A System for Distributed Event Detection in Wireless Sensor Networks

Georg Wittenburg, Norman Dziengel, Christian Wartenburger, and Jochen Schiller
Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany
{wittenbu,dziengel,wartenbu,schiller}@inf.fu-berlin.de

ABSTRACT

Event detection is a major issue for applications of wireless sensor networks. In order to detect an event, a sensor network has to identify which application-specific incident has occurred based on the raw data gathered by individual sensor nodes. In this context, an event may be anything from a malfunction of monitored machinery to an intrusion into a restricted area. The goal is to provide high-accuracy event detection at minimal energy cost in order to maximize network lifetime.

In this paper, we present a system for collaborative event detection directly on the sensor nodes. The system does not require a base station for centralized coordination or processing, and is fully trainable to recognize different classes of application-specific events. Communication overhead is reduced to a minimum by processing raw data directly on the sensor nodes and only reporting which events have been detected. The detection accuracy is evaluated using a 100-node sensor network deployed as a wireless alarm system on the fence of a real-world construction site.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks Distributed Systems [distributed applications]

General Terms

Algorithms, Experimentation, Measurement

Keywords

wireless sensor network, distributed event detection, in-network data processing, pattern recognition

1. INTRODUCTION

Wireless Sensor Networks (WSNs) [1] are used in increasingly complex application scenarios such as vehicle tracking [6], undersea monitoring [14], or classification of human motion sequences [19]. In all of these cases, a sensor network is used to detect events based on the raw data sampled by the various sensors on each sensor node. There are two predominant approaches to tackle the problem of event detection in WSNs: One option is to transmit the raw data as sampled by the sensors to the base station for centralized evaluation. This incurs the drawback of rapid energy depletion due to continuous data transmissions and thus shortens the lifetime of the WSN. Further, the time until all relevant data is transmitted may be significant and cause an unacceptable delay before an event is reported. The second option is to evaluate the raw data on each sensor node, report the detection result to the base station, and run a statistical evaluation on all reports. In this case, the overall detection accuracy suffers from the fact that the initial classification is based on the data of merely a single sensor node.

In this paper, we present a system for distributed event detection in WSNs that allows several sensor nodes to collaborate in order to identify application-specific events. Our system is capable of correctly identifying a configurable number of different classes of events, which can be freely trained on the deployed system. No formal event specification or expert knowledge about the event characteristics is necessary. Communication cost is reduced to a minimum as raw data is evaluated directly on the sensor nodes. Only small feature vectors, i.e., condensed but highly descriptive "fingerprints" of the sampled data, are locally exchanged between nodes; only the information about which event was detected is reported back to the base station.

Our general approach is to adapt algorithms from the field of pattern recognition to WSNs and train the deployed sensor network to recognize new events. The algorithms presented in this paper compose the feature vector using an automatically generated, application-specific selection of features across multiple sensor nodes. The extraction of features from the data samples is performed locally on each sensor. Hence, our approach combines the energy savings of local data processing with a distributed evaluation to yield high detection accuracy. Further, our system is not tied to any particular application scenario because the pattern recognition algorithms are not specific to the type of sensor used or the characteristics of the deployment area.

We evaluate our system on the example of a wireless alarm system consisting of 100 sensor nodes that were attached to the fence surrounding a real-world construction site. The task of the WSN was to detect security relevant incidents by recognizing four previously trained patterns in the lateral oscillation of the fence elements.

The contribution of this paper is threefold:

- We propose a system for event detection in WSNs based on distributed pattern recognition algorithms that can be thought about application-specific events by the means of supervised training.
- We quantitatively evaluate our system applied to a real-world use case in a major WSN deployment.
- We analyze our results in the context of prior lab experiments and compare our work qualitatively to similar approaches.

We expand upon our previous work, specifically upon the lessons learnt during a proof-of-concept deployment [18] and upon our lab prototype [5]. In [18], we evaluated the general validity of the concept of a WSN-based wireless alarm system. We deployed ten sensor nodes on a fence in the patio of our institute and programmed them with a heuristic classifier based on visual inspection of the raw data of typical events. In [5], we presented the fundamentals of our system and evaluated its accuracy in a small-scale lab experiment on the distributed recognition of human gestures. The results of this experiment encouraged us to deploy the system in a real-world setting and serve as a baseline for the discussion of the results obtained from the current deployment.

This paper is organized as follows: We begin with a discussion of related work in Section 2. In Section 3, we proceed with a brief introduction to pattern recognition, introduce the components and algorithms used in our system and detail the processes of training and event detection. In Section 4, we describe the experimental setup, introduce the metrics used in the evaluation, and present and discuss the experimental results. Finally, we conclude in Section 5.

2. RELATED WORK

Event detection has been a hot topic in WSN research in recent years. In this section, we summarize representative approaches and compare them qualitatively with our work.

2.1 Representative Approaches

In [6], Gu et al. describe the VigilNet project which has the objective of tracking vehicles, people and people carrying metallic objects. The sensor nodes employ threshold values on the readings from a magnetometer, a motion sensor and an acoustic sensor in order to classify events. The results are sent to a dynamically assigned group leader for evaluation and tracking. Tracking information about identified events is reported to the base station.

As part of the SensIT project, Duarte and Hu [3] evaluate several classification algorithms in a vehicle tracking deployment. Each sensor node gathers acoustic and seismic data and classifies events using features extracted from the frequency spectrum after performing a Fast Fourier Transform (FFT). The evaluation comprises three classification algorithms: k-nearest neighbor, ML and support vector machine. The classification result is sent to a fixed cluster head for evaluation and combined with reports received from other nodes for tracking a vehicle.

Tavakoli et al. [14] consider a scenario in which targets are tracked using an undersea acoustic sensor network. Similar to the previous two approaches, the sensor nodes report their local classification result to a cluster head which then in turn performs an evaluation of the data and may report the outcome to a base station. Additionally to the number of incoming reports, the cluster head also considers the accuracy of these reports in the past.

The system proposed by Yang et al. [19] is aimed at recognizing human motions. It is a Body Area Network (BAN) consisting of eight sensor nodes attached to the body of a person who may perform one out of twelve actions. Features are extracted from an accelerometer and gyroscope and and classified on each node. If a local classification is promising, the data of all nodes is transmitted to the base station and classified once again. The classification process identifies an action by matching the linear representation of the extracted feature vector to one of several subspaces, each of which corresponds to one type of action.

Wang et al. [16] describe a habitat monitoring system that is capable of recognizing and localizing animals based on acoustics. They employ a cluster head with additional processing capabilities that may request compressed raw data from other nodes for centralized evaluation. Animals are identified by the maximum cross-correlation coefficient of an observed spectrogram with a reference spectrogram. Using reports from multiple sensor nodes, the animals are then localized in the field.

The distributed event detection system proposed by Martincic and Schwiebert [10] groups the sensor nodes into cells based on their location. All nodes in a cell transmit their data samples to a cluster head which averages the results and retrieves the averages from adjacent cells. Event detection is performed on the cluster heads by arranging the collected averages in the form of a matrix and comparing it to a second predefined matrix that describes the event. An event is detected if the two matrices match.

Li et al. [9] use the example of a coal mine surveillance system to evaluate a WSN that detects events in a 3D environment without relying on threshold values in the raw data. Their system works by aggregating data from the sensor nodes on the base station into a gradient data map of the entire deployment area. Events are detected by matching a time series of data map values against the known properties of certain events, e.g., a gas leakage or a water seepage.

2.2 Evaluation

A quantitative comparison of these approaches with our own work is not feasible due to different application scenarios and methodologies used in the respective evaluations. Large-scale outdoor deployments similar to our own were used by Gu [6] and Duarte [3], while Yang [19] and Wang [16] conducted measurements in a more controlled environment. Tavakoli [14], Martincic [10], and Li [9] simulate a WSN using synthetic events.

Instead, Table 1 shows a qualitative comparison of the approaches with our own work. The criteria we considered are processing requirements on the sensor nodes, infrastructure requirements with regard to network services, and the capabilities to support training and to perform distributed classification. The first two criteria reflect how demanding an approach is on the hard- and software of the WSN platform. The remaining two criteria describe whether the system can be trained to recognize application-specific events and whether the classification algorithm considers feature vectors from multiple nodes as opposed to feature vectors gathered only on the local node.

Criterion / **Approach**	Processing requirements	Infrastructure requirements	Training support	Distributed classification
Gu et al. [6]	+	(-)	-	o
Tavakoli et al. [14]	o	o	(-)	-
Yang et al. [19]	o	(+)	+	+
Duarte and Hu [3]	-	o	+	-
Wang et al. [16]	-	o	o	-
Martincic [10]	+	-	+	(+)
Li [9]	+	o	o	-
Our approach	+	+	+	+

Table 1: Comparison of Approaches

Sensor nodes capable of extensive data processing, e.g., calculating a FFT, are required by Duarte [3] and as cluster heads by Wang [16]. Tavakoli [14] and Yang [19] are not as clear about their requirements, but from their approach one can still assume that the necessary calculations are not trivially handled by a typical sensor node.

Most approaches require the WSN platform to provide a mechanism for cluster management and leader election. Gu [6] even provides a service for migrating cluster heads in order to support object tracking. Martincic [10] requires the WSN to be partitioned into cells and assigns the nodes to cells based on location information. Except for our own approach, Yang [19] and Li [9] are the only ones not to require a clustering service because in their approaches all nodes communicate directly with the base station.

Training of scenario-specific events is supported by the approaches proposed by Yang [19], Duarte [3], and Martincic [10]. Wang [16] and Li [9] can be considered to have partial training support, since they implement a classification based on user-defined patterns in the form of spectrograms or properties of time series. Gu [6] mostly relies on pre-defined threshold values instead of event-specific training data. Tavakoli [14] discusses event detection based on simulated event radii and hence does not focus on the aspect of training. It should be mentioned that support for event-specific training is not necessarily an advantage for all kinds of scenarios. In some cases, it may in fact be preferable to classify events with a purpose-built classifier based on expert knowledge rather than a generic classifier using training data from sampled exemplary events. This is especially true for scenarios in which training data for all types of events is very hard to generate.

Martincic [10], Yang [19], and Li [9] are the only approaches except ours to support classification based on feature vectors from multiple sensor nodes. However, the feature vectors used by Martincic [10] are limited to a single dimension per node, and Li [9] relies entirely on the base station for classification. All other approaches perform a classification based on the features collected on the local node and later aggregate the local classifications on a cluster head. Gu [6] differs from the other approaches in that it allows the cluster head to reevaluate a given classification by transmitting confidence information together with the classification result.

To summarize, the approach proposed in this paper is unique in that it supports training and classification based on features from multiple sensor nodes while at the same time only posing minimal demands on processing and infrastructure resources.

3. SYSTEM ARCHITECTURE

Before discussing the software components and algorithms used in our system, we provide a brief summary of the basics of pattern recognition. An in-depth introduction to this subject matter is available in Duda et al. [4] and Niemann [11].

3.1 Pattern Recognition

Pattern recognition is the process of classifying raw data, i.e., assigning a set of data samples to one of multiple classes. The process can be subdivided into the following three steps:

1. **Sampling:** Data describing the object of interest is gathered by a sensor and optionally preprocessed to eliminate background noise.

2. **Feature Extraction:** Features are extracted from the data and combined into a feature vector. The goal is to reduce the dimensionality of the data while preserving all characteristic information.

3. **Classification:** Feature vectors are classified, either using *a priori* knowledge from a preceding training session or by statistical means alone.

A great variety of features can be extracted from raw data. They range from simple statistical measures, such as minimum, maximum and average, to complex signal processing algorithms, such as computing the discrete Fourier transform [11]. Figure 1 shows an exemplary feature being extracted from a time series of normalized 8-bit acceleration values sampled at a 41.6 Hz. The event is discretized into time intervals and for each time interval the difference between the minimal and maximal acceleration value is computed. This results in a set of features that are simple to compute and adequately capture the change of acceleration intensity over time.

The most widely used algorithms for classification are (among many others) decision trees, neural networks, support vector machines, and k-nearest neighbors [4]. In light of the resource constraints on sensor nodes, we opted for a classification employing the prototype modeler [7] in our system. Figure 2 illustrates the classification process in a two-dimensional feature vector space. Four prototype vectors have been established by averaging the normalized training data. A feature vector is classified by finding the nearest prototype vector in terms of Euclidean distance. If the feature vector is close enough to the prototype vector, it is recognized; if not, it is ignored.

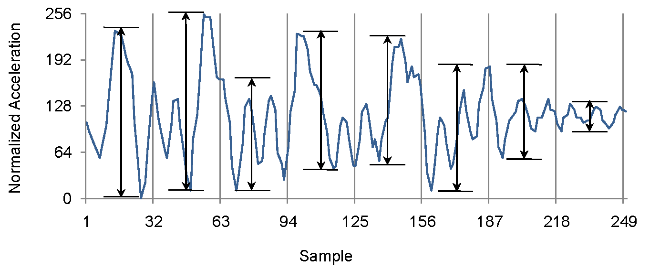

Figure 1: Exemplary Feature Extraction from Raw Acceleration Data

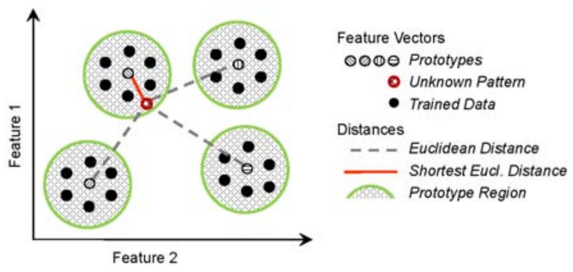

Figure 2: Classification by Finding the Nearest Prototype Vector

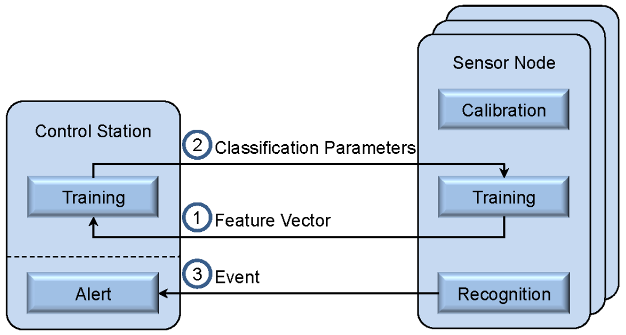

Figure 3: System Architecture

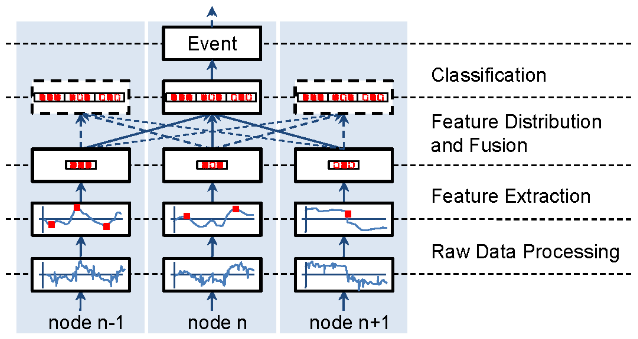

Figure 4: Stages of the Event Detection Process

3.2 System Components

The two principal components of our system are the sensor nodes and the control station as shown in Figure 3. The sensor nodes are deployed in a way that is suitable to detect the spatial extension of the physical effects of the events. For example, for an alarm system that detects trespassing at a fence the nodes are equipped with accelerometers and attached to the fence at fixed inter-node distances. Since the fence is constructed using interconnected fence elements, the lateral oscillation propagates along the fence and the nodes can sample the acceleration at different distances from the source of the event. In other application scenarios, different deployment strategies and different types of sensors may be used. However, all core components and algorithms are capable of general-purpose training and event detection, and require no application-specific adaptations if the system is deployed in different scenarios.

The information about which nodes are adjacent to each other is relevant in our system, because the nodes combine feature vectors from their respective neighbors in the classification. Thus, every node must know its position relative to its neighbors, e.g., which nodes are attached to neighboring fence elements. This information can be encoded into the unique node ID of each sensor node. For the example of the alarm system, it is sufficient to increment the node IDs along the fence. When receiving a packet, a node simply compares its own ID with the sender ID and deduces the relative position of the sender on the fence. For more complex scenarios, additional information can easily be encoded into the node ID, e.g., to allow the node to identify neighbors in a two-dimensional grid or to identify nodes attached to specific parts of machinery.

The control station is required to support and coordinate the training, but is not part of the event detection during the deployment. The training requires considerable memory and processing resources and is thus performed on PC-style hardware rather than on sensor nodes. During the deployment, the control station may optionally be used to serve as a base station for event reporting.

The system works in three distinct phases: The **calibration phase** begins when the user initially turns on the deployed sensor nodes. In this phase, each sensor node adjusts all attached sensors to the local environmental conditions at the deployment site by taking a fixed number of samples to measure the background noise of the observed physical quantity. For instance, a microphone measures the noise caused by wind or machinery operating nearby. This information is used later on as threshold values to switch between low and high resolution sampling when capturing an event and to segment the signal into event-specific time series of raw data. For sensors that have a deployment-specific default value, e.g., accelerometers with regard to the direction of the pull of gravity, the values for the default position are established. The calibration procedure may be repeated at any point in time during the deployment if deemed necessary by the sensor node, e.g., after observing a continuous drift in sampled values or an increase in background noise.

The **training phase** is initiated by the user via the control station, typically as soon as all sensor nodes have been deployed in the field. The purpose of this phase is to train the WSN to recognize application-specific events by providing a set of exemplary training events for each event class. In training mode, the system is exposed to several exemplary events of each class. The sensor nodes transmit feature vectors extracted from sampled raw data to the control station. The control station selects a subset of all available features that is particularly well suited for event detection, calculates a prototype vector for each event class, and transmits these parameters back to all sensor nodes. This process is depicted by arrows (1) and (2) of Figure 3 and described in more detail in Sections 3.3 and 3.4.

Once the training is complete, the system enters the **detection phase**. Except for occasional local recalibrations (described above), the WSN remains in this phase throughout the deployment. In the detection phase, the sensor nodes share their locally extracted features with other nodes in the vicinity via broadcasts. Each node combines features from multiple nodes into a feature vector which is then either classified or rejected, i.e., ignored, using the prototype modeler. Detected events that require logging or user interaction are reported to the base station (or control station) of the WSN, possibly via a gateway node. This process is depicted in Figure 4. It corresponds to arrow (3) of Figure 3 and is described in more detail in Sections 3.5, 3.6 and 3.7.

3.3 Training

The first step required to extract a feature from the raw data generated by a training event is to preprocess sampled data points to remove noise from the data stream. The data stream is then segmented into chunks that correspond to event candidates by using the threshold values with hysteresis that were established during the calibration phase. For features whose values are known to fit into a fixed range, the segment is normalized to the value range of the word size of the WSN platform in order to achieve the highest possible accuracy in subsequent calculations. Finally, features are extracted from the normalized segments of raw data. Depending on the algorithm corresponding to each feature, this can be computationally cheap (e.g., average) or expensive (e.g., discrete Fourier transform).

As all machine learning-based approaches, the overall accuracy of the event detection increases with the number of training events per class. The process of teaching the system for a specific application scenario may be too time consuming for end users. However, for a hypothetical commercial version of our system, we consider it save to assume that it would be part of the business model of the manufacturers of the WSNs to support their costumers with pre-trained, high-quality training data for major application scenarios. This corresponds to the current practices in the industry, e.g., regarding the provisioning of training data for hand writing or speech recognition applications.

All training events are carried out at the same location and observed from the perspective of a particular fixed sensor node. This fixed node should ideally be the node that is closest to the physical source of the event. As the nodes in the vicinity respond to the training events and send their features to the control station, the position of the nodes in relation to the fixed node is stored as an attribute of the features. The recorded features are thus no longer tied to particular nodes, but rather to nodes with a specific relative position to the fixed node. The intuition behind this approach is that each node together with its neighboring nodes has its own unique view of the physical effects related to an event. If the observed effects match those recorded at the fixed node during the training, the classification succeeds even if the effects are observed by a different node at a different position. It is thus possible to detect events independently of their position in the field.

Our approach relies on two basic assumptions about the setup of the deployed WSN: 1) the sensor nodes must be

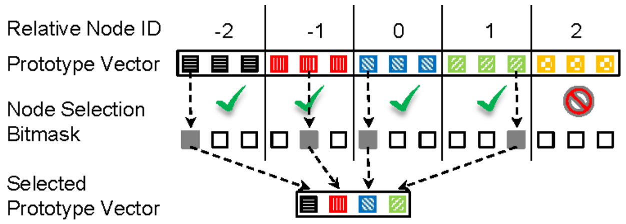

Figure 5: Composition of Prototype Vector using Feature Vectors from Multiple Sensor Nodes

placed uniformly within the deployment area, and 2) the physical effects of the events to be observed must propagate evenly in all parts of this area. The first assumption can be achieved easily in most application scenarios of our system. For the scenario evaluated in this work, the condition is naturally satisfied by attaching one sensor node per equally-sized fence element. Other common types of deployment, such as deploying sensor nodes in a line or in a grid on the ground, also meet this requirement. If a uniform deployment of the sensor nodes is not possible, the event detection can be be adapted to operate on features related to relative coordinates in the deployment area. However, this additional abstraction complicates the evaluation, and hence we focus on a scenario in which the assumption holds.

The second assumption is strongly related to which classes of events are to be detected. Generally, the assumption is valid for those scenarios in which the physical effects of an event are transmitted naturally via the environment. This is the case for events with a descriptive noise, temperature, or vibration patterns as found in use cases such as sniper detection [13], wild fire monitoring [2], or volcano monitoring [17]. Other classes of events require an appropriate deployment of the WSN. For example, when monitoring the movement of a fence as in our scenario, we have to ensure that the fence elements are physically interconnected for the acceleration patterns of an event to propagate from one fence element to the next. However, having to ensure the propagation characteristics as part of the deployment has certain drawbacks which we discuss in Section 4.4.

The parameters required for the classification component of our system, the prototype modeler, are the prototype vectors that correspond to the different classes of events and the size of the prototype regions surrounding the prototype vectors. The latter are necessary to discard the feature vectors that correspond to untrained events (cf. Fig. 2). In addition to this set of standard parameters, we also need to consider the special constraints present in WSNs: Features from certain sensor nodes may be inadequate for classification purposes because a node may only sporadically register the physical effects due to its distance to the source of the event. Additionally, it is not advisable to use all available features for classification because of the processing overhead incurred by each additional feature. Hence, the system needs to select a set of suitable features to be used in the classification before calculating prototype vectors.

3.4 Feature Selection

Once the training is complete, the control station considers how many feature vectors were received from each node. If this value is below a configurable threshold for more than

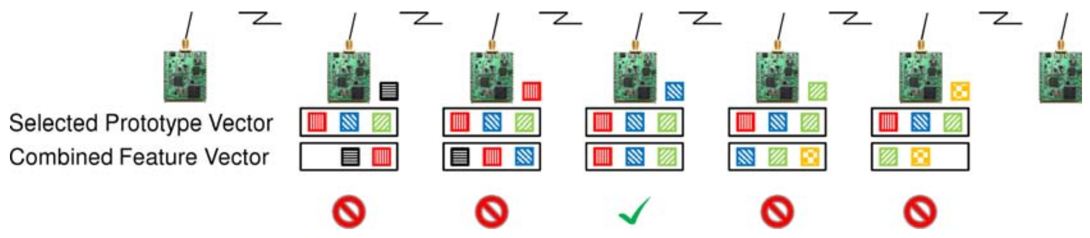

Figure 6: Event Classification and Rejection based on the Physical Location of the Sensor Nodes

one of the events, the algorithm discards all features from this node, because the physical effects of the event are not pronounced enough at this relative position to reliably trigger the segmentation. The remaining features are evaluated with regard to their potential to differentiate between the trained classes of events. The goal is to find the set of features that is the most suitable one to form a distinct prototype vector for each event class.

In order to evaluate the quality of a feature set we employ the Leave-one-out Cross Validation (LOOCV) algorithm [8]. Like all cross validation algorithms it divides the set of feature vectors into two subsets. The training vectors contained the first subset are averaged to calculate prototype vectors. The second set is used as input to check the classification error of the prototype vectors. The strategy used by LOOCV to subdivide the set is to only have a single element in the second set. LOOCV iterates over all possible subdivision and averages the classification errors. This average serves as quality metric for feature sets.

To find the optimal selection of features S, we iteratively select a feature f from the set of available features F and calculate the quality metric LOOCV($S \cup \{f\}$). If the quality is superior to any other choice of f, f is moved from F to S, otherwise it remains in F. We repeat this process until no addition to S can significantly improve the quality. The resulting selection of features S is stored in a bitmask that describes which feature from which node is to be used for event detection.

Now that we have established which features are to be used for classification, we calculate the final prototype vectors for each event class by averaging the training vectors and projecting the averaged vectors to the selected features as depicted in Figure 5. The prototype vectors are then transmitted to the sensor nodes together with information about the respective prototype regions and the bitmask that describes which features from which node were selected. As only a subset of the original features is used in the prototype vectors, the sensor nodes can omit the calculation of all unused features and reduce processor and memory usage.

3.5 Distributed Event Detection

The initial steps of the distributed event detection resemble those used during the training: The sensors gather samples at a preset frequency and the data is preprocessed, segmented, and – depending on the feature – normalized. Only features used in the prototype vectors are extracted.

The extracted feature vector is transmitted via broadcast to all sensor nodes in the 1-hop neighborhood. Forwarding the feature vectors to more distant nodes is usually not required because for most scenarios the radio range of a sensor node is significantly larger than the spatial expansion of the physical effects caused by an event. For those rare cases in which this assumption does not hold, the system needs to be configured to distribute the extracted feature vector in the n-hop neighborhood (for a scenario-specific n known to all nodes) via limited flooding.

The process of extracting and broadcasting feature vectors upon measuring the effects of an event runs in parallel on all sensor nodes (cf. Fig. 4). Different sensor nodes may transmit their feature vectors at different times depending on when the physical effects drop below the threshold of the segmentation algorithm on each particular node. Hence, if one node broadcasts its feature vector while another node is still processing the raw data, the transmission of feature vectors may fail. To solve this problem, every node that currently has a feature vector available rebroadcasts its own feature vector upon receiving a previously unknown feature vector from another node. Feature vectors are discarded after a successful classification or after a timeout.

Once all required feature vectors have been received, each node performs a feature fusion by combining the feature vectors based on the bitmask and the relative position of the sender (cf. Fig. 5). In other words, each node builds its own view of the event using feature vectors from its neighboring nodes. Obviously, the combined feature vector is different for each node, because the respective neighboring nodes perceive the event differently depending on their location. Ideally, only the prototype modeler running on the node whose view of the event matches the view trained at the fixed node will classify the event correctly, while the event is rejected on the other nodes.

3.6 Event Rejection

Since the WSN may be exposed to unknown events, our system needs a mechanism to reject feature vectors which cannot be matched to a prototype vector with a sufficient level of confidence. In fact, the rejection of feature vectors can be expected to be the rule rather than the exception. If a trained event occurs anywhere in the deployment area, only the node whose relative position to the source of the event matches that of the fixed node during the training should ideally classify the event as belonging to one of the trained classes. All other nodes that were exposed to the physical effects of the event but are located at different relative positions should reject the event, because their view of the event differs from the trained pattern. In case a misdetection occurs and a node falsely reports an event, it is up to the base station to gracefully handle the situation. In our system, we implemented the classification fusion by adding information about the distance between feature and prototype vectors to the data that is reported by each node and only counting the result with the smallest distance at the base station.

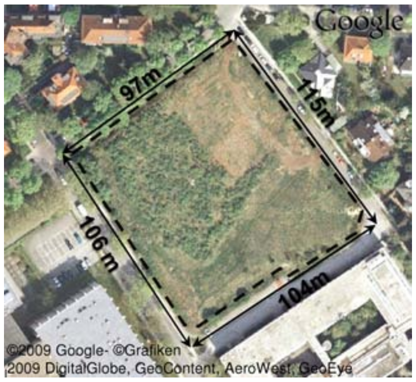

Figure 7: Construction Site with Highlighted Run of Construction Fence

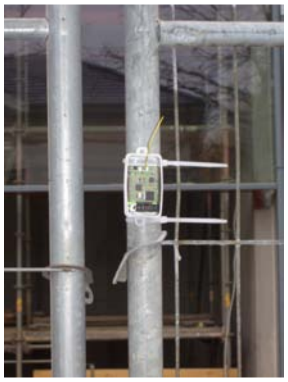

Figure 8: ScatterWeb MSB Sensor Node Attached to a Construction Fence

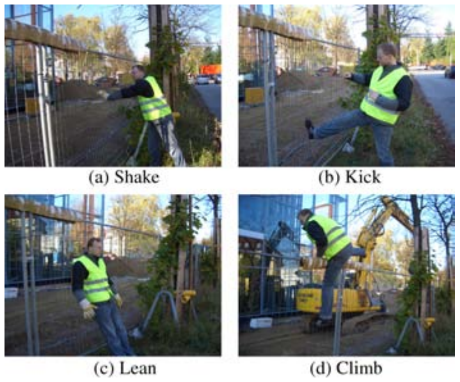

Figure 9: Four Different Classes of Events

Event rejection works by checking whether a feature vector is located within a prototype region as established during the training (cf. Fig. 2). Feature vectors that are located within a prototype region are classified as matching the corresponding prototype vector; feature vectors that are not located within any prototype region are rejected. The key parameter to the event rejection algorithm is thus the size of the prototype regions. If the region is too small, events will be rejected incorrectly; if it is too big, untrained events will be falsely classified as a trained one. In preliminary experiments, we evaluated several options, and decided to use the distance between the prototype vector and the most distant training feature vector of the same event class as the radius of the prototype region.

3.7 Event Reporting

Once a feature vector has been successfully classified as belonging to an event class, it needs to be reported to the base station of the WSN for logging or to alert the user. Most routing protocols are suitable for accomplishing this task, each with a different level of communication overhead. We decided against running the event detection system with routing and instead opted for simply flooding the network with notifications about the detected events. Compared to reactive routing protocols this is no disadvantage as they have to flood the network anyway to setup a route. Compared to proactive protocols it is only a disadvantage if the rate of detected events is high in relation to the maintenance traffic of the routing protocol. Furthermore, additional energy can be saved by configuring the WSN to only report high priority events, e.g., security breaches in case of an alarm system.

3.8 Example

With all the components in place, we now give a brief example encompassing the entire event detection process: Let's consider a small sensor network consisting of seven nodes deployed equidistantly along a straight line as illustrated in Figure 6. For simplicity, each sensor only extracts a single feature from the sampled raw data. The nodes have been programmed to recognize a previously selected three-dimensional prototype vector. The corresponding node selection bitmask is configured in such a way that the features of the node itself and its left and right neighbors (as identified by the node ID) are part of the prototype vector. Let's now assume that the previously trained event occurs in the center of the deployment area and that its physical effects are registered by the sensors on the five central nodes. Each node extracts the feature from the raw data, the value of which differs depending on the physical location of the sensor node (illustrated by the different colors of the squares next to each node). As the nodes proceed to broadcast the features, each node stores the features indicated by the node selection bitmask, i.e., those of its left and right neighbor. Once all features are present, each node compares the combined feature vector of the indicated features with the stored prototype vector. The comparison results in a match only on the central node which is deployed at the location of the event. At all other nodes, the combined features vector and the prototype vector do not match and the event is rejected. Finally, the central node reports the detected event to the base station of the network.

4. PERFORMANCE EVALUATION

In order to evaluate our system, we conducted a three-day real-world field test. Our application scenario was to deploy a WSN as a wireless alarm system on the fence of a construction site. This section contains a description of the experimental setup followed by a brief introduction to the metrics that we considered and finally presents and discusses the results of the experiment.

4.1 Experimental Setup

We attached 100 ScatterWeb MSB sensor nodes [12] to the fence elements of a construction site. The exact run of the fence is depicted in Figure 7. At the time of the deployment, the construction of a four-story hotel building was nearing its completion.

As shown in Figure 8, we attached one sensor node to the left-hand metal tube of each 3.5 m × 2 m fence element using wheather-proof junction boxes. ScatterWeb MSBs are equipped with a Texas Instruments MSP430 16-bit microcontroller with 5 KB of RAM and 55 KB of FLASH memory, a ChipCon 1020 radio transceiver operating at 868 MHz and a Freescale Semiconductor MMA7260Q 3-axis accelerometer that we used to measure the movement of the fence elements. We took special care to attach the sensor nodes to the fence elements uniformly in order to ensure that all of them are

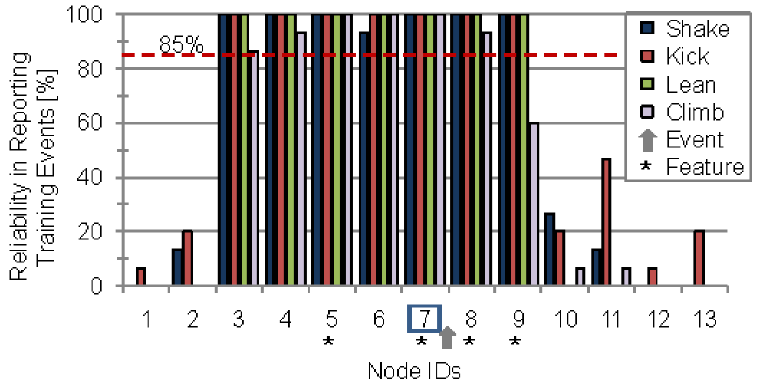

Figure 10: Node Selection by Reliability of Event Reporting

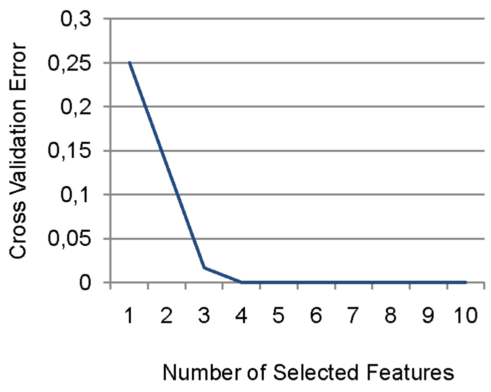

Figure 11: Expected Error against Number of Features

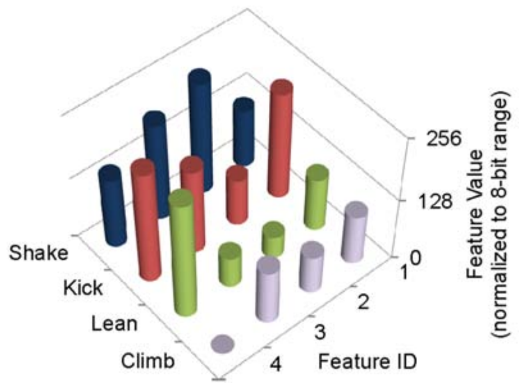

Figure 12: Comparison of Prototype Vectors

exposed to comparable acceleration values. Based on the experience from previous experiments, we set the sensitivity of the accelerometer to 2 G and the threshold for the feature selection to consider a node as sufficiently reliable to 85%.

We exposed the WSN to four different classes of events depicted in Figure 9. The events were shaking the fence, kicking against the fence, slightly leaning against the fence, and climbing over the fence. For legal reasons we were not actually allowed to climb over the fence, so we climbed up and down on the outer side instead. For the training, we chose a region of the fence that was free of any external obstructions and trained the WSN with 15 events of each class. The fixed node during the training was node #7. Afterwards we carried out 15 events of each class at the location where the training took place and another 15 events of each class at two other locations.

We collected the following metrics while exposing the system to the different classes of events (with TP = True Positive, TN = True Negative, FP = False Positiv, FN = False Negative):

- Sensitivity $= \frac{\#TP}{\#TP+\#FN}$, also called recall, corresponds to the proportion of correctly detected events.
- Specificity $= \frac{\#TN}{\#TN+\#FP}$, corresponds to the proportion of correctly ignored events.
- Positive Predictive Value (PPV) $= \frac{\#TP}{\#TP+\#FP}$, also referred to as *precision*, corresponds to the probability that correctly detecting an event reflects the fact that the system was exposed to a matching event.
- Negative Predictive Value (NPV) $= \frac{\#TN}{\#TN+\#FN}$, corresponds to the probability that correctly ignoring an event reflects the fact that the system was not exposed to a matching event.
- Accuracy $= \frac{\#TP+\#TN}{\#TP+\#TN+\#FP+\#FN}$, corresponds to the proportion of true results in the population, i.e., the sum of all correctly detected and all correctly ignored events.

4.2 Results

We begin with a close look at the training process before proceeding to evaluate the classification accuracy. Figure 10 shows the per-node reliability of reporting training events for each event class. We observe that seven nodes were reproducibly affected by the events and reported feature vectors to the control station with a reliability above 85%. We also note that the events do not propagate equally in both directions on the fence. Events were carried out next to the fixed node #7 (arrow) and we observed the propagation characteristic to be uneven, with five affected nodes to the left and two affected nodes on the right. From this diagram follows that features from the nodes #3 to #9 are candidates for the feature selection.

Figure 11 depicts the cross validation error as calculated by the LOOCV algorithm against the number of selected

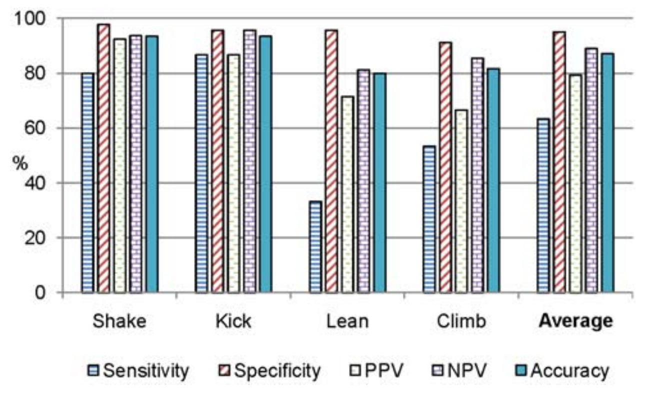

Figure 13: Detection Accuracy by Event Class after Feature Fusion

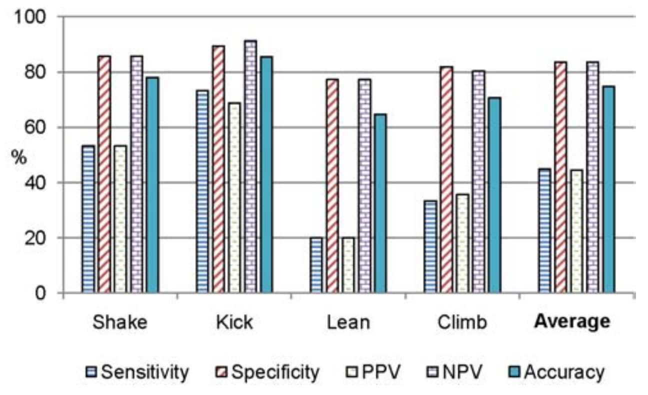

Figure 14: Detection Accuracy by Event Class after Classification Fusion

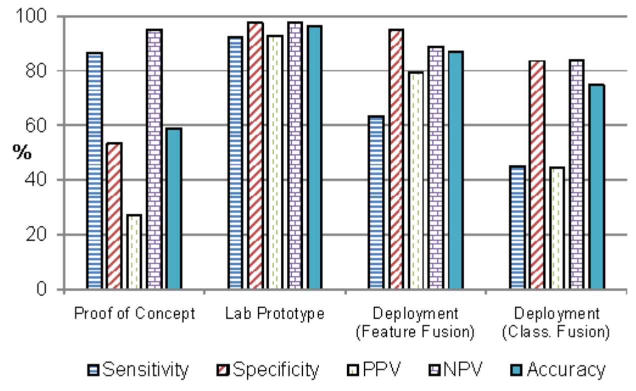

Figure 15: Comparison with Proof of Concept and Lab Prototype

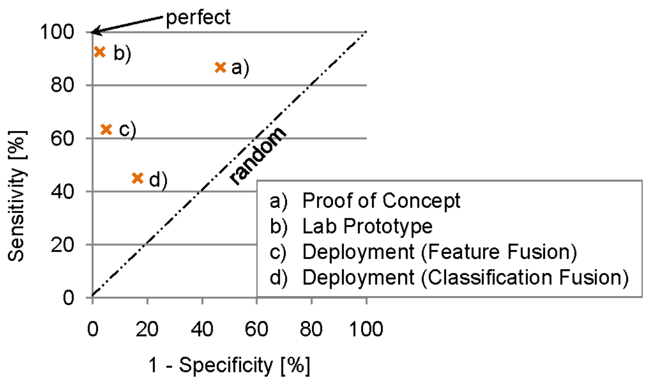

Figure 16: ROC Analysis of All Discussed Systems

features. The diagram shows how the cross validation error decreases and hence the quality of the selected features improves as the feature selection iteratively adds the best possible features to the set of selected features. While the cross validation error is still above 10% for two features, it drops rapidly and is negligible for more than four features. Hence, the quality of the selected features cannot be improved for more than four features. Adding more features beyond this point would merely increase the cost of the event detection process in terms energy consumption and resource usage. Thus, exactly these four features are used to create the prototype vectors for classification.

The prototype vectors for each event class are depicted in Figure 12 with their values normalized to an 8-bit range. Each of the four prototype vectors consists of four selected features. The four features are from four different nodes and of two different types: Feature #1 is a histogram-based feature (cf. [5]) from node #5; features #2 to #4 are intensity features (cf. Fig. 1) from nodes #7, #8 and #9. Each prototype vector differs from any other one in at least one feature. Hence, they span the entire four dimensional feature space and provide a good basis for the classification algorithm. Furthermore, it is noteworthy that the selected nodes (highlighted with stars in Fig. 10) are close to the location of the event. As we can assume that events are more pronounced in the vicinity of their source, this is an indication that our feature selection algorithm is able to select significant features, in spite of the unevenness in the propagation characteristics.

With the prototype vectors in place, we can now proceed to evaluate the accuracy of the event detection. Figure 13 shows the metrics for each type of event as reported by the feature fusion on node #7 which was the fixed node in the training. The shake and kick events are detected reliably with all metrics above 80% and accuracies of 93.3%. Detecting lean or climb events is not as accurate as the previous two events. Sensitivity is comparatively low for these two event classes, while specificity remains high. This is an indication that too many events are falsely rejected due to the prototype regions being too small. We conclude that our 15 training runs for each of these two classes were too similar to each other and thus the prototype regions only enclose part of the required space. Looking at the averages of all four events, the overall accuracy of the system after feature fusion is 87.1%.

Figure 14 shows the same metrics for the classification fusion that is required to filter out false positives from the neighboring nodes. While specificity, NPV and accuracy remain more or less stable, sensitivity and PPV decrease considerably. This decrease results from two separate effects: First, whenever a correct classification is falsely rejected on the node #7 while an incorrect classification is reported from another node, the incorrect classification is counted at the base station. Second, a few nodes report an incorrect classification with a distance metric that is smaller than that of the correct classification. In spite of these problems, the system achieves an overall accuracy of 74.8%.

The results based on the measurements at the other two locations exhibit similar trends as the previous ones, however, at a lower overall level of accuracy. We attribute these results to problems with the setup of the experiment and omit a quantitative evaluation. Instead, we discuss the problems separately in Section 4.4.

4.3 Discussion

The results presented in the previous section indicate that both the training including the feature selection and the event detection work as intended. Event rejection suffers slightly from the uniformity of the training data for two event classes and would have benefited from a more exhaustive training. Event detection accuracy is acceptable for events that are carried out at the same location as during the training.

In order to verify the results from this experiment, we compare them to our previous results from [18] and [5]. In our proof-of-concept work [18], we exposed ten sensor nodes attached to a fence to six different events. The system did not support autonomous training at that time. Instead, we relied on a custom-built heuristic classifier implemented in the FACTS middleware [15] which was manually configured to classify events based on visible patterns in the raw data. For the initial evaluation of our current system in a controlled lab setting [5], we trained three sensor nodes to cooperatively recognize four different geometric shapes based on the acceleration data measured by the sensor. The four shapes, comprising a square, a triangle, a circle and the capital letter U, were drawn on a flat surface by physically moving the sensor nodes and classified using a cooperative fusion classifier. We had three persons move one sensor nodes each along these shapes for a total of 160 runs.

From Figure 15 we can see that the current results do not reach the same level of accuracy as measured under lab conditions. Compared to the proof-of-concept implementation, whose results correspond to those of the feature fusion of the current system, we achieve a 28.8% higher accuracy with the supervised training and automated feature selection in place. Figure 16 highlights the relation between sensitivity and specificity in our experiments. It shows a comparison of the current system and our two prior experiments. Under lab conditions, our system performs almost perfectly, followed by our current results recorded after feature fusion.

It would certainly be beneficial for the evaluation to have more data points available to ensure the statistical relevance of our findings. Unfortunately, the experiments require a great amount of time to conduct and cannot be automated because real persons are required to create events. However, as the findings of our experiment follow those from prior small-scale tests as well as those from our lab experiments, we feel that the evaluation is reasonable well suited to judge the performance of our system.

4.4 Lessons Learned

While evaluating the data gathered from the experiments, we observed two factors that had a negative impact on the accuracy of the event detection process.

We trained the sensor network with 15 sample events of each class. One class was trained after the other starting with the shake and kick events and then followed by the lean and climb events. As the training progressed, our test subjects became more and more familiar with the setup and with generating events on the fence for the sensor nodes to detect. This resulted in the sample events growing more and more similar to each other, which in turn decreased the size of the prototype regions of the latter two classes of events. This in turn resulted in the lower sensitivity in the experiments of the lean and climb events as discussed in the previous section.

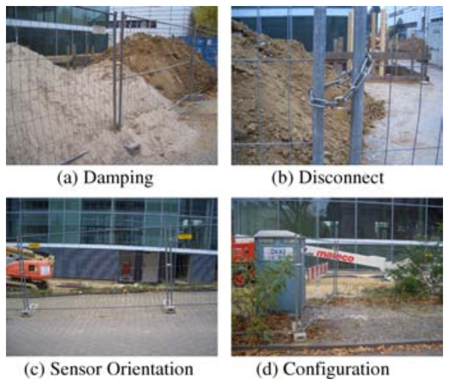

Figure 17: External Influences on Event Patterns

There are two possibilities to work around this problem: One is to significantly increase the numbers of test subjects and/or sample events. This has the drawback that the calibration requires more time to conduct. Alternatively, we can change the training process to train one sample event of each class and then repeat this step several times. This way, we can avoid any bias in the size of the prototype regions without having to commit any additional resources to the training phase.

The second factor that had a negative impact on our results were irregularities in the construction of the fence itself. Figure 17 illustrates how several external factors change the way in which the physical effects of the events propagate along the fence, e.g., the oscillation of the fence elements is dampened or interrupted by the way the fence is set up. Furthermore, some sections of the fence were deployed with an inclination or in a non-linear configuration. This violates the assumption built into our algorithms that the physical effects of the events to be observed propagate evenly in all parts of the deployment area. Given the highly accurate results in our prior lab experiments, we underestimated the effects that these heterogeneous conditions would have on the results.

Again, there are two possibilities to deal with this problem: The first possibility is to make it a constraint of the system to only be deployed in scenarios in which uniform propagation characteristics can be guaranteed. In a fence monitoring scenario, this would imply that greater care than usual must be taken by the construction workers to properly connect the fence elements to each other. However, this constraint is unpractical for a production-level system as it may require additional training of the workers. A better alternative would be to train the sample events on several locations of the deployed system and combine the data reported by sensor nodes in different parts of the deployment area when calculating the prototype vectors.

We were unable to repeat these parts of the experiment with our current deployment because no additional time in the construction schedule could be allocated to us. However, we believe that both issues can be addressed with only minor changes to our system, and we plan to incorporate the necessary adaptations in our next deployment.

5. CONCLUSIONS

In this paper, we have presented a system for distributed event detection in WSNs that allows for several sensor nodes to collaborate in order to identify which application-specific event has occurred. Our approach is unique in that it strikes a compromise between classification accuracy and resource requirements. It makes use of a dynamic subset of the nodes for event detection and does not rely on any services provided by more powerful nodes or a central base station. Using a large-scale deployment in a realistic setting, we have shown that our system is a viable option for correctly identifying different classes of previously trained events.

Since fall 2009, our research on distributed event detection in WSNs continues as part of the project "AVS-Extrem" funded by the German Federal Ministry of Education and Research. We are currently in the process of redesigning our sensor node platform in order to reduce energy consumption during periods of continuous, low-frequency sampling. Our overall goal in this project is to refine our system to achieve lifetimes and levels of accuracy that are suitable for production-level deployments in the context of construction site and freight terminal monitoring.

6. ACKNOWLEDGMENTS

This work was funded in part by the German Federal Ministry of Education and Research (BMBF, Project "AVS-Extrem"). We gratefully acknowledge the support of Seminaris Hotel- & Kongressstätten-Betriebsgesellschaft mbH and Hermann Kirchner Hoch- und Ingenieurbau GmbH.

7. REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8), Aug. 2002.

[2] D. M. Doolin and N. Sitar. Wireless Sensors for Wildfire Monitoring. In *Proc. of SPIE Symp. on Smart Structures & Materials / NDE '05*, San Diego, CA, USA, Mar. 2005.

[3] M. F. Duarte and Y. H. Hu. Vehicle Classification in Distributed Sensor Networks. *Journal of Parallel and Distributed Computing*, 64(7), July 2004.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, 2nd edition, Nov. 2000.

[5] N. Dziengel, G. Wittenburg, and J. Schiller. Towards Distributed Event Detection in Wireless Sensor Networks. In *Adjunct Proc. of 4th IEEE/ACM Intl. Conf. on Distributed Computing in Sensor Systems (DCOSS '08)*, Santorini Island, Greece, June 2008.

[6] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A. Tirumala, Q. Cao, T. He, J. A. Stankovic, T. Abdelzaher, and B. H. Krogh. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *Proc. of 3rd Intl. Conf. on Embedded Networked Sensor Systems (SenSys '05)*, San Diego, CA, USA, Nov. 2005.

[7] A. Kalton, P. Langley, K. Wagstaff, and J. Yoo. Generalized Clustering, Supervised Learning, and Data Assignment. In *Proc. of 7th ACM Intl. Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2001.

[8] R. Kohavi. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proc. of 14th Intl. Joint Conf. on Artificial Intelligence (IJCAI '95)*, San Mateo, USA, 1995.

[9] M. Li, Y. Liu, and L. Chen. Non-Threshold based Event Detection for 3D Environment Monitoring in Sensor Networks. In *Proc. of 27th Intl. Conf. on Distributed Computing Systems (ICDCS '07)*, Toronto, Canada, June 2007.

[10] F. Martincic and L. Schwiebert. Distributed Event Detection in Sensor Networks. In *Proc. of Intl. Conf. on Systems and Networks Communications (ICSNC '06)*, Tahiti, French Polynesia, Oct. 2006.

[11] H. Niemann. *Klassifikation von Mustern.* Springer, 1st edition, July 1983.

[12] J. Schiller, A. Liers, and H. Ritter. ScatterWeb: A Wireless Sensornet Platform for Research and Teaching. *Computer Communications*, 28, Apr. 2005.

[13] G. Simon, M. Maróti, Ákos Lédeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor Network-Based Countersniper System. In *Proc. of the 2nd Intl. ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, Baltimore, MD, USA, Nov. 2004.

[14] A. Tavakoli, J. Zhang, and S. H. Son. Group-Based Event Detection in Undersea Sensor Networks. In *Proc. of 2nd Intl. Workshop on Networked Sensing Systems (INSS '05)*, San Diego, CA, USA, June 2005.

[15] K. Terfloth, G. Wittenburg, and J. Schiller. FACTS - A Rule-Based Middleware Architecture for Wireless Sensor Networks. In *Proc. of the 1st Intl. Conf. on COMmunication System softWAre and MiddlewaRE (COMSWARE '06)*, New Delhi, India, Jan. 2006.

[16] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao. Target Classification and Localization in Habitat Monitoring. In *Proc. of IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP '03)*, Hong Kong, China, Apr. 2003.

[17] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation (OSDI '06)*, Seattle, WA, USA, Nov. 2006.

[18] G. Wittenburg, K. Terfloth, F. L. Villafuerte, T. Naumowicz, H. Ritter, and J. Schiller. Fence Monitoring - Experimental Evaluation of a Use Case for Wireless Sensor Networks. In *Proc. of 4th European Conf. on Wireless Sensor Networks (EWSN '07)*, Delft, The Netherlands, Jan. 2007.

[19] A. Yang, S. Iyengar, S. S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari. Distributed Segmentation and Classification of Human Actions Using a Wearable Motion Sensor Network. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW '08)*, Anchorage, AK, USA, June 2008.