Watchdog: Confident Event Detection in Heterogeneous Sensor Networks

Matthew Keally[†], Gang Zhou[†], Guoliang Xing[‡]
[†]Computer Science Department, College of William and Mary
[‡]Department of Computer Science and Engineering, Michigan State University

***Abstract*—Many mission-critical applications such as military surveillance, human health monitoring, and obstacle detection in autonomous vehicles impose stringent requirements for event detection accuracy and demand long system lifetimes. Through quantitative study, we show that traditional approaches to event detection have difficulty meeting such requirements. Specifically, they cannot explore the detection capability of a deployed system and choose the right sensors, homogeneous or heterogeneous, to meet user specified detection accuracy. They also cannot dynamically adapt the detection capability to runtime observations to save energy. Therefore, we are motivated to propose Watchdog, a modality-agnostic event detection framework that clusters the right sensors to meet user specified detection accuracy during runtime while significantly reducing energy consumption. Through evaluation with vehicle detection trace data and a building traffic monitoring testbed of IRIS motes, we demonstrate the superior performance of Watchdog over existing solutions in terms of meeting user specified detection accuracy and energy savings.**

I. Introduction

Wireless sensor network deployments have been widely used for event detection in military surveillance [1], ambulatory medical monitoring [2], and vehicle tracking [3]. These event detection scenarios usually require high accuracy to achieve application goals. For example, urban planners may wish to monitor traffic flow at a troublesome intersection [3] with less than 5% false positive and false negative vehicle detection rates. A high false positive rate may precipitate a costly and unneeded road expansion. Similarly, a high false negative rate in detection may cause the planners to cancel a proposed road expansion, leading to worsening traffic conditions. Such an event detection application must meet a user's event detection accuracy requirements with a long deployment lifetime. When a framework makes event detection decisions that meet a user's accuracy requirements in terms of desired false positive and false negative rates, we say it is *confident*.

Several challenges exist to provide a confident event detection framework:

- How to cluster the right sensors in order to meet user detection requirements? Previous work has shown that event detection with individual sensors can exhibit up to 60% false positive and false negative rates while sensor collaboration through clustering can significantly reduce such inaccuracies [1]. In order to cluster the right sensors, we must first determine how to differentiate the sensing capabilities of individual sensors and sensor clusters in a specific deployment. Secondly, how can the detection capability of a specific deployment be obtained? Lastly, if the detection capability of a specific deployment exceeds the user detection requirements, how to obtain a subset of the detection capability to save energy and still meet the user requirements?
- Since many real deployments use multiple sensor modalities, how to efficiently perform collaboration between heterogeneous sensors to meet specific user requirements? Instead of relying on different sensing models for different modalities, how to create a generic solution that can work easily and efficiently with a wide range of deployments and sensor modalities?
- How to adapt the detection capability to runtime sensor observations? Some runtime observations may easily yield a confident event detection decision with a small, energy-efficient cluster of sensors. However, other runtime observations may require more detection capability. With different runtime observations requiring different detection capabilities, how to form different clusters with different detection capabilities? Furthermore, how can these clusters of varying capability collaborate to perform confident event detection within a deployment?

Existing approaches for event detection do not provide a holistic solution with respect to addressing these challenges. Sensing coverage approaches [4] [5] only provide best effort detection and do not cluster the right sensors to meet user detection requirements. Many machine learning [6] approaches cluster sensors to save energy, but do not provide confident event detection. Application or modality-specific sensing models [7] [8] can determine theoretical detection accuracy, but this accuracy is not always achieved during runtime since these models fail to account for sensing irregularity [5]. In all of these existing approaches, the detection capability of a deployment is not fully explored to cluster the right sensors to meet user requirements during runtime event detection in an energy efficient way. Therefore, we are motivated to address these issues, and our main contributions are:

- With trace data from a vehicle detection application, we show the drawbacks of existing solutions. We motivate the need for a holistic framework that provides confident event detection with user-defined accuracy, addresses

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

heterogeneous sensor fusion, and reduces energy usage.

- We propose Watchdog, an event detection framework, which is able to fully explore the available event detection capability of a specific deployment. Watchdog is also able to cluster the right sensors to enforce user-defined event detection accuracy during runtime.
- Watchdog is able to dynamically adjust its detection capability to runtime observations. For observations that can easily yield confident event detection decisions, an energy efficient *sentinel* sensor cluster is used. For more difficult observations for which the *sentinel* cluster is not able to make a confident decision, a less energy efficient but more powerful *reinforcement* sensor cluster is used. Watchdog coordinates the two clusters so that user detection requirements can always be met with maximum energy savings.
- Watchdog is designed as a generic framework, whose performance is evaluated in two scenarios: a vehicle detection application using trace data and a building traffic monitoring application using IRIS motes. The performance evaluation shows that Watchdog can always meet user-specified detection accuracy with reduced energy usage, while in many cases existing solutions cannot.

The rest of this paper is organized as follows: We present related work in Section II and motivate our Watchdog design in Section III. We describe our detailed Watchdog design in Section IV and present its performance evaluation in Section V. Finally, we present conclusions and future work in Section VI.

II. Related Work

Many event detection approaches attempt to address in-situ sensing reality but do not cluster the right sensors to meet user detection requirements. In sensing coverage approaches [9] [4] [10] [11] [5] [12], energy savings is emphasized by ensuring at least k nodes are awake to cover a detection location, leaving all other nodes asleep. The authors of [13] use statistical models to remove outlier sensor data as noise. Regions of similar sensor data are detected in [14]. In [15] and [16], event detection is provided for multiple modalities along with a sleeping scheme to save energy. Other solutions use machine learning techniques to address in-situ sensing reality, such as feature classification [17] [18] [19] [6] [20], Hidden Markov Models [21] [22], or both [23]. Contrasting with Watchdog, none of these approaches cluster the right sensors to provide confident event detection with energy savings.

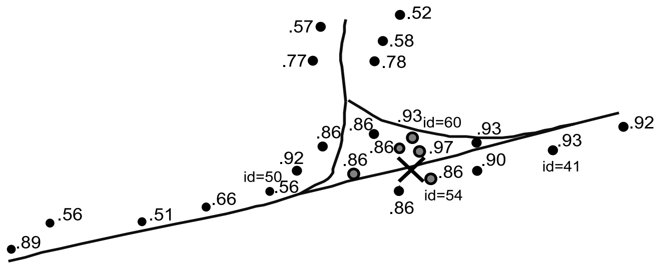
Another group of event detection solutions use a modality-specific sensing model that allows for predicting system or sensor cluster accuracy. An accelerometer-oriented sensing model is used in [24]. A sensing model for camera-based target localization is presented in [25]. Specific sensing models for acoustic, magnetic, and PIR sensors are presented in [26]. A signal attenuation-based model is used in [8] [27] [28] [29] [30] [31], which gives a false positive rate and false negative rate for a given modality and set of training data, allowing for data fusion between multiple sensors [32]. An attenuation-based model is also used in [33], where nodes in a cluster collaborate in detection to provide energy savings. Both [34] and [35] rely on mathematical sensing models to detect events, using a probabalistic noise distribution. More general sensing models, such as disc-based [36] [37] [38] [7], can apply to a wider range of sensing modalities with reduced accuracy. In comparison with Watchdog, these model-driven approaches are modality-specific and do not reflect sensing irregularities [5] in real deployments. None explore the detection capability of a specific deployment to cluster the right sensors and provide confident event detection.

III. Motivation

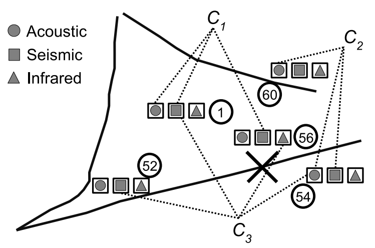
In this section, we demonstrate the need for a new approach to confident event detection with reduced energy consumption. Our goal is to provide confident event detection at a critical point, such as monitoring vehiclular traffic flow, detecting soldiers crossing a bridge, or health monitoring with body sensor networks. As an example, we utilize the Wisconsin SensIT experiment [3] to perform vehicle detection at a specific location. The SensIT experiment consists of a 75 node network with acoustic, seismic, and infrared sensors. With the trace data, we quantify the differences of detection accuracy among individual sensors and sensor clusters and analyze their impact on existing event detection solutions. Unlike existing approaches, we conclude that performance differences between different sensors and sensor clusters cannot be ignored in confident event detection.

In the trace data analysis, we define a target location at the “X” along the road in Figure 1. Data is aggregated into time intervals of 100ms length. A time interval is defined as an event time interval when the vehicle is present within 2 meters of the target location. With this in mind, we determine vehicle detection accuracy for individual sensors and sensor clusters using the method that we present in Section IV-B and we plot the results in Figure 1.

In Figure 1 (a), we first observe that sensors with the same distance to the target location may exhibit different detection accuracies. For example, nodes 41 and 50 are both 80m from the target location, but their detection accuracies are different, 93% and 56%, respectively. This is because while accuracy generally decreases with distance from the target location, terrain changes and environmental conditions still produce irregularities in sensor performance, which is consistent with the findings in [5]. This observed sensing irregularity can cause modality-specific sensing models to suffer, such as [8]. For example, a signal attenuation model for acoustic sensors [8] derives the same acoustic signal receiving power for sensors with the same distance to the target location. Therefore, the same detection accuracy is statistically derived for nodes with the same distance (node 41 and 50 in our example). This signal attenuation model cannot articulate the accuracy differences among sensors, such as determining which sensor is 93% accurate and which is 56% accurate in our example. For this reason, the system performance suffers and the required detection accuracy can not always be met, which we further demonstrate in Section V-B.



(a) Individual acoustic sensors, labeled by accuracy.



(b) Sensor clusters: with 100% accuracy.

Fig. 1. Sensor and cluster event detection performance with vehicle trace data. The target location is marked by the "X" on the road.

In Figure 1 (a), we also observe that not all sensors within the 25m sensing range provide the same detection accuracy. For example, even though both node 60 and 54 are within the 25m sensing range of the target location, they have different detection accuracies, 93% and 86%, respectively. This observed sensing difference can cause sensing coverage-based schemes to suffer. For example, in [1], only one of multiple sensors with the sensing range is enabled at a time to provide sensor coverage (or 1-coverage) for energy savings. For our example, this means that either node 60 or 54 can be turned on to provide such 1-coverage. However, it is clear that using node 60 will provide 7% points better accuracy than with node 54. Unfortunately, sensing coverage schemes have no knowledge of such subtle but important detection accuracy differences, and hence cannot provide confident event detection.

Figure 1 (b) illustrates that different sensor clusters are able to provide the same detection accuracy. For example, clusters C_1, C_2, and C_3 (consisting of different sensor modalities) can all provide 100% detection accuracy even though individual sensors cannot. As shown in [39], clustering sensors can produce a synergistic effect, allowing sensors with complimentary detection strengths in different scenarios to collaborate. Exploring the detection capability of a deployment by evaluating the performance of different sensor clusters allows the most energy efficient clusters to be chosen to confidently detect events. However, such thorough exploration is not achieved by existing works and thus user-defined accuracy requirements cannot be met with reduced energy usage.

While more negative impacts on existing works can be observed in Figure 1, we leave it to individual readers due to space limitations. From the trace data analysis, it is very clear that existing approaches have difficulty meeting user required detection accuracy. This is due to lack of detailed detection accuracy knowledge of individual sensors and sensor clusters. Therefore, it is imperative to design a scheme that can provide confident event detection with user-defined accuracy, address in-situ sensing reality, and reduce energy usage.

IV. Watchdog Design

In our Watchdog architecture, depicted in Figure 2, computationally limited nodes with sensors are connected through a link to a more powerful aggregator. Nodes collect sensor data and return observations to the aggregator, which makes event detection decisions. Our architecture is structured to solve the challenges that arise from providing confident event detection through the use of the Local Aggregation, Cluster Generation, Sentinel and Reinforcement Selection, and Runtime Event Detection modules.

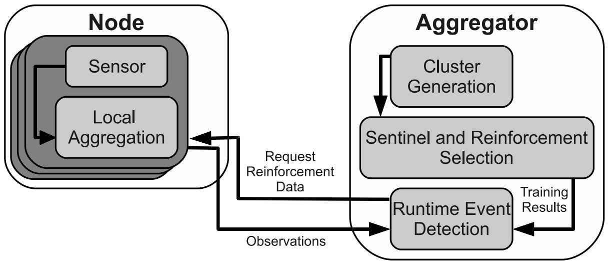


Fig. 2. Watchdog design with node and aggregator components.

The Local Aggregation module, located on sensor nodes, is used to provide efficient collaboration between heterogeneous sensors. Sensor data is aggregated such that observations from different sensor modalities can be compared and easily fused at the aggregator to make cluster-level detection decisions.

In Cluster Generation, we explore the detection capability of a deployment by determining the detection capabilities of individual sensors and sensor clusters within the deployment. We also use Hidden Markov Models [40] to determine accuracy between clusters of heterogeneous sensors.

In Sentinel and Reinforcement Selection, clusters are selected that adapt the detection capability of a specific deployment to runtime observations. Using the deployment detection capability determined by Cluster Generation, a subset of that capability is selected such that the user requirements can be met. A cluster of low-power *sentinel* sensors is selected to meet the user detection requirements for many runtime observations, when event detection decisions are easy. For more difficult event detection decisions where more detection capability is needed, a cluster of *reinforcement* sensors is selected to ensure the user detection requirements are met.

In Runtime Event Detection, the detection capability is adapted to runtime observations using the clusters selected in Sentinel and Reinforcement Selection. Specifically, a low-power set of sentinel sensors make easy event detection decisions to meet user accuracy requirements. When the sentinel sensors determine that more detection capability is needed, a second set of reinforcement sensors are used to make a confident detection decision.

Currently, we make use of offline training to determine detection models for different sensor clusters and assume that training data is representative of runtime data. In the evaluation, different data are used for training and runtime detection, which demonstrates negligible impact on performance. However, if computation and energy concerns permit, online training can allow for periodic detection model updates.

A. Local Aggregation

On a sensor node, the Local Aggregation module allows nodes to aggregate data locally at regular intervals, allowing for reduced radio communication and heterogeneous sensor fusion. The module is flexible to allow incorporation of different widely used aggregation algorithms. The aggregation interval length is selected such that an event can be captured. For each sensor j, aggregated data is converted to discrete observations at each aggregation interval t: $O_{j,t} \in \{1...m\}$, where m is the same for all sensors so that readings from different modalities can be easily compared. The aggregator fuses observations from each sensor j in a sensor cluster C_i to form an observation $O_{C_i,t}$ for that cluster. The fused observations can then be used by the aggregator to determine sensor cluster accuracy or make runtime detection decisions.

B. Cluster Generation

In Cluster Generation, we determine the detection capabilities of individual sensors and different sensor clusters, exploring the detection capability of a specific deployment. To do this, we determine the detection accuracy of sensor clusters of each possible size (size is 1 for individual sensors) in the network using training trace observations. Ideally, we wish to generate all possible clusters of each size to completely explore the deployment detection capability. However, if computing resources are limited, we can compute M random clusters of each possible size from which to choose sentinels and reinforcements. Algorithm 1 describes the Cluster Generation process: (1) First, to compute accuracy for a given cluster C_i in the set of generated clusters C, we train a Hidden Markov Model for that cluster. A thorough explanation of HMMs and training is provided by [40]; (2) Second, we determine a cluster event decision at each training data aggregation interval using the trained HMM; (3) Finally, we compare the cluster event detection decision at each interval with measured ground truth to determine the cluster detection accuracy. Ground truth can be collected via trace data, monitored through video recording, or provided by an upper layer application such as with [5].

Step 1. To distinguish events from noise and to help determine event detection accuracy, we train a Hidden Markov Model for each cluster. We use HMMs as our event detection mechanism since HMMs require little initial configuration and are built upon the premise of determining hidden states (events) from a sequence of known observations (sensor readings) [40]. Furthermore, HMMs allow aggregated sensor readings from different sensor modalities to be easily fused, providing a generic framework that is adaptable to many

Algorithm 1 Cluster Generation

```
Input: Set of all sensors in network $N$, user-defined false
  positive rate $u$.FP and negative rate $u$.FN, training obser-
  vations $O = \{O_{C_i,t} | C_i \subset N, 1 \leq t \leq T\}$, ground truth
  $G = \{G_t | 1 \leq t \leq T\}$, number of clusters for each cluster
  size $M$
Output: Set of clusters $C = \{C_i | C_i \subset N\}$
  Randomly generate $M$ clusters for each size $k(1 \leq k \leq |N| - 1)$, add to $C$
  for all clusters $C_i \in C$ do
    Train HMM $C_i.\lambda$ for $C_i$ with Baum-Welch using $O_{C_i}$
    for Aggregation interval $t(1 \leq t \leq T)$ do
      Determine event probability $\gamma_t$ with $C_i.\lambda$ and $O_{C_i}$
      if $\gamma_t \geq .5$ then $E_t = 1$ else $E_t = 0$
      Compare system event decision $E_t$ with $G_t$
      Update $C_i$.OA.FN, $C_i$.OA.FP, $C_i.\gamma$.FN, $C_i.\gamma$.FP
    end for
  end for
```

application scenarios. At each aggregation interval t, we define two hidden states in cluster HMMs: $E_t = 0$ for non-events and $E_t = 1$ for events. We also provide a sequence of known fused sensor observations for each cluster C_i at each aggregation interval: O_{C_i}. In the Watchdog context, a trained cluster HMM $C_i.\lambda$ for cluster C_i assumes that events are correlated with particular cluster observations and noise is correlated with all other observations. For instance, in the Wisconsin SensIT data trace, seismic sensor data may produce very high readings when a vehicle is close to the sensor, but lower readings when the vehicle is farther away or not present; a trained HMM will capture this correlation. HMMs also assume that events and non-events are correlated with time and make use of transition probabilities to further predict the likelihood of an event at each aggregation interval.

Steps 2 and 3. With a trained HMM for each cluster, we can determine a cluster's event decision E_t for each aggregation interval t. E_t is derived from the cluster's event probability γ_t at each training aggregation interval t. To determine γ_t for each aggregation interval, we use the forward algorithm [40] in conjunction with the trained cluster HMM $C_i.\lambda$ and a cluster observation sequence O_{C_i}. The cluster determines an event occurred at interval t ($E_t = 1$) if $\gamma_t \geq .5$ and no event occurred ($E_t = 0$) if $\gamma_t < .5$. We can then use the cluster's event decision sequence $E = \{E_t | 1 \leq t \leq T\}$ to compare with known ground truth $G = \{G_t | 1 \leq t \leq T\}$ at each aggregation interval to determine cluster training accuracy. If, at aggregation interval t, the event detection decision is equal to the ground truth ($E_t = G_t$), then the cluster made a correct decision at t. Otherwise, the decision was a false positive or false negative.

Event Probability Discussion. We can compute the overall accuracy for each cluster C_i by comparing all event detection decisions E_t to ground truth G_t to determine the overall false negative rate C_i.OA.FN and the overall false positive rate C_i.OA.FP. However, a cluster with an overall low false positive

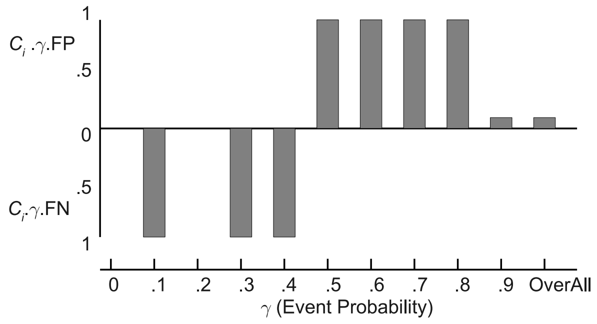


Fig. 3. Event probability breakdown for a cluster C_i with a 6% overall false positive rate and no overall false negative rate. For each .1 event probability range, the associated false positive rate $C_i.\gamma$.FP and false negative rate $C_i.\gamma$.FN are shown as bars. All ranges that have no observations yield a false positive or false negative rate of 1, since no accuracy can be determined for that range and hence we assume the worst.

or false negative rate may have all its incorrect decisions result from event probabilities that hover near .5. During runtime detection, it is likely that an event probability near .5 will result in an incorrect decision. Consequently, it is beneficial to differentiate the accuracies between event probabilities. During runtime detection, possible bad decisions made by sentinels due to middle-range event probabilities can be caught and reinforcements can be used to meet the user requirements.

To study the correlation between event probability and detection accuracy, for each cluster C_i, we break down each training event probability γ_t into p ranges of size $1/p$. For each range we compute false positive rates $C_i.\gamma$.FP and false negative rates $C_i.\gamma$.FN. Figure 3 shows an event probability breakdown of a cluster C_i from the Wisconsin vehicle trace data with 97% overall accuracy with $p = 10$ probability ranges. From the figure, it is clear that all negative event decisions have an event probability in the $[0, .1)$ and $[.2, .3)$, ranges, while all event decisions have a probability in the $[.9, 1]$ range. During runtime detection, the event probability breakdown for the sentinel cluster is used to determine if an event probability γ_t does not meet user false positive and false negative requirements and that reinforcement observations should be collected to make a confident decision.

C. Sentinel and Reinforcement Selection

With the deployment detection capability explored by determining accuracy for all generated clusters, we choose a subset of the deployment to remain awake during runtime detection as sentinels and reinforcements to make confident detection decisions. We choose sentinels such that all negative event decisions can be made with confidence: that the user's false negative requirement is met by sentinels. Since communication is the most energy intensive operation in wireless sensor networks [41], we minimize energy usage by selecting a sentinel cluster with sensors on the fewest number of nodes, for only one radio transmission is needed to report observations from multiple sensors on the same node in one aggregation interval.

Since sentinels are only concerned with determining the lack of an event with confidence, we leave more difficult observations to the more powerful reinforcements when negative event decisions cannot be confidently made by sentinels. Therefore we choose reinforcements so that both the user's false positive and false negative requirements are met. We also ensure that the combined sentinel and reinforcement clusters are located on the fewest number of nodes to save energy. The reinforcement cluster has at least one sensor that is not in the sentinel cluster in order to ensure there is some added benefit from sampling reinforcement data. The sentinel and reinforcement selection algorithm is given in Algorithm 2.

Algorithm 2 Sentinel and Reinforcement Selection

```
Input: Set of all sensors in network $N$, set of trained clusters
  $C$, user-defined false positive rate $u$.FP and negative rate
  $u$.FN
Output: Sentinel sensors $s$, Reinforcement sensors $r$
  /*Sentinel Selection*/
  $s$.FN=1; $s$.numNodes=$|N|$; $s = N$;
  for all clusters $C_i \in C$ do
    /*Meet user FN with least energy*/
    if $C_i$.OA.FN$\leq u$.FN and $C_i$.numNodes$\leq s$.numNodes
    then
      $s = C_i$
    end if
  end for
  /*Reinforcement Selection*/
  $r$.FP=1;$r$.FN=1; $r$.numNodes=$|N|$; $r = N$;
  for all clusters $C_i \in (C - s)$ do
    /*Meet user FP and FN with least energy*/
    if ($s \cup C_i$).numNodes$\leq r$.numNodes and $C_i$.OA.FP$\leq u$.FP
    and $C_i$.OA.FN$\leq u$.FN then
      $r = C_i$
    end if
  end for
```

D. Runtime Event Detection

In Runtime Event Detection, sentinels and reinforcement sensors sample observations at each aggregation interval while all other nodes are asleep. The aggregator dynamically determines an event detection decision E_t for each interval t using sentinel or reinforcement observations. From training observations, a default observation value is determined for each sensor, which is associated with non-events. To save energy, a node only transmits observations when at least one of its sensors makes a non-default observation. If the aggregator does not receive an observation from a sentinel or reinforcement sensor when such an observation is needed, it assumes the default observation value. The Runtime Event Detection algorithm is described in Algorithm 3.

As shown in the algorithm, for each runtime aggregation interval t, sentinels determine an event probability γ_t using the same method performed in Cluster Generation except runtime observations are used. If $\gamma_t < .5$, the sentinels can confidently determine that no event has taken place ($E_t = 0$) since the sentinels were selected such that the user's false negative requirement is always met. However, if $\gamma_t \geq .5$, we must check

Algorithm 3 Runtime Event Detection

```
Input: Sentinels $s$, reinforcements $r$, runtime observation for
    $s$ for the current aggregation interval $O_{s,t}$, may also receive
    runtime observations for $r$ for the previous and currrent
    aggregation intervals $O_{r,t-1}$, $O_{r,t}$
Output: Event detection decision for the current aggrega-
    tion interval $E_t$ and for the previous interval $E_{t-1}$ if
    $E_{t-1}$=UNDECIDED
  if $E_{t-1}$=UNDECIDED then
    /*Make a confident decision at $t-1$ using $r$*/
    Determine $\gamma_{t-1}$ using HMM $r.\lambda$ and $O_{r,t-1}$
    if $\gamma_{t-1} \geq .5$ then $E_{t-1} = 1$ else $E_{t-1} = 0$
  end if
  Determine $\gamma_t$ using HMM $s.\lambda$ and $O_{s,t}$
  if $\gamma_t < .5$ then
    $E_t = 0$    /*$s$ confidently determines no event at $t$*/
  else if $\gamma_t \geq .5$ and $s.\gamma$.FP$\leq u$.FP then
    $E_t = 1$    /*$s$ confidently determines an event at $t$*/
  else if $\gamma_t \geq .5$ and requested $O_{r,t}$ has been received then
    /*Make a confident decision at $t$ using $r$*/
    Determine $\gamma_t$ using HMM $r.\lambda$ and $O_{r,t}$
    if $\gamma_t \geq .5$ then $E_t = 1$ else $E_t = 0$
  else
    /*A confident decision cannot be made at $t$ using $s$*/
    $E_t$=UNDECIDED; request $O_{r,t}$ and $O_{r,t+1}$
  end if
```

if the sentinels meet the user's false positive requirement for the given probability range in which γ_t falls into, $s.\gamma$.FP. If the user false positive requirement is met, $s.\gamma$.FP $\leq u$.FP, the sentinels can confidently determine that an event has occurred ($E_t = 1$). Otherwise, when $s.\gamma$.FP $> u$.FP, then the user false positive requirement is not met, E_t is undecided, and more detection capability is required by requesting reinforcement observations. The aggregator sends a request message to retrieve reinforcement observations for intervals t and $t+1$ when a confident decision cannot be made by the sentinels. The reinforcement observations for t will be returned at the end of interval $t+1$. Piggybacking reinforcement observations for interval $t+1$ along with the observations for t will allow the aggregator to use reinforcement observations to make a decision for $t+1$ if the sentinels are not confident for $t+1$. Another reinforcement observation request message for interval $t+1$ would not be necessary.

When sentinel observations are returned during an interval t for the previous interval $t-1$, the aggregator can make a confident decision, since the sentinels meet the user accuracy requirements. γ_t is determined using the reinforcement observations and an event, $E_{t-1} = 1$, is confidently determined if $\gamma_t \geq .5$. Otherwise, $E_{t-1} = 0$.

To illustrate Runtime Event Detection, an example is presented in Figure 4. In the figure, the sensors on node 1 are sentinels while the other two sensors on nodes 60 and 61 are reinforcements. During the first interval $t = 1$, no sensors report non-default observations, so the base station determines

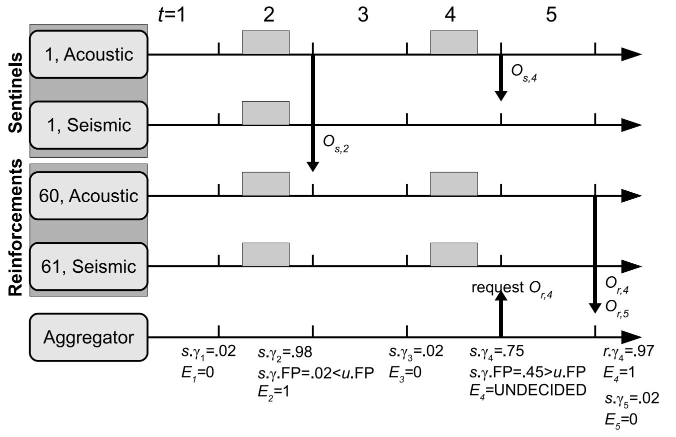
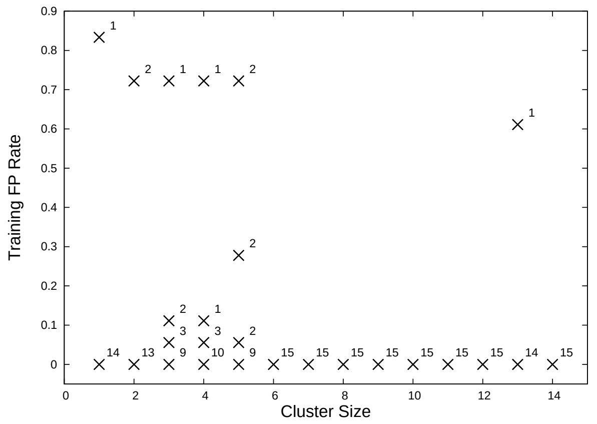


Fig. 4. Runtime detection timeline with sentinel and reinforcement event decisions, where u.FN = u.FP = .05. Gray areas indicate sensor readings that trigger non-default observations. Aggregator-determined event probabilities are indicated by γ_t and event decisions are indicated by E_t. Radio transmissions due to non-default observations are indicated by the arrows.

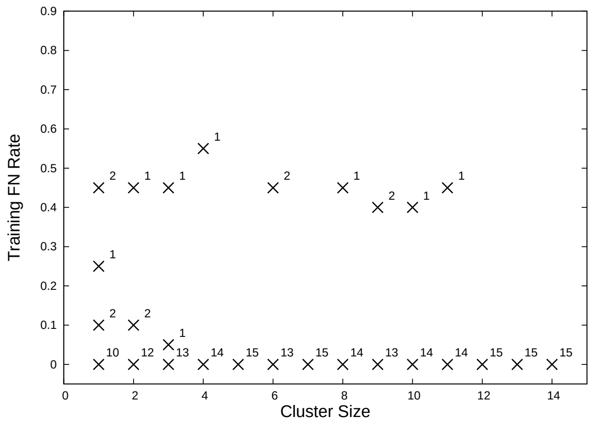
an event probability of .02. Since the sentinels have been determined to meet the overall false negative requirement, s.OA.FN $\leq u$.FN = .05, the decision is confident. A similar decision also occurs at $t = 3$. At $t = 2$, the sentinels capture an event and report their observations via radio, yielding an event probability of .98. The false positive rate for sentinels when $\gamma_t = .98$ was determined during training as .02, so this is a confident decision (.02 $\leq u$.FP = .05). At $t = 4$ the seismic sentinel sensor does not capture the event, and the sentinel false positive rate for the current observation and event probability was determined from training as .45. Since .45 is greater than u.FP = .05, the aggregator could not make a confident decision and more detection capability is needed. Therefore, reinforcements are signaled to return their data for $t = 4$ at the end of interval $t = 5$. At $t = 5$, the reinforcement data yields a confident event decision for $t = 4$ since sentinels always meet the user requirements and the sentinel data determines that no event has occurred.

V. Evaluation

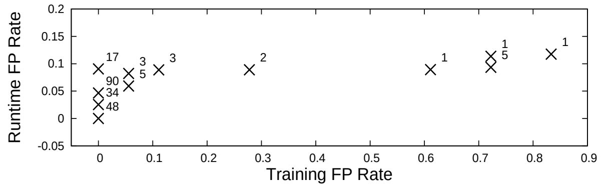
Watchdog is designed as a generic framework, so we evaluate its performance in two different application scenarios: vehicle detection using trace data and a building traffic monitoring application using IRIS motes. Our evaluation is conducted through three aspects. First, we demonstrate that Watchdog is able to explore the detection capability of a specific deployment and cluster the right sensors to meet user detection requirements. Next, we compare against a sensing coverage-based framework and illustrate that Watchdog achieves a significantly higher performance. Finally, we compare against a detection framework which uses a modality-specific sensing model and show that Watchdog can adapt the detection capability to runtime observations and always meet user detection requirements while the model-driven approach cannot. In the experiments, for Watchdog, we generate $M = 15$ clusters for each possible cluster size, and for each cluster, we aggregate sensor readings at each interval into two observations: 0 and



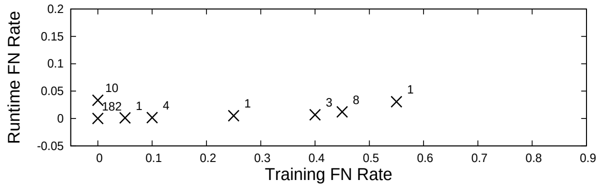
(a) Training False Positive Rate



(b) Training False Negative Rate



(c) Training vs. Runtime Detection False Positive Rate



(d) Training vs. Runtime Detection False Negative Rate

Fig. 5. Cluster Training and Runtime Detection. An integer besides the "x" denotes the number of clusters that give the corresponding FP or FN rate.

1. Energy is measured as usage of CC2420 radios [42] and is derived as power consumption (W) $\times$ time (s).

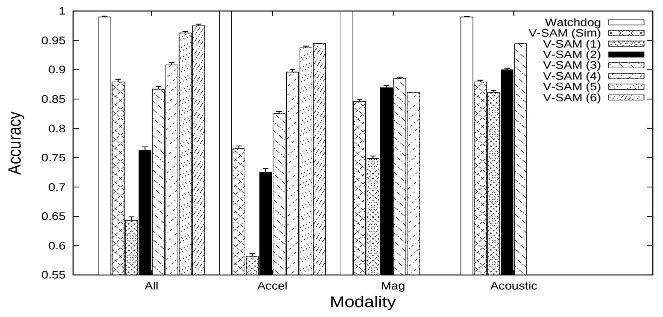
A. Exploring Detection Capability and Meeting Requirements

In this group of experiments, we show that by exploring the detection capability of a specific deployment, Watchdog can choose the right sensor clusters to meet user-defined false positive and false negative rates. We place five IRIS motes with attached MTS310 sensorboards (2-axis accelerometer, 2-axis magnetometer, acoustic, light sensors) [43] on the main entrance door of the Computer Science building to monitor the traffic pattern of when people are most often entering and leaving the building. We define an event and measure the ground truth as the time period during which someone opens the door and walks through (either entering or exiting), with the door automatically closing behind. We obtained ground truth via video recording of the building entrance and sampled data at 20ms intervals using the heterogeneous sensors on the mote sensorboards. Using the collected trace data, in Figure 5 (a) (b), we plot the number of clusters for each cluster size that achieve the same training false positive or false negative rate. In Figure 5 (c) (d), we plot cluster training performance compared with runtime performance. In Figure 5 (a) (b), there are only a limited and discrete number of false positive and false negative rates that the deployed system can support. To that end, a user can only require a false positive or false negative rate that can be supported by the system. For example, most sensors and sensor clusters have false positive and false negative rates near zero, while only a few experience false positive rates greater than 70% or false negative rates greater than 45%. This set of cluster performances is determined by the sensor hardware and local sensing reality where the system is deployed. Different scenarios may produce different false positive and false negative rates for each cluster.

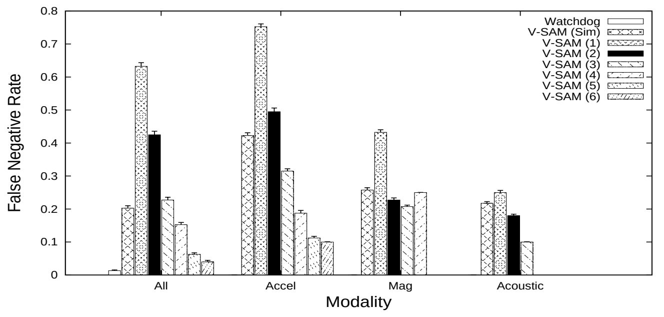
In Figure 5 (a) (b), we also observe that even in a small deployment with "5 IRIS $\times$ 6 sensors each = 30 sensors", there are a large number of sensor clusters available to meet user specified false positive or false negative rate. As shown in Figure 5 (a), there are exactly 3+3+2=8 sensor clusters that demonstrate a 5% false positive rate in the training data and there are 189 sensor clusters in Figure 5 (a) that demonstrate smaller than a 5% false positive rate. So, in total, 8+189=197 different sensor clusters can be chosen to meet the user-specified 5% false positive rate.

In Figure 5 (c) (d), we observe that during runtime detection, Watchdog is able to meet the false positive or false negative rate explored during training. For example, Figure 5 (c) shows that 48 clusters with a training false positive rate of 0% achieve this performance during runtime; Figure 5 (d) shows that 182 clusters with a false negative rate of 0% also demonstrate no false negatives during runtime. In Figure 5 (c) (d), we also observe that clusters with higher training false positive or false negative rates achieve significantly better runtime performance: 6 clusters with a training false positive rate of 72% achieve a runtime false positive rate of 10%, and 13 clusters with a training false negative rate greater than 20% achieve a runtime false negative rate of 5% or less.

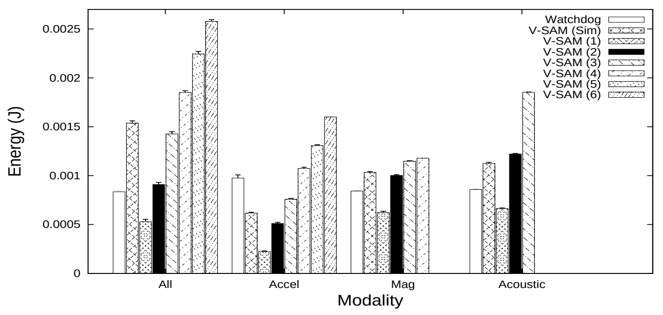
To summarize, these data illustrate that Watchdog is able to cluster the right sensors to meet user requirements during runtime. Plus, many clusters of different sizes exist to meet user-required accuracy. This allows for freedom in sentinel and reinforcement selection to adapt the detection capability



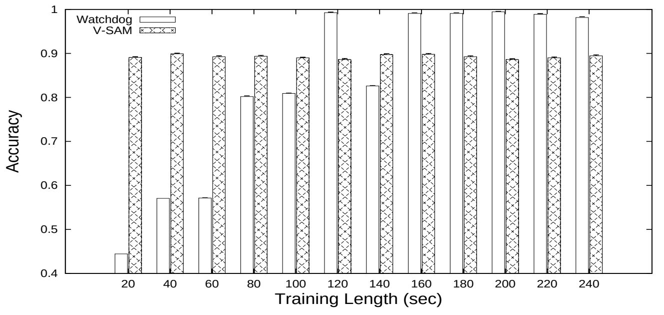
(a) Accuracy Comparison



(b) False Negative Comparison



(c) Energy Comparison



(d) Training Length Accuracy

Fig. 6. Watchdog and V-SAM comparison for different modalities, levels of V-SAM coverage, and training lengths.

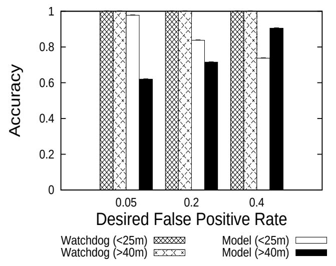
to runtime observations and maximize energy savings.

B. Comparison with V-SAM

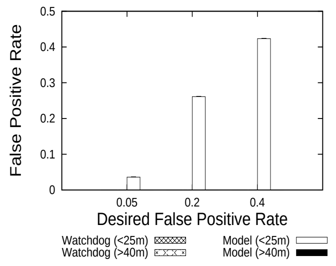
We compare Watchdog to the most recent sensing coverage framework that addresses sensing irregularity, V-SAM [5]. V-SAM measures data similarity between sensors, and keeps awake only a cluster of sensors whose members sample dissimilar data. This similarity is recomputed at every update interval along with a new sleep schedule. V-SAM detects an event if energy readings of awake sensors surpass a dynamic noise threshold, causing all sensors to wake up to monitor the event. A node only transmits a packet if it detects at least one event during an update interval. In this experiment, we use the building traffic monitor application. Besides the default V-SAM similarity-based coverage approach, we force k-coverage on V-SAM to illustrate performance under different levels of sensor coverage. We set the Watchdog aggregation interval and the V-SAM update interval to 4s and give V-SAM the same training data as Watchdog with performance compared using runtime data. Though V-SAM cannot provide guaranteed accuracy, we set the Watchdog user requirements to the lowest false positive and false negative rates determined from training. Evaluation results are presented in Figure 6 with 95% confidence intervals over 20 runs. In Figure 6 (a) (b), We observe that Watchdog outperforms V-SAM in every configuration: all modalities, individual modalities, and varying levels of V-SAM coverage. Although using higher k-coverage and similarity-based coverage helps improve V-SAM performance, it is always outperformed by Watchdog, which consistently demonstrates close to 100% detection accuracy in Figure 6 (a) and close to zero false negatives in Figure 6 (b). None of the Watchdog or V-SAM configurations experience statistically noticeable false positives, so false positive rates are not illustrated. Watchdog can consistently outperform V-SAM because Watchdog fully explores the detection capability of individual sensors and sensor clusters in a deployed system and cluster the right sensors to meet user requirements. However, V-SAM has no detailed knowledge of detection accuracy, so the most accurate sensors may be excluded while poor performing sensors may become involved in detection decisions.

In Figure 6 (c), we observe that Watchdog is much more energy efficient than V-SAM. As shown in Figure 6 (c), Watchdog energy consumption is relatively constant for all modalities and for each modality, hovering around 9×10^{-4} J, while V-SAM energy consumption (when achieving good performance) varies within $10 \times 10^{-4} \sim 26 \times 10^{-4}$ J. Even though Watchdog may use more energy than 1 or 2-coverage V-SAM, Watchdog achieves about 35% points better accuracy compared with those V-SAM configurations. Watchdog is significantly more energy efficient than V-SAM since Watchdog fully explores the detection capability of individual sensors and sensor clusters. Hence, Watchdog can use this knowledge to adapt sensing capability to runtime observations while making confident detection decisions, but V-SAM cannot.

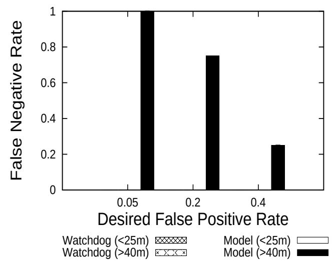
Training Length. In Figure 6 (d), we observe that for Watchdog to achieve the aforementioned superior detection accuracy and energy efficiency compared with V-SAM, only a short training length is needed. As shown in Figure 6 (d), when the training length increases, Watchdog performance improves quickly, surpasses V-SAM performance, and converges to near perfect accuracy after about 2 minutes, which is reasonably short for real applications. Even though V-SAM requires little



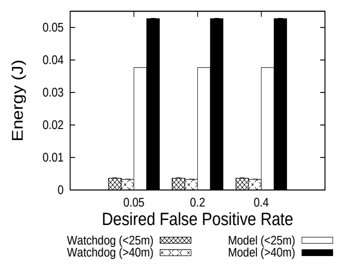
(a) Accuracy Comparison



(b) False Positive Comparison



(c) False Negative Comparison



(d) Energy Comparison

Fig. 7. Watchdog and modality-specific sensing model comparison with sensors located within 25m of, or more than 40m from, the target location.

training, which is invisible in Figure 6 (d), it demonstrates much lower detection accuracy and much higher energy usage than Watchdog. Since the training length is short, the use of periodic retraining can handle environmental changes.

C. *Comparison with a Modality-Specific Sensing Model*

In this section, we compare Watchdog with a classical model-driven event detection solution [8] that uses a modality-specific sensing model. In [8], a signal attenuation model is used to estimate signal energy for targets of different distances with a Gaussian noise distribution model. Given user-defined false positive rate, the model-driven implementation in [8] can derive an event detection threshold for the average energy readings of all sensors in a cluster. Clusters are formed by including all sensors within a fixed distance (fusion range) to a target location with all others put to sleep. For fair comparison, we use the same Wisconsin SensIT experiment trace data [10] used in [8] and make use of acoustic sensors to detect vehicles at a target location. An event occurs when a vehicle is located within 2 meters of the target location. Data is sampled at 4960 Hz and we make use of the AAV8 run for training and the AAV11 run for runtime detection. Ground truth of the vehicle location is provided in the trace. Further details on the experimental setup are given in [3]. For Watchdog and the model-driven scheme, we set an 100ms aggregation interval and compare with varying levels of desired false positive rates since the model-driven scheme in [8] cannot take user-defined false negative rates as input. Our evaluation is conducted in two scenarios: when the target location is well within the sensing range of all sensors, and when the sensors are located at the fringe of the detection range. In the first scenario, we use 5 acoustic sensors < 25m to the target location; in the second, we use 7 acoustic sensors with distances > 40m from the target location. The results are plotted in Figure 7.

For the <25m scenario, we observe from Figure 7 (b) that Watchdog always meets the user false positive requirement while the model-driven scheme cannot. For instance, in Figure 7 (b), the model-driven scheme has a 28% false positive rate when 20% is required, and gives a 42% false positive rate when 40% is required. We also observe from Figure 7 (a) that Watchdog yields perfect accuracy, while model-driven accuracy drops when the desired false positive rate increases. Watchdog performs better than the model-driven scheme because Watchdog always chooses sentinels and reinforcements that meet user requirements for confident event detection. The model-driven scheme does not exploit such subtle but important information.

For the >40m scenario, we also observe that Watchdog always meets user requirements but the model-driven scheme performs poorly or even fails. For example, when user requires a 5% false positive rate, the model-driven approach experiences very low accuracy, 67% in Figure 7 (a), and a very high false negative rate, 100% in Figure 7 (c). This is because for a low desired false positive rate, the model-driven detection threshold is set too high to detect any events. We also find in Figure 7 (c) that requesting higher false positive rates does not help much. The poor performance of the model-driven scheme and the good performance of Watchdog can be explained with the same reasons attributed to the <25m scenario.

For both scenarios, Watchdog is found to consume significantly less energy than the model-driven scheme as shown in Figure 7 (d). This is because the model-driven scheme in [8] has a very simple energy saving scheme: nodes within the 25m "fusion range" are awake and nodes beyond the range all sleep. On the contrary, Watchdog adapts the detection capability to runtime observations through the use of sentinels and reinforcements for more aggressive energy savings. In Figure 7 (d), we also observe that the model-driven scheme consumes more energy in the >40m scenario than the <25m scenario. This is because 7 nodes are used instead of 5.

TABLE I
ADAPTING DETECTION CAPABILITY TO RUNTIME OBSERVATIONS.

Sentinel FP/FN (%)	Reinforc. FP/FN (%)	Reinforc. Requests (%)
9.5/0.0	0.0/0.0	21

Adapting Detection Capability. Using the <25m scenario we illustrate in Table I how Watchdog adapts the detection capability to runtime observations. With desired false positive and false negative rates of 0%, a sentinel cluster is selected with a 9.5% false positive rate and 0% false negative rate. A more powerful reinforcement cluster is selected with a 0% false positive and false negative rate. During runtime, 79% observations are comparatively easy and hence confident decisions are entirely made by sentinels. When the sentinels make a decision that does not meet user requirements (for the 21% more difficult observations), reinforcements are used to make a confident decision. The reduction in radio transmissions made by using only the sensors necessary to meet user requirements ensures significant energy savings.

VI. Conclusions and Future Work

Existing works do not provide a holistic solution with respect to clustering the right sensors for confident event detection, heterogeneous deployments, and adaptation of detection capability during runtime. Consequently, we present Watchdog, a generic event detection framework which can function in a wide array of applications and deployments. Unlike existing approaches, Watchdog can obtain the detection capability of a specific deployment and use this knowledge to cluster the right sensors to perform confident event detection. With a short training length, Watchdog chooses sentinel and reinforcement sensors which adapt the detection capability to confidently detect events while saving energy. Our evaluation demonstrates that Watchdog largely exceeds the detection accuracy of existing approaches with reduced energy consumption. Our evaluation also demonstrates that Watchdog always meets user detection requirements when in many cases existing approaches cannot. In future work, we plan to extend our confident Watchdog framework to provide distributed detection with online training for environmental adapability.

References

[1] T. He, S. Krishnamurthy, L. Luo, T. Yan, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. Stankovic, T. Abdelzaher, J. Hui, and B. Krogh, "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance," in *ACM Transactions on Sensor Networks*, 2006.

[2] E. Shih, A Shoeb, and J. Guttag, "Sensor Selection for Energy-Efficient Ambulatory Medical Monitoring," in *ACM MobiSys*, 2009.

[3] M. Duarte and Y. Hu, "Vehicle Classification in Distributed Sensor Networks," in *Journal of Parallel and Distributed Computing*, 2004.

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

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

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

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

[8] Z. Yuan, R. Tan, G. Xing, C. Lu, Y. Chen, and J. Wang, "Fast Sensor Placement Algorithms for Fusion-based Target Detection," in *IEEE RTSS*, 2008.

[9] 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 Transactions on Sensor Networks*, 2005.

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

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

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

[13] Y. Zhuang, L. Chen, X. Wang, and J. Lian, "A Weighted Moving Average-Based Approach for Cleaning Sensor Data," in *IEEE ICDCS*, 2007.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] B. Greenstein, C. Mar, A. Pesterev, S. Farschi, E. Kohler, J. Judy, and D. Estrin, "Capturing High-Frequency Phenomena Using a Bandwidth-Limited Sensor Network," in *ACM SenSys*, 2006.

[21] R. K. Ganti, P. Jayachandran, T. Abdelzaher, and J. Stankovic, "SATIRE: A Software Architecture for Smart AtTIRE," in *ACM MobiSys*, 2006.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] G. Xing, C. Lu, R. Pless, and J. O'Sullivan, "Co-Grid: an Efficient Coverage Maintenance Protocol for Distributed Sensor Networks," in *ACM/IEEE IPSN*, 2004.

[28] G. Xing, J. Wang, K. Shen, Q. Huang, X. Jia, and H. So, "Mobility-assisted Spatiotemporal Detection in Wireless Sensor Networks," in *IEEE ICDCS*, 2008.

[29] 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.

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

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

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

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

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

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

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

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

[38] 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.

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

[40] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *IEEE*, 1989.

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

[42] "CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

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