On State Fusers Over Long-Haul Sensor Networks

Katharine Brigham
and B. V. K. Vijaya Kumar
Carnegie Mellon University
Pittsburgh, PA 15213
Email: {kbrigham,kumar}@ece.cmu.edu

Nageswara S. V. Rao
Oak Ridge National Laboratory
Oak Ridge, TN 37831
Email: raons@ornl.gov

Qiang Liu
and Xin Wang
Stony Brook University
Stony Brook, NY 11794
Email: {qiangliu,xwang}@ece.sunysb.edu

***Abstract*—We consider a network of sensors wherein the state estimates are sent from sensors to a fusion center to generate a global state estimate. The underlying fusion algorithm affects the performance measure $Q_{CC}(\tau)$ (with subscripts CC indicating the effects of the communications and computing quality) of the global state estimate computed within the allocated time τ. We present a probabilistic performance bound on $Q_{CC}(\tau)$ as a function of the distributions of state estimates, communications parameters as well as the fusion algorithm. We present simulations of simplified scenarios to illustrate the qualitative effects of different fusers, and system-level simulations to complement the analytical results.**

I. Introduction

We consider a long-haul network of sensors which collect information on targets and generate state estimates, such as position and velocity. The sensors periodically communicate state estimates to a remote fusion center, which combines them to generate global state estimates that correspond to targets. However, due to the long-haul nature of the network, messages from the sensors may suffer from random delays and losses, which could negatively impact the accuracy of the fused result. Even with perfect communications, a major issue encountered with fusing sensor tracks is dealing with correlations of state estimate errors across the sensors. In particular, the cross-covariance, which is a measure of such correlations, may be nonzero due to a variety of factors such as common process noise, and correlated measurement noise across sensors. The cross-covariance is a key component in several fusion strategies such as the best linear unbiased estimation (BLUE) and optimal weighted least squares (WLS) fusion rules [1], but explicit estimation of the cross-covariance can be quite involved [2]. There are generally two approaches to the fusion of state estimates; one approach attempts to fuse estimates with unavailable cross-covariance, and the other approach requires knowledge of the cross-covariance [3]. In [1], Li and Zhang provide an extensive list of mathematical expressions for the estimation of the cross-covariance for linear and nonlinear observations. In the case of unavailable cross-covariance, one may try to estimate the cross-covariance or fuse the data without it. Several methods for doing so are described in [4] (e.g., estimation by time averaging, fusion using pseudo-measurements, etc.). Another method for fusing data when the cross-covariance is unavailable is the Covariance Intersection (CI) algorithm, which will be described in more detail in Section III-B.

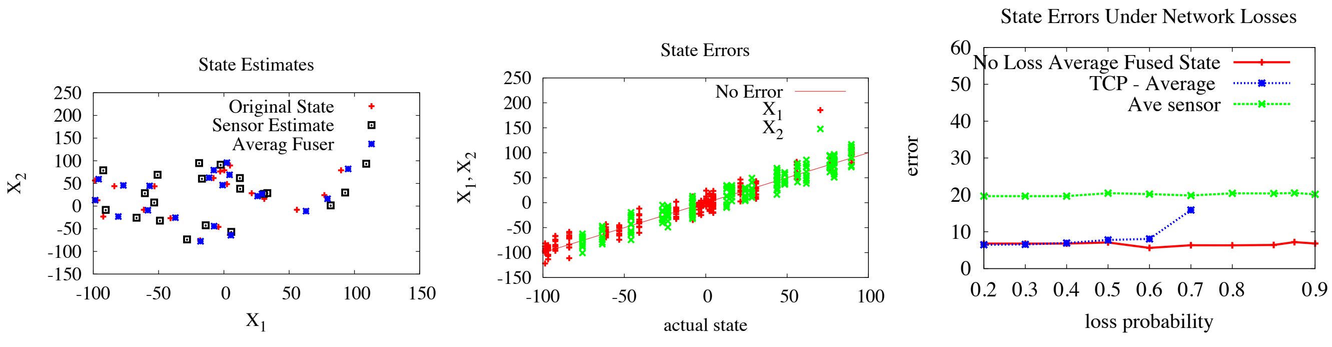
Learning approaches may bypass the need to compute the cross-covariance by implicitly incorporating the correlations into the fuser function that is estimated from measurements. It is noted that the cross-covariance could actually be computed off-line if the underlying dynamic system were linear and time-invariant [5]; however, in many target tracking applications, the underlying system is often nonlinear. In this work, we explore learning-based fusers for nonlinear systems; in particular, we investigate the use of artificial neural networks (ANNs) for multisensor fusion. ANNs possess the capability of modeling arbitrary mappings [6], as long as a sufficient number of training samples are available from the same distribution. This will also provide us with the ability to use nonlinear functions for fusing the data, which may potentially yield better results than with linear fusion.

ANNs have been previously been proposed for target tracking applications, e.g., for improving data association [7], filtering [8]–[10], and measurement fusion [11], to name a few. Chowdhury [12] and Fong et. al. [13] propose using ANNs for sensor fusion, where the neural networks are used to determine the weights for linearly combining sensor state estimates. We further explore ANNs for nonlinear sensor fusion.

The remaining sections are organized as follows. In Section II, we provide an analytical formulation of the fusion problem. In Section III, we will briefly describe the basic formulations of some traditional fusers and how a neural network can be employed for nonlinear fusion. In Section IV, we present an example of a nonlinear target-tracking system with state-dependent noise, and quantitatively compare the performance of learning-based fusers against the more conventional methods of track fusion. We then follow with concluding remarks.

II. Problem Formulation

Let $\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_M$ be the state estimates generated by the sensors and sent to the fusion center. Of them, $\hat{X}^W_{i_1}, \ldots \hat{X}^W_{i_k}$ arrive within a time-window $[T, T+W]$ at the fusion center, and used as input to a fusion algorithm, which generates a single global estimate to correspond to the underlying target. For allocated time τ to the fusion algorithm, which runs either after all expected state estimates are received or when the time window expires, let $Q_C\left(\hat{X}^W_{i_1}, \ldots \hat{X}^W_{i_k}; \tau\right)$ denote the performance measure, of the output, normalized to interval $[0, 1]$, where 0 represents no error and 1 represents the highest error. The expected quality of k received state estimates is



(a) original and estimated states (b) state error distributions (c) state errors

Fig. 1. Average fuser under unbiased independent errors and network losses.

given by

$$\bar{Q}_C(k,\tau) = \int Q_C\left(\hat{X}^W_{i_1},\dots \hat{X}^W_{i_k};\tau\right) dP_{\hat{X}^W_{i_1},\dots \hat{X}^W_{i_k}}. \tag{1}$$

By combining the computing and communications parts, the quality of global estimate generated by the network in response to the state estimates $\hat{X}_1, \hat{X}_2, \dots, \hat{X}_M$ is $Q_{CC}\left(\hat{X}_1,\dots \hat{X}_M;\tau\right)$. We are interested in the probability of ensuring quality δ of the final estimate, given by

$$\mathbf{P}\left\{Q_{CC}\left(\hat{X}_1,\dots \hat{X}_M;\tau\right) < \delta\right\}. \tag{2}$$

We decompose this quantity by conditioning on the state estimates as follows

$$\mathbf{P}\left\{\mathbf{Q_{CC}}\left(\hat{\mathbf{X}}_\mathbf{1},\dots \hat{\mathbf{X}}_\mathbf{M};\tau\right) < \delta\right\} \tag{3}$$
$$= \int \mathbf{P}\left\{Q_{CC}\left(\hat{X}_1,\dots \hat{X}_M;\tau\right) < \delta | \hat{X}_1,\dots \hat{X}_M\right\} dP_{\hat{X}_1,\dots \hat{X}_M}.$$

The following lower bound is derived in [14]:

$$\begin{aligned} &\mathbf{P}\left\{Q_{CC}\left(\hat{X}_1,\dots \hat{X}_M;\tau\right) < \delta | \hat{X}_1,\dots \hat{X}_M\right\} \\ &\geq \left(1-\frac{\bar{Q}_C(k;\tau)}{\delta}\right)\mathbf{P}\left\{t_{\hat{X}^W_{i_1}},\dots t_{\hat{X}^W_{i_k}} \in [T, T+W]\right\}, \end{aligned} \tag{4}$$

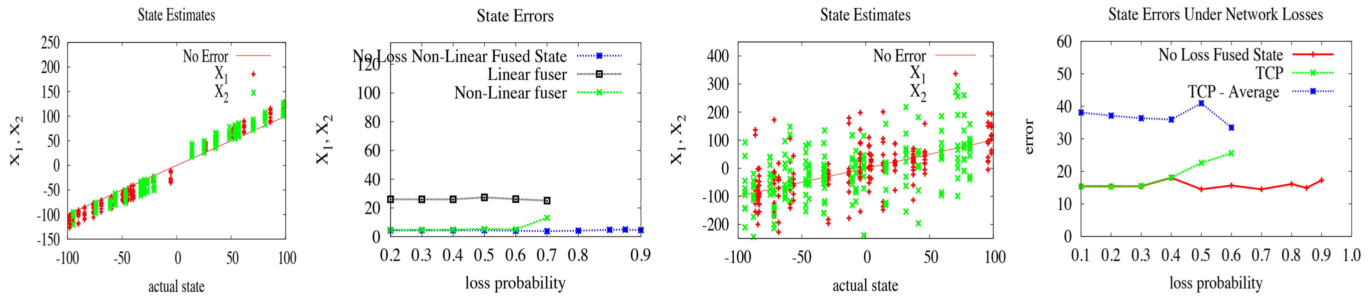
where $t_{\hat{X}^W_{i_j}}$, $j = 1,2,\dots,k$, denotes the time at which the state estimate $\hat{X}^W_{i_j}$ arrives at the fusion center. This expression demonstrates the contributions of randomness due to: (a) state estimates reflected in $\bar{Q}_C(k;\tau)$ and, (b) communications network parameters reflected in the term $\mathbf{P}\left\{t_{\hat{X}^W_{i_1}},\dots t_{\hat{X}^W_{i_k}} \in [T, T+W]\right\}$. This decomposition shows the separation between the computation and communications parts, which can be analyzed somewhat independently. In this paper, we focus on the first term, which depends on the distribution of the state estimates and the fusion algorithm. The second term does not depend on the state estimates but depends on the properties of the network, such as latency and loss rate.

We consider simplified simulations of a single 3D target to illustrate the effects of communications and computations on the fused state estimate. Here, the states are generated uniformly within $[-A, A]^3$ area as shown in Fig. 1(a) for two of the three coordinates. The communication losses are simulated by using TCP message delivery rates computed based on connection loss probabilities. In this example, round-trip time (RTT) is 1 second corresponding to about 10,000 mile connection, and the computation time-window τ is 10 seconds.

(a) *Average Fuser - Unbiased independent errors:* The sensor errors have zero mean and are statistically independent in the range $[-B, B]$ and are shown in Fig. 1(b). For this case, the fuser averages the state estimates that arrive within the time-window, and provides a substantial improvement in the state error as shown in Fig. 1(c); the Euclidean distance error of the fused estimate is around 8 compared to the average sensor error around 20. Thus, the fusion of sensor estimates is a good choice in this case if there are no network losses. As the communications loss probability is increased, TCP losses lead to the degradation of state estimate as shown in Fig. 1(c). When the loss rate exceeds 0.7 no messages are received within the window at the fusion center in the 20 instances we simulated.

(b) *Nonlinear Fuser - Biased errors:* To illustrate the effects of the fuser, we consider that the sensor errors have a bias as shown in Fig. 2(a) where negative state values have a negative bias and positive state values have a positive bias. In this case, the average fuser is no longer as effective, but a nonlinear fuser that applies a correction based on the sign of coordinate and then computes an average, leads to a much improved state estimate as shown in Fig. 2 (b). This fuser is more complex and better performing than above; nevertheless, its performance also degrades with the connection loss probability in a qualitatively similar manner.

(c) *Linear Fuser - Unbiased errors with different variances:* Then we consider a case where sensor errors have zero bias but have different variances as shown in Fig. 2(c). In this case a linear fuser with coefficients inversely proportional to the sensor error covariances performs better than the average fuser as illustrated in Fig. 2(d). And, the effects of communications losses are quite



(a) profiles of biased sensor errors (b) state errors - nonlinear fuser (c) profiles of sensor errors - different variances (d) state errors - linear fuser

Fig. 2. Linear and nonlinear fusers under network losses.

similar to previous cases.

As illustrated in these cases, the fusion method determines the quality of estimate and execution time. However, these fusers are not applicable in practical scenarios since errors are correlated and the detailed knowledge needed to implement the above fuser is very rarely available. We consider general classes of such fusers in the next section.

III. State Estimate Fusion Algorithms

In this work, we will assume that the cross-covariance is unavailable. We consider two linear fusers for comparison with nonlinear and linear learning-based fusers: one that assumes the sensor errors are uncorrelated, and one that does not. Note that the formulas provided here are for fusing two state estimates, which are given as an example; however, these fusion algorithms are not limited to only fusing two state estimates.

A. *Linear Fuser with Cross Covariance*

This linear fuser is optimal in the linear minimum mean-square error (LMMSE) sense. The fused state estimate $\hat{X}_F$ and its error covariance P_F are defined for two sensors [2] as:

$$\hat{X}_F = \hat{X}_1 + (P_1 - P_{12})(P_1 + P_2 - P_{12} - P_{21})^{-1}(\hat{X}_2 - \hat{X}_1) \quad (5)$$

$$P_F = P_1 - (P_1 - P_{12})(P_1 + P_2 - P_{12} - P_{21})^{-1}(P_1 - P_{21}) \quad (6)$$

where $\hat{X}_i$ and P_i are the state estimates and error covariance from sensor i, respectively, and $P_{ij} = P_{ji}^T$ is the error cross-covariance between sensors i and j. If the sensor errors are uncorrelated, this fuser reduces to a simple convex combination of the state estimates:

$$P_F = (P_1^{-1} + P_2^{-1})^{-1} \quad (7)$$

$$\hat{X}_F = P_F(P_1^{-1}\hat{X}_1 + P_2^{-1}\hat{X}_2) \quad (8)$$

However, if the sensor errors are correlated but the cross-covariance is unavailable, one may assume that the cross-covariance is zero in order to apply this linear fuser, but the result will be suboptimal.

B. *Covariance Intersection (CI) Algorithm*

Another sensor fusion method is the covariance intersection (CI) algorithm. The intuition behind this approach comes from a geometric interpretation of the problem. If one were to plot the covariance ellipses for P_F (defined as the locus of points $\{\mathbf{y} : \mathbf{y}^T P_F^{-1} \mathbf{y} = c\}$ where c is some constant), the ellipses of P_F are found to always lie within the intersection of the ellipses for P_1 and P_2 for all possible choices of P_{12} [15]. The intersection is characterized by the convex combination of sensor covariances:

$$P_F = (\omega_1 P_1^{-1} + \omega_2 P_2^{-1})^{-1} \quad (9)$$

and the corresponding sensor fusion for the CI algorithm is

$$\hat{X}_F = P_F\left(\omega_1 P_1^{-1}\hat{X}_1 + \omega_2 P_2^{-1}\hat{X}_2\right), \quad \omega_1 + \omega_2 = 1 \quad (10)$$

where $\omega_1, \omega_2 > 0$ are weights to be determined (e.g., by minimizing the determinant of P_F).

Recently, Wang and Li [3] proposed a fast CI algorithm where the weights are found based on an information-theoretic criterion so that ω_1 and ω_2 can be solved for analytically as follows:

$$\omega_1 = \frac{D(p_1, p_2)}{D(p_1, p_2) + D(p_2, p_1)} \quad (11)$$

where $D(p_A, p_B)$ is the Kullback-Leibler (KL) divergence from $p_A(\cdot)$ to $p_B(\cdot)$, and $\omega_2 = 1 - \omega_1$. When the underlying estimates are Gaussian, the KL divergence can be computed as:

$$D(p_i, p_j) = \frac{1}{2}\left[\ln\frac{|P_j|}{|P_i|} + d_X^T P_j^{-1} d_X + Tr(P_i P_j^{-1}) - k\right] \quad (12)$$

where $d_X = \hat{X}_i - \hat{X}_j$, k is the dimensionality of $\hat{X}_i$, and $|\cdot|$ denotes the determinant. Note that if $P_1 = P_2$, then $\omega_1 = \omega_2 = 0.5$, and the resulting fused estimate will be equivalent to that from Eq. (8) but with an inflated error covariance matrix (increased by a factor of 2). This version of the CI algorithm will be used for a quantitative comparison against the nonlinear fusers in Section IV.

C. Learning-Based Fusers

There are a number of fusers that can be trained to combine state estimates as many types of regression analysis methods exist that can be used to learn or compute the parameters of the fusing function we wish to estimate. In this work, we look at the use of ANNs for fusing the state estimates as they are known to be able to approximate any continuous function, given sufficient parameters.

1) Artificial Neural Network (ANN) Fuser: We consider a simple three-layer feedforward neural network, whose overall architecture is shown in below in Fig. 3. This network consists of an input layer, a hidden layer, and an output layer, interconnected by weights (to be determined) which are represented by the arrows between the layers. The inputs $\hat{X}_1, ..., \hat{X}_M$, for example, can be the state estimates from the sensors, and the outputs $\hat{X}_F^{(1)}, ..., \hat{X}_F^{(N)}$ are the global (fused) state estimates for N states. There is also a bias unit (not shown in Fig. 3) that is connected to each node in addition to the input nodes.

The nodes in the hidden layer are referred to as hidden nodes. The output of the j^{th} hidden node, a_j, is given by

$$a_j = g_1(\mathbf{w}_j^T \hat{X} + b_j) \tag{13}$$

where $\mathbf{w}_j = [w_{1j}, ..., w_{Mj}]^T$ and b_j are the weight vector and bias for the j^{th} hidden node, respectively. $\hat{X} = [\hat{X}_1, ..., \hat{X}_M]^T$ is a vector of input features (e.g., the state estimates from each sensor), and $g_1(\cdot)$ is a nondecreasing function called the activation function, which is typically a bounded function such as the sigmoid. A simple diagram illustrating this node function is shown in Fig. 4.

If we concatenate all of the hidden node outputs a_j into a vector $\mathbf{a} = [a_1, ..., a_L]^T$, then a single fused output of our network is given by

$$\hat{X}_F^{(i)} = g_2(\mathbf{w}_i^T \mathbf{a} + b_i) \tag{14}$$

where $\mathbf{w}_i = [w_1^{(i)}, ..., w_L^{(i)}]^T$ is the weight vector for the hidden node outputs, b_i is the bias for output i, and $g_2(\cdot)$ is an activation function.

When the target outputs are known, a well-known approach to determining the neural network parameters is called backpropagation. Backpropagation is based on gradient descent; the weights are initialized with random values and are iteratively updated to reduce the error (according to some user-defined error function, e.g., the mean-squared error). Once the network parameters are learned (from training data), new data can simply be fed into the neural network to obtain fused outputs.

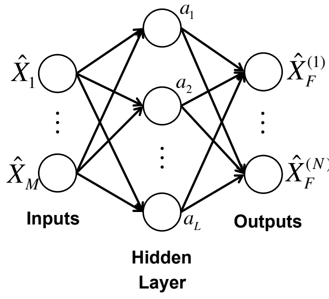


Fig. 3. Example architecture of a simple three-layer neural network.

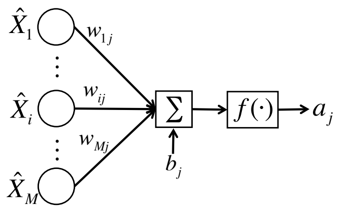


Fig. 4. Node function diagram. The inputs are multiplied by weights for that hidden node, summed, and then passed through a function to produce a hidden node output, a_j.

IV. System Simulation

The potential of neural networks for estimating the fusing function is quantitatively evaluated by comparing the performance results to conventional methods through simulation. In these simulations, we consider the simplified case where we have synchronous sensors and only one target with perfect data association so that we may focus only on the fuser performance. We model two sensors tracking a ballistic target in the exo-atmospheric coast phase whose trajectory is determined by a nonlinear state-space model. These sensors generate state estimates of the target's position and velocity, which are subsequently sent to the fusion center where they are combined to produce a final state estimate of the target. State-dependent errors are introduced with the use of a simple radar model in simulating the sensor measurements.

The state-space model of a ballistic coast target has the form

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ \mathbf{a} \end{bmatrix} \tag{15}$$

where $\mathbf{x} = [\mathbf{p}^T \quad \mathbf{v}^T]^T$ is the state vector consisting of the target's position $\mathbf{p} = [x \ \ y \ \ z]^T$ and velocity $\mathbf{v} = [\dot{x} \ \ \dot{y} \ \ \dot{z}]^T$ in the Earth-centered inertial (ECI) coordinate system (i.e., the coordinate system does not rotate; it is fixed relative to the "fixed stars", and its origin is at the center of the Earth) [16].

In the coast phase, gravity is considered to be the dominating force acting on a ballistic target, so the total acceleration is $\mathbf{a} = \mathbf{a}_G$, where $\mathbf{a}_G$ is the gravitational acceleration. The following is an expression for $\mathbf{a}_G$ that assumes a spherical Earth model [16]:

$$\mathbf{a}_G = -\frac{\mu}{\|\mathbf{p}\|^3}\mathbf{p} \tag{16}$$

where $\mathbf{p}$ is the target position vector from the Earth's center to the target, $\|\mathbf{p}\| = \sqrt{x^2 + y^2 + z^2}$ is its length, and $\mu = 3.986012 \times 10^5$ km^3/s^2 is the Earth's gravitational constant. The continuous-time model of the system is given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\mu x/r^3 \\ -\mu y/r^3 \\ -\mu z/r^3 \end{bmatrix} \tag{17}$$

where $r = \sqrt{x^2 + y^2 + z^2}$. An algorithm for computing the state propagation can be found in [17].

In tracking applications, the target dynamics are usually modeled in Cartesian coordinates, while the measurements are typically available in sensor coordinates (most often spherical coordinates) [18]. We simulate the measurements following the simulation in [19] for a ballistic coast target. The measurement model is given by $\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v}$, and the measurements of the range (r), elevation (E), and azimuth (A) of the target are computed as follows:

$$\mathbf{z} = \begin{bmatrix} r \\ E \\ A \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \tan^{-1}\left(z/\sqrt{x^2 + y^2}\right) \\ \tan^{-1}(x/y) \end{bmatrix} + \mathbf{v}, \tag{18}$$

where $\mathbf{v}$ is white Gaussian noise with covariance $R = diag\left(\begin{bmatrix}\sigma_r^2 & \sigma_E^2 & \sigma_E^2/\cos^2(E)\end{bmatrix}\right)$. As with the simulations in [19], the azimuth error is set to be a function of the elevation and the elevation error.

A simplified radar model is used to generate state values for σ_r and σ_E so that the errors are state-dependent and correlated across sensors. We will only consider the error that is dependent on the signal-to-noise ratio (SNR) for both the range and angle measurement accuracy since they usually dominate their overall respective radar error [20]. From [20], we have the following relationship between the standard deviation of the SNR-dependent random range and angle measurement errors and the SNR:

$$\sigma_r, \sigma_E \propto \frac{1}{\sqrt{SNR}} \tag{19}$$

The SNR (from the well-known radar range equation) is inversely proportional to r^4, where r is the range from the sensor to the target. To simplify, we assume a number of the radar parameters from the radar range equation are constant (e.g., the radar pulse duration, antenna gain, etc.) so that

$$\sigma_r, \sigma_E \propto r^2 \tag{20}$$

The range and elevation error of a ballistic target/satellite tracking phased array radar, the Cobra Dane, are published in Table 1 of [21] as 15ft and $0.05°$, respectively. These parameters are used to find reasonable values for scaling the σ_r and σ_E used in these simulations to generate the state-dependent measurement noise.

A. Generating State Estimates

Since the measurement noise is additive in spherical coordinates, a bias is introduced into the state estimates in Cartesian coordinates. Zhao et. al. [22] developed a recursive BLUE filter for a linear system that is theoretically optimal (in the mean-squared error sense) among all linear unbiased filters in Cartesian coordinates. This filter was used to generate the state estimates in these simulations to account for the converted measurements.

B. Simulation Setup

The training and testing data were generated from random initial positions and velocities, with standard deviations 100m and 5m/s, respectively (for each coordinate), about the mean $\mathbf{p} = [100,\ 2000,\ 4500]^T$ (km) and $\mathbf{v} = [1,\ 3,\ -6]^T$ (km/s). Starting from each initial state (position and velocity), a 180 second trajectory was generated for each data sample using the state-space model described earlier. Each sensor processes its own measurements using the BLUE filter, with fixed, but different values for what the sensor believes its error is in range and elevation (i.e., each sensor generated its state estimates using a fixed measurement error covariance matrix). For sensor one, we set $\sigma_{r_1} = 15$ft, $\sigma_{E_1} = 0.1°$, and for sensor two, $\sigma_{r_2} = 20$ft, $\sigma_{E_2} = 0.05°$.

Twenty training trajectories were generated along with one test trajectory, and the features input into the neural network were the state estimates from each sensor at the current time step. For the three-layer feedforward neural network described earlier, we need to specify its architecture by selecting activation functions and the number of hidden nodes. The sigmoid function is perhaps the most widely used activation function as it possesses a number of desirable properties (e.g., it is differentiable, smooth, nonlinear, and saturating), and it also admits a linear model if the network weights are small [6]. We will use the sigmoid $g_1(x) = \frac{1}{1+e^{-x}}$ at the hidden layer and a linear function at the output layer so that a single output of our network is given as

$$\hat{X}_F^{(i)} = \sum_{j=1}^{L} w_j^{(i)} \left(g_1(\mathbf{w}_j^T \hat{X} + b_j) \right) + b_i \tag{21}$$

The number of hidden nodes needed depends on the complexity of the function we are trying to estimate. Using too few hidden nodes may yield a poor approximation to the actual function. Using too many hidden nodes result in overfitting the data so that while the neural network may precisely give the desired outputs for the training data, it may not generalize well to unseen data. Unfortunately, there is no precise method that provides the optimal number of hidden nodes needed to properly model the data. Therefore, we also investigate the impact of the number of hidden nodes has on the ANN fusion. Lastly, we will also look at using only linear activation functions (i.e., $g_1(x) = x$ in Eq. (21)) in the ANN to compare nonlinear and linear fusion using learning-based fusers.

V. Results

Following [22], the filter was initialized with an effectively infinite initial state error covariance and a highly inaccurate initial state estimate. We present results for the position MSE starting at 120s (averaged over 100 simulations). Table I shows the average MSE for the ANN fuser for different numbers of hidden nodes. It can be seen that as the number of hidden nodes increases, the MSE decreases, but continuing to increase that number does not necessarily have a positive impact on the performance. However, it can be seen the performance of the fuser does not degrade much, which suggests that the

TABLE I
POSITION MSE FOR THE NONLINEAR ANN FUSER WITH DIFFERENT NUMBERS OF HIDDEN NODES.

# Hidden Nodes	Position MSE
2	2047.42
5	3.44
10	0.80
20	0.66
30	0.70
50	0.76

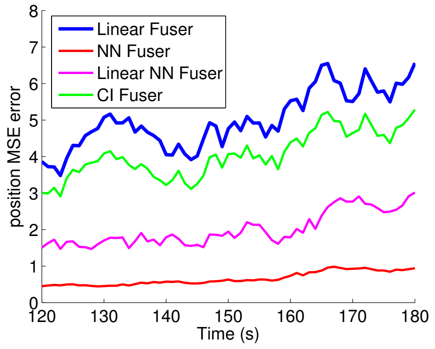


Fig. 5. Position MSE over time for linear and nonlinear fusers.

training data may be a good representation of our data as not much overfitting seems to occur. Fig. 5 shows the position MSE over time for the different fusers (linear, Covariance Intersection, and the ANN fuser with 20 hidden nodes). The resulting performance of the ANN fuser in these simulations shows promise for using nonlinear, learning-based fusers for improving the overall system performance.

VI. CONCLUSIONS

In this work, we explored the use of nonlinear fusers with multisensor fusion. We presented a probabilistic performance bound on the quality of a system as a function of several system components: the distribution of the state estimates, communication parameters, as well as the fusion algorithm. For optimal sensor fusion, detailed knowledge of the state estimates is typically required but may be unavailable (such as the cross-correlation of the errors across sensors), in which case one may use learning approaches to implicitly incorporate unknown information into the fuser function. The performance results from system-level simulations of a ballistic coast target with state-dependent errors demonstrate the potential for nonlinear fusers to improve the overall system performance.

ACKNOWLEDGMENTS

This work is funded by the Mathematics of Complex, Distributed, Interconnected Systems Program, Office of Advanced Computing Research, U.S. Department of Energy, and SensorNet Project of Office of Naval Research, and is performed at Oak Ridge National Laboratory managed by UT-Battelle, LLC for U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES

[1] X. R. Li and P. Zhang, "Optimal linear estimation fusion - part iii: Cross-correlation of local estimation errors," in *Proc. 2001 Int. Conference Information Fusion*, 2001.
[2] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT: YBS Publishing, 1995.
[3] Y. Wang and X. Li, "Distributed estimation fusion with unavailable cross-correlation," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 48, no. 1, pp. 259 –278, jan. 2012.
[4] K. Kim, "Development of track to track fusion algorithms," in *American Control Conference, 1994*, vol. 1, june-1 july 1994, pp. 1037 – 1041 vol.1.
[5] C.-Y. Chong, S. Mori, W. Barker, and K.-C. Chang, "Architectures and algorithms for track association and fusion," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 15, no. 1, pp. 5 –13, jan 2000.
[6] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley-Interscience, 2000.
[7] S. S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Boston: Artech House, 1999.
[8] M. K. Sundareshan and F. Amoozegar, "Neural network fusion capabilities for efficient implementation of tracking algorithms," *Opt. Eng.*, vol. 36, no. 3, pp. 692–707, 1997.
[9] J. Zhongliang, X. Hong, and Z. Xueqin, "Information fusion and tracking of maneuvering targets with artificial neural network," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 5, jun-2 jul 1994, pp. 3403 –3408 vol.5.
[10] S. Gezici, H. Kobayashi, and H. Poor, "A new approach to mobile position tracking," *Proc. 5th IEEE Int. Conf. Universal Personal Communications*, pp. 204–207, Mar 2003.
[11] N. Yadaiah, L. Singh, R. Bapi, V. Rao, B. Deekshatulu, and A. Negi, "Multisensor data fusion using neural networks," in *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, 0-0 2006, pp. 875 –881.
[12] F. Chowdhury, "A neural approach to data fusion," in *American Control Conference, 1995. Proceedings of the*, vol. 3, jun 1995, pp. 1693 –1697 vol.3.
[13] L.-W. Fong and C.-Y. Fan, "Multisensor fusion algorithms for maneuvering target tracking," in *E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on*, dec. 2006, pp. 80 –84.
[14] N. S. V. Rao, K. Brigham, V. K. Bhagavathula, Q. Liu, and X. Wang, "Effects of computing and communications on state fusion over long-haul networks," in *15th International Conference on Information Fusion*, 2012.
[15] S. J. Julier and J. K. Uhlmann, *General Decentralized Data Fusion with Covariance Intersection*, ser. Handbook of Multisensor Data Fusion. Boca Raton, FL: CRC Press, 2001.
[16] X. Li and V. Jilkov, "Survey of maneuvering target tracking. part ii: Motion models of ballistic and space targets," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 46, no. 1, pp. 96 –119, jan. 2010.
[17] M. Yeddanapudi, Y. Bar-Shalom, K. R. Pattipati, and S. Deb, "Ballistic missile track initiation from satellite observations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 3, pp. 1054–1071, 1995.
[18] Z. Zhao, X. Li, V. Jilkov, and Y. Zhu, "Optimal linear unbiased filtering with polar measurements for target tracking," in *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, vol. 2, 2002, pp. 1527 – 1534.
[19] T. Kerr, "Streamlining measurement iteration for ekf target tracking," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 27, no. 2, pp. 408 –421, mar 1991.
[20] G. R. Curry, *Radar System Performance Modeling*. Artech House Publishers, 2004.
[21] E. Filer and J. Hartt, "Cobra dane wideband pulse compression system," in *Proceedings of IEEE EASCON*, 1976, pp. 26–29.
[22] Z. Zhao, T. Rong Li, and V. Jilkov, "Best linear unbiased filtering with nonlinear measurements for target tracking," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 40, no. 4, pp. 1324 – 1336, oct. 2004.