Exploiting Sensing Diversity for Confident Sensing in Wireless Sensor Networks

Matthew Keally, Gang Zhou, Guoliang Xing[†], Jianxin Wu[‡]
College of William and Mary, [†]Michigan State University, [‡]Nanyang Technological University

***Abstract*—Wireless sensor networks for human health monitoring, military surveillance, and disaster warning all have stringent accuracy requirements for detecting or classifying events while maximizing system lifetime. We define meeting such user accuracy requirements as *confident* sensing. To perform confident sensing and reduce energy, we must address sensing diversity: sensing capability differences among heterogeneous and homogeneous sensors in a specific deployment. We are among the first to explore the impact of sensing diversity on sensor collaboration, exploit diversity for sensing confidence, and apply diversity exploitation for confident sensing coverage. We show that our diversity-exploiting confident coverage problem is NP-hard for any specific deployment and present a practical solution, Wolfpack. Through a distributed and iterative sensor collaboration approach, Wolfpack maximizes a specific deployment's capability to meet user detection requirements and save energy by powering off unneeded nodes. Using real vehicle detection trace data, we demonstrate that Wolfpack provides confident event detection coverage for 30% more detection locations, using 20% less energy than a state of the art approach.**

I. Introduction

Many applications for body sensor networks [1], sensing coverage [2], and classification [3] all have stringent user accuracy requirements and demand long system lifetimes. A system with wearable sensors for fall detection [4] in the elderly must be extremely accurate, lest an injury-causing fall go undetected as a false negative. Similarly, an acoustic sensor network used to localize enemy snipers on a battlefield [5] may misclassify a friendly soldier's gunshot as an enemy sniper, with such a false positive resulting in a friendly fire incident. Consequently, the need exists for *confident* sensing, where the user's desired false positive and false negative rates are met for event detection, classification, or coverage.

In providing confident sensing, we must first address sensing diversity. Sensing diversity encompasses the sensing capability differences between different sensors of the same modality as well as among different modalities. The causes of sensing diversity can be linked to the in-situ reality of a specific deployment [6] as well as hardware differences [7], especially in the case of cheap off-the-shelf motes.

Many works ignore sensing diversity entirely and assume all sensors have similar sensing capabilities [2] [8]. Other works attempt to overcome sensing diversity by correcting for the differences in readings from different sensors [7]. Instead, we take advantage of such sensing differences to provide sensing confidence. Several challenges exist in providing confident sensing through exploitation of sensing diversity:

- Learning Sensing Diversity. Machine learning techniques, many of which can be utilized to learn the sensing capabilities of a deployment, greatly differ in terms of accuracy and complexity. A learning method for energy constrained sensor networks should provide enough accuracy to meet user requirements but still require low computation and communication overhead.
- On-demand sensor collaboration. In many deployments, individual sensors are sufficient to provide confident event detection, and hence collaboration is not needed. Through learned sensing diversity, it is important to determine when single sensors are sufficient and when collaboration is necessary. When sensor collaboration is neeeded, it is also important to determine the right sensors to collaborate, for such careful collaboration can save valuable computation and communication resources.
- Distributed and online diversity exploitation. Distributed learning and exploitation of sensing diversity allows for decreased bandwidth and energy usage as well as increased scalability. Furthermore, a distributed scheme allows for more efficient adaptation to environmental dynamics, for only portions of the network that do not meet user requirements need to be updated during runtime to ensure these requirements are met.

In this paper, we first capture and explore sensing diversity with a sensor network deployment for vehicle detection. We show that sensing capabilities greatly differ among sensors in a real deployment and identify when sensor collaboration is needed. When collaboration is needed, we show that arbitrary sensor collaboration often fails to meet user accuracy requirements, not to mention a joint consideration of accuracy and energy efficiency. We explore different machine learning techniques for on-demand collaboration and identify one appropriate for energy and computationally constrained sensor networks. These results provide key insights into protocol design for collaborative sensing.

We are among the first to take advantage of explored sensing diversity to provide sensing confidence and apply this approach for confident sensing coverage, such as in a fixed sensor network deployment for vehicle detection. We formally define and theoretically analyze our confident coverage problem when diversity is explored. We show that a specific case of our diversity-exploiting confident coverage problem is NP-hard

This work was supported in part by NSF grants ECCS-0901437, CNS-0916994, and CNS-0954039.

and propose *Wolfpack*, a distributed event detection framework that exploits sensing diversity for use in practical deployments. With machine learning, Wolfpack determines only the sensing capability needed to meet user detection requirements and save energy, only collaborating sensors when individual sensors are not accurate enough. During runtime, Wolfpack adapts its detection capability to adjust for environmental changes that cause a drop in accuracy and run the risk of not meeting the user requirements. Our main contributions are:

- We explore the fundamental challenges in addressing sensing diversity and its impact on collaboration for confident sensing using two different machine learning techniques.
- Through theoretical analysis and our practical Wolfpack design, we exploit sensing diversity to provide sensing confidence and apply it to sensing coverage.
- Our evaluation in a vehicle detection application demonstrates that Wolfpack achieves confident coverage for 30% more locations while using 20% less energy than a state of the art approach.

This paper is organized as follows: We present related work in Section II and explore sensing diversity in Section III. We formally define our confident coverage problem with theoretical analysis in Section IV and present our Wolfpack confident coverage design for practical system deployment in Section V. We analyze the performance of Wolfpack in Section VI, and present conclusions and future work in Section VII.

II. Related Work

Some works ignore both sensing confidence and diversity. These include k-coverage approaches [8] [2] [9] [10] [11] that rely on k nodes to be awake within the sensing range of a target location. In [12] [13], multiple modalities collaborate to detect events along with a sleeping scheme to save energy. Similarly, disc-based sensing models [14] [15] [16] [17] [18] address neither sensing confidence nor diversity.

Other works attempt to meet user accuracy requirements through theoretical modality-specific sensing models and data fusion-based [19] collaboration. Specific sensing models for coverage with acoustic, seismic, and infrared sensors are presented in [3]; models also exist for structural health monitoring with accelerometers [20] as well as camera-based event detection [21]. Signal attenuation-based models are described in [22] [5] [23] which give false positive and false negative rates for a given modality and training data set, allowing for data fusion between a cluster of sensors. Another collaboration scheme using a signal attenuation model [24] incorporates a sleeping scheme to save energy. A noise distribution model is used for event detection in [25] and [26]. However, modality-specific sensing models adopted in these works must train each modality individually, making heterogeneous collaboration difficult. Furthermore, such models often rely on sensing assumptions that do not account for reality.

Some approaches attempt to address sensing diversity by accounting for sensing differences in different sensors but cannot provide sensing confidence. This includes works [6] [27] that use a similarity metric to ensure enough nodes are awake within the sensing range of a target location as well as those [7] that calibrate sensors based on differences in their readings. Some approaches use machine learning to provide collaboration [28] [29] [30] [31] [32] [33] [34], but these works do not fully explore the effects of sensing diversity on collaboration nor do they provide sensing confidence. Other work [1] addresses sensing diversity in providing sensing confidence but does not provide lightweight and decentralized collaboration appropriate for wireless sensor networks.

III. Exploring and Exploiting Sensing Diversity

We explore how to take advantage of sensing diversity and on-demand sensor collaboration to provide confident sensing. We make use of the Wisconsin SensIT vehicle detection trace data [28], with 23 nodes deployed along a road with each node containing an acoustic, seismic, and infrared sensor. Vehicles make 20 passes along a road through the network with ground truth provided via a GPS trace. Trace data of raw sensor energy is provided for each sensor at a sampling rate of up to 4960Hz. We provide this unmodified real sensor data and ground truth as input to a trace-driven wireless sensor network simulation run on a PC. While the sensor data and ground truth is real, we simulate communication behavior and assume each node is a low power mote-class device equipped with an 802.15.4 radio, such as the Crossbow IRIS [35]. While we are aware that radio communication is often lossy in wireless sensor networks, we focus on sensing accuracy, not communication quality, and assume reliable communication.

We use the trace sensor data sampled at 100ms intervals and a total trace length of 6763 intervals. We classify sensor and sensor cluster readings into events when a vehicle is detected and non-events when no vehicle is present. To learn sensing diversity, collaborate sensors, and perform event detection, we choose machine learning, which can address the complexity and heterogeneity of sensor data. We first identify sensing diversity among sensors of the same modality as well as with different modalities. Next, we illustrate the effects of sensing diversity on collaboration, determining when and how to collaborate sensors such that user requirements can be met. Finally, we compare two different machine learning techniques for learning sensing diversity and locate one appropriate for low power sensor networks.

A. Identifying Sensing Diversity

In this subsection, we use k-means clustering [36] with $k = 2$ classifications: trace data for each sensor is clustered into mean event and non-event centroids. As vehicles pass through the deployment area, each sensor reading is classified by determining its closest centroid. Using vehicle location ground truth and classified data, we plot the sensing range for all acoustic, seismic, and infrared sensors in Figure 1. Since sensing diversity encompasses sensing capabilities within a specific deployment, accounting for in-situ reality, some sensors have a sensing range of 0m, indicating that the vehicle does not pass close enough to the sensor to be detected.

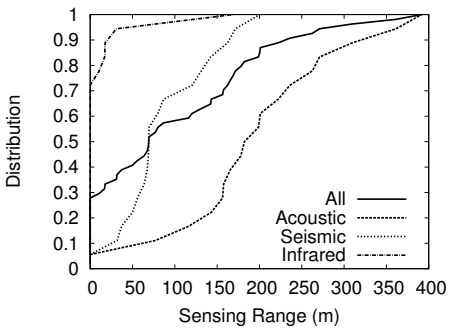

Fig. 1: Sensing range differences in Wisconsin deployment.

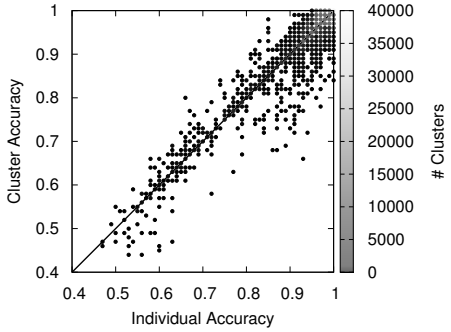

Fig. 2: Nearest Centroid: cluster accuracy and best individual member accuracy.

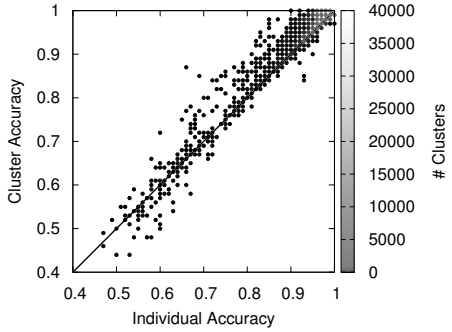

Fig. 3: Fisher's Linear Discriminant: cluster accuracy and best individual member accuracy.

Diversity within the same modality. Figure 1 demonstrates that sensors of the same modality experience significant differences in event detection performance. For example, 10% of acoustic sensors can detect the vehicle at 400m, while another 10% can only detect vehicles at ranges up to 50m. Similarly, 10% of seismic sensors have a range of 40 meters or less while 10% have a range greater than 200m. Similar differences can be observed for infrared sensors. This diverseness in sensing capability can be linked to the quality of the sensor itself as well as the properties of the local environment such as terrain, weather, and other obstacles [6]. Due to sensing diversity, a single sensor may even perform differently in different environments. However, these observations are largely ignored in traditional sensing approaches. In [1], it is demonstrated that traditional sensing approaches, such as sensing coverage, use too little or too much sensing capability to detect events and either exhibit poor accuracy or waste energy. Other approaches such as [14] [8] also do not account for sensing diversity with respect to individual sensors, and thus fail to provide sufficient accuracy to meet user requirements.

Diversity among different modalities. Figure 1 also illustrates the differences between different sensing modalities. The sensing range of acoustic sensors is extremely varied, with 50% of acoustic sensors exhibiting a sensing range of at least 200m, with a miniumum and maximum range of 0 and 400m, respectively. Conversely, infrared sensors have little variance, with 95% of sensors exhibiting a sensing range of 30m. Many existing detection coverage approaches [37] [23] rely on modality-specific sensing models, making sensor collaboration difficult in heterogeneous deployments. Therefore, significant sensing diversity exists in real deployments which should be addressed or exploited in confident sensing.

B. Impact of Diversity on Collaboration and Accuracy

In our study of sensor collaboration and accuracy, we compare Nearest Centroid, a variant of k-means clustering, with another learning technique, Fisher's Linear Discriminant [36]. Both techniques allow for sensor readings to be combined into tuples for cluster-based collaboration and classification. In Nearest Centroid, sensor data is used to form two classification centroids, one for events and one for non-events, except that unlike k-means, ground truth is used in centroid formation. Like k-means, new data is classified by determining the closest centroid. While Nearest Centroid assumes that each reading in a tuple of sensor cluster data is independent, Fisher's Linear Discriminant attempts to capture the dependencies among different sensors in a cluster. For example, sensors in proximity to each other usually yield correlated readings. Fisher's Linear Discriminant attempts to find a projection (i.e. linear combination) of sensor cluster readings that maximizes the separation of event readings from non-event readings.

Using the vehicle detection trace, we select 103 detection locations with a 10m radius throughout the deployment area along a road on which vehicles pass. We detect vehicles at these target locations and classify such sampling intervals as events. For each target location, we form random clusters of size 2 through 25 with up to 30 clusters of each size, using sensors within 100m of each target location. We perform classification with both Nearest Centroid and Fisher's Linear Discriminant and use ground truth for each individual sensor or cluster reading to determine accuracy. For each generated cluster we compare the detection accuracy of the best individual member sensor as a singleton cluster to the generated cluster accuracy and plot the results in Figures 2 and 3, where darker points indicate fewer than 50 overlapping clusters.

1) On-Demand Collaboration: We derive the guidelines for when sensor collaboration is needed and when it is not. First, we demonstrate that when an individual sensor meets the user detection requirements for a target location, collaboration is unnecessary. Using Figures 2 and 3, we show that in over 36,000 cases for both Fisher's Linear Discriminant and Nearest Centroid, individual sensors have perfect accuracy. Individual sensors have over 95% accuracy in 2,000 more cases. In such cases where the user requirements fall in this 5% points acccuracy range, collaboration is not needed for the requirements can be met by a single sensor and valuable communication and computational overhead is saved.

When an individual sensor cannot meet the user requirements for a given target location, collaboration is needed, but there are individual sensors that can be excluded from the collaboration process to reduce the search space size. In Figures 2 and 3, with this specific deployment and trace data, it is clear that no individual sensor can boost cluster accuracy as a cluster member by more than 20% points (We define this exclusion boundary as the sensitivity threshold). These sensors can be excluded from detection and collaboration, for

any cluster consisting entirely of sensors below this threshold will not meet the user detection requirements.

Lastly, we show that when a sensor has a detection accuracy above the sensitivity threshold, but below the user requirements, it is a candidate for collaboration in a sensor cluster. Figures 2 and 3 depicts over 2,500 cases where all individual sensors in a cluster exhibit less than perfect accuracy but the cluster accuracy is equal to 100%.

TABLE I: Cluster vs. individual sensor performance.

Cluster Type	Nearest Centroid	Fisher's Linear Discriminant
Good	3894	4606
Bad	1424	412
Neutral	36919	37219

2) Collaboration Accuracy and Complexity of Learning: Now that we have determined when sensor collaboration is required, we analyze how to learn the sensing capabilities of sensors and sensor clusters. We first show that when collaboration is needed, we must cluster sensors carefully instead of randomly. In Figures 2 and 3, cluster performance from randomly generated clusters can be classified into three categories: "good", "bad", and "neutral", whose classification totals can be found in Table I. Good clusters perform better than their best individual member sensors, bad clusters perform worse than their best individual member sensors, and neutral clusters perform the same as their best individual member sensors. It is clear that if the right sensors are not chosen for collaboration, cluster performance can suffer, for over 1,800 clusters exhibit worse performance than the best individual member sensor. However, clustering carefully can yield a cluster that performs accurately and better than the best individual member sensor.

We next compare two machine learning collaboration techniques, Nearest Centroid and Fisher's Linear Discriminant and demonstrate that Fisher's Linear Discriminant has better sensor collaboration performance. Figures 2 and 3 and Table I show that Fisher's Linear Discriminant is able to eliminate most of the "bad" clusters and increase the number of "good" clusters since it accounts for the data dependencies among individual sensors in the cluster. While Fisher's Linear Discriminant improves clustering performance by eliminating many bad clusters, its greater computational requirements make it unattractive for use in sensor networks for distributed event detection, as most motes have microcontroller speeds of less than 10 MHz [35]. Given n training observations and a cluster size of k, Nearest Centroid incurs a training cost of $O(n \cdot k)$ while Fisher's Linear Discriminant incurs a much greater training cost of $O(k^3 + nk^2)$. Although, from Figure 2, Nearest Centroid has many more "bad" clusters than Fisher's Linear Discriminant, its best performing clusters have comparable results to that of Fisher's Linear Discriminant. By collaborating sensors wisely, we can use Nearest Centroid to classify events in a distributed, node-centric fashion, achieving high performance and low complexity.

IV. Diversity-Exploiting Confident Coverage

Confident sensing through exploitation of sensing diversity can apply to many problems such as assisted living [4] or military surveillance [5]; in this paper we apply confident sensing to sensing coverage for vehicle detection in a static wireless sensor network. Here, we define our diversity-exploiting confident coverage problem and show that it is NP-hard. To provide confident coverage, a set of sensor clusters must be found that can meet user detection accuracy requirements for all desired detection locations in an energy conscious manner. We first define a set of nodes $N = \{n_1, ..., n_n\}$. Each node $n_j \in N$ contains k_j sensors, forming the set S of all sensors: $S = \{s_1^1, s_1^2, ..., s_1^{k_1}, s_2^1, s_2^2, ..., s_2^{k_2}, ..., s_n^1, ...s_n^{k_n}\}$, where s_j^i is the ith sensor on node j. We also define a set of detection locations $L = \{l_1, ..., l_l\}$ which a user wishes to cover.

Users can specify the accuracy of detection for all locations in L in terms of desired false positive and false negative rates, ufp, ufn, respectively. With machine learning, cluster C_i of one or more sensors can quantify its sensing diversity by determining its false positive rate and false negative rate for each location l_k: $fp(C_i, l_k)$ and $fn(C_i, l_k)$. If cluster C_i meets the user requirements ufp and ufn, for location l_k, we say that location l_k is *covered* by C_i.

A deployment has a set of possible sensor clusters $C = \{C_i | C_i \subseteq S\}$. With the learning techniques we have discussed, we can quantify cluster detection capabilities through a function $f : C_i \rightarrow 2^L$ that maps a cluster $C_i \in C$ to a subset of locations in L indicating the coverage of the locations by the sensors in C_i. In this section, we assume that possible covering clusters are already generated and focus only on cluster selection for energy savings. In Section V, we describe our combined cluster generation and selection process.

Our goal is to find the set of clusters $C^* \subseteq C$ that meets the user requirements for all locations while residing on the fewest number of nodes, hence using the least amount of energy due to active node power consumption. We need to find a set of clusters $C^* \subseteq C$ such that all locations in L are covered,

$$\bigcup_{C_i \in C^*} f(C_i) = L \quad (1)$$

subject to minimizing the total number of nodes contained by the clusters in C^*:

$$\text{minimize } |\{n_j \in N\}| \text{ where } s_j^k \in C_i \text{ for some } C_i \in C^* \text{ and some } k \quad (2)$$

We now demonstrate that our cluster selection problem is NP-hard by showing that a special case of our problem is in fact the known NP-hard Set Cover problem [38]. In the special case, we assume that each node $n \in N$ has only one sensor. That is, the set of all sensors in the deployment is represented as $S = \{s_1, s_2, ..., s_n\}$, where a sensor s_i is the only sensor on node i. We also assume that the set of possible clusters C contains only clusters with one sensor: $C = \{\{s_i\} | s_i \in S\}$. Using the mapping of clusters to covered locations, $f : C \rightarrow 2^L$, the set of all possible clusters, C, is equivalent to: $C = \{L' | L' \subseteq L\}$, where C represents a collection of subsets of L. In this instance, the optimization problem can be rewritten to find the set of clusters $C^* \subseteq C$ such that:

$$\bigcup_{L' \in C^*} L' = L \quad (3)$$

subject to minimizing the number of clusters in C^*. This special case is equivalent to the Set Cover problem, demonstrating that the general case of our clustering problem is also NP-hard. Since we wish to solve our confident coverage problem in a distributed manner, we plan to formally define a greedy solution and derive an approximation ratio in future work.

V. Wolfpack Framework Design

In this section, we propose Wolfpack, a practical, distributed solution to our confident coverage problem defined in Section IV. For each defined detection location, Wolfpack assigns a sensor cluster which meets the user detection requirements; these sensor clusters are formed in parallel at the start of deployment and updated when needed during runtime. While many clustering schemes exist, such as leader election [3], Wolfpack clusters sensors on nodes based on their sensing capabilities and only clusters the sensors needed to meet the user requirements, placing unused nodes to sleep. A cluster is formed for each detection location by incrementally adding a member node and one or more of its sensors until the user requirements are met. Nodes are added to a cluster in decreasing order of the learned detection capability of their sensors. In some cases, a single sensor residing on a single node may be enough to meet the user requirements. We now describe the diversity aware clustering process in Section V-A and how clusters can be adaptively updated during runtime in Section V-B if they fail to meet user detection requirements.

A. Distributed Diversity-Aware Clustering

Nodes first quantify their sensing diversity and then compete to declare themselves as cluster heads, with the nodes that have the most sensing capability winning the competition. If a cluster formed solely from a cluster head node is not enough to meet the user requirements for its target location, the most capable member nodes join the cluster one at a time until the user requirements are met.

For event detection training and cluster formation, each active node maintains a history of recent observations for all of its sensors. Each node also maintains an application-level feedback mechanism, such as a vehicle tracking application, to provide event ground truth in a manner similar to [6]. Since we demonstrate in Section III that communication range of many sensor motes [35] is at least twice that of the sensing range, we cluster sensors within one communication hop of each detection location (fusion range) to save bandwidth and energy resources. We now describe the details of our distributed clustering scheme from a node and event-driven perspective.

Exploring and quantifying diversity. Using Nearest Centroid, each node n_j explores its sensing diversity by training singleton clusters with each of its sensors s_j^m for each of the locations l_i in its fusion range. For each trained sensor and location, n_j determines the detection false positive and false negative rates $fp(s_j^m, l_i)$ and $fn(s_j^m, l_i)$. A sensor s_j^m is *sensitive* to location l_i, or $sensitive(s_j^m, l_i)$, if its false positive and false negative rates fall within 20% of the user requirements. This sensitivity threshold is an empirical rule that may vary with different deployment scenarios, for our choice of threshold is due to our study in Section III which shows that no individual sensor can improve detection accuracy by more than 20% when added to an existing sensor cluster.

Algorithm 1 Event Handler: Backoff Timer Fires

Input: Node n_j, sensitive locations L_S for node n_j
Output: Cluster head or cluster member declaration for n_j

```
for all l_i ∈ L_S do
  if No cluster exists for l_i then
    Create cluster C_i
    Set n_j as cluster head
    Compute fp(C_i, l_i) and fn(C_i, l_i)
  else
    Add n_j to existing cluster C_i
  end if
  Broadcast the following in a packet:
    Cluster C_i covers l_i with fp(C_i, l_i) and fn(C_i, l_i)
  if fp(C_i, l_i) > ufp or fn(C_i, l_i) > ufn then
    /*More collaboration is needed*/
    Broadcast observation history for cluster C_i
  end if
end for
```

Each node then determines how much each of its sensors can contribute towards meeting the user requirements for each sensitive location: Δfp_j and Δfn_j, which are real numbers between 0 and 1. Values closer to 0 indicate that the sensor s_m^j contributes very little towards meeting the user requirements for location l_i, while the maximum possible values $1 - ufp$ and $1 - ufn$ indicate that the user requirements are met. Δfp_j is defined in Equation 4 (Δfn_j is similar).

$$\Delta fp_j(s_j^m, l_i) = 1 - \max\{fp(s_j^m, l_i), ufp\} \quad (4)$$

A node quantifies its sensing diversity by calculating its *importance*, which is the sum of all contributions on a node for all of its sensors and sensitive locations. The more important a node, the more valuable it is towards meeting the user requirements for locations within its fusion range. More important nodes are more likely to have very capable sensors which will become members of many different clusters covering different detection locations. Each node sets a backoff timer based on its importance value to declare itself as a cluster head for its sensitive locations, where greater importance values result in shorter timers. Therefore, clusters can usually be formed from a small number of important nodes to cover all locations, thus reducing the number of active nodes needed to meet the user requirements. Importance is defined in Equation 5 as:

$$I(n_j) = \sum_{m=1}^{|S_j|} \sum_{i=1}^{|L_j|} \left(\Delta fp_j(s_j^m, l_i) + \Delta fn_j(s_j^m, l_i)\right) \quad (5)$$

Importance-based competition. When a cluster head timer fires on node n_j (Algorithm 1), node n_j declares itself as the head for all sensitive locations not yet declared covered by

another cluster head, creating a cluster C_i for each undeclared sensitive location $l_i \in L_S$. If a cluster member timer fires on node n_j, the node adds itself to an existing cluster C_i containing other member nodes. In both cases, the declaring node n_j adds to the cluster only its sensitive sensors that increase the Δfp_j and Δfn_j contribution towards meeting the user requirements for each location l_i. The declared cluster C_i is trained using Nearest Centroid learning and the observation history of all sensors in C_i.

Algorithm 2 Event Handler: Receive Declaration Packet

Input: Node n_j, sensitive locations L_S for node n_j, declaration cluster C_i for location l_i

Output: Node n_j sets member timer if user requirements are not met for C_i

```
if n_j ∉ C_i and l_i ∈ L_S then
    if n_j is competing to be a member covering l_i then
        Stop timer on node n_j for l_i
    end if
    if fp(C_i, l_i) > ufp or fn(C_i, l_i) > ufn then
        /*More collaboration is needed*/
        Update I(n_j) using Eqns. 5 and 6
        Set timer using I(n_j); n_j competes to join C_i
    end if
end if
```

After a node n_j has its backoff timer fire and it declares itself as a head or member, the node broadcasts a packet to all neighbors for each declared cluster C_i with the cluster false positive and false negative rates, $fp(C_i, l_i)$ and $fn(C_i, l_i)$ and the location the cluster covers, l_i. If a declared cluster C_i does not yet meet the user requirements ufp or ufn, more collaboration is needed by recruiting member sensors, so node n_j broadcasts its member sensor observation history to its neighbors to allow neighbors to compete to form a new cluster including observations from node n_j.

If a node n_j receives a cluster head or cluster member declaration packet for an existing cluster C_i and location l_i, node n_j may perform one of two actions (Algorithm 2). First, if node n_j is competing to become a member of cluster C_i, node n_j has lost the member competition to the broadcasting node and cancels its backoff timer. Second, if the declared cluster C_i does not meet the user requirements ufp or ufn, and n_j is sensitive to l_i, then n_j attempts to add itself as a cluster member. Node n_j determines its contribution towards meeting the requirements at l_i if it adds itself as a member of cluster C_i. For each of its sensitive sensors s_j^m, node n_j updates Δfp_j and Δfn_j as performed in Equation 6 and sets an importance-based backoff timer to compete with other nodes to join C_i based on Equation 5.

$$\Delta fp_j(C_i, s_j^m, l_i) = fp(C_i, l_i) - \max\{fp_j(C_i \cup \{s_j^m\}, l_i), ufp\} \quad (6)$$

Example. We present an example of the distributed clustering process in Figure 4. During initialization, in Figure 4 a), nodes n_1 and n_2 explore their diversity by determining which of their sensors are sensitive to each of the two locations, l_1 and l_2. From sensitivity and contributions towards meeting the user requirements Δfp_j and Δfn_j, node n_1 quantifies its diversity, calculating an importance value greater than that of node n_2, thus setting a shorter backoff timer. With its shorter backoff timer, node n_1 declares itself as the cluster head for all its sensitive locations: l_1 and l_2 in Figure 4 b), but the cluster C_2 for location l_2 does not meet the user requirements. Node n_2 sets a backoff timer to join C_2, where its timer fires and in Figure 4 c) the new C_2 meets the user requirements.

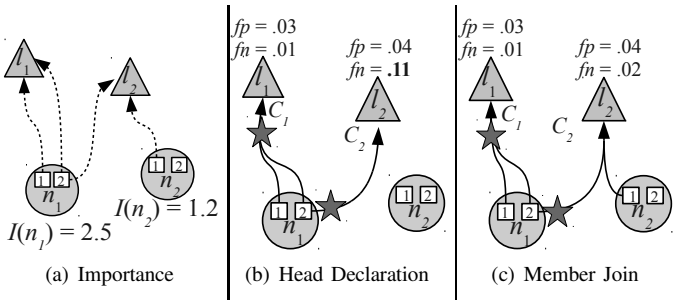

Fig. 4: Distributed clustering with two nodes n_1, and n_2, each with two sensors, two detection locations l_1 and l_2, and user requirements $ufp = ufn = 0.05$. n_1 has greater sensing capability and importance: it becomes the cluster head for C_1 and C_2 (indicated by stars). n_2 is then added as a member to C_2 since the user requirements are not met by n_1 alone.

B. Runtime and Adaptive Coverage

During runtime, after clusters have been formed to cover each location, cluster member nodes transmit sensor readings collected at each sample interval to their assigned cluster heads. Each cluster head makes a detection decision for each of its covered locations at each sample interval using its learned detection model and collected sensor data. To save bandwidth and transmission energy, we employ a scheme similar to [1], where a member node will only transmit sensor readings that are closest to the learned event centroid. A cluster head assumes the non-event centroid value if no transmission is received from a member sensor.

A cluster currently covering a location may experience a drop in performance, running the risk of not meeting the user detection requirements. Such a performance drop may be due to changes in background noise or to the properties of the event. In these cases, the existing cluster is dissolved and a new, more accurate cluster is formed in its place that meets the user requirements. Such an approach allows Wolfpack to adapt to environmental changes over time.

All cluster heads maintain moving accuracy using observation history and ground truth, allowing a cluster to detect a short term drop in $fp(C_i, l_i)$ and $fn(C_i, l_i)$. When such a performance drop is detected, the cluster head broadcasts an update message containing the location the cluster covers. Upon receiving the update message, all current cluster members stop transmitting samples to the cluster head. All nodes that have been awake long enough to have full observation histories compete to form a new cluster as in Section V-A.

A new, updated cluster may change with respect to the old cluster in three ways. First, the new cluster may consist of

the same exact sensors as the old cluster, but with new event and non-event centroids. Second, a newly formed cluster may also reside on the same nodes as the old but contain different sensors. Third, a new cluster may also reside on different nodes than the previous cluster. In future work, we will predict how a cluster changes during an update, reducing energy and computational overhead in generating a new cluster.

For each cluster that is formed to cover a location, a subset of nodes is selected as candidate nodes from among all sleeping and non-member sensors. Such candidate nodes remain awake and sample data so that during an update candidates may be selected to become member nodes if the current nodes cannot meet the user requirements.

VI. Evaluation

We evaluate our scheme using the Wisconsin SensIT vehicle detection trace data [28], with the same trace-driven simulation methodology described in Section III. We choose 79 detection locations within the deployment area along the road where vehicles pass: the first pass is used for initial training and the subsequent 10 passes are used for runtime detection. The vehicle path deviates slightly with each pass, creating environmental dynamics during runtime.

We compare our Wolfpack confident coverage approach to V-SAM [6], a state of the art coverage scheme, which in contrast to conventional coverage approaches, attempts to address sensing diversity by keeping awake sensors that sample similar data. For both Wolfpack and V-SAM, we set the user requirements to 5% for both false positive and false negative rates and set a 100m fusion range to collaborate sensors and to detect events at each location. For Wolfpack, we use Nearest Centroid for clustering and collaboration if not explicitly specified. For comparison, we illustrate the performance of Wolfpack with and without adaptive coverage (AC) and V-SAM with both similarity coverage and forced k-coverage, where 1 to 3 nodes are awake to cover each location. We also test V-SAM with each possible k of n decision fusion rule: at least k of n awake nodes within the fusion range must detect an event in order for V-SAM to detect an event.

A. Meeting User Requirements

In this section, we demonstrate that Wolfpack can meet the user requirements for nearly all detection locations, while V-SAM cannot. From Table II, Wolfpack with Adaptive Coverage exhibits 99.8% accuracy and meets 98.7% of the detection locations while the best V-SAM configurations have 94.8% accuracy and meet only 74.7% locations. In analyzing performance by detection location, Figure 5 demonstrates that Wolfpack has perfect accuracy for 85% of locations while V-SAM has much higher variance, with 25% of locations exhibiting less than 90% accuracy. In comparison with V-SAM, Wolfpack is able to quantify the sensing diversity in this deployment and choose the nodes with the most capable sensors to confidently detect events at each location. Conversely, V-SAM experiences poor accuracy, with false negative rates as high as 39.8% since it neither correctly learns the capabilities of each sensor nor collaborates carefully. V-SAM places too many nodes to sleep for lower k-coverage levels and fails to detect events while increasing coverage levels only slightly increases detection accurarcy.

TABLE II: Overall accuracy for Wolfpack and V-SAM.

	Acc. (%)	FP (%)	FN (%)	Loc. Met (%)
Wolfpack	**99.8**	**0.0**	**0.4**	**98.7**
Wolfpack, No AC	99.2	**0.0**	1.9	93.7
Wolfpack, Fisher's	**99.8**	**0.0**	0.6	96.2
Wolfpack, No Collab.	99.5	**0.0**	1.4	88.6
V-SAM, Sim-cov, 1/n	93.4	**0.0**	15.1	67.1
V-SAM, 1-cov, 1/n	94.2	**0.0**	13.0	68.4
V-SAM, 2-cov, 1/n	94.6	1.6	9.7	74.7
V-SAM, 2-cov, 2/n	92.2	**0.0**	19.0	59.5
V-SAM, 3-cov, 1/n	94.8	2.2	8.0	57.0
V-SAM, 3-cov, 2/n	92.7	**0.0**	17.8	60.8
V-SAM, 3-cov, 3/n	84.6	**0.0**	39.8	31.6

Table II shows that Wolfpack with Adaptive Coverage is able to increase the locations covered by 5% points due to adaptation to environmental dynamics during runtime. Furthermore, the table demonstrates that by collaborating carefully and choosing different sensors, Nearest Centroid classification is able to slightly outperform Fisher's Linear Discriminant by covering 2.5% points more locations. The table also demonstrates that in most cases, single sensor clusters are enough to meet the user requirements, for when collaboration is disabled, the number of locations met decreases by about 10% points.

B. Exploiting Sensing Diversity

TABLE III: Sensor, modality, and node makeup per cluster.

No. Sensors				
Acoustic	Seismic	Infrared	No. Nodes	No. Clusters
1	0	0	1	46
1	1	0	1	21
0	1	0	1	12
0	2	0	2	5
2	0	0	2	4
2	1	0	2	1
0	1	1	1	1

Using Table III, which depicts the node and sensor makeup of each cluster, we show that acoustic sensors are the most capable and are chosen in 80% of all clusters, with seismic and infrared sensors selected in 44% and 1% of all clusters. The table also illustrates that 64% of all clusters consist of a single sensor and 89% reside on a single node, emphasizing that Wolfpack only collaborates sensors when a single sensor cluster is not good enough to meet user requirements. When collaboration is disabled in Table II, the relatively small 10% points reduction in locations that meet user requirements is explained by most clusters residing on a single node. Only 36% of all clusters consist of multiple sensors and 11% of all clusters consist of multiple nodes.

In addition to exploiting sensing diversity among sensors of different modalities, Watchdog exploits sensing diversity among sensors of the same modality. Figure 6 shows that while nearly 60% of all nodes are not used to detect events at

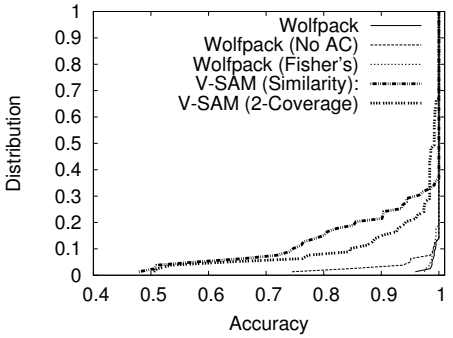

Fig. 5: Location accuracy CDF, illustrating the performance difference of each detection location.

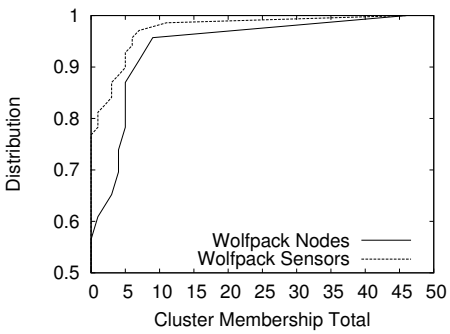

Fig. 6: Cluster membership CDF of nodes and sensors for all detection locations.

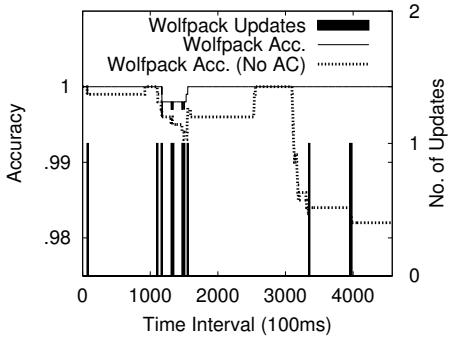

Fig. 7: Location updates and accuracy during runtime.

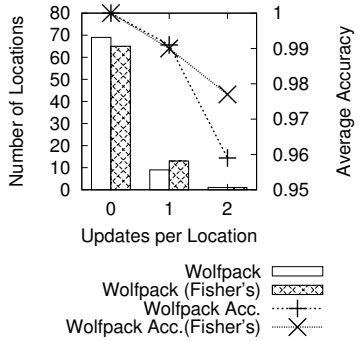

Fig. 8: Updates per location.

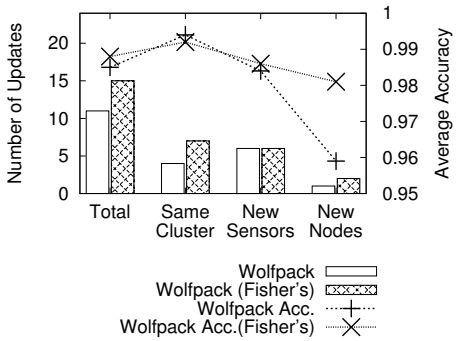

Fig. 9: Number and accuracy of update types.

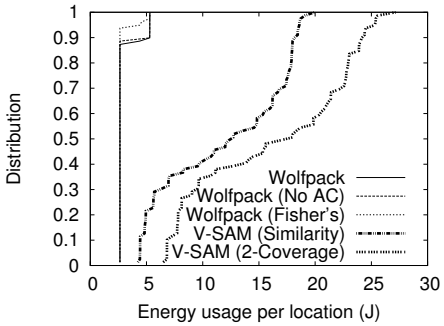

Fig. 10: CDF of energy usage per detection location for Wolfpack and V-SAM.

any location, one node is used to detect events at 46 different locations. Since each node has one sensor of each modality, some sensors of the same modality are much more capable than others, for Wolfpack heavily relies on a few nodes and sensors to confidently detect events. Similar behavior in cluster selection is witnessed for individual sensors.

C. Adaptive Coverage

Using Figure 7, we demonstrate that Wolfpack with Adaptive Coverage is able to update and maintain accuracy when the environment changes, leading to an increase in locations covered by 5% points in Table II. The figure shows that there are three distinct time periods where environmental dynamics cause an Adaptive Coverage update: intervals 50, 1100, and 3300. At these time intervals, the overall accuracy from Wolfpack with Adaptive Coverage maintains or drops only slightly while a greater decrease is witnessed for Wolfpack without Adaptive Coverage.

Most clusters are able to tolerate moderate environmental dynamics and require no updates, as illustrated in Figure 8, with insignificant differences between Nearest Centroid and Fisher's Linear Discriminant. Over 10 locations experience only 1 update and only 1 location experiences 2 updates. Those locations that experience greater environmental changes experience lower detection accuracy: as the number of updates increases, accuracy drops from 100% to 95% with Nearest Centroid. Locations with significant environmental changes also incur cluster updates that move the cluster to different sensors or different nodes, as illustrated in Figure 9. As the complexity of the update increases, the accuracy decreases with Nearest Centroid from nearly 100% using the same sensors to 95% using different nodes. Fortunately, Figure 9 also shows that most updates form the same cluster (with different centroids) or use different sensors on the same nodes, maintaining higher accuracy than when new nodes are used.

D. Active Nodes and Energy Usage

By exploiting sensing diversity and selecting the nodes with the most capable sensors to cover each detection location, Wolfpack can avoid both too little and too much active node coverage. In Table IV, all 3 Wolfpack configurations maintain 10 nodes awake at all times, while bested only slightly by V-SAM with similarity coverage. However, Wolfpack covers nearly all detection locations while similarity coverage V-SAM only covers 67%. Increasing levels of V-SAM coverage have more nodes awake, peaking at nearly 17 out of 23 total nodes awake. The large V-SAM standard deviation in Table IV show that V-SAM puts too many nodes to sleep to accurately capture all events, and once an event is captured, an unnecessarily large number of nodes are awoken to monitor the event.

Consequently, due to fewer active nodes, Wolfpack uses less energy than V-SAM, illustrated by total energy usage in Table IV. For both schemes we measure energy usage as active node sampling time and transmission energy as defined in [39]. With only 10 nodes awake and very few communications due to most clusters residing on the same node, the most costly Wolfpack configuration, Nearest Centroid with Adaptive Coverage, uses 26.558J. This is compared with 32.933J for the most energy efficient V-SAM configuration, similarity coverage. As coverage is increased with V-SAM, energy usage also increases, for more nodes are awake, with energy usage nearly twice that of Wolfpack for 3-coverage V-SAM.

In Figure 10, we plot the CDF of energy usage per location for both Wolfpack and V-SAM. The figure demonstrates that

TABLE IV: Active nodes and energy usage for Wolfpack and V-SAM.

	Active Nodes		Energy		
	Avg.	SD	Total (J)	Radio (J)	Active (J)
Wolfpack	10.0	**0.0**	26.558	0.177	26.381
Wolfpack, No AC	10.0	**0.0**	26.548	0.168	26.381
Wolfpack, Fisher's	10.0	**0.0**	**26.456**	**0.075**	26.381
V-SAM, Sim-cov	**9.9**	5.0	32.933	6.771	**26.162**
V-SAM, 1-cov	10.8	4.6	34.759	6.195	28.564
V-SAM, 2-cov	13.9	3.8	43.393	6.644	36.749
V-SAM, 3-cov	16.7	3.2	50.959	6.797	44.162

for all locations, Wolfpack has very low energy consumption while for many locations, V-SAM can be very energy consuming. For all Wolfpack configurations, almost 90% of all locations are covered by a cluster that uses less than 4J of energy, with all locations using less than or equal to 5J. With similarity coverage, the most energy efficient V-SAM configuration, only 25% of locations use less than or equal to 5J, while 50% use more than 15J and others use nearly 20J.

VII. Conclusion and Future Work

Existing work does not exploit sensing diversity in order to provide confident sensing. Thus, we explore how to use sensor collaboration to take advantage of sensing diversity. Through trace-driven study, we explore sensing diversity, when sensor collaboration is needed, and how to perform sensor collaboration. We formally define a diversity-exploiting confident coverage problem and demonstrate that it is NP-hard. For practical sensor network deployments, we also propose Wolfpack, a confident and distributed event detection coverage scheme which exploits sensing diversity to meet user detection requirements and save energy. Using real trace data for a vehicle detection application, Wolfpack outperforms existing approaches in terms of meeting user requirements, energy, and environmental adaptability. In future work, we plan to investigate how to duty cycle the most capable nodes and sensors to balance the energy consumption of the network while ensuring user requirements are still met.

References

[1] M. Keally, G. Zhou, and G. Xing, "Watchdog: Confident Event Detection in Heterogeneous Sensor Networks," in *IEEE RTAS*, 2010.

[2] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration for Energy Conservation in Sensor Networks," *ACM TOSN*, 2005.

[3] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. Stankovic, T. Abdelzaher, and B. Krogh, "Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments," in *ACM SenSys*, 2005.

[4] Q. Li, J. Stankovic, M. Hanson, A. Barth, and G. Zhou, "Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information," in *BSN*, 2009.

[5] P. Volgyesi, G. Balogh, A. Nadas, C. Nash, and A. Ledeczi, "Shooter Localization and Weapon Classification with Soldier-Wearable Networked Sensors," in *ACM MobiSys*, 2007.

[6] J. Hwang, T. He, and Y. Kim, "Exploring In-Situ Sensing Irregularity in Wireless Sensor Networks," in *ACM SenSys*, 2007.

[7] R. Tan, G. Xing, X. Liu, J. Yao, and Z. Yuan, "Adaptive Calibration for Fusion-based Wireless Sensor Networks," in *IEEE INFOCOM*, 2010.

[8] T. Yan, T. He, and J. Stankovic, "Differentiated Surveillance for Sensor Networks," in *ACM SenSys*, 2003.

[9] Z. Abrams, A. Goel, and S. Plotkin, "Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks," in *ACM/IEEE IPSN*, 2004.

[10] C. Hsin and M. Liu, "Network Coverage using Low Duty-Cycled Sensors; Random and Coordinated Sleep Algorithms," in *ACM/IEEE IPSN*, 2004.

[11] S. Kumar, T. Lai, and A. Arora, "Barrier Coverage With Wireless Sensors," in *ACM MobiCom*, 2005.

[12] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events," in *ACM/IEEE IPSN*, 2005.

[13] M. Malinowski, M. Moskwa, M. Feldmeiera, M. Laibowitz, and J. Paradiso, "CargoNet: A Low-Cost MicroPower Sensor Node Exploiting Quasi-Passive Wakeup for Adaptive Asynchronous Monitoring of Exceptional Events," in *ACM SenSys*, 2008.

[14] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong Barrier Coverage of Wireless Sensor Networks," in *ACM MobiHoc*, 2008.

[15] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks," in *IEEE TOC*, 2002.

[16] E. B. Ermis and V. Saligrama, "Adaptive Statistical Sampling Methods for Decentralized Estimation and Detection of Localized Phenomena," in *ACM/IEEE IPSN*, 2005.

[17] W. Wang, V. Srinivasan, B. Wang, and K. Chua, "Coverage for Target Localization in Wireless Sensor Networks," in *ACM/IEEE IPSN*, 2006.

[18] N. Shrivastava, R. Mudumbai, U. Madhow, and S. Suri, "Target Tracking with Binary Proximity Sensors: Fundamental Limits, Minimal Descriptions, and Algorithms," in *ACM SenSys*, 2006.

[19] P. Varshney, *Distributed Detection and Data Fusion*, Springer, 1996.

[20] G. Hackmann, F. Sun, N. Castaneda, C. Lu, and S. Dyke, "A Holistic Approach to Decentralized Structural Damage Localization Using Wireless Sensor Networks," in *IEEE RTSS*, 2008.

[21] V. Isler and R. Bajcsy, "The Sensor Selection Problem for Bounded Uncertainty Sensing Models," in *ACM/IEEE IPSN*, 2005.

[22] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor Network-Based Countersniper System," in *ACM SenSys*, 2004.

[23] N. Bisnik, A. Abouzeid, and V. Isler, "Stochastic Event Capture Using Mobile Sensors Subject to a Quality Metric," in *ACM MobiCom*, 2006.

[24] G. Yang, V. Shukla, and D. Qiao, "A Novel On-Demand Framework for Collaborative Object Detection in Sensor Networks," in *IEEE INFOCOM*, 2008.

[25] Y. Rachlin, R. Negi, and P. Khosla, "Sensing Capacity for Discrete Sensor Network Applications," in *ACM/IEEE IPSN*, 2005.

[26] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based Sensor Selection Heuristic for Target Localization," in *ACM/IEEE IPSN*, 2004.

[27] S. Subramaniam, V. Kalogeraki, and T Palpanas, "Distributed Real-Time Detection and Tracking of Homogeneous Regions in Sensor Networks," in *IEEE RTSS*, 2006.

[28] M. Duarte and Y. Hu, "Vehicle Classification in Distributed Sensor Networks," *JPDC*, 2004.

[29] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madded, and H. Balakrishnan, "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring," in *ACM MobiSys*, 2008.

[30] K. Lorincz, B. Chen, J. Waterman, G. Werner-Allen, and M. Welsh, "Resource Aware Programming in the Pixie OS," in *ACM SenSys*, 2008.

[31] A. Singh, C. Ramakrishnan, I. Ramakrishnan, and D. Warren, "A Methodology for In-Network Evaluation of Integrated Logical-Statistical Models," in *ACM SenSys*, 2008.

[32] A. Benbasat and J. Paradiso, "A Framework for the Automated Generation of Power-Efficient Classifiers for Embedded Sensor Nodes," in *ACM SenSys*, 2007.

[33] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, "SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-right Mobile Environments," in *ACM MobiSys*, 2008.

[34] P. Zappi, C. Lombriser, T. Steifmeier, E. Farella, D. Roggen, L. Benini, and G. Troster, "Activity Recognition from On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection," in *EWSN*, 2008.

[35] "XBOW Mote Specifications," http://www.xbow.com.

[36] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.

[37] G. Xing, R. Tan, B. Liu, J. Wang, X. Jia, and C. Yi, "Data Fusion Improves the Coverage of Wireless Sensor Networks," in *ACM MobiCom*, 2009.

[38] V. Vazirani, *Approximation Algorithms*, Springer, 2004.

[39] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Power TOSSIM: Efficient Power Simulation for TinyOS Applications," in *ACM SenSys*, 2004.