Sensor Registration Using Neural Networks

HAIM KARNIELY
Bar-Ilan University
Israel

HAVA T. SIEGELMANN
Technion
Israel

One of the major problems in multiple sensor surveillance systems is inadequate sensor registration. We propose a new approach to sensor registration based on layered neural networks. The nonparametric nature of this approach enables many different kinds of sensor biases to be solved. As part of the implementation we develop some modifications to the common network training algorithm to tackle the inherent randomness in all components of the training set.

Manuscript received October 28, 1997; revised February 10, 1999; released for publication November 24, 1999.

Refereeing of this contribution was handled by S. Shrier.

IEEE Log No. T-AES/36/1/02588.

Authors' addresses: H. Karniely, Dept. of Mathematics and Computer Science, Bar-Ilan University, Ramat-Gan, Israel, E-mail: (haim@interwise.co.il); H. T. Siegelmann, Dept. of Industrial Engineering and Management, Technion, Haifa, Israel, E-mail: (iehava@ie.technion.ac.il).

I. INTRODUCTION

A. Multiple Sensor Surveillance Systems

Surveillance systems appear in many civilian and military applications. A surveillance subsystem is a key part of air traffic control (ATC) systems, air defense systems, naval command and control systems and lately even some car fleet management systems. The objective of a surveillance system is to calculate and display the states of objects in the real world based on "raw data" received from sensors. The objects of interest ("targets") could be aircraft, ships, cars, etc. The sensors could be radar, sonar, or any other sensors, and the state of an object could be its position, velocities, and possibly additional characteristics such as identification.

In order to give a complete description of the "real world" the system is required to obtain target trajectories. That is, to separate a set of unidentified measurements into subsets in such a manner that all measurements within a particular subset are originated by the same target, while measurements that belong to different subsets are associated with different targets. Using techniques of interpolation, extrapolation, and prediction of trajectory data, the state vector can be determined at any time, as to build *tracks* from targets. The process of *data association* into targets and building their tracks is known in the literature as *multiple target tracking*.

Fig. 1 illustrates the main functions performed in the tracking process. This is a repetitive process by which new measurements received from the sensors are compared to and associated with tracks that were build from former measurements. These measurements are then used to update the tracks they are associated to, or they initiate new tracks if necessary. The multiple target tracking process is described in detail in Section IIA.

Many surveillance systems must utilize multiple sensors for getting enough raw data needed for their processing. Different sensors usually have different coverage areas due, for example, to their geographic locations, and a system that receives data from several sensors is able to track targets in larger areas. Moreover, even when a target is located in an area that is covered by several sensors, the utilization of multiple sensors increases the probability of this target being detected, as well as the frequency at which measurements of this target will be reported to the surveillance system.

B. Systematic Errors in Multiple Sensor Surveillance System

The measurements reported by the sensors include errors. These errors can be thought of as consisting of two components: a random component ("noise"), and systematic component ("bias"). In most multiple

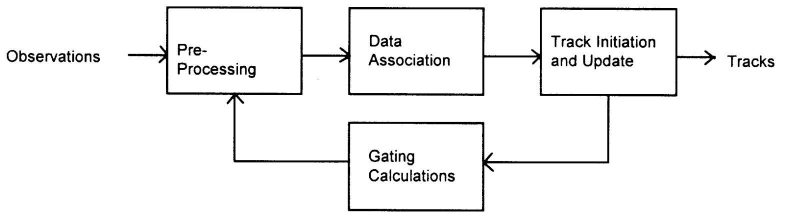


Fig. 1. Main functions of conventional multiple target tracking algorithm.

target tracking systems, track state estimation as well as the estimation of track covariance matrices used in gating calculations and measurement-to-track correlation criteria are based on Kalman filtering techniques [19, 20]. One of the assumptions on which these calculations are based is that a measurement $y(k)$ can be presented as a linear combination of the components of the target state vector $x(k)$ corrupted by uncorrelated noise. Following the notation in [4, ch. 2]

$$y(k) = H^{*}x(k) + \nu(k)$$

where H is the measurement matrix, and $\nu(k)$ is a zero-mean white Gaussian measurement noise with a known covariance. $E[\nu(k)] = E[y(k) - H^{*}x(k)] = 0$. The existence of systematic errors in sensor measurements implies that $E[y(k) - H^{*}x(k)]$ is greater than zero. This results in damages to the performance of track update, gating calculations, and measurement-to-track correlation functions. For these functions to perform adequately, the systematic errors should be removed in an earlier processing phase, namely in the preprocessing phase. The process of removing the systematic errors of the sensor is referred to as *sensor registration*, and these errors are sometimes referred to as registration errors.

In a single sensor system, if the size of the systematic errors is of an order similar to the random measurement error, their effect on the accuracy of the *tracks* will be of small significance. That is, the tracks will be biased a bit. Moreover, since these errors will usually offset measurements of targets that are close to one another in the same direction, their position relative to one another will be unchanged. Different sensors, however, will have different biases which will result in different offsets in measurements of the same target, or targets close to one another (see Fig. 2). Biases that are of small significance in a single sensor system, may cause a serious degradation in the performance of a multiple sensor system (as a result of their effect on the measurement-to-track correlation and track update functions). The discrepancy between measurements of the same target received from two different sensors, may cause the system to "believe" that they originated from two different targets, as measurements from one sensor will fail to correlate to a track initiated from measurement from the other sensor (see Fig. 3). Even if the system identifies that the measurements from the different sensors belong to the same target, the attempts to combine them into a single track (representing that target) may cause this track to "zigzag" as a result of false maneuver detection, and may even result in the loss of tracking of the target (see Fig. 4). The primary goal of sensor registration in multiple sensor systems is therefore to achieve *relative* sensor alignment. That is, to achieve a situation in which the sensors are aligned with respect to one another. For example, in an aircraft tracking system with two sensors, a situation in which measurements from both sensors include an offset

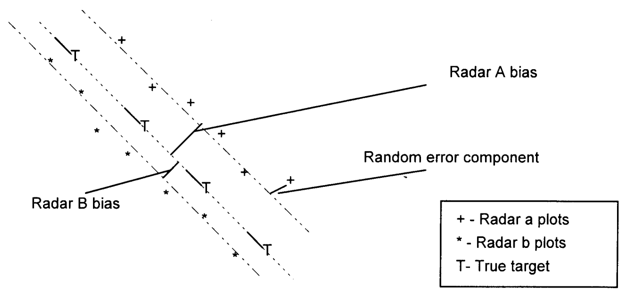


Fig. 2. Systematic and random measurement errors.

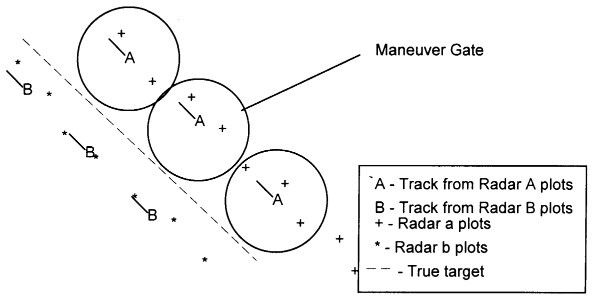


Fig. 3. Multiple tracks for single target.

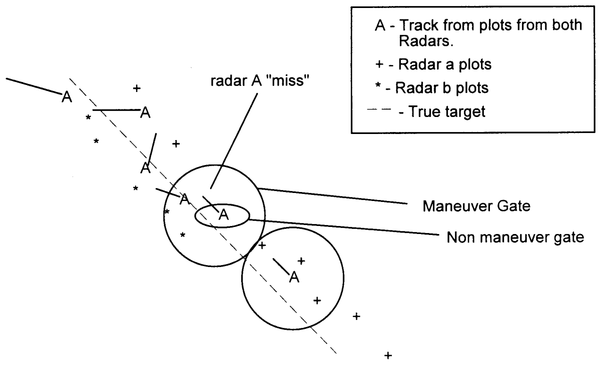


Fig. 4. Loss of track stability.

of 1 km to the south, is much better than a situation in which the measurements from one sensor include an offset of 0.5 km to the south and measurements from the other sensor include an offset of 0.5 km to the north.

A proper solution of the registration problem is essential for the success of a multiple sensor surveillance system. As stated by Dana [1] and previously by Fischer, Muehe, and Cameron [2]: "…attempts to net multiple sensors into a single surveillance system have met with limited success, due in large part to the failure to register adequately the individual sensors."

C. Registration Procedures

Even though sensor registration is considered a prerequisite for multiple target tracking, there have been only few publications on the registration problem. This problem is considered one of the least understood and documented aspects of multiple sensor systems [4]. A possible reason for the limited number of publications is the fact that most research in this field is performed by a small number of commercial companies and government institutions which are not interested in the publication of their developments.

Common solutions to the registration problem are based on statistical methods: averaging, least squares estimation, and a generalized least squares estimation (GLSE) [1]. These solutions are based on the assumption that the systematic errors stem from a fixed set of offsets, most commonly fixed offsets in the range and angle measurements or inaccuracies in sensor location, and they attempt to estimate the sizes of these offsets from pairs of correlated *plots* (measurements) from two different sensors. In reality, however, other causes for systematic errors may exist. For example, different fixed errors in different regions are a reasonable form of systematic errors, caused possibly by different weather conditions, or other conditions in different areas. When the assumptions of the existing solution do not hold, these solutions will provide poor performance and they might even add new errors to sensor measurements.

D. Neural Networks

Neural networks (NNs) are commonly used as nonparametric function approximators, and are used to learn input–output mappings. Here we concentrate on one particular technique, the "supervised" learning process. This process involves two phases: a training phase and a working phase. In the training phase the network is presented with a *training set* composed of input–output pairs. Each pair consists of an input pattern, and a *target value* that the network is expected to produce for this input. The network is to "learn" how to solve the problem by generalizing from the data it is presented with. In the working phase the network is presented with new inputs and is required to produce outputs according to what it has learned. An important feature of NN solutions is that they are nonparametric (model-free) solutions. The only information on the function being estimated that is used in the function approximation process is the input–output pairs in the training set. Specifically, no assumptions are made on the form of this function. For further details on NNs and NN training refer to [7, 8].

E. Our Contribution

We propose a new nonparametric approach to sensor registration; that is, we utilize the great power of NNs as general estimators [15–17] for the task of sensor registration. This approach does not require the nature of the systematic errors to be known in advance, and can therefore solve different kinds of sensor biases. This is a significant advantage since not all the sources of registration errors are currently known. Moreover, as sensor technology improves, new sources of registration errors will become significant, making a model-free registration procedure even more valuable.

In our system a NN is taught to correct plots from one sensor, so that they agree with the picture generated by the other (reference) sensor.[1] In the training phase, the network inputs are measurements from the first sensor, and matching measurements from the second (reference) sensor serve as the target values. During the training phase the network will "build" an internal representation of the bias correction algorithm. When training is finished, the network will be able to receive a biased plot from the first sensor as its input, and produce the same plot with the biases removed from it as its output.

Neural networks for registration purpose is indeed novel. In our application, both the input and the target value presented to the network in the training phase contain random noise components. This is not the case in most NN applications, where it is assumed that target values are accurate. As part of our research we develop modifications to the learning rate and error function used in the network training algorithm that compensate for the existence of random noise components in the target values of the training set.

We experimented with our NN application in models from the area of aircraft tracking systems that use radar sensors. We used simulated scenarios to test the performance of the network for different kinds of biases and compare it with that of the classical GLSE

[1]Note that our primary goal is to achieve *relative* sensor alignment. Therefore, if we get one sensor to agree with the second sensor, the overall picture generated by the system will improve tremendously, even if the reference sensor has biases of its own.

(generalized least squares estimation) registration procedure. Our methodology has produced good results for a wide range of biases, some of which were not solvable at all using previous methods.

We should note that while our research was concentrated on the registration problem in aircraft tracking systems that use radar sensors, similar approaches are probably applicable for other areas in which data from several sensors is integrated into a single system. Moreover, our adjustments to the error function and learning rate may be applicable to a large class of network learning problems in which target values in the training set contain random noise components.

F. Organization of this Paper

Section II provides an overview of the processing performed in a multiple target tracking system, describes quantitative requirements for sensor registration, lists known sources of registration errors, overviews existing registration procedures, and describes the main differences between these procedures and our new approach. Section III describes our solution in further detail, covering both practical implementation issues and aspects of NN training which we addressed as part of our research, concentrating on our modifications to the error function and learning rate. Section IV describes the method we developed for evaluating network performance. Section V describes our results and Section VI provides a summary of our current work and discusses its limitations as well as subjects for future work.

II. PRELIMINARIES: THE REGISTRATION PROBLEM

A. Multiple Target Tracking

This subsection defines some basic terms used throughout this work, and provides a general overview of the main functions performed by a multiple target tracking system. This overview can provide a better understanding of the role of sensor registration in the overall framework of a multiple target tracking system.

1) *Definitions*:

Target. A target is an object of interst in the real world (e.g., a car, an aircraft, etc.).

Plot/Measurement. A plot is a raw measured data reported by a sensor. The plot includes positional coordinates (of a single target) and is sometimes extended by more data such as Doppler speed, identification data, etc. A plot is usually corrupted by random noise and theoretically can be even a false alarm. Here we use the terms plot and measurement interchangeably.

Track. A track is an entity built by a tracking system to represent an existing target in the real world. It consists of smoothed state vector components that are position coordinates, velocity coordinates, maneuver data, etc.

2) *Main Phases of Processing in Multiple Target Tracking Systems*:

Preprocessing of measurement data: This initial processing of the "raw" measurement data received from the sensors prepares it for the next processing stages. Preprocessing could for instance include coordinate conversion and transformation, or filtering of false alarms in areas known to be problematic. This is also the phase in which the correction of registration errors should be performed.

Data association: The goal of this phase is to perform plot-to-track assignments (also known as plot-to-track *correlation*). The basic criteria used when deciding whether to assign a plot to a track is whether the plot is located within or out of a gate surrounding this track (see Fig. 2). The gate is calculated from the covariance matrices of the track and the plot. The covariance matrix of a plot is calculated from an error model of the reporting sensor. The covariance matrix of the track is calculated in the track update phase by using a Kalman filter and it expresses the uncertainty in the position of the track resulting from both errors in the plots it was built from and from the errors in target dynamics modeling. Much effort has been to solve conflicts arising when more than one plot (reported by a single sensor at a given time frame) falls inside the gate of a single track or when one or more plots fall in the gates of several tracks. Classical approaches, such as the nearest neighbor attempt to achieve unique plot-to-track pairing, where the best data association hypotheses in terms of maximizing likelihood is selected and this decision is irrevocable. Other approaches, such as *joint probabilistic data association* (JPDA [23–25]), associate the track with a weighted average of all (or several) of the plots with which it can be associated. In *multi hypothesis tracking* [21, 22] the data association process can result in several hypothesis and future plot data can be used to resolve uncertainty.

Track initiation and update: The processing in this phase depends upon the plot-to-track associations made by the data association phase. Plots which were not associated to any existing track serve as candidates for initiating new tracks. Plots that are associated to an existing track are used to update its state estimate. The track and/or the plot are extrapolated/interpolated to a common reference time. Then they are weighted together to achieve the new track state estimate. Track extrapolation and interpolation are enabled due to target motion modeling. These calculations are based on a priori knowledge of target dynamics. However, more than one model can be used to describe target motion, as done by interactive multiple model (IMM) algorithms [26, 27]. The weighting of the track and plot is based on the track and plot covariance matrices. The covariance matrix of the track (for the next processing cycle) is also estimated in this phase.

Gating calculations: The covariance matrices from the previous phase are used for calculating the gates for the next phase of data association. Gating considerations are also part of maneuver detection algorithms (see Fig. 2). The decision that a target is performing a maneuver, may have a significant effect on the processing performed in the track update phase.

B. Quantitative Requirements for Registration

The damages described in Subsection IB (multiple tracks for a single target, track zigzagging, etc.) occur when, as a result of registration errors, plots fall outside the "nonmaneuver gate"[2] of tracks that represent their targets. As a result, it is either falsely assumed that the target which this track represents is performing a maneuver, or it is falsely assumed that the plot originated from a different target than the track. The following analysis is based on Dana [1].

The nonmaneuver gate is defined by the following criteria:

$$\zeta = (Z - T)(\Sigma p + \Sigma t)^{-1}(Z - T)^T < G$$

where

Z is the plot ("raw sensor measurement") position in system coordinates (approximately normally distributed),

T is the track ("target state estimate") position system coordinates (approximately normally distributed, plot or track are predicted to a common reference time),

Σp is the plot covariance matrix which is calculated according to the sensor error model,

Σt is the track covariance matrix, which is calculated as part of the tracking process,

G is selected from the central χ^2 distribution tables so that in probability 99% of a plot that measures the target that the track represents will pass the criteria.

This criteria is designed so that at a required probability (typically 99%) a plot belonging to the same target as the track will pass it.[3] The existence of systematic errors reduces this probability and the reduction is largest when a plot from sensor 1 is compared with a track built from plots from sensor 2. The size of the reduction in this situation is determined by the size of $\lambda = b^T(\Sigma p1 + 0.5\Sigma p2)^{-1}b$, where $b = E[Z - T]$, is the difference between the plot and track resulting from the systematic errors. Dana continues to derive the maximal acceptable range, azimuth, and location offsets. We stop our overview of his analysis at this point since we are interested in the part of this analysis that relates to general bias scenarios.

C. Sources of Registration Error

There are several known sources of registration errors in radar-based systems. Some of these sources are [2] range offsets, range scaling, atmospheric refraction, azimuth offsets, azimuth errors due to antenna tilt, evaluation offsets, evaluation errors due to antenna tilt, time offsets and time scaling, inaccurate radar location, and coordinate conversion errors. There are sensor-level corrective measures that are taken in order to overcome several of these error sources. However, even when all sensor-level measures are taken the discrepancies between plots from different sensors are still large enough to cause the problems described in Subsection IB. In order to achieve relative sensor alignment a system-level phase of registration is required. The need for this second phase can be explained either by the accuracy of the sensor-level measures being insufficient, or by the existence of additional error sources which these measures are not designed to deal with.

D. Registration Procedures

Existing procedures for relative sensor registration aim at improving the accuracy of the solution for a subset of the known sources of registration errors. Registration procedures differ in the offsets that they attempt to estimate (the error sources that are considered significant) and in the way the sizes of these offsets are estimated. Dana [1] provides an overview of the registration problem, derives quantitative requirements for registration, describes existing registration procedures and resupports the GLSE method previously suggested by Fischer, Muehe, and Cameron [2] and rejected on computational grounds. He also demonstrates the advantages of this solution approach over a group of earlier registration procedures. In his article, Dana describes registration procedures designed to deal with angle offsets, range offsets, combinations of angle offsets and offsets in sensor location, and combinations of angle offsets and range offsets. Blom Hogendoorn and van Doorn [3] describe a registration procedure developed for ATC systems. Their method is designed to deal with range, angle, position, and range gain (range scaling) offsets.

E. Parametric Versus Nonparametric Approaches

Each of the existing registration procedures is designed to correct a specific set of sensor offsets. The main advantage of a NN-based solution is that

[2]This gate is used to test whether the difference between the track and plot can be accounted for by the random measurement errors in the plot and the uncertainty in the track's position estimate.

[3]It is assumed that the probability that the sensor misses the target represented by the track but reports a measurement of a new target (or false alarm) that is close enough to the track to pass this criteria is negligible.

the same network can be trained to solve different kinds of systematic errors. This advantage is a result of the nonparametric nature of NN solutions. Standard procedures assume that a certain model describes the systematic errors, and attempts to estimate the size of (a small number of) parameters of this model of their "training phase" (parameter estimation phase). In contrast, the NN training process can be viewed as estimating both the error model and its parameters. This is because the number of free parameters in the NN (network weights) provides the flexibility required to model many different bias natures.

The flexibility of our NN solution does have a price. A procedure that makes specific assumptions on the nature of the systematic errors will generally have certain advantages over the NN solution if (and only if) its assumptions hold. In this work the NN solution was compared with the GLSE procedure designed to solve angle and range offsets. The performance of the GLSE procedure in certain range and azimuth offset bias scenarios is better than that of the NN solution. However, the results of the NN solution are still acceptable. In such scenarios the GLSE procedure also required less training data than the NN solution (50–100 plot pairs instead of 150–200 plot pairs). For training sets of 200 plot pairs the NN solution is a little more sensitive to target distribution than the GLSE solution. For the AR2 bias scenario (see Appendix A), the performance of the "modified error function" NN solution for different training sets varied from 91% to 100% (the average was 98.4%), while the GLSE solution provided 100% of adequate registration for all training sets. Another advantage of the GLSE solution over the NN solution is that it requires less central processing unit (CPU) time for its training phase. However, while a more specific solution approach can be expected to provide better performance when its assumption hold, it should be noted that the very reason we suggest a NN solution is the fact that it cannot always be guaranteed that these assumptions in fact hold.

III. NEURAL NETWORK SOLUTION

A. Solution Overview

1) *Network Input and Output Values*: We define the goal of the network to correct a plot from one sensor (the "unregistered sensor") to agree with the picture generated by a second (reference) sensor. As stated before, our primary goal in the registration process is to achieve *relative* sensor alignment. Thus this goal is adequate even if the reference sensor has biases of its own.

The input to the network is a plot from the unregistered sensor, and the output is the correction that should be applied to this plot. Both the input and the output correction are in "system coordinates." That is, they are both represented in Cartesian coordinates relative to a common reference point. We must emphasize that the Cartesian representation of network input and output is specific to our implementation while the general solution approach can support any other form of input/output representation. The definition of the network's output as the correction to be applied to the input plot (rather than the corrected plot) is preferred because of scaling reasons. Proper scaling was found to be crucial to the success of the training process, and this definition enabled the network to "concentrate" on the "important" part of the corrected plot.[4]

2) *Application Considerations*: The training set used to train the network is constructed from pairs of measurements reported by two sensors which measure same targets. Each pattern/target pair consists of a measurement from the unregistered sensor (as the pattern) and the difference between the two measurements (as the target value). The pairing of these measurements can be performed on the basis of identification data contained in the measurements (e.g., if and only if (IFF) systems), by the results of the plot-to-track correlation processing of the target system after an initial level of registration is achieved by the use of external means (e.g., radar responders), or even by manual association performed by the system operator.

In practice, changes in the environment (e.g., changes in weather conditions) may result in changes in the systematic errors of the sensors, resulting in a time-dependent nature of the registration errors. The changes in the registration errors, however, happen over a much larger time scale than the changes in target positions, velocities and maneuvers (handled by *track update* processing). The common solution is to periodically collect new sensor data and reestimate the systematic errors. In our NN approach this translates to periodically retraining the network with recently collected data. The nonparametric nature of this solution approach will allow answering for changes in both the size and the nature of sensor systematic errors.

B. Modified Learning Rate and Error Function

The development of a NN-based solution to the registration problem required addressing different aspects of NN training. Some of the problems we dealt with are inherent to any network training problem, while others are a result of the particular nature of the registration problem. One subject which

[4]If we denote the unregistered plot as P' and the registered plot as P then $\|P' - P\| \ll \|P\|$. However, the correction $(P' - P)$ is the component of P that results from the registration errors that we are attempting to correct. The isolation of this component and its scaling to a $[-1,1]$ range resulted in significant improvement in the performance of the network.

required special attention was the existence of random error components in both patterns and target values in the training set. We developed two methods that improved the performance of the network training algorithm in the presence of such errors. These methods involve using a *modified error function* and using a *different learning rate for each pattern* and are described in detail in this subsection. Other interesting aspects of our implementation are described in IIIC.

1) *Gradient Descent Learning*: The network training process is based on minimizing an error function with respect to the network weights. It is aimed at finding a weight vector W', that minimizes $E(W)$, where W is the vector of all network weights, and E is the error function. The error $E(W)$ can be viewed as an estimate of the "distance" between the function that is performed by a network with weights W, and the "target function," which the network is required to learn. This "estimate" is based on the network performance for the patterns in the training set S.

$E(W) = E(S,W)$ usually takes the form $E(S,W) = \Sigma E(pi,W)$ where the sum runs over all $pi \in S$. The error for a single pattern $E(pi,W)$ is a measure of the difference between Oi, the network output for pattern pi, and Di, the target value for pi. A popular selection of $E(pi,W)$ is the sum of squares error $E(pi,W) = \|Di - Oi\|^2$, where $\| \;\|$ denotes the second norm.

The training process is an iterative process of weight updating. The required update at each step of training is determined using the gradient descent (steepest descent) principle. That is, the minimum of $E(w)$ is sought by moving in the direction opposite to the gradient of E with respect to W. The backpropagation algorithm [10] is used to calculate the gradient of E with respect to $W(\Delta W)$ and then the weight vector is updated by setting $Wi_{+1} = Wi - \eta\Delta W$ (η is called the learning-rate parameter). An error function that takes the form $E(S,W) = \Sigma E(pi,W)$, such as the "sum of squares" error function, treats all patterns in the training set equally. For such a function, ΔW takes the form $\Delta W = \Sigma \Delta Wi$, where ΔWi is the derivative of $E(pi,W)$ with respect to W.

2) *Measurement Errors in Patterns and Target Values*: For the registration problem both the patterns and targets contain a noise component. The size of these error components is different from pattern to pattern. Moreover, we have prior knowledge on the covariance matrices of these errors for each pattern.[5] Each pattern/target pair include one plot from each of the sensors. Each of the plots can be viewed as consisting of three components:

$$P1 + T + b1 + e1, \qquad P2 = T + b2 + e2$$

where $P1, P2$ are the two plots, T is the true target position, $b1, b2$ are the systematic error component in each of the plots, and $e1, e2$ are the random noise components in each of the plots.

Our goal is to teach the network to get a plot from sensor 1 to agree with sensor 2, that is, to map $T + b1$ to $T + b2$. The size of the systematic error typically changes very little for close plots (measured by the same sensor), thus the systematic error at locations $T + b1 + e1$, $T + b1$, would be almost the same. Since our input plot is $T + b1 + e1$, the required output for this input is $T + b2 + e1$. Since the plot we have for the taret value is $P2 = T + b2 + e2$, we can say that, for the purpose of our learning problem, the random error in the target value is $e2 - e1$, and its covariance matrix is $(\Sigma p_{i1} + \Sigma p_{i2})$, where Σp_{i1}, Σp_{i2} are the covariance matrices of the random error components of the two plots $e1, e2$.

In practice, we defined the output value for our network as the required correction $(b2 - b1)$[6] and used $P2 - P1$ as the target value. This definition does not however change the random error component in the output value $(e2 - e1)$ as in this case instead of the "true" target value $(b2 - b1)$ the target value we use is $[(b2 + e2) - (b1 + e1)]$.

3) *Modified Learning Rate*: The difference between the output and target value for a "noisy pattern" may be a result of the noise in the target value, rather than an error of the network. We therefore want to learn more from these pattern/target pairs that include less noise. One way to achieve this goal is to use a different learning rate for each pattern. In the weight update stage, instead of $W_{n+1} = W_n + \eta\Delta W = W_n + \eta(\Sigma\Delta Wi)$, we use $W_{n+1} = W_n + \Sigma\eta i \Delta Wi$, where ηi is selected to be smaller for patterns with a larger noise component.

Our first selection of ηi was $\eta i = \eta/\mathrm{tr}(\Sigma p_{i1} + \Sigma p_{i2})$. The learning rate was proportional to the inverse of the sum of the "size" (variance) of the error components in the target values. A practical problem that arose when implementing this solution was that huge learning rates were reached for patterns with small random errors. In order to avoid this problem we used $\eta i = \eta/(b + \mathrm{tr}(\Sigma p_{i1} + \Sigma p_{i2}))$, where the constant b was added to avoid ηi going to infinity for small covariance.

Using a different learning rate for each pattern is equivalent to using a weighted error function in which each pattern is associated with a different weight, as

$$[\Sigma\eta i\|Di - Oi\|^2]' = \Sigma\eta i[\|Di - Oi\|^2]'.$$

[5]It is a basic assumption in surveillance systems that the sensors have a known error model from which measurement covariance matrices can be estimated. For radars it is usually assumed that range and angle measurements are randomly distributed with known variances.

[6]This was done because of scaling considerations.

Low and Webb suggested the use of weighted error functions for classification problems [5, 6] in order to compensate for differences in the distribution of patterns in the training set and in the actual population, and deal with different penalties associated with different kinds of misclassifications. In our problem the different weight assigned to each pattern is aimed at learning more from patterns with less noise.

4) *Modified Error Function*: The above approach assigns a different weight to each pattern. Our next approach is to assign different weights not only different patterns, but also to different elements of the output vector in each pattern. In sensor registration, our goal is to maximize the probability of a plot from sensor 1 to correlate to a track built from plots from sensor 2. This is determined by the size of $\lambda = b[(\Sigma p1 + 0.5\Sigma p2)^{-1}]b^T$ (see IIB). Both $\Sigma p1$ and $\Sigma p2$ are different from pattern to pattern, as they depend on the sensor-target geometry. The nature of the sensor random errors is such that the variance of the range measurement is typically much less than that of the angle measurement. Depending on the sensor-target geometry, this implies that in some cases the difference in the x-component more dominant in λ and in others, the difference in the y-component will be more dominant.

Since we actually want to minimize λ for each of the patterns we should replace the standard error function $\|Di - Oi\|^2$ with $\lambda i = (Di - Oi)[(\Sigma p1 + \Sigma p2)^{-1}] \cdot (Di - Oi)^T$ (the 0.5 multiplier is omitted from $\Sigma p2$ since Di, Oi represent two plots and not a plot and a track). $\Sigma p1$, $\Sigma p2$ are generally not diagonal. However in order to avoid mixed coefficients and simplify computations, we used only the diagonal components of $\Sigma p1 + \Sigma p2$.

Thus for two-dimensional data instead of the standard squared error function for a single pattern:

$$E([Xo, Yo], [Xd, Yd]) = (Xd - Xo)^2 + (Yd - Yo)^2$$

we used

$$E'i(([Xo, Yo], [Xd, Yd]) = (Xd - Xo)^2/Vxi + (Yd - Yo)^2/Vyi$$

where

$$Vx = \Sigma p1(1,1) + \Sigma p2(1,1)$$
$$Vy = \Sigma p1(2,2) + \Sigma p2(2,2).$$

It is clear that Ei' is differentiable, that $Ei'(O,D) \to 0$ as $O \to D$ and that $Ei'(O,D) = 0$ when $D = O$. $E' = \Sigma E'i$ is therefore adequate to serve as an error function. Again, to avoid Ei going to infinity when Vx or $Vy \to 0$, we limit them from below by replacing Vx with $\max(Vx, b)$ and Vy with $\max(Vy, b)$. If the error variances of all the components of the target value vector are equal, using such a modified error function is equivalent to using a different learning rate for each pattern, as described in the first part of this subsection.

Our results show that both using a different learning rate for each pattern and using a weighted error function each produced improved results in comparison with backpropagation algorithm using the classical "sum of squares" error. From these two methods, the weighted error function produced the best results.

C. Other Aspects of our NN Solution

1) *Network Architecture*: In our research we used a feed-forward network with 2 hidden layers. We followed the suggestion of Sontag [13] that networks with two hidden layers are more suited for complex problem such as control related problems. The activation function we used for the two hidden layers was *tanh*, and the output layer was linear.

Several runs were made to test the performance of networks with different numbers of neurons in each layer. Layer sizes of 11.73 for the first layer and 5.29 for the second layer were tested. As a result we selected an 11-5 architecture which was the smallest architecture which produced good results. Two of the larger architectures, 11-29 and 31-29, produced results that were only slightly better and the smaller architecture was selected for simplicity. The range of architectures that was tested was selected according to sizes commonly used in similar problems. As an afterthought it is possible that smaller architectures could have been used. For us, they did not load so well, but perhaps would have worked with extra NN tricks and effort.

2) *Training Termination*: The fact that both input and target values in the training set contain a random error component implies that even if the network learns to correct the bias perfectly the network output will still not match the target value exactly. Moreover, because of the different sizes of the random error components in different plots in the training set, we cannot define the goal for our training process as a target value for the average squared error. We tested several criteria for termination of training including a "χ^2 test for goodness of fit to a theoretical distribution" and a test of the percent of plots that fall inside a certain gate around their target value. However our final approach was to go on training as long as the network performance continues to improve. This approach provided the best results when combined with a gradual reduction of the learning rate as described in the next subsection.

3) *Dynamic Adjustment of the Basic Learning Rate*: It is generally accepted that the learning rate should be adjusted during the learning process[7] [11, 12]. After testing several schemes for modifying the learning

[7]In this subsection we refer to the adjustment of the "basic" learning rate. This basic learning rate can then be modified as described in Section V.

rate, we adopted the following principle. If the performance of the network (measured by the percent of all the patterns for which the output value falls "close enough" to the target value[8]) does not improve for a certain number of epochs, the learning rate parameter is reduced by 50%. This scheme provided improved performance in comparison to using a constant high learning rate and quicker convergence than using a constant low learning rate.

4) *Using Sensor Coordinates as Additional Network Inputs*: Our motivation for adding sensor coordinates as network inputs was that the intuitive translation of most known radar biases to system coordinates involves using the coordinates of the sensors. This is true for example for range offsets, azimuth offsets, regional disturbances in a sector centered at the sensor, range and azimuth scaling offsets, etc. At first, this additional data seemed to improve performance. However, after the modified error function and modified learning rate were implemented, and proper input scaling applied, adding this data no longer contributed to network performance and in certain scenarios even resulted in slightly worse performance. It seems that in our case the advantages that can be gained from this "free data" do not justify the required increase in the number of adjustable parameters (network weights).

IV. NETWORK/REGISTRATION PROCEDURE PERFORMANCE EVALUATION

The network was tested using simulated scenarios. The use of simulated scenarios enabled testing the performance of the network for different kinds of biases and enabled accurate evaluation of network performance (as the true biases simulated in the training set can be used in the evaluation process).

The evaluation of existing registration procedures is based on the difference between the actual and estimated sizes of the offsets in range and angle measurements [1]. The evaluation of NN-based algorithm is usually based on the average squared error over a reference set. Both these methods cannot be used "as is" for our problem. The first because it is adequate only for range and angle offset biases and the second because it does not match the quantitative requirements for registration procedures. In this section we develop an evaluation method that uses a reference set, but measures performance over this set in a way that reflects the requirements for a registration procedure (see IIB).

A. Performance Evaluation Using an Exhaustive Reference Set

The result of a NN training process is the selection of network weights. "What the network has learned" is therefore not available in a "readable format." The only way to determine if the network has correctly generalized the training data is by running the network over a reference set (different than the training set) and evaluating the performance of the network for the set.

Since the network is trained using simulated scenarios, we can generate for each training set an exhaustive reference set that simulates the same biases (but with no random errors). For our reference set we selected plots from a tight grid that covers all the surveillance area (our area of interest). The function performed by the network is continuous. It is most reasonable to assume that the function representing the difference between plots from the two sensors at a given location is also continuous (or at least continuous almost everywhere). Therefore, if our grid is tight enough, the performance for a plot in any point can be approximately estimated by the average of the performance for the four corners of the square in the grid it is located in.

B. Measuring Performance for a Given Reference Set

In order to evaluate the performance of the network, we propose a single "overall grade" calculated from the results for all the patterns in this reference set. This grade was designed to meet the following requirements.

1) The performance of a registration procedure is measured in terms of its contribution to the success of the plot-to-track correlation process.

2) The same grade can be used for all kinds of systematic errors (this implies, for example, that it cannot be defined it terms of the maximal acceptable offsets in range and/or angle measurements).

3) While defining a single grade for the whole reference set, this grade should be sensitive to the existence of specific areas where the performance of the registration procedure is inadequate.

Following IIB the performance of a registration procedure for a given plot in the reference set, should be measured by the probability P of plots in that location succeeding to correlate to their tracks after they are "corrected" using this procedure. This probability is determined by[9]:

$$\lambda = b(\Sigma p1 + 0.5\Sigma p2)b^T$$

where $b = N1 - P2$,[10] $N1$ is the network output when presented with the plot $P1$ from the "unregistered"

[8]Where "close enough" is defined by a $\lambda i < M$ criteria.

[9]If we want to limit the probability for plots from any sensor correlation a track from the other sensor we could use $\lambda = \max(b(\Sigma p1 + 0.5\Sigma p2)b^T, b(0.5\Sigma p1 + \Sigma p2)b^T)$. Initial tests showed that this alternative measure does not modify results significantly.

[10]Note that both $N1$ and $P2$ are from the reference set and therefore do not contain random measurement errors.

sensor as input, and $P2$ is the required output (the plot from the reference sensor).

Overall network performance is then measured by the percent of the reference set (representing entire region) in which the probability P of a successful correlation (determined by λ) is larger than a required percent; in our case it is 90%. This definition of the "overall grade" matches all the requirements specified above.

Note that while the reference set that we use does not include random measurement errors, the criteria we use measures the probability of success over all possible values of random measurement errors. Rather than simulating random measurement errors and testing if $(b^*)(\Sigma p1 + 0.5\Sigma p2)(b^*)^T < G$, where b^* contains both specific random errors and the portion of the systematic errors still left after the network processing, we test if $\lambda = b(\Sigma p1 + 0.5\Sigma p2)b^T < g$, where b isolates the systematic error component and g is calculated based on the analysis presented in Subsection IIB to guarantee the required probability of success in the correlation process in the presence of random measurement errors.

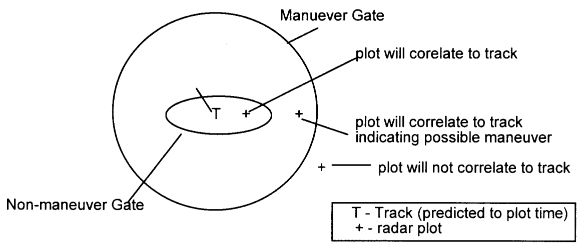
V. RESULTS

The goal of the tests we performed is twofold: to demonstrate the feasibility of a NN-based registration approach as well as its flexibility to deal with different kinds of sensor biases, and to evaluate the contribution of our modifications to the standard error function and learning rate to the performance of the backpropagation training algorithm. We tested the network-based registration procedures for different kinds of biases and compared their performance with that of the GLSE registration procedure developed to solve range and angle measurement offsets. The GLSE procedure was selected as a reference algorithm as it was shown to be superior to a group of earlier registration procedures, and a detailed description enabling its implementation can be found in [1].

The remainder of this section is organized as follows. Subsection VA describes the different kinds of biases we simulated in our tests; Subsection VB describes the simulation generator used to generate the training sets and reference sets; Subsection VC describes the test procedure used to test each of the bias scenarios; Subsection VD presents the performance of the different algorithms for each of the bias scenarios.

A. Kinds of Biases Used in Our Simulations

The bias scenarios we tested are combinations of angle offsets, range offsets, position offsets, range scaling offsets, and "disturbance sectors" with different values of both range and angle offsets. In this subsection we define each of these kinds of biases, a description of the specific biases used in our tests is given in Appendix A.



Manuever Gate
plot will corelate to track
plot will correlate to track indicating possible maneuver
+ plot will not correlate to track
Non-maneuver Gate
T
+
T - Track (predicted to plot time)
+ - radar plot

Fig. 5. Plot-to-track correlation gates.

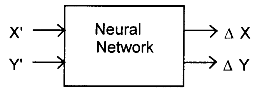


Fig. 6. Network input and output.

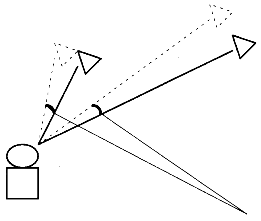


Fig. 7. Fixed offsets in angle measurements.

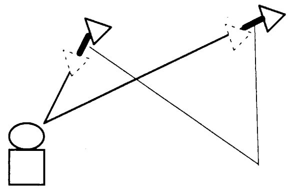


Fig. 8. Fixed offsets in range measurements.

Let X_0, Y_0 denote sensor position.

Let X_i, Y_i, R_i, θ_i denote the measured values of target position, target range from the sensor, and target angle relative to the sensor including random errors but without systematic errors.

Let X_i', Y_i', R_i', θ_i' denote the values of plot position, range from the sensor, and angle relative to the sensor as reported by the sensor.

If an *angle offset* of $\Delta\theta$ exists then for every plot $\theta_i' = \theta_i + \Delta\theta$ (see Fig. 7).

If a *range offset* of ΔR exists then for every plot $R_i' = R_i + \Delta R$ (see Fig. 8).

If a *range scaling* offset of r exists then $R_i' = rR_i$. (To visualize this error imagine the bias in Fig. 8 with the size of ΔR being a linear function of the distance of the target from the sensor).

If *position offsets* of ΔX, ΔY exist then $X_i' = X_i + \Delta X$, $Y_i' = Y_i + \Delta Y$ (see Fig. 12).

If *sectorial biases* of sizes $\Delta\theta_1$, ΔR_1 exist for the sector defined by angles θ_0, θ_1 then:

$$\begin{array}{llll} \text{If} \quad \theta_0 < \theta_i < \theta_1 & \text{then} \quad \theta_i' = \theta_i + \Delta\theta_1, & R_i' = R_i + \Delta R_1 \\ \text{Otherwise} & \theta_i' = \theta_i + \Delta\theta, & R_i' = R_i + \Delta R \end{array}$$

(see Fig. 13).

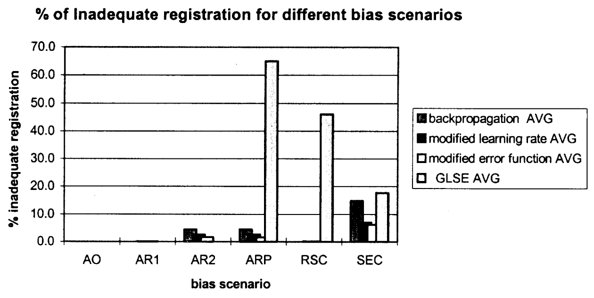


Fig. 9. Error percents for different training algorithms and bias scenarios.

If we superimpose all these systematic errors we receive

$$X_i' = X_0 + \Delta X + r(R_i + \Delta R')\cos(\theta_i + \Delta\theta')$$

$$Y_i' = Y_0 + \Delta Y + r(R_i + \Delta R')\sin(\theta_i + \Delta\theta')$$

where $\Delta\theta' = \Delta\theta_1$ if $\theta_0 < \theta_i < \theta_1$, and $\Delta\theta$ otherwise, $\Delta R' = \Delta R_1$ if $\theta_0 < \theta_i < \theta_1$, and ΔR otherwise.

B. Simulation Generation Tool

In order to generate training sets and reference sets for different bias and target scenarios we built a simulation tool that simulates the behavior of biased sensors. This program simulates the measurements that would be received from a pair of sensors with given biases for a given target set.

The inputs to the simulation generation program are two files: a sensor file and a target file. The sensor file includes a description of the sensors, their locations, the standard deviations of the range and azimuth measurements of each sensor, and sensor biases. The biases supported by the simulation program include all the kinds of biases described in the previous subsection: range, azimuth, and position offsets, range scaling factors, and sectorial biases. The target file defines the locations of the targets, which determine where plots are measured.

The simulation program calculates the range and azimuth from each of the sensors to the target, and adds normally distributed random errors to the range and azimuth measurements of each of the sensors (according to the variances defined in the sensor file). Then, the simulator adds systematic errors to the "measurements" according to the target locations and the biases defined in the sensor file. The resulting simulated measurements are written into the output file. This process is repeated for each of the targets in the target file.

C. Test Procedure

How well a registration procedure is able to learn a given bias depends on the kind of bias as well as the amount, quality, and distribution of the data in the training set. We wanted our simulated training sets to resemble as much as possible real training sets, built from plot pairs collected from a working system. For this reason, a random error component was added to the plots in the training set. The amount of the plot pairs (pattern/target pairs) in the training sets was also limited to an amount of the same order as the amounts used by "classical" procedures (200 pairs). This amount is limited in real systems by the time that would be required to collect the data.

In order to avoid dependence of our results on a specific target scenario we used 50 target files to test each bias scenario. Each of these target files contained 200 randomly distributed targets (Fig. 9).

For each bias scenario and each of the target files we generated (using our simulator) a training set of simulated plot pairs that would be measured by sensors with the defined biases for that target file. The network was trained with each of these 50 training sets separately, using three training algorithms: a normal backpropagation algorithm, an algorithm that uses a modified learning rate, and an algorithm using a modified error function. The latter two algorithms were designed to take advantage of prior knowledge on the covariances of the random error components in patterns and target values, as described in Subsection IIIB. The GLSE algorithm was also run for each of these training sets (Fig. 10).

After training (using a given training set) was finished, network performance was evaluated using an exhaustive reference set, as described in Section IV. The performance of a given algorithm for a given training set, was measured by the percent of the reference set for which the probability that a plot (in the same position as the plot in the reference set) will succeed to correlate to its track is at least 90% (for more details see Section IV). The performance of a given algorithm for a given bias scenario is measured by the average (and standard deviation) of its performance over the 50 training sets which were generated for this bias scenario.

D. Simulation Results

Table I and Fig. 9 present the average performance of the three NN training algorithms as well as the performance of the GLSE registration procedure for the different bias scenarios. The entries in the table display the percent of the exhaustive reference set for which registration is still inadequate after corrections are made using a trained network or the results of the GLSE algorithm. We chose to display this percent, rather than the percent for which registration is adequate since this presentation makes the differences in performance of the different algorithms clearer. Each line in Table I displays the result for one of the tested bias scenarios. For each of the four registration

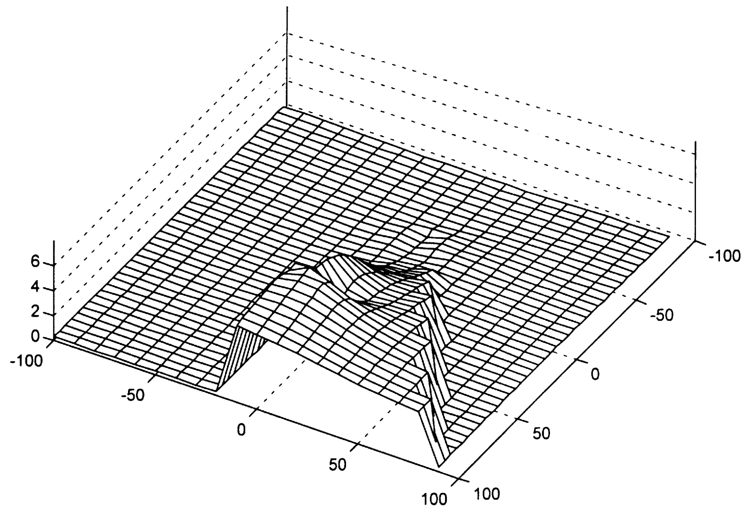


Fig. 10. GLSE normalized error graph.

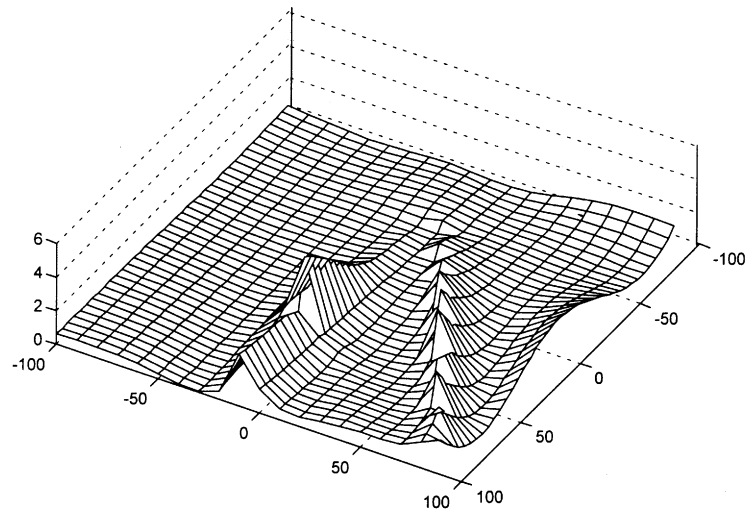


Fig. 11. Modified error function normalized error graph.

TABLE I
Error Percents for Different Training Algorithms and Bias Scenarios

	Backpropagation		Modified Learning Rate		Modified Error Function		GLSE	
Sensors	AVG	STD	AVG	STD	AVG	STD	AVG	STD
AO	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AR1	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.0
AR2	4.5	3.0	2.6	2.3	1.6	1.3	0.0	0.0
ARP	4.5	3.1	2.6	2.2	1.5	1.2	65.0	0.7
RSC	0.0	0.0	0.1	0.2	0.1	0.1	46.0	0.6
SEC	14.8	3.9	7.1	2.5	6.2	1.8	17.6	0.3

procedures the average error percent and standard deviation over the 50 training sets are presented.

Table I contains results of the following bias scenarios (described in detail in Appendix A):

AO	angle offsets only,
AR1	angle, range offsets 1 (range offsets 0.1 km),
AR2	angle, range offsets 2 (range offsets 0.2 km),
ARP	angle, range, position offsets,
RSC	angle, range offsets + range scaling offset,
SEC	angle, range offset + sectorial biases (sector with different angle, range offset).

The AO bias scenario simulates a situation in which both sensors have offsets in angle measurements (but no offsets in range measurements). This bias is adequately corrected for all the reference set by both the network-based algorithms and the GLSE algorithm for all of the tested training sets.

The AR1 and AR2 scenarios simulate offsets in both angle and range measurements of both sensors. The AR1 scenario simulates smaller range offsets (0.1 km), while the AR2 scenario simulates larger range offsets (0.2 km). In the AR1 scenario the performance of all the algorithms is almost perfect. The AR2 bias scenario demonstrates the possible advantages of the GLSE procedure when its assumptions hold, as well as the improvement in the performance of the NN-based algorithm as a result of our adjustments to the learning rate and error function. In this bias scenario the GLSE algorithm provides perfect results for all of the training sets, while the standard backpropagation algorithm fails on average to correct 4.5% of the reference set, with a standard deviation of 3%. The "modified learning rate" and "modified error function" fail (on average) to correct adequately 2.6% and 1.6% of the reference set, with standard deviations of 2.3% and 1.3%, respectively. The standard deviations indicate that for this bias scenario, the network-based algorithms are more sensitive to specific target distribution in the training set than the GLSE procedure. While this sensitivity is significantly reduced by our modification to the error function, the solution of this problem for a general bias scenario may require the use of "on-line" measures of training effectiveness as discussed in Section VI.

The biases simulated in the ARP scenario are similar to those in the AR2 scenario, except that constant offsets are added to target positions (simulating the results of inaccurate sensor location). This scenario provides the clearest example of the sensitivity of the GLSE algorithm to the existence of error sources other than those assumed when it is implemented. The GLSE algorithm, which corrected perfectly the biases in the AR2 scenario for all of the target scenarios, fails to adequately correct 65% of the reference set when fixed position offsets are added. The performance of the NN-based algorithms is almost identical to that in the previous scenario.

The RSC scenario simulates a situation in which in addition to fixed range and angle offsets, a range scaling offset exists. The performance of all NN-based algorithms for this scenario is good. They all correct adequately 99.9% of the reference set for almost all the tested scenarios. The GLSE algorithm is again unable to deal with a bias of an unexpected nature, and fails to correct adequately 46% of the reference set.

The SEC scenario is perhaps the most interesting one. In this scenario one of the sensors was assigned certain fixed offsets in range and angle measurements

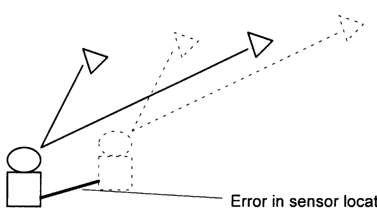


Fig. 12. Position offsets as result of inaccurate sensor location.

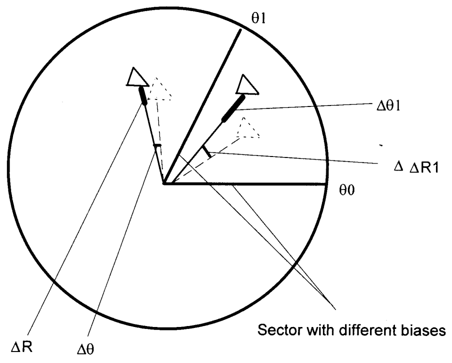


Fig. 13. Sectorial biases.

for a specific sector (−5°–35°) and other offsets for the rest of the region. Figs. 10 and 11 display the size of λ which is a "normalized error" (see Fig. 8) for all plots in the reference set, after they are corrected using the results of the GLSE and modified error function algorithms that were "trained" with one of the training sets. While the GLSE algorithm fails to adequately correct all of the disturbance sector, most of the errors of the modified error function algorithm are near the borders of the "disturbance sector." The GLSE algorithm uses all plots in the training set to estimate one set of radar range and angle offsets. Therefore, the estimates it reaches are a weighted average of the offsets inside and outside of the disturbance sector. Because the majority of the plots in the training set are outside the sector (about 80%), the offsets estimated by the GLSE algorithm provide acceptable results outside the sector, but unacceptable results for almost the whole sector. The standard backpropagation algorithm is also unable to correct this bias scenario, while the modified learning rate and modified error function algorithms provide significantly better results. The average percents of inadequate registration of the four algorithms are: GLSE—17.6% (standard deviation: 0.3%), backpropagation—14.8% (standard deviation: 3.9%), modified learning rate—7.2% (standard deviation: 2.5%), and modified error function 6.2% (standard deviation: 1.8%).

Measuring Performance Without Sector Borders: Since the sizes of the offsets change sharply near sector borders it is unreasonable to expect any procedure to identify these borders exactly (unless the training set includes an excessive amount of data from the area of these borders). We therefore remeasured

TABLE II
Average Percents of Inadequate Registration When Ignoring Sector Borders

Algorithm\nullified area	Normal	−2 km	−5 km	−10 km
backpropagation	14.76	13.70	12.43	10.33
modified learning rate	7.05	5.84	4.54	2.79
modified error function	6.21	4.99	3.68	2.13
GLSE	17.65	16.93	16.12	14.29

the performance of the different training algorithms when ignoring 2 km, 5 km, and 10 km regions around sector borders. The average performances are given in Table II. While the improvement in the performance of the GLSE algorithm is proportional to the percent of the disturbance sector that was removed, and the improvement in the performance of the standard backpropagation algorithm is only slightly larger, the improvement of the modified learning rate and modified error function algorithms is greater, indicating again that most of the areas for which registration is inadequate are near sector borders.

VI. DISCUSSION

Adequate sensor registration is considered a prerequisite for multiple-sensor multiple-target tracking. In this paper we presented a new nonparametric approach for sensor registration. While standard parametric approaches may provide better results when their assumptions hold, the main advantage of this new approach is in its ability to provide a reasonable (though may be not optimal) solution for many different and unspecified kinds of sensor biases.

The registration problem was formulated as a NN training problem. A method to evaluate registration performance for a general bias scenario was developed. The network was tested against the GLSE procedure in several bias scenarios, and the greater flexibility of the NN solution was demonstrated. As part of the implementation of this NN solution we addressed the problem of random noise components in the target values of the training set. We developed two adjustments to the network training algorithm designed to compensate for the existence of these random error components and demonstrated their contribution to training performance. Of these two adjustments, our modification to the sum of squares error function was shown to produce the best results.

We note that the NN-based approaches require significantly more processing time for the training phase than standard registration procedures. In this paper we made no attempts to optimize this aspect of the NN-based algorithms; in comparison with the time required to collect the training data, this processing time is not unreasonable. It is nonetheless an issue to be addressed in future works.

The goal of this work was to demonstrate the feasibility of a NN solution as well as its potential advantages over existing registration procedures. The next step should be the implementation of our approach for a commercial tracking system. The remainder of this section discusses subjects that require further study and suggests possible directions for future research.

The analysis of the reasonable/expected training sets for a given surveillance system is a difficult problem. The required amount of training data as well as the sensitivity of the training algorithm to the geographical distribution of the data in the training set may depend on the actual sensor biases. Future research may attempt to define conditions on the amount and/or geographical distribution of the data in the training set that guarantee training convergence for different kinds of biases. If such conditions are defined attempts can be made to control the data used to form training sets (e.g., by filtering the collected plot pairs if enough data is available). In the absence of any prior knowledge, it is generally recommended to collect as much data as can be collected under the time limitations that are defined for this process.

A related subject is the need of an on-line estimate of the quality/effectiveness of training the network with a given training set. This need exists when a priori conditions for "adequate training sets" cannot be defined. In this case an on-line measure of training performance can be used to decide, after training with a given training set is finished, whether to use the new weights (if the quality of the solution is good) or to ignore the current training set and reset network weights to their previous values. Such an estimate exists for the GLSE registration procedure [1], for the specific biases it is designed to deal with. A possible approach for on-line evaluation of a NN registration procedure is to use a cross-validation scheme such as that suggested for some earlier NN problems [9, 18]. The implementation of such an approach for the registration problem is left for future research.

Another subject which requires further study is an answer for possible (slow) changes in sensor biases. This subject is addressed in [3] for a standard registration procedure. A possible solution to this problem in a NN environment is to implement a "smart retraining" scheme. Such retraining is required to take advantage of the "knowledge" already present in network weights, while maintaining the ability to adjust to changes in sensor biases. NN retraining techniques have been suggested by previous researchers (see for example [14]). The implementation of these techniques to the registration problem is yet another direction for future research.

Finally we would like to emphasize that from the point of view of NN learning problems, our modification to the sum of squares error function and the learning rate can be used for different learning problems in which random error components of different sizes appear in the target values of the training set. This will occur whenever the target values in the training sets are measured using sensors for which a random error model exists. An example for such a problem is the prediction problem in tracking systems. Our approach may be used in an attempt to train a network to predict the position of the target using a certain number of previous measurements as inputs, and their successive measurement as the target value.

APPENDIX A. DETAILED BIAS SCENARIOS

In this subsection we provide a detailed description of each of the sensor bias scenarios that we use in our tests.

Angle Only Biases (AO)

Sensor	σ_R (km)	σ_θ (deg)	Range Offset	Angle Offset	X Offset	Y Offset	Range Scaling
Sensor 1	0.1	0.25	0.0	0.57	0.0	0.0	1.0
Sensor 2	0.1	0.25	0.0	−0.46	0.0	0.0	1.0

$$\theta'_{i1} = \theta_i + 0.57^\circ$$
$$\theta'_{i2} = \theta_i - 0.46^\circ.$$

Angle and Range Offset Biases 1 (AR1)

Sensor	σ_R	σ_θ	Range Offset	Angle Offset	X Offset	Y Offset	Range Scaling
Sensor 1	0.1	0.25	−0.1	0.46	0.0	0.0	1.0
Sensor 2	0.1	0.25	−0.1	−0.23	0.0	0.0	1.0

$$\theta'_{i1} = \theta_{i1} + 0.46^\circ$$
$$\theta'_{i2} = \theta_{i2} - 0.23^\circ$$
$$R'_{i1} = R_{i1} + 0.1$$
$$R'_{i2} = R_{i2} + 0.1.$$

Angle and Range Offset Biases 2 (AR2)

Sensor	σ_R	σ_θ	Range Offset	Angle Offset	X Offset	Y Offset	Range Scaling
Sensor 1	0.1	0.25	−0.2	0.46	0.0	0.0	1.0
Sensor 2	0.1	0.25	−0.2	−0.23	0.0	0.0	1.0

$$\theta'_{i1} = \theta_{i1} + 0.46^\circ$$
$$\theta'_{i2} = \theta_{i2} - 0.23^\circ$$
$$R'_{i1} = R_{i1} + 0.2$$
$$R'_{i2} = R_{i2} + 0.2.$$

Angle Range and Position Offsets (ARP)

Sensor	σ_R	σ_θ	Range Offset	Angle Offset	X Offset	Y Offset	Range Scaling
Sensor 1	0.1	0.25	−0.2	0.46	−0.3	0.0	1.0
Sensor 2	0.1	0.25	−0.2	−0.23	0.5	0.3	1.0

$$X'_{i1} = X_1 - 0.3 + (R_i - 0.2)\cos(\theta_{i1} + 0.46^\circ)$$
$$Y'_{i1} = Y_1 + (R_{i1} - 0.2)\sin(\theta_{i1} + 0.46^\circ)$$
$$X'_{i2} = X_2 + 0.5 + (R_i - 0.2)\cos(\theta_{i2} - 0.23^\circ)$$
$$Y'_{i2} = Y_2 + 0.3 + (R_{i2} - 0.2)\sin(\theta_{i2} - 0.23^\circ).$$

Range Scaling Bias (RSC)

Sensor	σ_R	σ_θ	Range Offset	Angle Offset	X Offset	Y Offset	Range Scaling
Sensor 1	0.1	0.265	−0.1	0.24	0.0	0.0	0.99
Sensor 2	0.1	0.265	0.1	−0.47	0.0	0.0	1.0

$$\theta'_{i1} = \theta_{i1} + 0.24^\circ$$
$$\theta'_{i2} = \theta_{i2} - 0.47^\circ$$
$$R'_{i1} = 0.99(R_{i1} - 0.1)$$
$$R'_{i2} = R_{i2} + 0.1.$$

Sectorial Bias (SEC)

Sensor	σ_R	σ_θ	Range Offset	Angle Offset	X Offset	Y Offset	Range Scaling
Sensor 1	0.1	0.15	0.0	0.17	0.0	0.0	1.0
Sensor 2 −5.7°–30°	0.1	0.15	0.0	0.8	0.0	0.0	1.0
Sensor 2 elsewhere	0.1	0.15	0.1	0.46	0.0	0.0	1.0

$$\theta'_{i1} = \theta_{i1} - 0.17^\circ$$
$$R'_{i1} = R_{i1}$$
if $-5.7^\circ < \theta_{i2} < 30^\circ$ then
$$\theta'_{i2} = \theta_{i2} + 0.8^\circ;\ R'_{i2} = R_{i2}$$
otherwise:
$$\theta'_{i2} = \theta_{i2} + 0.46^\circ;\ R'_{i2} = R_{i2} + 0.1.$$

REFERENCES

[1] Dana, M. P. (1990)
Multiple sensor registration: A prerequisite for multiple sensor tracking.
In Y. Bar-Shalom (Ed.), *Multitarget-Multisensor Tracking: Advanced Applications*.
Norwood, MA: Artech House, 1990, 155–185.

[2] Fischer, W. S., Muehe, C., and Cameron, A. (1980)
Registration errors in a netted air surveillance system.
Tech Note 40, MIT Lincoln Labs, 1980.

[3] Blom, H. A. P., Hogendoorn, R. A., and van Doorn, B. A. (1992)
Design of a multisensor tracking system for advanced air traffic control.
In Y. Bar-Shalom (Ed.), *Multitarget-Multisensor Tracking: Applications and Advances*, Vol. 2.
Norwood, MA: Artech House, 1992.

[4] Blackman, S. S. (1990)
Association and fusion of multiple sensor data.
In Y. Bar-Shalom (Ed.), *Multitarget-Multisensor Tracking: Advanced Applications*.
Norwood, MA: Artech House, 1990, 187–217.

[5] Lowe, D., and Webb, A. R. (1991)
Optimized feature extraction and the Bayes decision in feed forward classifier networks.
IEEE Transactions on Pattern Analysis and Machine Intelligence, **13**, 4 (1991), 355–364.

[6] Lowe, D., and Webb, A. R. (1991)
Exploiting prior knowledge in network optimization: An illustration from medical prognosis.
Network, **1** (1991), 299–323.

[7] Bishop, C. M. (1995)
Neural Networks for Pattern Recognition.
Oxford: Clarendon Press, 1995.

[8] Herz, J., Krogh, A., and Palmer, R. G. (1991)
Introduction to the Theory of Neural Computation.
Reading, MA: Addison-Wesley, 1991.

[9] Stone, M. (1974)
Cross-validatory choice and assessment of statistical predictions.
Journal of the Royal Statistical Society, **B36** (1974).

[10] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988)
Learning internal representations by error propagation.
Parallel Distributed Processing: Explorations in Micro Structure of Cognition, Vol. 1.
Cambridge, MA: MIT Press, 1988, ch. 8.

[11] Jacobs, R. A. (1988)
Increased rates of convergence through learning rate adaptation.
Neural Networks, **1** (1988), 195–307.

[12] Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T., and Alkon, D. L. (1988)
Accelerating the convergence of the backpropagation method.
Biological Cybernetics, **59** (1988), 257–263.

[13] Sontag, E. D. (1992)
Feedback stabilization using two-hidden-layer nets.
IEEE Transactions on Neural Networks, **3** (1992), 981–990.

[14] Pratt, L. Y. (1994)
Non-literal transfer among neural network learners.
In R. J. Mammone (Ed.), *Artificial Neural Networks for Speech and Vision*.
London: Chapman & Hall, 1994, 143–169.

[15] Cybenko, G. (1989)
Approximation by superposition of a sigmoidal function.
Mathematics of Control, Signals and Systems, **2** (1989), 303–314.

[16] Funahashi, K. (1989)
On the approximate realization of continuous mappings by neural networks.
Neural Networks, **2** (1989), 183–193.

[17] Hornik, K., Stinchcombe, M., and White, H. (1989)
Multilayer feedforward networks are universal approximators.
Neural Networks, **2** (1989), 359–366.

[18] Hansen, L. K., and Salmon, P. (1990)
Neural network ensembles.
IEEE Transactions on Pattern Analysis and Machine Intelligence, **12**, 10 (1990), 993–1001.

[19] Kalman, R. E. (1960)
A new approach to linear filtering and prediction problems.
Journal of Basic Engineering (Mar. 1960), 123–139.

[20] Balakrishnan, A. V. (1984)
Kalman Filtering Theory.
New York: Springer (University Series in Modern Engineering), 1984.

[21] Singer, R. A., Sea, R. G., and Housewright, R. B. (1974)
Derivation and evaluation of improved tracking filters for use in dense multi-target environments.
IEEE Transactions on Information Theory, **IT-20** (July 1974), 423–432.

[22] Reid, D. B. (1979)
An algorithm for tracking multiple targets.
IEE Transactions on Automatic Control, **AC-24** (Dec. 1979), 843–857.

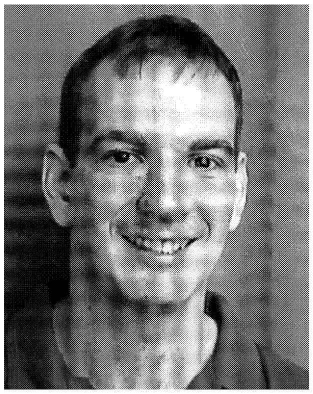
[23] Bar-Shalom, Y., and Tse, E. (1975)
Tracking in a cluttered environment with probabilistic data association.
Automatica, **11** (1975), 451–460.

[24] Bar-Shalom, Y. (1974)
Extension of the probabilistic data association filter to multi-target tracking.
In *Proceedings of the 5th Symposium on Non-Linear Estimation*, San Diego, Sept. 1974, 16–21.

[25] Formann, T. E., Bar-Shalom, Y., and Scheffe, M. (1983)
Sonar tracking of multiple targets using joint probabilistic data association.
IEEE Journal of Oceanic Engineering, **OE-8** (July 1983), 173–184.

[26] Blom, H. A. P., and Bar-Shalom, Y. (1984)
The interacting multiple model algorithm for systems with Markovian switching coefficients.

[27] Blom, H. A. P. (1984)
An efficient filter for abruptly changing systems.
In *Proceedings of the 23rd IEEE Conference on Decision and Control*, 1984, 217–224.



Haim Karniely received his B.Sc. in mathematics and computer science in 1990 and his M.Sc. in computer science in 1996, both from Bar-Ilan University, Ramat-Gan, Israel.

Between 1991 and 1996 he served in the IAF as a Technical Officer. He is currently the software development manager at InterWise Ltd., developing live e-learning applications.

Hava Siegelmann received the B.A. from the Technion (Summa Cum Laude) in 1988, M.Sc. from the Hebrew University in 1992, and Ph.D. from Rutgers University, New Brunswick, NJ, in 1993, all in computer science.

Currently, she is a senior lecturer on the faculty of Industrial Engineering and Management at the Technion.

Dr. Siegelmann has published in a variety of journals, including *Science*, *Theoretical Computer Science*, *Journal of Computer and Systems Science*, *IEEE Transactions on Information Theory*, *IEEE Transactions on Systems, Man, and Cybernetics*, and *IEEE Transactions on Neural Networks*. Her new book, *Neural Networks and Analog Computation: Beyond the Turing Limit* was published by Birkhauser as part of the Progress in Theoretical Computer Science Series. She was a 1995–1997 ALON Fellow (Israeli Presidential Young Investigator).