Distributed Target Classification and Tracking in Sensor Networks

R. R. Brooks, Sr. Research Associate, Applied Research Laboratory, Pennsylvania State University

P. Ramanathan, Professor, Electrical and Computer Engineering, University of Wisconsin

A. M. Sayeed, Professor, Electrical and Computer Engineering, University of Wisconsin

Abstract—The highly distributed infrastructure provided by sensor networks supports fundamentally new ways of designing surveillance systems. In this paper, we discuss sensor networks for target classification and tracking. Our formulation is anchored on location-aware data routing to conserve system resources, such as energy and bandwidth. Distributed classification algorithms exploit signals from multiple nodes in several modalities and rely on prior statistical information about target classes. Associating data to tracks becomes simpler in a distributed environment, at the cost of global consistency. It may be possible to filter clutter from the system by embedding higher-level reasoning in the distributed system. Results and insights from a recent field test at 29 Palms Marine Training Center are provided to highlight challenges in sensor networks.

Index Terms—Sensor Networks, Classification, Tracking, Collaborative Signal Processing, Location Aware Routing

I. INTRODUCTION

Sensor networks are an emerging technology that promises unprecedented ability to monitor and instrument the physical world [1, 2, 3]. Sensor networks consist of a large number of inexpensive wireless devices (nodes) densely distributed over the region of interest. Nodes have wireless connectivity and are tied to a backbone command network, such as the Internet. They are typically battery powered with limited communication and computation abilities [4]. Each node is equipped with a variety of sensing modalities, such as acoustic, seismic, and infrared.

Many challenges must be overcome to implement practical sensor networks. Two critical areas are: (i) efficient networking techniques, and (ii) collaborative signal processing (CSP) to efficiently process distributed information gathered. These problems are interconnected. For example, the utility of combining sensed data across nodes depends on network characteristics, such as latency. However, characteristics of the data exchanged, such as volume, affects network performance.

In this paper, we discuss a CSP framework for target classification and tracking in sensor networks. Our framework uses *location-aware data routing* that limits the scope of CSP to relevant subset of nodes conserving network resources, such as energy and bandwidth. Existing centralized algorithms for classification/tracking could be adapted for decentralized CSP using location-based formulations. However, CSP algorithms need to make efficient use of the sensor nodes' limited communication and computation abilities. The framework presented in this paper illustrates one approach for leveraging existing tracking and decision-making techniques within the constraints of sensor networks. We refer the reader to [4,5,6] for related work.

Thematic Example: Tracking a Single Target. We consider as an example tracking a target using a sensor network. Subsequent sections elaborate on the component problems of data routing, target classification and tracking.

Each object in the sensor field generates a time-varying spatial signature field that is sensed using multiple modalities [7]. A moving object is a spatial peak in a signature field that moves over time. Tracking a target involves tracking the peak location over time. To enable distributed tracking, the sensor field is divided dynamically into spatial cells. Within each cell, a manager node coordinates CSP tasks.

The approach presented has five basic steps for collaborative detection, classification, and tracking of a target moving through a sensor field.

1. Cells near potential target trajectories are put on alert. Nodes within cells collaborate to determine if a target is present.
2. When a target is detected, the cell becomes active. If classification finds a target of the desired type, tracking is initiated.
3. Tracking includes estimating target location, direction and speed for predicting future target positions.
4. Based on the predictions data from the active cell is sent to other cells; alerting them and facilitating CSP.
5. When the target is detected in an alerted cell that cell becomes active and the process repeats.

II. LOCATION AWARE ROUTING AND PROCESSING

Collaborative detection, classification, and tracking require data exchange between sensor nodes over an *ad hoc* wireless network with no central coordination of medium access. Instead, a fully distributed protocol regulates access. Furthermore, the communication range is very limited due to energy, size, and environmental constraints. Information must be forwarded from node to node to reach nodes outside the immediate vicinity.

In conventional wireless networks, data is exchanged between specific nodes. Even when nodes move, the connection remains between the same nodes. In contrast, in sensor networks, information exchange is between nodes in the same geographic region or concerning data with specified attributes. As nodes and targets move, the set of nodes involved in exchanges changes to reflect the new geographic regions and data attributes. As a result, it is widely accepted that traditional Internet Protocol (IP) based networking is not suited to sensor networks.

Data-centric and location-centric networking are alternatives to IP for data exchange in sensor networks [7,8,9]. In the *data-centric approach* [9,10], sensor nodes publish or subscribe to data with attributes defined in the communications request. Other nodes may not immediately have the data to respond. They note subscriptions for future use. When data whose attributes match the subscription becomes available, nodes transmit the data. Subscribed nodes receive published data over the network. The advantage of the data-centric approach is that no data is exchanged before events of interest occur. However, the nodes must periodically renew subscriptions, and the network must maintain routes from all publishing nodes to the subscribed nodes.

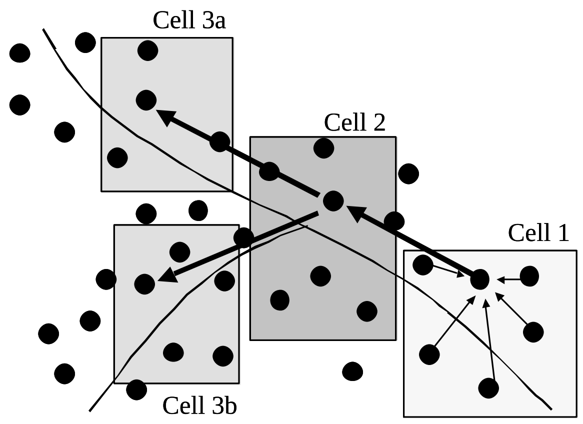

Figure 1. Location-centric approach for target tracking.

In the *location-centric approach*, geographic cells play the role of nodes in IP networks [7,8]. Depending on the data requests, cells are created and tasked as needed. A manager node is created and tasked with coordinating activities in the cell as needed. Nodes in cells collaboratively decide when events of interest occur. If other cells are needed, the manager node creates and tasks new cells. For example, consider target tracking in Figure 1. A target enters the field and cell 1 forms to track it. Data is shared locally and the manager node creates cell 2 to maintain surveillance. The manager of cell 2 creates cells 3a and 3b to continue the track as appropriate, since the target may take alternative paths. In Section 3, we will discuss how cell sizes can be determined dynamically and one example of how data can be shared. All decisions are made dynamically with local information.

Diffusion routing is a solution proposed to efficiently route data in the data-centric approach. In diffusion routing, nodes exchange two types of control messages, *interests* and *reinforcements*. Interests are data subscriptions. They are diffused to convey node interests. This dissemination sets up *gradients* within the network "drawing" relevant data to interested nodes. Data flows to interested nodes along multiple paths. The network reinforces paths using control messages. Over time, data is sent only along reinforced paths [9]. Enhancements to this approach can take advantage of location information [10].

UW-routing is a location-centric approach developed at the University of Wisconsin [8]. Unlike diffusion routing, routes are not established and maintained until data needs to be communicated. To forward data from cell to cell, a Route Request (RREQ) is diffused through the network. A cell is addressed by its geographic location and this information limits data propagation. Also, as the RREQ propagates, state information is temporarily deposited in the network to identify an efficient route from source to destination cells in a distributed manner. When the RREQ reaches a node in the addressed cell, it responds with a RREP control message. The RREP message is routed to the source cell using the state information from the propagation of the RREQ. When the RREP message reaches the source cell, a single path to the destination cell is established. This path is used to send data from the source cell to nodes in the destination cell. In the destination cell, data is diffused to all nodes in the cell by the manager node.

III. TARGET TRACKING

Centralized tracking [11] using sensor networks is possible, but has numerous drawbacks. Sending time series data through the network introduces latency and synchronization issues. It also consumes energy and network bandwidth, while potentially introducing a single point of failure. Associating sensor readings to tracks suffers from combinatorial explosion when multiple sensors are used. It becomes ambiguous when sensors have overlapping ranges, disagree, or when multiple targets are present [12].

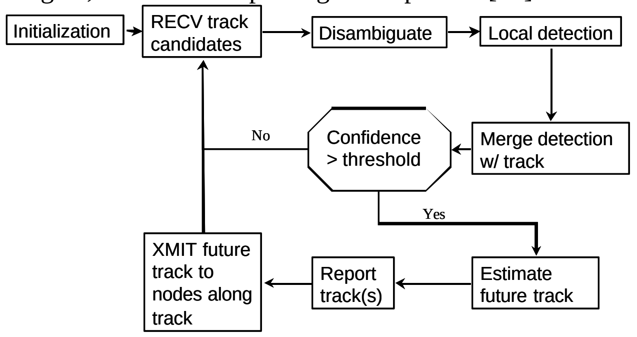

Figure 2. Flowchart of the processing performed at any given node to allow distributed target tracking.

Figure 2 gives a flowchart of the data flow at each node in our distributed tracking approach. Multiple threads execute concurrently and the system is a peer-to-peer network. All nodes execute the same logic:

1. *Initialization* declares node attributes to the location-centric network.
2. *Candidate track information* describing approaching targets is continuously received and stored in temporary priority queues.

3. *Local detection and parameter estimation* provide inputs to the tracking algorithm.
4. *Detections are merged* with the track that best fits the current data. Target attributes from the candidate track record are projected forward to the time of the current detection and compared with the current data.
5. *Confidence threshold* is set so that when no candidate tracks adequately match the current detection, a new track record is created.
6. *Estimate future track* from recent information and update the track record.
7. *Report track update to user community* (outside the scope of this paper).
8. *Transmit updated track* record to regions along the target trajectory. Using multiple regions of varying size can provide fault tolerance. Queues containing precise regions are considered first.

Local parameter estimation is done using a location centric approach. Closest Point of Approach (CPA) data is shared locally. The CPA is a robust statistic and easily detected. It corresponds to the signal peak in Figure 4. Cells form dynamically within a limited space-time window. The manager node is chosen as the sensor node with the strongest signal in the space-time window. Linear regression using the trigonometry of node locations is used to estimate target position, velocity and heading. In the numerical results presented below, typically one CPA event from each of three modalities of four to five nodes was used in this calculation (12 to 15 total). The results in [13] show this to be a reliable technique.

Given local detection information and a list of tracks, data association is required to map the detection to a track. In the results we present here, we used a simple Euclidean metric computing the difference of the last target track estimate (position, velocity, and heading) projected forward to the time of current detection. In a multi-target tracking scenario with *n* targets and *n* tracks, [12] states that centralized association requires at least *n* (*n*-1) comparisons. In the distributed case, the manager node is the only one performing comparisons. It has one detection and at most *n* candidate tracks (possibly fewer) and thus at most *n* comparisons are required. Hence, the combinatorial explosion that exists in the centralized case does not occur.

When a track is continued, the manager node defines a new cell. The cell position encloses the region the target is likely to traverse. The cell size is a function of observed target velocity. The track information packet is routed to the new cell. The tracking process repeats at this point.

Figure 3 shows example target tracks from a field test at 29 Palms Marine Training Ground. Prototype hardware easily handled the sensing, processing, and networking requirements. Network latency and packet dropping were not significant during this test. Due to a microphone deployment issue, over 50% of the CPA events detected were false positives. The linear regression for velocity estimation found no correlation among false alarm CPA's and thus translated false positives into target tracks of zero velocity, which effectively removed them from the tracking system.

The left hand image in Figure 3 shows data from the software used in that test. The target track diverges and continues through only part of the field. To correct these deficiencies, various data association and track estimation techniques were tested. The middle image shows an Extended Kalman Filter (EKF), similar to the one in [14, 15], added to the track estimation process. It reduced track divergence and targets were tracked for a longer distance. The EKF implicitly assumes a target with linear motion and does not enforce global consistency.

The right image in Figure 3 shows results using "lateral inhibition." Before continuing a track, nodes whose current readings match a candidate track broadcast their intention to continue the track. They wait for a period of time proportional to the log of their goodness-of-fit metric. During this wait, they can receive messages from other nodes that fit the candidate track better, in which case, they drop their continuation. If no other node has a better fit, the node continues the track. In our tests, this approach reduced track divergence more than the other approaches. It does not assume a linear trajectory and enforces some global consistency.

Table 1 contains error information for the techniques in Figure 3. Tracks produced using the EKF appear to be the most accurate, with lateral inhibition not being significantly worse. Lateral inhibition does have a significant advantage in reducing the tendency of tracks to diverge since it does not assume any target trajectory.

	RMS for tracks from Nov 08 2001			
	Live Data	EKF	Lateral Inhibition	EKF & LAT
Averaged	18.108328	8.877021	9.361643	11.306236
Track Summed	81.456893	52.775338	13.535534	26.738410
	RMS for track beginning at Nov_08_14.49.18.193_2001			
	Live Data	EKF	Lateral Inhibition	EKF & LAT
Averaged	14.977790	8.723196	9.361643	8.979458
Track Summed	119.822320	183.187110	18.723287	35.917832

Table 1. Root mean square error comparison for the data association and track estimation techniques discussed. The top set of numbers is for all target tracks collected on Nov. 8, 2001. The bottom set of numbers is for the target run in Figure 3. In each set, the top row is the average error for all tracks made by the target during the run. The bottom row sums the error over all the tracks. Since these tests were of a target following a road, the EKF filter has an advantage since it assumes a linear trajectory. Lateral inhibition still performs well, although it is non-parametric.

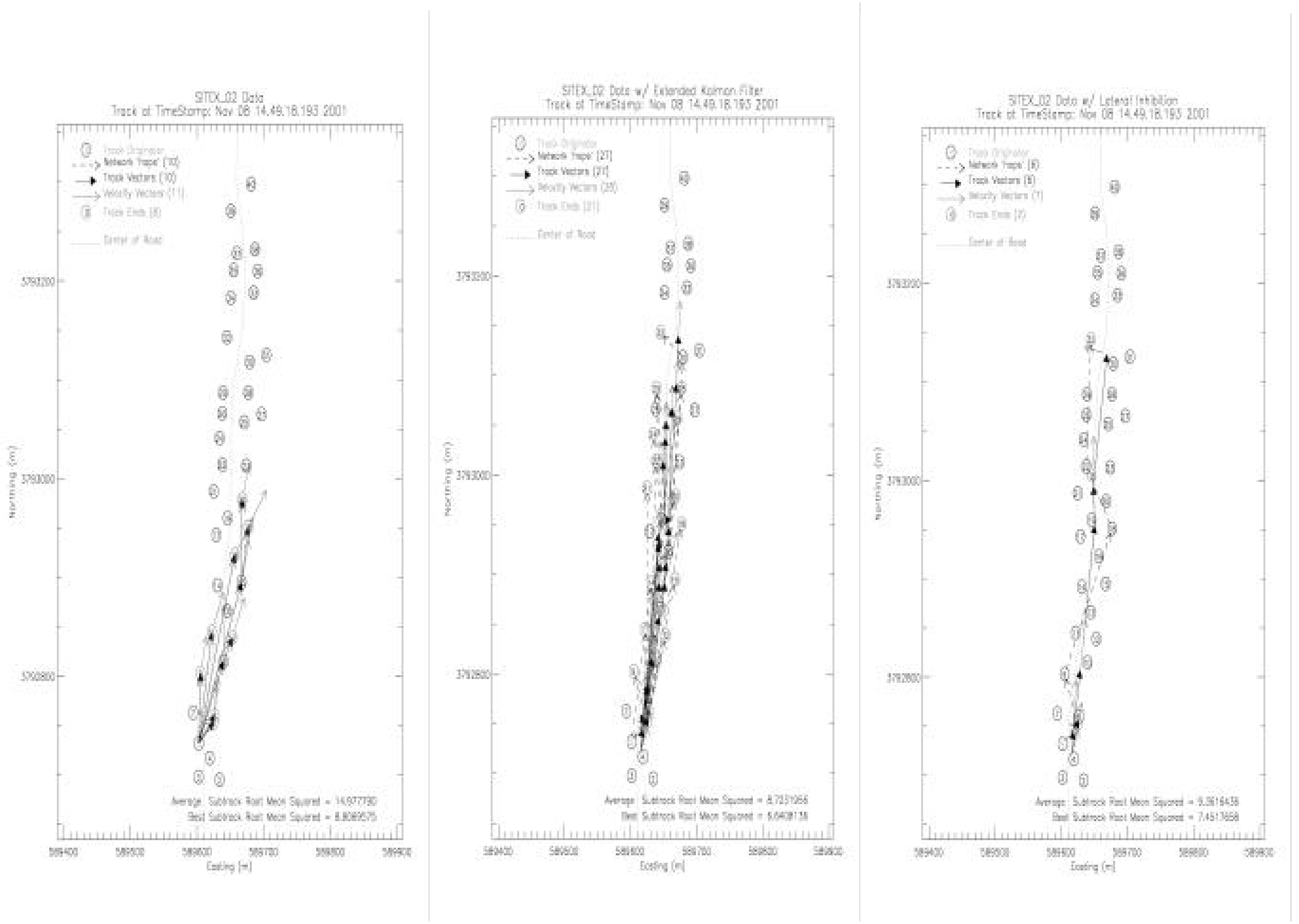

Figure 3. Tracks of the same single target at 29 Palms. Axes are UTM coordinates. Circles are sensor nodes. The faint curve through the nodes is the middle of the road. Dark arrows are the reported target tracks. Dotted arrows connect the manager nodes that formed the tracks. From left: no filtering, EKF, and lateral inhibition.

		CPA size	40		
		Inhibition size	56		
	Track packets	Track pack size	CPA packets	Inhib. packets	Total
EKF	852	296	59	0	254552
Lateral inhibition	217	56	59	130	21792
EKF & Lateral inhibi	204	296	59	114	69128
Centralized	0	0	240	0	9600

Table 2. Data transmission requirements for the different data association techniques. The total is the number of bytes sent over the network. The EKF requires covariance data and previous data points. Angle gating and lateral inhibition require less data in the track record. Data is from the tracking period shown in Figure 3.

Table 2 compares the network traffic incurred by the approaches shown in Figure 3 with the bandwidth required for a centralized approach using CPA data. CPA packets had 40 bytes, and the lateral inhibition packets had 56 bytes. Track data packets vary in size, since the EKF required three data points and a covariance matrix. The table shows that lateral inhibition requires the least network bandwidth due to reduced track divergence.

Note from Table 2, that in this case centralized tracking required less than half as many bytes as lateral inhibition. This data is somewhat misleading. The data shown is from a network of 40 nodes with an Internet gateway in the middle. As the number of nodes and the distance to the gateway increases, the number of packet transmissions will increase for the centralized case. For the other techniques, the number of packets transmitted will remain constant. Recall the occurrence of tracking filter false positives in the network, which was more than 50% of the CPA's during this test. Reasonably, under those conditions the centralized data volume would more than double over time and be comparable to the lateral inhibition volume. Note as well that centralized data association would involve as many as 24 to 30 CPA's for every detection event in our method. When association requires O(n^2) comparisons [12] this becomes an issue.

IV. TARGET CLASSIFICATION

In this section we outline a CSP approach to target classification based on node measurements within a cell. We discuss fusion of measurements from a purely decision theoretic viewpoint, followed by distributed application of the fusion techniques in sensor networks, along with the associated communication and computation burden. We discuss Gaussian classifiers that assume Gaussian data – the general fusion principles presented here also apply to arbitrary classifiers. Gaussian classifiers exploit only the second-order statistics of the (possibly non-Gaussian) data and are an attractive choice requiring estimation of mean vectors and covariance matrices, instead of arbitrary joint probability densities, for different target classes.

A. *Single Measurement Classification*

Event detection. Classifiers operate on feature vectors extracted from time series data corresponding to an event. Node detection algorithms extract data segments. Energy detectors are typically used for event detection. At each instant, the detector monitors signal energy in a given time window. Events are declared when the energy exceeds a threshold. The threshold is dynamically updated based on background noise statistics to maintain a constant false alarm rate. Once a node detects an event (e.g., the presence of a moving vehicle), it stores a time series segment corresponding to the event. As illustrated in Figure 4, the time series segment is extracted from the interval in which the energy first exceeds the threshold (event onset) and then drops below it (event offset) due to the target passing by the node. The time instant of the maximum reading signifies the CPA. Time series data may also be used for target localization algorithms (see, e,g. [16, 17]). Simpler localization algorithms that exploit energy decay profile are also possible [7].

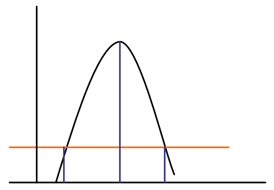

Figure 4. Illustration of event detection by thresholding the energy detector output. The horizontal line represents the threshold. The maximum reading corresponds to CPA time.

Lower dimensional **feature vectors** are extracted from time series. Feature vector selection is important as it impacts classifier performance [18]. In our work, spectral (Fourier) features were used since vehicle signatures exhibit dominant harmonic characteristics [7]. Each event yields multiple feature vectors that are collapsed into a single effective one, the mean, for example.

Gaussian classifiers. Let $\mathbf{x}$ denote an N dimensional complex-valued event feature vector. Suppose there are M target classes, $w_m \in \Omega = \{1,\cdots,M\}$. We assume each event consists of a single target. A classifier C assigns **x** to one of the target classes. We focus on maximum likelihood (ML) classifiers corresponding to equal prior probabilities for different classes [16]. The ML classifier is given by $C(\mathbf{x}) = \arg\max_{j=1,\cdots,M} P(\mathbf{x}\,|\,w_j)$ which assigns the class with the largest likelihood $P(\mathbf{x}\,|\,w_j)$ to **x.** In Gaussian classifiers, the feature vectors are modeled as complex Gaussian with mean $\boldsymbol{\mu}_j = E_j[\mathbf{x}]$ and covariance matrix $\mathbf{S}_j = E_j[\mathbf{x}\mathbf{x}^H]$, where the superscript H refers to complex conjugate transpose and $E_j[.]$ denotes ensemble average over the j-th class. The likelihood function then takes the form $P(\mathbf{x}\,|\,w_j) = \frac{1}{p^N\,|\,\mathbf{S}_j\,|}\exp(-(\mathbf{x}-\boldsymbol{\mu}_j)^H\,\mathbf{S}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j))$.

Training and testing. Designing the Gaussian classifier corresponds to determining the mean vectors and covariance matrices for the different classes. This is done from available training data. The training and performance assessment is usually done through cross-validation [18] in which the available data is split into multiple groups of training and testing subsets. For each group, the mean vectors and covariance matrices for all classes are estimated from the training sets. The estimated parameters are then used for assessing the performance of the classifier using the testing set. The results of the experiments generate an M by M confusion matrix whose elements $[\mathbf{CM}]_{ij} = n_{ij}$ represent the number of vectors from w_i classified as w_j. Two performance metrics are typically computed from the confusion matrix. The probability of correct detection for class m is given by $PD_m = n_{mm}\Big/\sum_{j=1}^{M} n_{mj}$ and the probability of false alarm is computed as $PFA_m = \frac{1}{M-1}\sum_{k=1,k\neq m}^{M}\frac{n_{km}}{\sum_{j=1}^{M} n_{kj}}$ which signifies the probability that an event is labeled from class m when the true underlying class is different.

B. *Multiple Measurements*

Suppose that multiple measurements are available for each event. These measurements may be from different sensing modalities at a particular node or from multiple measurements at different nodes. Suppose K measurements are available. Let $\mathbf{x}_k$, k =1, …, K, denote the feature vector for the k-th measurement. The classifier now operates on all K measurements to decide the class for the event $C(\mathbf{x}_1,\cdots,\mathbf{x}_K) = \arg\max_{j=1,\cdots,M} P(\mathbf{x}_1,\cdots,\mathbf{x}_K\,|\,w_j)$.

The classifier can combine the information from different measurements in two ways: 1) *Data fusion* in which the classifier jointly operates on the feature vectors of all measurements, or 2) *Decision fusion* in which the classifier combines the decisions of the component classifiers for each measurement. From a purely decision theoretic viewpoint, the choice between data versus decision fusion depends on the statistical relation between the different measurements, as elaborated next.

Data Fusion. If the different measurements yield correlated information, data fusion is needed in general for best performance. Suppose that, for any given class, the different measurements are jointly Gaussian. That is, for class j, the concatenated feature vector $\mathbf{x}^c = \left[\mathbf{x}^T{}_1, \cdots, \mathbf{x}^T{}_K\right]^T$ is characterized by the mean vector $\boldsymbol{\mu}_j^c$ and the covariance matrix $\mathbf{S}_j^c$. The ML classifier based on data fusion can then be designed and tested in the same way as the single-measurement classifier described in Section IV-A by using concatenated feature vectors.

Decision Fusion. If the different measurements are statistically independent, the likelihood function factors as $P(\mathbf{x}_1, \cdots, \mathbf{x}_K \mid w_j) = \prod_{k=1}^{K} P(\mathbf{x}_k \mid w_j)$ which suggests combining the *decisions* of the component classifiers (for different measurements) to make the final decision. Either hard or soft decisions may be combined [18, 19]. We limit our discussion to soft decision fusion. There are a variety of possibilities for decision fusion, all of which stem from successive bounds for the factored likelihood [19]. The *sum rule* is robust [19]: $C(\mathbf{x}_1, \cdots, \mathbf{x}_K) = \arg \max_{j=1,\cdots,M} \left[\sum_{k=1}^{K} P(\mathbf{x}_k \mid w_j)\right]$. Other rules based on the median, minimum or maximum of the component likelihoods may also be used. Multiple measurement classifiers based on decision fusion can be designed from training data by estimating the mean vectors and covariance matrices for component classifiers as described in Section IV-A.

C. *Different Forms of CSP in Sensor Networks*

We now discuss the application of decision and data fusion of multiple measurements in sensor networks. As mentioned earlier, multiple measurements may be from different modalities at a single node or from different modalities at different nodes. We consider a single cell consisting of P nodes. Each node can sense in K different modalities. Let $\mathbf{x}_{p,k}$ denote an event feature vector for the k-th modality at p-th node, and let $C_{p,k}$ denote the corresponding component classifier. For concreteness, we consider K=2 modalities and P=3 nodes with the third node being the manager node. Let C denote the overall CSP classifier at the manager node. We discuss various CSP classifiers in the order of increased communication and computational burden on the network.

Single Node Multiple Modality (SN, MM). This is the simplest form of CSP since it is limited to data in multiple modalities at a single node (no communication burden). The final classifier takes the form $C(C_{1,1}(\mathbf{x}_{1,1}), C_{1,2}(\mathbf{x}_{1,2})) \to m$ for decision fusion and $C(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}) \to m$ for data fusion. The latter imposes a higher computational burden since it involves KN dimensional joint processing as opposed to N dimensional component processing in the former.

Multiple Node Single Modality (MN, SM). This form of CSP involves higher communication burden since data or decisions from P nodes are shared. The final classifier is of the form $C(C_{1,1}(\mathbf{x}_{1,1}), C_{2,1}(\mathbf{x}_{2,1}), C_{3,1}(\mathbf{x}_{3,1})) \to m$ for decision fusion and $C(\mathbf{x}_{1,1}, \mathbf{x}_{2,1}, \mathbf{x}_{3,1}) \to m$ for data fusion. Decision fusion entails communication of P-1 decisions to the manager node that jointly processes the P component decisions. Data fusion involves communication of N dimensional event feature vectors from P-1 nodes to the manager node that jointly processes the PN dimensional concatenated feature vector.

Multiple Node Multiple Modality (MN, MM). This is the most general form of CSP that entails the highest communication and computational burden. In this case, various forms of CSP are possible.

a) Decision fusion across modalities and nodes. The final decision is given by $C(C_{1,\bullet}, C_{2,\bullet}, C_{3,\bullet}) \to m$ where $C_{p,\bullet} \equiv C_{p,\bullet}(C_{p,1}, C_{p,2})$ denotes the component decision at p-th node formed by fusing the decisions of classifiers for the K (=2) modalities at that node. This sub-case entails the least communication and computational burden since only decisions need to be communicated to and processed by the manager node.

b) Data fusion over modalities and decision fusion over nodes. The final decision is given by $C(C_{1,\bullet}, C_{2,\bullet}, C_{3,\bullet}) \to m$ where $C_{p,\bullet} \equiv C_{p,\bullet}(\mathbf{x}_{p,1}, \mathbf{x}_{p,2})$ denotes the component decision at p-th node formed via data fusion over modalities at that node. Compared to the last sub-case, this one entails higher computational burden at individual nodes. One possibility, intermediate to the above two sub-cases, is in which data fusion is performed over nodes in modality 1 and decision fusion in modality 2. The final decision is given by $C(C_{\bullet,1}(\mathbf{x}_{1,1}, \mathbf{x}_{2,1}, \mathbf{x}_{3,1}), C_{\bullet,2}(C_{1,2}, C_{2,2}, C_{3,2})) \to m$.

c) *Data fusion over modalities and nodes.* The final decision is $C(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{3,1}, \mathbf{x}_{3,2}) \to m$ which entails the highest communication and computational burden since K, N-dimensional event feature vectors are communicated from each of the P-1 nodes to the manager node that jointly processes the final PKN dimensional concatenated event feature vector. Note that if the measurements at different nodes and in different modalities are independent, this sub-case reduces to a).

Numerical Results. We briefly present some (MN, SM) numerical results using data collected in field experiments of the DARPA SenseIT program. The results are based on N=50 dimensional FFT features derived from acoustic measurements. Classification between wheeled versus tracked vehicles is performed. The tracked data corresponded to Amphibious Assault Vehicle (AAV) whereas the wheeled data corresponded to Dragon Wagon (DW) and Humvee (HV) vehicles. Concatenated and component covariance matrices for the two classes were estimated at three nodes within a cell from training data collected during the experiments. Due to limited training data, synthetic test data for the three nodes was then generated using the eigenvalue decomposition of the estimated correlation matrices and white Gaussian background noise was added to yield an SNR of 20dB. This experiment tests the ability to classify the vehicles based on second-order statistical information in the available data. The confusion matrix for the single node classifier (that operated on 50 dimensional feature vectors) is:

SN	Dec. = wheeled	Dec.=tracked
Class=wheeled	337	163
Class= tracked	120	380

which yields PD = 0.67, 0.76 and PFA = 0.24, 0.32 for the two classes (Average PD = 0.72). The confusion matrix for the data fusion classifier is:

MN – data fusion	Dec. = wheeled	Dec.=tracked
Class=wheeled	396	104
Class= tracked	83	417

which yields PD = 0.79, 0.83 and PFA = 0.17, 0.21 for the two classes (Average PD = 0.81). The confusion matrix for the decision fusion classifier (sum rule) is:

MN – dec. fusion	Dec. = wheeled	Dec.=tracked
Class=wheeled	342	158
Class= tracked	63	437

which yields PD = 0.68, 0.87 and PFA = 0.13, 0.32 for the two classes (Average PD = 0.78). As evident, data fusion performed the best with decision fusion in between single node and data fusion. In particular, the decision fusion classifier performs nearly as well as the data fusion classifier but with significantly lower communication and computational burden. The data fusion classifier requires communication of 50 dimensional vectors from each node to the manager node, compared to the communication of scalars (decisions) in decision fusion. Furthermore, the data fusion classifier computes 150 dimensional quadratic forms of the concatenated feature vector, whereas the decision fusion classifier simply uses the sum of the three scalar decisions. We direct the readers to [7] for the performance of other types of classifiers on real data.

Pros and Cons of Data versus Decision Fusion

1. Decision fusion is preferable due to lower communication and computational burden. It also requires lesser amount of data for training. This is particularly important when limited training data is available as it enables more accurate estimation of classifier parameters (covariance matrices).
2. Data fusion can potentially yield the best performance at the cost of higher communication and computational burden if measurements are sufficiently correlated.
3. Data fusion across modalities (no communication burden) and decision fusion across nodes is attractive.
4. Decision and/or data fusion may not yield sufficient improvement in performance if inconsistencies between multiple measurements are present, such as due to malfunctioning nodes. Some recent results indicate that decision fusion might perform better in such a fault-tolerant context [20].
5. Measurements yielding complementary performance should ideally be combined. For instance, modalities M1 and M2 may both be effective for classifying AAV versus DW but not AAV versus HV, whereas modality M3 may be useful for classifying AAV versus HV. Combining M1 and M3 (or M2 and M3) would likely be more beneficial than combining M1 and M2.

In general, measurements from different nodes within a cell will exhibit a combination of dependent (correlated) and independent (uncorrelated) components. The optimal classifier performs data averaging over the correlated components to improve the effective signal to noise ratio (SNR) and decision averaging over uncorrelated measurements to reduce the inherent statistical variation in the signal. Some recent work shows that for targets modeled as zero-mean stochastic (Gaussian) signals, the decision fusion classifier incurs a relatively small loss in effective SNR compared to the optimal classifier even in the presence of correlated measurements [21]. Thus, the decision fusion classifier, which is clearly the attractive choice in view of the communication and computational burden, is also a robust choice from a decision theoretic viewpoint.

V. Issues and challenges

We presented CSP methods for target classification and tracking in distributed sensor networks. These algorithms exploit multiple sensing modes gathered at different nodes. Significant savings are possible in power and bandwidth consumption by processing time series locally. Significant information can be distilled from the time series. Location aware routing limits data distribution to regions directly affected by the data. Results based on field tests show these approaches are feasible. Further research is needed to determine the operational limitations of these approaches.

As CSP techniques often rely on prior statistical information about the signals, an overriding challenge is to make CSP algorithms robust and/or adaptive to variations in environmental conditions that can significantly influence statistical signal characteristics [7]. For example, the presence of a strong wind can radically influence acoustic measurements. Similarly, different vehicle operating conditions, such as gearshifts and acceleration must also be taken into account. Finally, the effect of Doppler shifts can also be quite pronounced in acoustic and seismic measurements due to the relatively slow speed of wave propagation in such modalities [7].

The choice between decision versus data fusion depends on the statistical correlation between different measurements. Thus, algorithms for determining the subset of nodes for data versus decision fusion could significantly enhance the efficiency of CSP algorithms. One simple approach may be based on the observation that feature vectors from different nodes provide snapshots of the target signal at different times. Thus, nodes in close proximity will be highly correlated, whereas sufficiently spaced nodes will be weakly correlated. A simple measure of the degree of correlation between nodes could be derived from the knowledge of the bandwidth of the target signal and the location of the nodes relative to the target (e.g., a stationary stochastic signal decorrelates after a time interval inversely proportional to its bandwidth). Recent work on a related topic is reported in [22].

Tracking results indicate that using laterally inhibited distributed tracking is currently about as efficient as centralized tracking in network resource consumption. Lateral inhibition is simpler computationally and scales better. In large-scale networks it is likely to be the better alternative. Work still needs to be done on optimizing packet and cell sizes. Work is also needed to fully realize the ability of the distributed system to support target classes with different dynamics and maintain multiple track hypotheses [23].

Finally, the classification and tracking algorithms presented here primarily apply to a single target or multiple targets that are separated sufficiently in space and/or time. Tracking multiple closely spaced targets is a challenging problem that relies on classification algorithms. Single-target classification algorithms can be extended to deal with multiple targets. A key problem is the interference between signals from different targets. In a multiple target classifier, each component classifier for a particular target class must also suppress interference from targets from other classes. Subspace-based methods may be leveraged in this context (see, e.g., [24] and references therein).

VI. Acknowledgments and Disclaimer

Dr.s Brooks and Ramanathan's participation was partially supported by the Defense Advanced Research Projects Agency (DARPA), and administered by the Army Research Office under Emergent Surveillance Plexus MURI Award No. DAAD19-01-1-0504. Profs. Ramanathan and Sayeed's work was partially supported by the DARPA SenseIT program under grant no. F30602-00-2-0555. Dr. Brooks' efforts also partially sponsored by the Defense Advance Research Projects Agency (DARPA) Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-99-2-0520 (Reactive Sensor Network).

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory (AFRL), and Army Research Office (ARO).

VII. References

[1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, Instrumenting the world with wireless sensor network, Proc. ICASSP'2001, Salt Lake City, UT, 2001, pp. 2675-2678.

[2] J. Agre and L. Clare, An Integrated architecture for cooperative sensing networks, Computer, vol. 33, pp. 106-108, May 2000.

[3] S. Kumar, F. Zhao and D. Shepherd, Eds., Special Issue on Colllaborative Signal and Information Processing in Microsensor Networks, IEEE Signal Processing Magazine, March 2002.

[4] D. L. Hall and J. Llinas, "An Introduction to Multisensor Data Fusion," Proc. IEEE, vol. 85, no. 1, pp. 6-23, Jan 1997.

[5] M. Liggins II, C-Y Chong, I. Kadar, M. G. Alford, V. Vannicola, and S. Thomopoulos, "Distributed Fusion Architectures and Algorithms for Target Tracking," Proc. IEEE, vol. 85, no.1, pp. 95-107, Jan 1997.

[6] B. Dasarathy, "Sensor Fusion Potential Exploitation – Innovative Artchitectures and Illustrative Applications, Proc. IEEE, pp. 24-38, Jan. 1997.

[7] D. Li, K. Wong, Y. Hu and A. Sayeed. (2002) Detection, Classification, Tracking of Targets in Micro-sensor Networks, IEEE Signal Processing Magazine, pp. 17-29, March 2002.

[8] P. Ramanathan, K.-C. Wang, K. K. Saluja, and T. Clouqueur, "Communication support for location-centric collaborative signal processing in sensor networks," *Proc. of DIMACS Workshop on Pervasive Networks*, May 2002.

[9] J. Heidemsnn, F. Silva, C. Intanagonwiwat, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," *Proc. Sym. on Operating Sys. Princ.*, pp. 146-159, Oct. 2001.

[10] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed energy conservation for ad-hoc routing," *Proc of Mobicom*, July 2001.

[11] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*,, Academic Press, Boston, 1988.

[12] D. L. Hall, *Mathematical Techniques in Multisensor Data Fusion*, Artech House, Boston, 1992.

[13] D. S. Friedlander and S. Phoha, "Semantic Information Fusion for Coordinated Signal Processing in Sensor Networks," *Int. J. High Performance Computing Applicat*. vol. 16, no. 3, pp.235-242, Fall 2002

[14] R. Brooks, C. Griffin, and D. Friedlander, "Self-organized distributed sensor network entity tracking," *Int. J. High Performance Computing Applicat* vol. 16, no. 3, pp.207-220, Fall 2002.

[15] R. R. Brooks and S. S. Iyengar, *Multi-sensor Fusion: Fundamentals and Applications with Software*, Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[16] J. Chen, K. Yao, and R. Hudson, Source Localization and Beamforming in Microsensor Networks, IEEE Signal Processing Magazine, pp. 30-39, March 2002.

[17] P. Boettcher and G. Shaw, A distributed time-difference of arrival algorithm for bearings-only target localization, in Proc. 4th Int. Conf. Information Fusion, pp. TuC3-9-TuC3-14, Montreal, CA, August 2001.

[18] R. Duda, P. Hart, and D. Stork, Pattern Classification, 2nd Edition, Wiley, 2001.

[19] J. Kittler, M. Hatef, R. Duin, J. Matas, On Combining Classifiers, IEEE Trans. Pattern Anal. Machine Intelligence, vol. 20, no. 3, pp. 226-238, March 1998.

[20] [T. Clouqueur, P. Ramanathan, K. Saluja and K-C. Wang, Value-Fusion Versus Decision-Fusion for Fault-tolerance in Collaborative Target Detection in Sensor Networks, 4th Int. Conf. Information Fusion , Montreal , CA.

[21] A. D'Costa and A. M. Sayeed, "Data Versus Decision Fusion in Wireless Sensor Networks," to be presented at ICASSP 2003.

[22] F. Zhao, J. Shin, and J. Reich, Information-Driven Dynamic Sensor Collaboration, IEEE Signal Processing Magazine, pp. 61-72, March 2002.

[23] J. Moore, T. Keiser, R. R. Brooks, S. Phoha, D. Friedlander, J. Koch, A. Reggio, and N. Jacobson, "Tracking Targets with Self-Organizing Distributed Ground Sensors," 2003 IEEE Aerospace Conference, Invited Paper, March 2003.

[24] J. Kittler, A method for determining class subspaces, Information Processing Letters, pp. 77-79, June 1977.