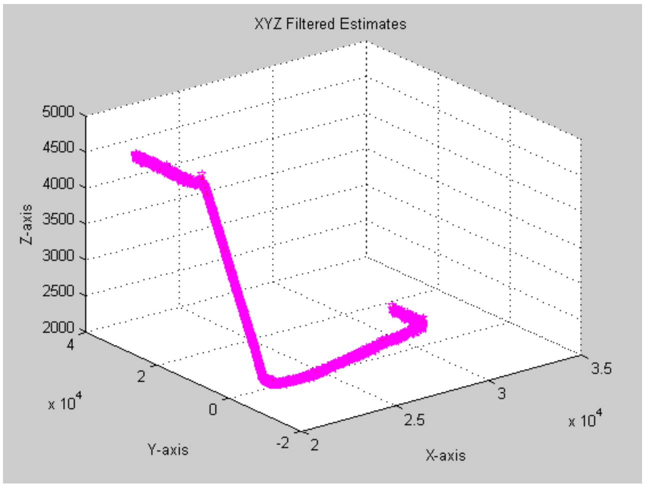


Fig. 15. Target 4 Trajectory

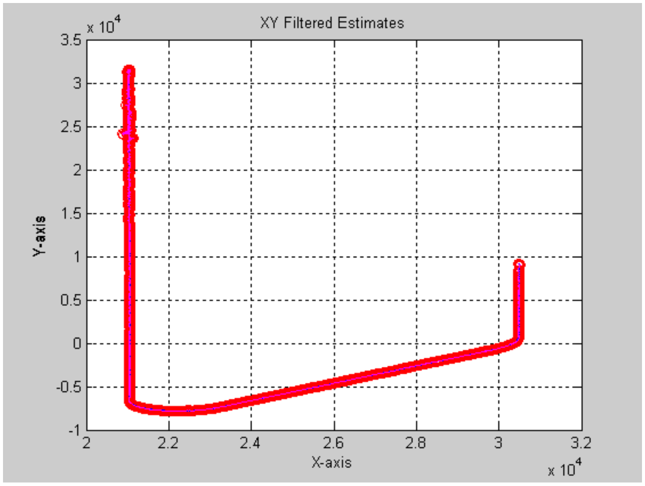


Fig. 16. Target 4 XY Trajectory

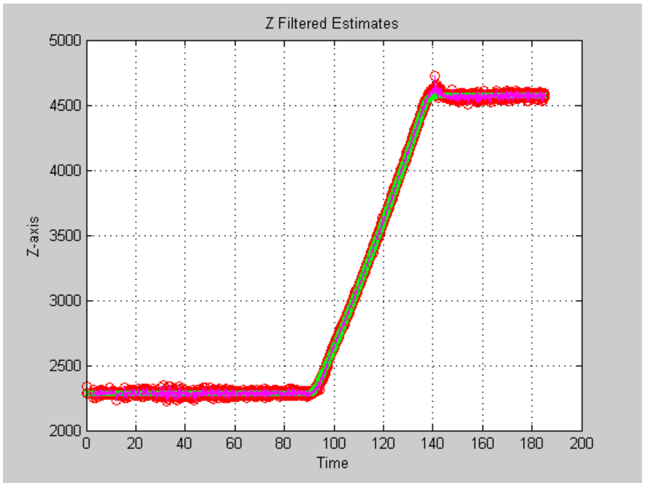


Fig. 17. Target 4 Z Trajectory

Fig. 18 shows the root mean squared error (RMSE) for the position of target 4 in meters. The average position RMSE taken from Table 1 for target 4 is 36 meters. The peak position error during the most severe maneuver was approximately 150 meters. An outlier approximately equal to 400 meters per second is shown in the plot. This needs to be investigated why it occurred on a straight track.

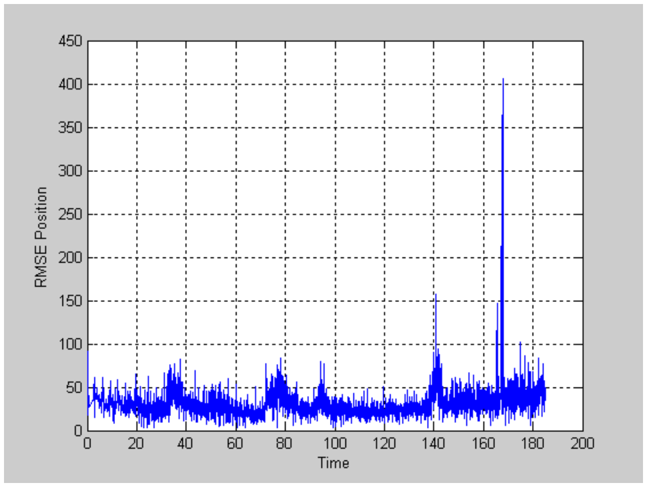


Fig. 18. Target 4 Position RMSE

Fig. 19 shows the RMSE for the velocity of target 4 in meters per second. The average RMSE taken from Table 1 for the velocity of target 4 is 24 meters per second. The peak velocity error during the most severe maneuver was approximately 90 meters per second. There is an outlier near the end of the scenario at 210 meters per second. This needs to be investigated why it occurred on a straight track.

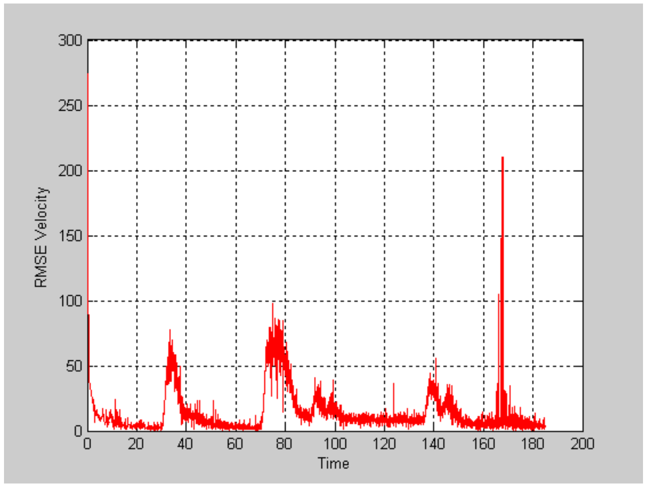


Fig. 19. Target 4 Velocity RMSE

Fig. 20 shows the 3 dimensional target trajectory for target number 5. The target begins at 1500 meters in altitude as it moves towards the sensor located at the origin and then climbs to 4500 meters as it moves away from the sensor. There are three turns during this scenario in the XY plane, one 45 degree turn and two 90 degree turns. Fig. 21 and 22 show the 2 dimensional plots for the XY and Z trajectories, respectively. Overlayed on the figures are the 100 Monte-Carlo run estimates in both the XY and Z planes. This particular scenario was the most distant target to track from the sensor across all six targets. In order to keep a high SNR the largest integration time waveforms were utilized to produce these results.

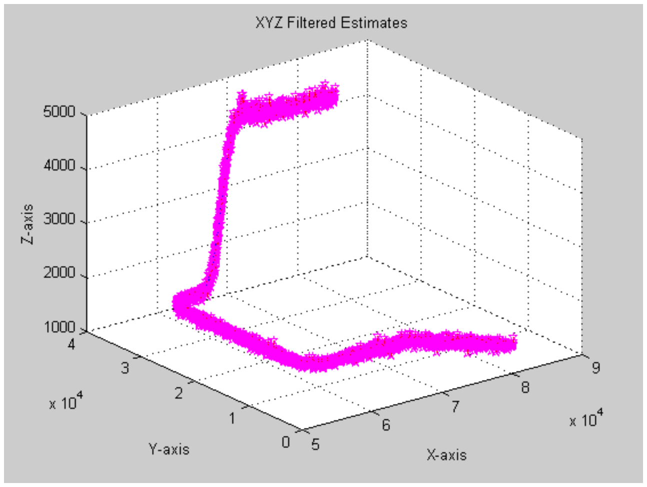


Fig. 20. Target 5 Trajectory

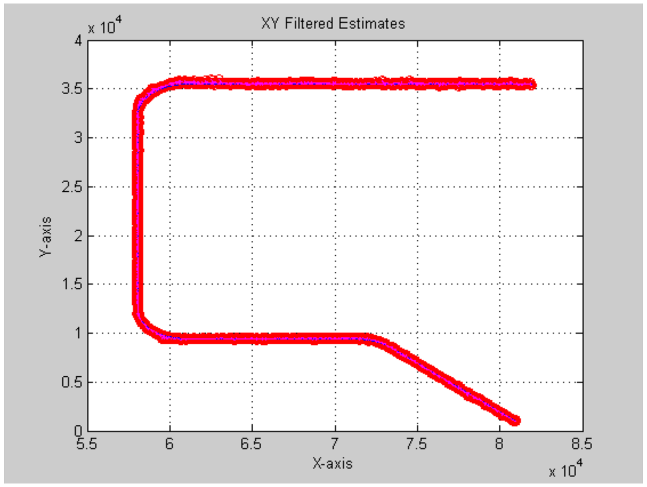


Fig. 21. Target 5 XY Trajectory

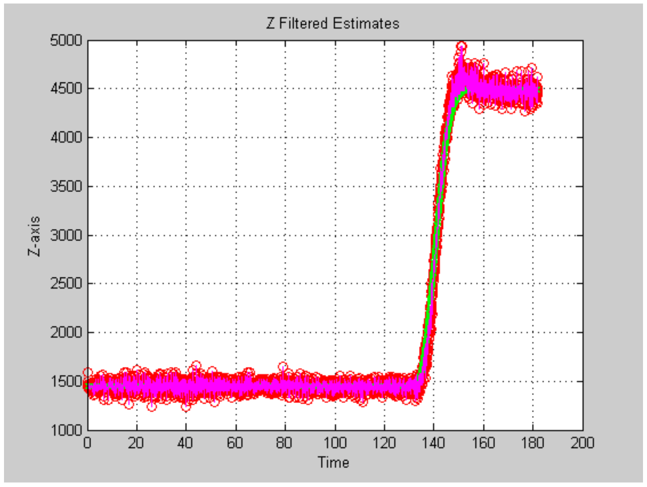


Fig. 22. Target 5 Z Trajectory

Fig. 23 shows the root mean squared error (RMSE) for the position of target 5 in meters. The average position RMSE taken from Table 1 for target 5 is 142 meters. The peak position error during the most severe maneuver was approximately 650 meters.

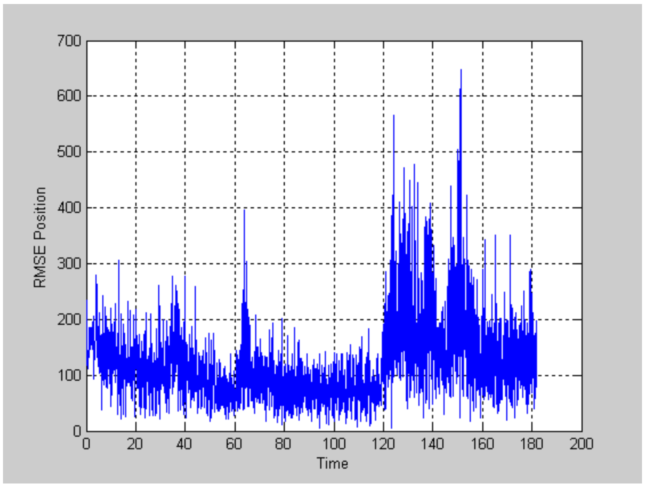


Fig. 23. Target 5 Position RMSE

Fig. 24 shows the RMSE for the velocity of target 5 in meters per second. The average RMSE taken from Table 1 for the velocity of target 5 is 70 meters per second. The peak velocity error during the most severe maneuver was approximately 275 meters per second. There is an outlier at 20 seconds due to lack of averaging.

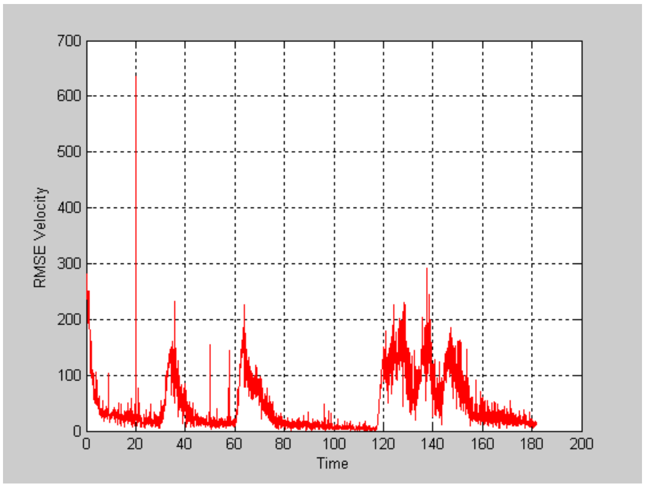


Fig. 24. Target 5 Velocity RMSE

Fig. 25 shows the 3 dimensional target trajectory for target number 6. The target begins at 1550 meters in altitude and descends to 800 meters as it moves towards the sensor located at the origin and then moves away from it. There are four turns during this scenario in the XY plane, two 90 degree turns, a 135 degree turn, and finally a 45 degree turn. This scenario is the most stressing with the target executing up to 7g-turns in the horizontal and vertical plane. Figs. 26 and 27 show the 2 dimensional plots for the XY and Z trajectories, respectively. During the second 90 degree turn the target pitches downward and pulls a 7g dive while executing the turn. In Fig. 27 it is shown how quickly the descent is executed on the order of seconds. Overlayed on the figures are the 100 Monte-Carlo run estimates in both the XY and Z planes.

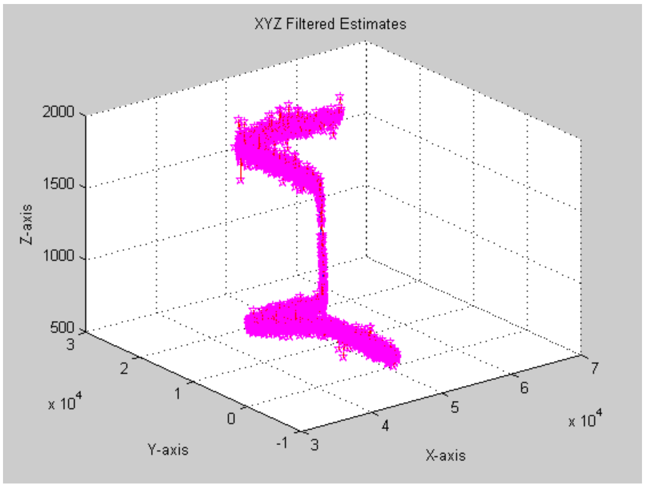


Fig. 25. Target 6 Trajectory

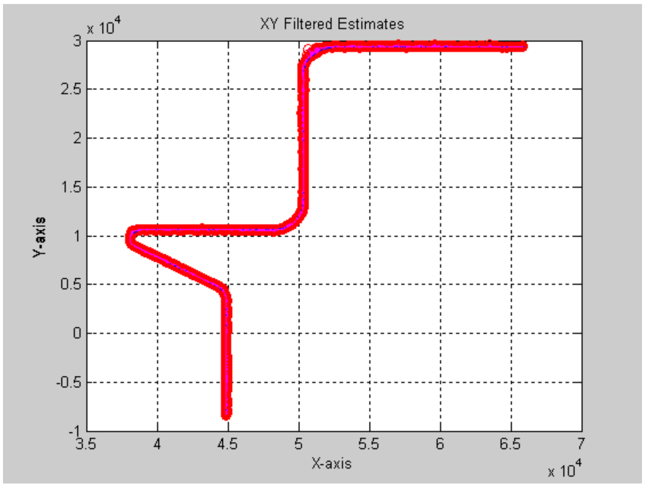


Fig. 26. Target 6 XY Trajectory

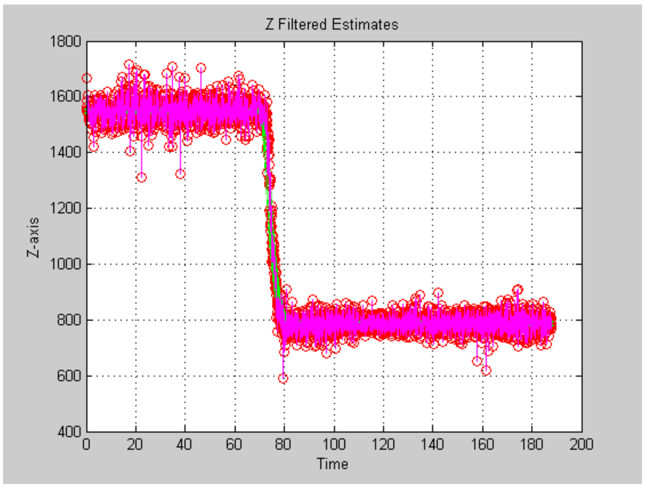


Fig. 27. Target 6 Z Trajectory

Fig. 28 shows the root mean squared error (RMSE) for the position of target 6 in meters. The average position RMSE taken from Table 1 for target 6 is 86 meters. The peak position error during the most severe maneuver was approximately 420 meters. There is an outlier of 720 meters due to lack of averaging.

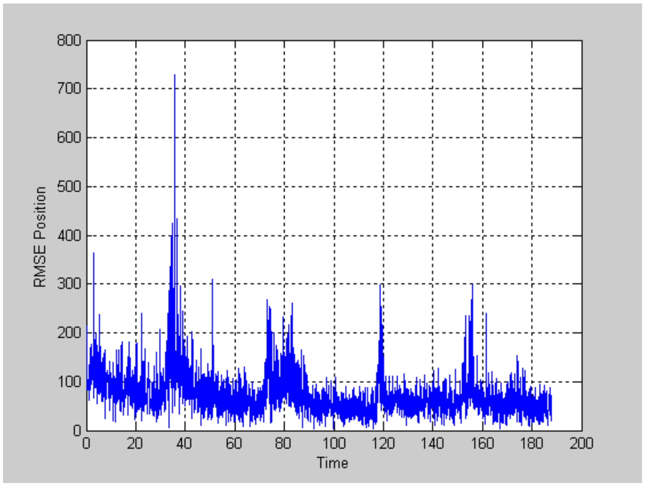


Fig. 28. Target 6 Position RMSE

Fig. 29 shows the RMSE for the velocity of target 6 in meters per second. The average RMSE taken from Table 1 for the velocity of target 6 is 73 meters per second. The peak velocity error during the most severe maneuver was approximately 300 meters per second. There is an outlier equal to 600 meters per second due to lack of averaging.

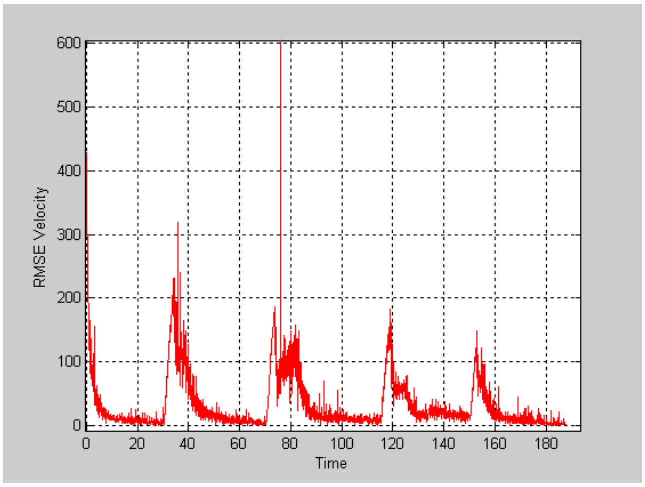


Fig. 29. Target 6 Velocity RMSE

V. CONCLUSIONS

In this paper, we discussed the use of a neural extended Kalman filter embedded in an IMM architecture for air target tracking problem. The NEKF uses a neural network to adapt on-line to unmodeled dynamics or nonlinearities in the target trajectory. This on-line adaptation provides for a robust state estimation for tracking applications because the maneuvers do not have to be known beforehand. The NEKF is a generic state estimator that can be used to estimate any state vector such as position, velocity, magnetic moment, frequency signatures, etc... A set of preliminary results on the Benchmark II from 1996 were presented in a tabular form. Also, plots of RMSE errors and Monte-Carlo estimates were shown to demonstrate the NEKF IMM tracking capability.

V. REFERENCES

[1] A Stubberud,., H. Wabgaonkar. 1990 “Approximation and Estimation Techniques for Neural Networks,” *Proceedings of the 28th Conference on Decision and Control*, (December), Honolulu, Hawaii, pp. 2736-2740.

[2] R.N. Lobbia, S.C. Stubberud, and M.W. Owen, “Adaptive Extended Kalman Filter Using Artificial Neural Networks,” The International Journal of Smart Engineering System Design, Vol. 1, pp. 207-221, 1998.

[3] W. Blair and G. Watson, “Benchmark II Problem for Radar Resource Allocation and Tracking Maneuvering Targets in the Presence of ECM”, NSWCDD Technical Report – 96, September 1996.

[4] S. Blackman, Multiple-Target Tracking with Radar Applications, Artech House, 1986.

[5] S. Blackman and R. Popoli, Design and Analysis of Modern Tracking Systems, Artech House, 1999.

[6] A. Gelb, Applied Optimal Estimation, M.I.T. Press, 1974.

[7] M. Santina, A. Stubberud, and G. Hostetter, Digital Control System Design, Saunders College Publishing, 1994.

[8] Y. Bar-Shalom and X. Li, Estimation and Tracking: Principles, Techniques, and Software, Artech House, 1993.

[9] R. Hecht-Nielsen, Neurocomputing, Addison-Wesley Publishing Company, 1990.

[10] S. Singhal and L. Wu, “Training Multilayer Perceptrons with the Extended Kalman Algorithm,” Advances in Neural Information Processing System I, D.S. Touretzky (ed.) Morgan Kaufmann, 1989, pages 133-140.

[11] E. Daeipour, Y. Bar-Shalom, and X. Li. 1994 “Adaptive Beam Pointing Control of a Phased Array Radar Using an IMM Estimator,” *Proceedings of the American Control Conference*, (June), Baltimore, Maryland, pp. 2093-2097.

[12] M. Owen and A. Stubberud, “NEKF IMM Tracking Algorithm,” *Proceedings of SPIE: Signal and Data Processing of Small Targets 2003*, volume 5024, Oliver Drummond, editor, San Diego, California, August, 2003.