Nonthreshold-Based Event Detection for 3D Environment Monitoring in Sensor Networks

Mo Li, *Member*, *IEEE*, Yunhao Liu, *Senior Member*, *IEEE*, and Lei Chen, *Member*, *IEEE*

Abstract—Event detection is a crucial task for wireless sensor network applications, especially environment monitoring. Existing approaches for event detection are mainly based on some predefined threshold values and, thus, are often inaccurate and incapable of capturing complex events. For example, in coal mine monitoring scenarios, gas leakage or water osmosis can hardly be described by the overrun of specified attribute thresholds but some complex pattern in the full-scale view of the environmental data. To address this issue, we propose a nonthreshold-based approach for the real 3D sensor monitoring environment. We employ energy-efficient methods to collect a time series of data maps from the sensor network and detect complex events through matching the gathered data to spatiotemporal data patterns. Finally, we conduct trace-driven simulations to prove the efficacy and efficiency of this approach on detecting events of complex phenomena from real-life records.

Index Terms—Distributed applications, data compaction and compression, query processing, wireless sensor networks.

1 Introduction

Wireless sensor networks (WSNs) have been widely studied for environment monitoring. In such monitoring applications, automatically detecting events is quite essential, e.g., for detecting vehicles or forest fires. Currently, the typical event detection method [12] relies on decisions made at the sensor node(s) based on predefined data thresholds for normal environments. The rationale behind such threshold-based approaches is that when events occur, there will be detectable changes in environmental data. Thus, an event can be captured once the observed sensory data exceed the predefined thresholds.

Our motivating scenario comes from the field study in a coal mine [15], where environment surveillance is carried out to ensure miners' safety. The amount of oxygen, gas, dust, temperature, humidity, and watery regions are monitored in a 3D space of underground tunnels in the mine. Several event detection tasks are essential to secure the safety of the miners, such as detecting *gas leakage*, *oxygen-enriched spots*, and *water seepage*. *Gas leakage* often occurs when the digging machines expose a source of gas in the mining process, and it often leads to a local increase in gas density. If a certain district of gas accumulates to critical explosive density, explosions could occur. *Oxygen-enriched spots* exist at the ventilative places where high oxygen density creates healthy environmental conditions for human beings. Indicating such areas provides important guidelines for the miners patrolling in the coal mine. *Water seepage* brings water into the coal mine tunnels, which corrodes the tunnel surfaces and threatens the tunnel's structural integrity.

• *The authors are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {limo, liu, leichen}@cse.ust.hk.*

Manuscript received 25 May 2007; revised 7 Feb. 2008; accepted 15 May 2008; published online 2 June 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-05-0236.
Digital Object Identifier no. 10.1109/TKDE.2008.114.

The events described above share the common characteristics that their occurrence results in trends in the development of environmental data rather than some instantaneous overrun of specified thresholds in individual sensor nodes. Hence, the threshold-based approaches work well for detecting simple events, but the complex events with spatiotemporal variety in the environment can hardly be captured by a simple cutoff method. An integrative view of the environment has to be established to extract the features of such events. For example, gas leakage usually leads to an expanding area of high gas density over time, which spatially follows a degrading form where the gas density decreases from the source of the leak. The water seepage can be categorized as a "fault event" with an apparent observation on the advances of the frontier between the dry area and the flooded area.

In order to accurately detect complicated events, we need a nonthreshold-based event detection approach. We intend to describe complex phenomena with certain spatiotemporal data patterns and detect events through matching the gathered data to such data patterns. The challenges for such a design are as follows: First, differing from threshold-based approaches, the environment data map has to be continuously maintained from real-time sensor readings, while conserving energy for battery-powered sensors [5], [6], [23], [25], [27] is a very critical issue. We need to restrain the data traffic and maintain the data map in an energy-efficient manner. Second, the communication quality of WSNs is poor, especially in the underground monitoring environments, such as a coal mine. We have to develop robust methods of data map construction so that the accuracy of the obtained data map could be preserved in a high loss rate network. Third, the 3D monitoring field raises nontrivial issues in abstracting the environment, which are not faced by previous works in 2D cases. Efficient data structures and modeling methods are required in a point of 3D view.

In this paper, we propose a 3D gradient data map using the space orthogonal polyhedra (OP) model. We build a multipath routing architecture to provide robust data delivery for the map construction. Instead of directly routing raw data to the sink before processing, a novel 3D aggregation algorithm is designed for map construction and update. We demonstrate the efficacy and efficiency of the proposed approach in trace-driven simulations using synthetic data sets derived from the raw data collected in our study in the real coal mine environment.

The rest of this paper is organized as follows: We briefly review the related work in Section 2. In Section 3, we describe the network architecture and construction of the gradient data map. The aggregation criteria and incremental data map construction techniques are also introduced. In Section 4, we describe the event feature patterns and illustrate how pattern-based event detection is performed on the data map. Experimental studies of our approach are given in Section 5. Finally, we conclude this work in Section 6.

2 Related Work

Event detection remains an essential task in various WSN applications. There are a number of recent works on event-oriented query processing in sensor networks. The COUGAR project [3] introduces a sensor database system and deals with three types of event queries: historical queries, snapshot queries, and long-running queries. The system employs threshold-based detection logic and encapsulates it into a set of asynchronous functions provided for users. Directed Diffusion [14] aims at addressing the event-based real-time queries by diffusing different event interests into the monitoring network and letting sensors report when occurrences of some specified events are detected. The Directed Diffusion approach does not explore the spatial or temporal correlations among the sensory data, and it relies on individual reports of sensor nodes according to the disseminated event interests. TinyDB [12] defines the event by a composition of various specified attribute thresholds. The event detection is carried out by comparing sensory readings of attributes with predetermined threshold values. TinyDB provides a distributed mapping method to construct contour maps of sensor network readings. Differing from our approach, the mapping process in TinyDB is only done in 2D fields and their work does not aim to provide event detection based on the data spatiotemporal patterns. DSWare [16] explores the correlation among different sensor observations for event detection. Events are grouped into two different types: atomic events and compound events. Confidence functions are employed to address compound events. Above works all focus on 2D scenarios.

In-network data aggregation has been intensively studied as an effective method to provide energy-efficient data collection [4], [17], [18], [19], [21]. LEACH [11] protocol constructs clustering hierarchy on the network and achieves data fusion at the cluster heads to reduce transmitted information. By rotating the cluster heads, energy dissipation is evenly distributed over the network. The TAG approach [19] builds a routing tree in the sensor network and statistical data are aggregated in the intermediate nodes. In [17], network tomography techniques have been applied to solve the problem of loss inference in data aggregation. Different from the approaches above, our approach explores the spatial correlations on the sensory data and achieves data aggregation through the combination of OP in the gradient data maps. Recently proposed contour mapping methods [12], [20], [24] share similar ideas with this work in visualizing the monitored fields for event detection. While those works utilize aggregation-based approaches to efficiently approximate the 2D field in contour maps, they provide no means to extend for 3D scenarios.

The multipath routing strategy has recently been suggested to provide robust data deliveries in sensor networks. Robust aggregation on it needs duplicate-insensitive data structures to carry information [7], [22]. In our work, we propose the space OP model as such a duplicate-insensitive data structure.

3 Three-Dimensional Gradient Data Map Construction

This section is organized as follows: In Section 3.1, we briefly describe the sensor network architecture and deployment of sensors in a 3D space. Then, in Section 3.2, we present the concept of 3D gradient data map. In Section 3.3, we introduce the OP and describe how to achieve in-network construction of the gradient data map by the space OP model. Sections 3.4 and 3.5 describe the aggregation criteria for the gradient data map construction and its incremental update. Finally, in Section 3.6, we extend our algorithm for random sensor deployments.

3.1 Network Architecture

In our coal mine monitoring scenario, sensor nodes are assumed uniformly deployed in 3D monitoring space with measured location information (later, we will release this constraint to extend our work into random deployments). This could be easily achieved by placing sensors along the safety props in the tunnel. Fig. 1 shows the environment in underground coal mine tunnels and the placement of sensors in our underground prototype system previously reported in [15]. A cubic grid can be established on this network and each sensor node accounts for the environment sensing in the cubic cell it resides in (as shown in Fig. 2a). The grid information is created at sink and disseminated throughout the network. Each sensor node, based on its location information, calculates the dimension and coordinates of the cubic cell it resides in.

The whole network is organized into multipath routing architecture. Sensor nodes are divided into different levels from the sink. The sensor nodes closer to the sink have lower levels. For each sensor node, the one level lower nodes are considered as parent neighbors, and the one level higher nodes are treated as child neighbors. Each node forwards the query messages originated in the sink to its child neighbors and sends the report messages to its parent neighbors. Thus, in the message relay process at each node, multirelayers are triggered for message forwarding. By the means of multipath routing, message redundancy is provided to ensure a more reliable message delivery in the lossy sensor network.

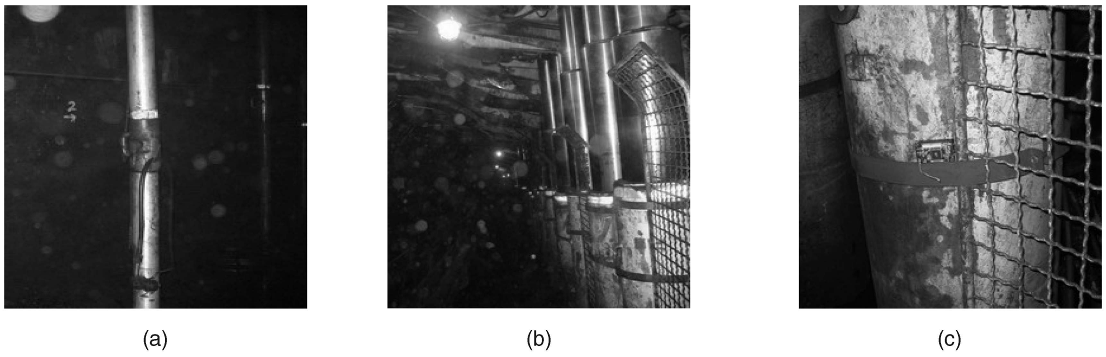


Fig. 1. The underground coal mine environment. (a) and (b) The safety props in the main tunnel and the working face. (c) The sensor node deployed on a safety prop.

The multipath routing architecture is constructed by a two-phase initialization process, *DIFFUSION* and *ECHO*. In the *DIFFUSION* phase, the sink originated indicator message is flooded into the network. Each sensor node estimates its hop count from the sink. When the *DIFFUSION* phase completes, each sensor node gets its level and discovers its parent neighbors and child neighbors. The *ECHO* phase is triggered by the highest level sensor nodes, which are farthest from the sink. *ECHO* messages are created by those nodes and flooded in the network. Hence, the total level count is captured by all the nodes and each node calculates its own operation schedule for each sampling cycle. Fig. 2b shows that sensor nodes in different levels share different schedules. Each node carries out sensing and processing at the beginning of a data sampling cycle. Nodes in different levels transfer data in different time slots within the same sampling cycle. Lower level nodes need to wait for the data from higher level nodes so that data aggregation could be implemented in each level. The sampling cycle interval D_s, data processing time T_p, and the total level count c determine the duration d of data transfer and aggregation for nodes in each level, $d = 2(D_s - T_p)/c$. For each node at the ith level, its data transfer and aggregation process lies in the slot of $[(c-i-1)d/2, (c-i+1)d/2]$.

While more redundancy of the aggregated data has been provided by multipath routing, data duplicates have been introduced at the time of multiple relaying. Therefore, we must employ duplicate-insensitive methods (such as [7] and [22]) to prevent error during data aggregation.

3.2 Three-Dimensional Gradient Data Map

Under the network architecture described in Section 3.1, we propose 3D gradient data map to describe the monitored environment. As mentioned above, the sensor nodes are deployed in a 3D space in the monitoring area and each sensor is responsible for sensing the environmental data within its unit cubic cell (we assume that the data within a unit cube have the similar values). Thus, we can aggregate the cubic cells with similar sensor readings into a cube cluster and construct the gradient data map at each sampling period within the network. The gradient data map consists of different clusters with their own geometric shapes and data distributions. The gradient data map is an approximation of monitored environment and reflects environmental data distribution at each sampling period.

We employ data aggregation in each sampling process and create partial gradient data maps from sensor readings. The partial data maps are merged as much as possible along the paths from sensors to the sink. At the sink, the gradient data map is built from a set of partial gradient data maps. Along with a sequence of sampling, a time series of approximated 3D gradient data maps is constructed at the sink, on which event detection is performed. Fig. 3 exhibits a partial gradient data map including three different cube clusters. We can simply use the average value of all sensor readings in the cube cluster to approximate the data of this cluster during the aggregation. However, that approach introduces large approximating errors. In our gradient data map, we compute the data distribution f of each cube cluster and represent each cube

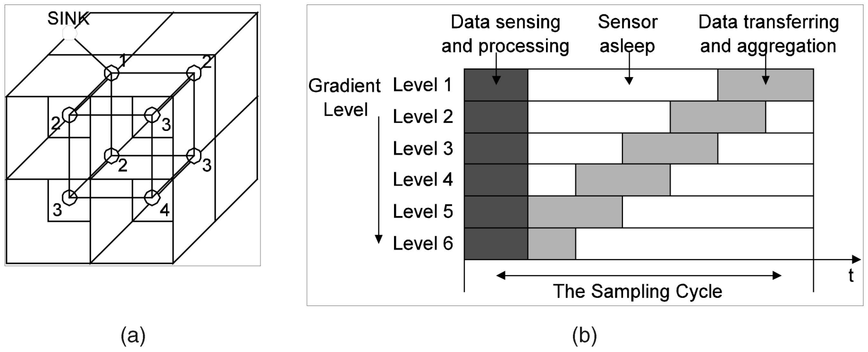


Fig. 2. (a) The multilevel network architecture. (b) Node schedules in this network.

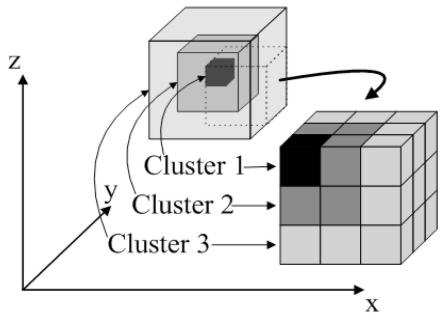


Fig. 3. Three-dimensional space gradient data map.

cluster by the geometric structure, called space OP. By manipulating two parameters of OP, which can be simply transmitted with little bandwidth, the sensor nodes are collaborated to construct the gradient data map in an in-network manner. The key operations in the aggregation process are estimating the similarity of different OP and merging the similar OP at each sensor node.

3.3 Space OP Model and Gradient Map Construction

We use the *space OP* model to describe different cube clusters. OP can capture the data distribution in 3D cubic space and only requires few parameter settings. The OP was first introduced in Constructive Solid Geometry (CSG). Aguilera and Ayala investigated the characteristics of OP [1] and presented the geometric models to represent OP as well as some basic geometric operations, which are summarized as follows:

Definition 3.1. *OP are polyhedra with all faces oriented in three orthogonal directions.*

In OP, all planes and lines are parallel to three orthogonal axes. The number of incident edges for each vertex can be only three, four, or six, which is referred as V3, V4, or V6, respectively [1]. An Extreme Vertices (EV) model has been proposed to represent OP.

Definition 3.2. *The EV model for OP is defined as a model that only stores all V3 vertices.*

Aguilera and Ayala proved that the EV model is a valid *B-Rep* model, i.e., it is complete and compact in the sense of geometry. Furthermore, they proposed the ABC-sorted EV model, which provides computational convenience for geometric operations.

Definition 3.3. *An ABC-sorted EV model is an EV model where V3 vertices are sorted first by coordinate A, then by B and then by C.*

Fig. 4 gives an example of the ABC-sorted EV model for the OP. The model is stored as a series of vertices (node 1 to node 16). Based on ABC-sorted EV models, the following geometric operations can be efficiently performed:

1. *Volume calculation*—To calculate the volume of OP. An $O(n)$ algorithm exists by accumulating the strip region between any consecutive different sections, where n is the number of vertices of the OP.
2. *Relationship checking*—To check the relationship of two OP: overlapping, adjacent, or separated. An $O(n)$ algorithm exists by sequentially checking the relationship of the sections of the two OP along some axis, where n is the number of vertices of the OP.
3. *Boolean operations*—To compute the union or intersection or difference of two OP. An $O(n)$ algorithm exists, which sequentially performs Boolean operations on the sections of the two OP along some axis.

Since a cube cluster is composed of multiple cubic cells, the geometric shape of clusters can be well modeled by OP, which is described by the geometric shape of the covered area and a data distribution function in this area. The partial gradient data map stored in each sensor node is represented as a list of OP depicted by the ABC-sorted EV model. The in-network construction of the gradient data map starts from each node sensing its environment and generating the OP model for its own cell. Each node receives the partial maps from all its child neighbors at the time slot of data aggregation. The OP from different partial maps form an active set S_p. Through investigating the relationship among OP within S_p, the sensor node estimates the similarity of OP and merges the *mergeable* OP. Fig. 5 illustrates the possible relationships between two OP. Finally, the partial maps are aggregated into a single map M_f, which includes disjointed OP. The lower level sensor node transfers M_f to its parent neighbors.

Algorithm 1 presents detail steps of partial map generation for each sensor node. A min-heap H is constructed containing possible mergers (line 1). Different OP pairs from

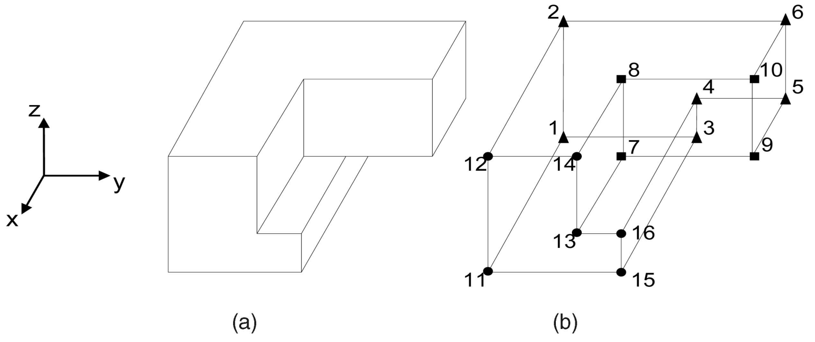


Fig. 4. XYZ-sorted EV model. (a) A hidden line representation of OP. (b) The order number of its XYZ-sorted EV model (V3 vertices on three planes).

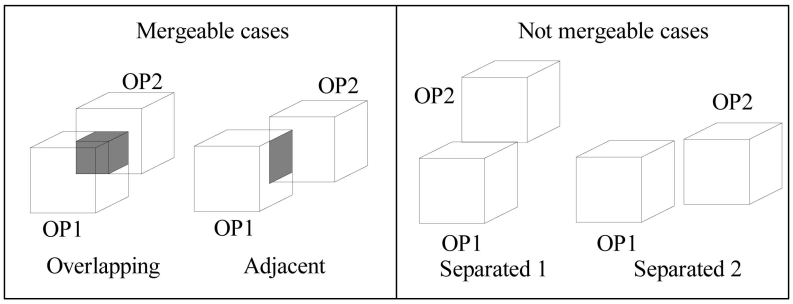


Fig. 5. The possible relationships of two OP.

the active set S_p are checked by function *checkMergeable*, which verifies whether two OP are *mergeable*. Two OP are mergeable when they are overlapping or adjacent and have enough similarity (as will be defined in Section 3.4). The function *createMerge* creates a merger for two mergeable OP and adds it into H. This merger contains a key value indicating the potential of merging the two OP. A smaller key value indicates a higher potential of merging, and this key value is used as the element key in H (lines 2-4). In later sections, we will introduce how the function *checkMergeable* and *createMerger* estimate the similarity of two OP and calculate the key value of the merger, respectively. After all the possible mergers have been added into H, the mergers are extracted from H by increasing order of the key values (line 6). The two OP in each merger are merged into a new OP by the function *merge* (line 7). The two merged OP are deleted from the active set S_p and all the merges related to these OP are deleted from the min-heap H (lines 8-11). The newly created OP is then inserted into the active set H and immediately checked whether it could be further merged (lines 12-15). After all mergeable OP are merged, we process the overlapping but not mergeable OP pairs and remove the intersection region from one of them so that no ambiguous regions exist in our generated map (lines 16-18).

Algorithm 1 Partial Map Generating
Input: the active set S_p
Output: the resulting map M_f

```
 1: construct an empty min-heap $H$, to contain mergers $\langle OP_1, OP_2 \rangle$;
 2: for each OP pair $OP_i$ and $OP_j$ $(i \neq j)$ in $S_p$ do
 3:   if $checkMergeable(OP_i, OP_j)$
 4:     $H.\ add(createMerger(OP_i, OP_j))$;
 5: while not $H.\ empty()$ do
 6:   merger $m\langle OP_1, OP_2 \rangle = H.\ extract()$;
 7:   create $OP = merge(OP_1, OP_2)$;
 8:   for any merger $m$ containing $OP_1$ or $OP_2$ do
 9:     $H.\ delete(m)$;
10:     $S_p.\ delete(OP_1)$;
11:     $S_p.\ delete(OP_2)$;
12:   for each $OP_k$ in $S_p$ do
13:     if $checkMergeable(OP, OP_k)$ 1
14:       $H.\ add(createMerger(OP, OP_k))$;
15:   $S_p.\ add(OP)$;
16: for each overlapping region $R = OP_i \cap OP_j$ in $S_p$ do
17:   $OP_i = OP_i - R$ or $OP_j = OP_j - R$;
18: return $M_f = S_p$;
```

3.4 Aggregation Criteria

To aggregate different partial maps, we need to adopt effective criteria for measuring the similarity of OP so that the resulting partial data map well approximates the actual data map.

The OP model used in our system represents a cluster of cubic cells with similar environmental data. We can use a specific data value to represent the data in the whole OP region, e.g., the average value of all the sensor readings in the OP. In such case, to check the similarity of two OP, we only need to check their representing values. However, a single data value can hardly reflect full-scale environmental conditions in the OP. Moreover, only investigating the representative value of OP will miss the important spatial information. For example, with the same value, OP occupying a larger space is still different from OP holding a smaller space. Thus, we can merge a tiny OP (OP with small space) into a much larger OP (OP with large space) even though their representative data values differ a lot, because the merging simplifies the data map representation without losing much accuracy. However, for the case in which two OP both occupy large spaces, merging them may greatly reduce the accuracy of the resulting gradient data map.

In our design, each OP is associated with a data distribution model, which describes the environmental data within this OP. There have been several techniques proposed to represent the data distribution with compression, such as wavelet transformation [26], histogram-based model [8], [10], and so forth. In this work, we choose to adopt linear regression (LR) model, which incurs constant communication overhead and facilitates the aggregation of OP. A function $v = f(x, y, z)$ is employed to approximate the data value in each spot in the OP, where x, y, and z correspond to the spot coordinate in the 3D space. Polynomial models can be utilized to formulate this approximation function. To reduce the computational overhead for resource constraint sensor nodes, we adopt the linear model $f(x, y, z) = c_0 + c_1x + c_2y + c_3z$, where the data distribution is approximated by a hyperplane in the 4D space built on $<x, y, z, v>$. In the sampling period, each node first computes its initial model for its cubic cell from its sensor reading. During the aggregation, a linear model is built by conducting LR over the whole OP area. For OP containing n cubes, n values are extracted for all cubes. Thus, we can get n 4-tuples $<x_1, y_1, z_1, v_1>$, $<x_2, y_2, z_2, v_2>, \dots, <x_n, y_n, z_n, v_n>$ from which we compute the coefficients of the linear model by solving the following equation:

$$Aw = b$$

$$A = \begin{pmatrix} 1 & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} z_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} x_i z_i \\ \sum_{i=1}^{n} y_i & \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 & \sum_{i=1}^{n} y_i z_i \\ \sum_{i=1}^{n} z_i & \sum_{i=1}^{n} x_i z_i & \sum_{i=1}^{n} y_i z_i & \sum_{i=1}^{n} z_i^2 \end{pmatrix}, \tag{1}$$

$$w = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}, \ b = \begin{pmatrix} \sum_{i=1}^{n} v_i \\ \sum_{i=1}^{n} v_i x_i \\ \sum_{i=1}^{n} v_i y_i \\ \sum_{i=1}^{n} v_i z_i \end{pmatrix}.$$

The parameters A, w, and b of the LR model are integrated and transmitted with the OP in the aggregation process, which take $O(1)$ cost to represent the data distribution over the OP. When merging two OP, OP_i and OP_j, we can compute the LR model of the resulting OP_{ij} from the LR models of OP_i and OP_j. By summing A_i and A_j, we get A_{ij}, so does b_{ij} with respect to b_i and b_j. The coefficients w_{ij} can be derived from the generated A_{ij} and b_{ij}. Therefore, we only transmit the parameters of the two matrices, instead of sampling the $<x, y, z, v>$ tuples to construct our LR model, which induces more overhead. The similarity of two OP is estimated based on the linear models of OP. An estimated error bound ε_{ij} is computed when aggregating two different OP by the following formula:

$$\varepsilon_{ij} = \frac{(1+\varepsilon_i)\Delta_i + (1+\varepsilon_j)\Delta_j}{R_i + R_j}, \tag{2}$$

where Δ_i represents the difference between the cumulates of f_{ij} and f_i on OP_i, and Δ_j represents the difference between the cumulates of f_{ij} and f_j on OP_j, i.e.,

$$\Delta_i = \left| \iiint_{OP_i} \big(f_{ij}(x,y,z) - f_i(x,y,z)\big) d\delta \right|,$$
$$\Delta_j = \left| \iiint_{OP_j} \big(f_{ij}(x,y,z) - f_j(x,y,z)\big) d\delta \right|. \tag{3}$$

ε_i and ε_j are the error bounds for f_i on OP_i and f_j on OP_j. Thus, $(1+\varepsilon_i)\Delta_i + (1+\varepsilon_j)\Delta_j$ gives the maximum difference when we substitute the former LR models on OP_i and OP_j with the aggregated one. R_i and R_j represent the cumulates of f_i on OP_i and f_j on OP_j, respectively, i.e.,

$$R_i = \iiint_{OP_i} f_i(x,y,z) d\delta, \ R_j = \iiint_{OP_j} f_j(x,y,z) d\delta. \tag{4}$$

This formula computes the error bound ε_{ij} after the aggregation, and it is then evaluated by a user-defined error bound ε. Only when ε_{ij} is not greater than ε, two OP are *mergeable*. Note that, in the above formula, the error bound is computed in a weighted manner, where the OP volume is the weight factor.

Based on the estimation of the LR model, we consider the data value as well as the volume of the OP when merging two different OP regions. We list all the functions *checkMergeable*, *createMerger*, and *merge* for merging two OP in Algorithm 2. The function *checkMergeable* first checks the relationship of the two OP. If they are overlapping or adjacent, the function further checks whether the error induced by merging is tolerable. The function *createMerge* computes the benefit of merging two OP, $\varepsilon_{ij}/\varphi_{ij}$, the error bound over reduced region, and takes it as the index key of the merger in the min-heap H in Algorithm 1. Thus, the merging with less error and larger reduced region will be conducted earlier. The function *merge* merges two OP by combining their regions and computing the new LR model for the resulting OP.

Algorithm 2 Merging Manipulations

```
Function $checkMergeable(OP_i, OP_j)$
1: if $OP_i$ and $OP_j$ are overlapping or adjacent
2:   compute $\varepsilon_{ij}$;
3:   if $\varepsilon_{ij} \le \varepsilon$
4:     return $TRUE$;
Function $createMerger(OP_i, OP_j)$
1: $\varphi_{ij} = volume(OP_i) + volume(OP_j) - volume(OP_i \cup OP_j)$;
2: if $\varphi_{ij} > 0$
3:   return merger $<OP_1, OP_2>$ with key $\varepsilon_{ij}/\varphi_{ij}$;
Function $merge(OP_1, OP_2)$
1: compute $OP = OP_i \cup OP_j$;
2: $A = Ai + Aj$;
3: $b = bi + bj$;
4: compute $w$ by $Aw = b$;
5: return $OP$ with $A$, $b$, and $w$;
```

3.5 Incremental Update of Gradient Data Map

When carrying out a sequential sampling, each sensor node needs to continuously update its partial gradient data map to reveal environmental status in real time. An effective and efficient criterion for map update can offer the ability of real-time monitoring while minimizing computational and communicational cost. Simply reconstructing a new data map in each sampling period is an easy yet costly approach, which may cause sensor nodes to quickly deplete their power due to the heavy communication traffic. Indeed, the consecutively constructed data maps vary not much due to the rareness of environmental events [9]. Thus, we employ an incremental method to update the partial gradient data map.

In the aggregation process, each sensor node keeps its previously constructed data map as M_p. The node receives the update units $U_1, U_2, \ldots, U_n$ from its child neighbors in each updating phase. The updated units are in fact OP with different data values out of the error bound ε from their previous statuses. Upon receiving these updated units, the node first constructs an update map M_u by aggregating these updated units. This process is similar to the aggregation of partial maps, with the same aggregation criteria. Before sending out the update map M_u, the node diminishes some reducible units to reduce the amount of sending data. In the diminishing phase, the update map M_u is compared with the previous data map M_p, and the similarity estimation is conducted for the OP in M_u, which can be enclosed by OP in M_p. Considering OP P_i in M_u enclosed by some OP P_j in M_p, we can calculate the estimated error bound ε_{ij} by (2) and evaluate it with a predefined error bound ε. If $\varepsilon_{ij} \leq \varepsilon$, the OP P_i is considered reducible and removed from M_u, since updating this OP will not bring significant impact to the final data map. After removing all reducible OP from M_u, the node sends the diminished update map as an update unit U to its parent neighbors and updates its current data map M_c by combining M_u and M_p in the manner described in Section 3.4.

Algorithm 3 illustrates this process. The algorithm receives the previous data map M_p and the updating units $U_1, U_2, \ldots, U_n$ from the child neighbors as inputs and outputs the updating unit U it sends to its parent neighbors and the current data map M_c it maintains for recording current environment status. The current data map M_c will be in the next round update taken as the previous data map M_p. This algorithm contains two phases. In the first phase (lines 1-8), the algorithm constructs the updating unit U by generating a partial map from the input OP of $U_1, U_2, \ldots, U_n$. Specifically, the algorithm filters the scrapped small OP generated from small environment variations or measurement noises by merging them into previous large and stable OP, which saves unnecessary communication cost for representing them (lines 3-7). In the second phase (lines 9-10), the algorithm combines the previous data map and the update partial map by merging their OP and obtaining the current data map M_c, reflecting current environment status.

Algorithm 3 Data Map Updating
Input: the previous data map M_p and the updating units $U_1, U_2, \ldots, U_n$
Output: the updating unit U and the current data map M_c

```
1:  active set $S_p = \{\text{all OP} \in U_i | i = 1, \ldots, n\}$;
2:  generate update map $M_u$ from $S_p$ (Algorithm 1);
3:  for each OP$_i$ in $M_u$ do
4:    for each OP$_j$ in $M_p$ do
5:    if OP$_i \subseteq$ OP$_j$
6:      if $checkMergeable(OP_i, OP_j)$
7:      drop $OP_i$ from $M_u$;
8:  $U = M_u$;
9:  active set $S'_p = \{\text{all OP} \in M_u \text{ and } M_p\}$;
10: generate current data map $M_c$ from $S_p$ (Algorithm 1);
11: return $U$ and $M_c$;
```

The data map updating technique exploits the stabilization of monitored environment where events rarely happen, so the communication cost could be largely saved without losing accuracy. Our experiments in Section 5 further prove this.

3.6 Adapting Random Sensor Deployments

To adapt a broader scope of WSN application scenarios, we extend our algorithm into random deployment of sensor networks. In such random sensor deployment, when the cubic grid is established, some of the cubic cells may contain multiple sensor nodes and some of them may be empty. For the cells with multiple sensors, our aggregation algorithm automatically aggregates their readings into the linear function f. However, those empty cells make the constructed gradient map incomplete, with OP of undetermined values. This problem also occurs under node failures and link losses.

We employ spatial interpolation on the server side to recover the complete map from the collected incomplete map. Each undetermined OP is estimated by spatial interpolation around its surrounding OP and gets merged into the most similar neighbor OP. By this means, we can finally construct a complete gradient data map for the random deployed sensor network. Algorithm 4 illustrates this procedure. Algorithm 4 is executed on the server side after the incomplete gradient data map has been collected from the network. Thus, the interpolation will not influence all in-network procedures described in previous sections.

Algorithm 4 Gradient Map Recovery
Input: the incomplete gradient data map M_{ic}
Output: the recovered complete gradient data map M_c

```
1:  for each undetermined OP region $P_i$ in $M_{ic}$ do
2:      compute $w_i$ from interpolation;
3:  $\varepsilon_i = \varepsilon$;
4:    for each adjacent OP $P_j$ do
5:      compute $\varepsilon_{ij}$;
6:      if $\varepsilon_{ij} < \varepsilon_i$
7:        $\varepsilon_i = \varepsilon_{ij}$;
8:      $k = j$;
9:    $merge(P_i, P_k)$;
10: return $M_c$;
```

4 Event Detection

When the sink receives the aggregated data map, we can perform the event detection based on the exhibited data pattern from the data map. Moreover, the spatiotemporal pattern revealed from the series of data maps provides us the dynamic progress of the event, which helps capture the event developments. In this section, we describe the event feature patterns and propose a formal method of utilizing the predefined feature patterns to detect a specific event.

In previous discussions, the term "data map" refers to the constructed gradient data map in some sampling period. For the purpose of event detection, the spatiotemporal data pattern is often investigated over a time series of data maps. For the convenience of description, without specification, we will later use "data map" referring to the spatiotemporal data map consisting of a time series of received data maps. Each data map in this series is referred to as a data map "snapshot."

4.1 Event Feature Patterns

The event feature pattern F is defined as a time series of snapshots L on the data map of some environmental attribute and a set of relationship R among them. $L = \{S_0, S_1, \ldots, S_n\}$, where S_i is a snapshot (t_i, M_i) on the data map composed of the time label t_i and current data map M_i. Here, $\Delta t = t_{i+1} - t_i$ is the sampling interval between two consecutive snapshots and the data map M_i consists of different OP $(P_{i1}, P_{i2}, \ldots, P_{im})$. Different OP are associated with different data values v_{ij}. The relationship R specifies the event feature pattern on series L. R describes the spatial relationship R_S by regulating the relationships $R(P_{ik}, P_{il})$ between different OP on the data map snapshot M_i and the temporal relationship R_T by regulating the relationships $R(M_i, M_j)$ between different data map snapshots.

The above definition of F describes the spatiotemporal trends on the data map of the specified event. By comparing the obtained aggregated data maps with the predefined feature patterns, we could accurately detect the ongoing development of certain events. We illustrate the event feature pattern in detail by describing two sample events as well as their specified feature patterns.

Spreading Event. For the event of gas leakage, as the gas spreads from the source spot, the distribution of the gas density follows the single source spreading event model in the data map of gas density. Spatially, in the data map snapshots, the leakage source bears the highest gas density value and the value falls along all directions from the source spot. Temporally, as time passes, the abnormal region expands and the gas density rises within the whole region. According to above observed features, we specify the spreading event feature pattern as follows:

The spreading event feature pattern F^s is determined by the user-specified snapshot series L^s and the relationship R^s on them, which are customized by the users: 1) T is a user-specified event duration, which defines the time interval between the first snapshot S_0 and last snapshot S_n in the snapshot series L^s, i.e., $T = t_n - t_0$. 2) The spatial relationship R^s_S regulates a series of nesting OP $\{P_{i1}, P_{i2}, \ldots, P_{iNi}\}$ for each M_i. N_i $(0 \leq i \leq n)$ is the user-specified spreading level, which specifies the number of nesting OP. P_{ik} occupies the hole region in P_{ik+1}. The difference of data values associated with the two OP $v_{ik} - v_{ik+1}$ is bounded by the user-specified degrading bound $[D_L, D_H]$, and the ratio of their volumes δ_k/δ_{k+1} is bounded by the user-specified scaling bound $[f_L, f_H]$, $(0 < f_L < f_H < 1)$. 3) For the temporal relationship R^s_T, the variation of data values between two consecutive data maps M_i and M_{i+1} is regulated by the user-specified variation factor vf, such that the data value variation for any spot p in the event region between M_i and M_{i+1} is $v_{pi} - v_{pi+1} \geq vf$. Another user-specified spreading factor sf $(0 < sf < 1)$ constrains the ratio of the volumes of event regions (composed of the nesting OP) in consecutive data maps M_i and M_{i+1}, such that $E_i/E_{i+1} \leq sf$. This factor indicates the spreading speed of the source. Fig. 6a illustrates the spreading event.

Fault Event. The fault event corresponds to those breaking out changes in terms of some attribute value. For instance, the underground water seepage can be categorized as a fault event, which induces a large flooded region on the tunnel floor, disturbs the normal work, and damages the working equipment. Moreover, in severe situations, the water destroys the tunnel structure and threatens the life of miners. For the fault event, the sensory readings in the fault region largely differ from those in the normal region. So, the event detection can be featured as a 0/1 detection on the data map by setting appropriate thresholds for two regions. However, this cannot be achieved by simply setting thresholds at individual sensors, because what we need is a big picture of the entire field and we aim to find the two

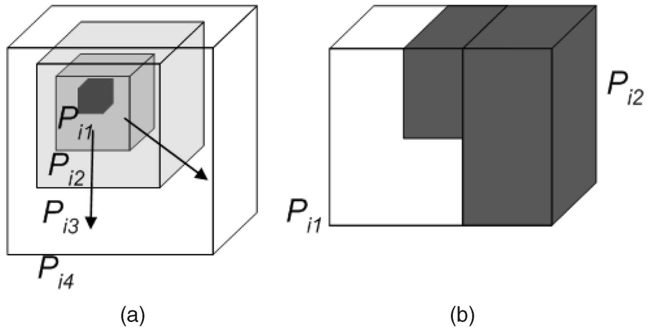


Fig. 6. Illustration of sample event feature patterns. (a) The spreading event pattern. (b) The fault event pattern.

distinct regions in a macroscopic level. This can only be achieved after a global data map of the field is obtained. We, on the server side, thus are able to observe the data pattern and detect the event. Any individual sensor node, without enough information on the global data distribution, cannot draw its local judgement about the event.

By specifying the relationship between the 0 attribute OP and the 1 attribute OP on the data maps, we can describe the feature pattern of fault event. The fault event feature pattern F^f is determined by the user-specified snapshot series L^f and the relationship R^f on them, which are customized by the users: 1) T is a user-specified event duration, which constrains that, in the snapshot series L^f, the time interval between the first snapshot S_0 and last snapshot S_n is $t_n - t_0 = T$. 2) The spatial relationship R^f_S regulates two adjacent OP, P_{i1} and P_{i2} in each M_i. P_{i1} is associated with value v_{i1} in the range $[b_1, b_1 + k]$ and P_{i2} with value v_{i2} in the range $[b_2, b_2 + k]$. $b_2 - b_1 \geq \Delta$, where Δ is a user-specified threshold. The volumes of both OP E_1 and E_2 should be larger than a user-specified region size bound E $(E > 0)$, which defines the scale of the event to be detected. Another user-specified parameter S_c sets the lower bound of the coincident plane area shared by the two OP. 3) The temporal relationship R^f_T regulates the event regions in consecutive data maps overlapping at least at a percentage of α, where $0 < \alpha < 1$ is a user-specified confidence factor. Fig. 6b illustrates the fault event.

4.2 Pattern-Based Event Detection

Once the event feature patterns have been specified, the sink continuously processes the received data maps and compares them with the predefined event feature patterns. Once a match between the pattern and the data map is found, the corresponding event is captured. Moreover, by tracking the spatiotemporal feature of the data map series, the development of current event could also be revealed.

We define that an instant snapshot A_t of the data map A matches S_i if and only if the OP in A_t match the OP in M_i and share the same spatial relationships $R(P_{ik}, P_{il})$. We define that the data map A matches pattern F if and only if from time t, there exists a series of map snapshots $\{A_t, A_{t+\Delta t1}, \ldots, A_{t+n\Delta t}\}$ from A, such that any snapshot $A_{t+i\Delta t}$ matches the corresponding feature snapshot S_i and all map snapshots obey the temporal relationships $R(A_{t+i\Delta t}, A_{t+j\Delta t})$.

Algorithm 5 illustrates how the sink processes the matching of data map series with predefined event feature patterns. First, each map snapshot in the data map series $\{A_t, A_{t+\Delta t1}, \ldots, A_{t+n\Delta t}\}$ is compared with the feature pattern snapshot according to the spatial relationship R_S (lines 2-7). Only after all snapshots match R_S, the temporal relationship R_T is checked on this data map series with event duration T (lines 8-13). If R_T holds, the feature pattern matching is achieved, and *TRUE* is returned (line 14). When any mismatch appears, the starting time t of the series is slid by Δt, and a new round of checking is processed (lines 4-7 and 10-13). If no pattern matching has been detected until the possible event time exceeds the duration T_A, *FALSE* is returned, which means no event is detected (lines 5, 6, 11, and 12).

Algorithm 5 Feature Pattern Matching

```
Input: the specified event feature pattern $F =<L, R>$ and
       the received data map series $A$
Output: the matching result (TRUE/FALSE)
 1: $t = 0$;
 2: for each snapshot $S_i$ $(0 \leq i \leq n)$ of $L$ do
 3:   if not $A_{t+i\Delta t}$ matches $S_i$ according to $R_S$
 4:     $t = t + \Delta t$;
 5:     if $t + T > T_A$
 6:       return FALSE;
 7:     goto 2;
 8: for each pair of map snapshots
    $A_{t+i\Delta t}, A_{t+j\Delta t}$ $(0 \leq i, j \leq n)$ do
 9:   if not $R(A_{t+i\Delta t}, A_{t+j\Delta t})$ conforms to $R_T$
10:     $t = t + \Delta t$;
11:     if $t + T > T_A$
12:       return FALSE;
13:     goto 2;
14: return TRUE;
```

5 Performance Evaluation

We conducted a field study by investigating the various environmental conditions in the D.L. Coal Mine. It is one of the most automated coal mines worldwide. We collected different sets of real data in the field and from historic records under normal and exceptional situations. In this section, we investigate the efficacy and efficiency of our proposed event detection mechanism by a trace-driven simulation using synthetic workload generated from the collected raw data.

5.1 Simulation Setup

We simulated the event scenarios in a sensor network with a 3D sensor deployment. The widely used Mica2 motes [13] are presumed as the underlying hardware standard. All numbers are 2-byte integers (including sensor readings and all coefficients). The size limit for the simulated packets is set to 60 bytes. Sensor nodes follow a CSMA strategy in the link layer transmission, and according to our experimental observations in the coal mine [15], when the channel is heavily reused (10+ transmitters or interferers for each transmission link), the maximum throughput for each link is limited below 16 packets per second. Any traffic exceeding this amount will be dropped or collided in the delivery.

An $N \times N \times N$ cubical grid topology is initiated with a sensor node placed at the center of each cubical grid. The parameter N indicates the diameter of this cubical grid network, ranging from 5 to 20 (default value is 10) in our experiments to explore the system scalability under different network sizes. Each sensor node has direct communication links with the six closest neighbor nodes surrounding it. The link quality is measured with the link loss rate q, which is the probability that a packet transmitted along the link gets lost. In our experiment, this parameter varies from 0 percent to 40 percent (default value is 10 percent) to explore the system reliability under different network conditions.

Three types of real-world historic sensory data for the underground environment have been collected in the coal mine as our data trace including gas density, oxygen density, and watery regions. However, due to the constraint on resource and environment, the original data are collected with rough granularity of time scale and incomplete sampling spots in the space scale. Based on the raw data on hand, we generate more detailed synthetic data sets for use in our experiment, which nevertheless mimic the original data characteristics and trends under normal conditions as well as during the period that events happen.

In the experiment, each data set contains all three kinds of environmental data as three different environmental attributes and lasts for 10,000 seconds, while each data sampling cycle interval is 100 seconds. The sample events described in Section 4.1 have been queried over the three different attributes. Event $E1$ refers to the spreading event of gas density, which occurs with gas leakage; Event $E2$ refers to the spreading event over oxygen density, which occurs around the oxygen-enriched spots; Event $E3$ refers to the fault event in watery regions when water osmosis happens. Note that though $E1$ and $E2$ both refer to the spreading events, due to the different principles of gas leakage and oxygen concentration, the two events have different spatio-temporal data trends and their feature patterns differ in the parameter settings. The above events are queried over the data set with a predetermined event frequency f, which depicts the monitoring workload. f is calculated as the duration of events over the total monitoring duration. In our experiment, we vary f from 0 percent to 40 percent (default value is 10 percent) to obtain a full view of the system efficiency under different workload.

We focus on two metrics for performance evaluation. The event *detection accuracy* measures the efficacy of the approach and the network *traffic overhead* tells the efficiency on energy consumption. The event *detection accuracy* is measured by two submetrics, *precision* and *recall*, which have been widely used in IR domain [2]. *Precision* describes the detection precision, which is defined as the percentage of accurately detected real events over all reported events; *recall* describes the detection completeness, which is defined as the percentage of successfully detected events over all occurred events. The network *traffic overhead* is measured by the total amount of messages (bytes) transmitted in the network in the monitoring duration.

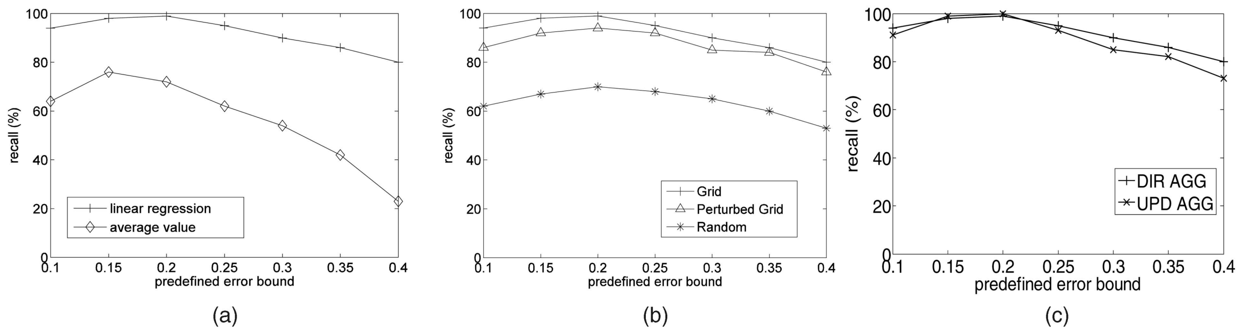


Fig. 7. The accuracy of our aggregation approach with different settings. (a) The detection recall of different aggregation criteria: average value aggregation versus LR. (b) The detection recall of different sensor placement: grid, perturbed grid, and random. (c) The detection recall of direct aggregation versus incremental update of the data map.

To further evaluate the performance gain of our approach, we did a comparative study investigating three possible approaches: 1) AGG: our proposed in-network data map aggregation approach described in Section 3; 2) SAT: the server side aggregation under TAG [19] framework, in which the network is organized into a tree structure and all data are forwarded to the sink. Data map is constructed in the server side. In the SAT approach, each sensor sends and relays data to only one parent node; and 3) SAM: the server side aggregation under multipath routing strategy [22], in which the network is organized into multipath routing style and the data map is constructed in the server side. In the SAM approach, each sensor sends and relays data to multiple parent nodes. The network architecture is the same as that in AGG.

5.2 Simulation Results

Since in all runs of experiments nearly 100 percent detection *precision* is consistently achieved in different approaches under different parameter settings, we omit this metric and only present the performance on *recall* in the experiment result part.

Before the comparative study on our AGG approach with the other two approaches, we first evaluate the performance gain of our approach under different settings.

5.2.1 Evaluating AGG Performance

Fig. 7 presents the recall of event detection under varied error bound ε (Section 3.4). Fig. 7a compares the recall of the aggregation based on LR adopted in AGG approach with the simple average value-based one. For the aggregation based on LR, the error bound refers to the user-defined error bound ε introduced in Section 3.4. For the aggregation based on average values, the error bound refers to the difference ratio between two OP. As the figure shows, the aggregation based on LR provides much higher percentage of recall, overcoming the aggregation based on average values by more than 20 percent. The best recall achievable in the average value-based aggregation is 78 percent when the error bound is set to be 0.15, while the best recall achievable in the LR-based aggregation is nearly 100 percent. This is mainly because the average value-based aggregation does not integrate the volume factor of OP in the aggregation process. It simply compares the average value of two different OP and accepts the merge if they have similar values no matter what their sizes are. Thus, the average value-based aggregation is vulnerable to the noises of the environment data.

Fig. 7b compares the recall in different sensor placements. "Grid" placement refers to the regular placement of sensors into 3D grids, which is the ideal case assumed in this paper. "Perturbed Grid" places sensors into grids but with a small random perturbation on their placement. It represents the most possible situation of manually placing sensors in the practical environment. "Random" placement deploys sensors in a purely random way. For the "random" placement, as the sensors might be deployed unbalanced, the network might be disconnected, so we place necessary relay nodes to connect the entire network. As Fig. 7b shows, the "Grid" placement achieves the highest recall rate, while the "Perturbed Grid" achieves slightly lower recall rate. The "Random" placement suffers from the highly unbalanced sensor samplings and, thus, has the lowest recall rate. Nevertheless, it achieves up to 70 percent detection recall when the error bound is properly set. We believe the "Perturbed Grid" placement of sensors gives the most representative results for the practical situations.

In Fig. 7c, we use DIR AGG to represent the method that crudely does data map aggregation and aggregates the data map in every sampling period. We use UPD AGG to represent the refined method that adopts the technique of data map updating described in Section 3.5. As the tolerable error bound is enlarged, the event detection recall in both methods varies slightly, while both strategies achieve perfect detection recall when the error bound is set in the range of [0.15, 0.2]. This shows that the proposed incremental updating technique does not significantly degrade the event detection accuracy.

Fig. 8 compares the network traffic overhead incurred by the original direct aggregation approach DIR AGG and the incremental update approach UPD AGG (see Section 3.5).

Fig. 8a plots the network traffic overhead for both methods under different error bounds. The traffic overhead decreases as the error bound ε is enlarged, but much more slowly when ε is large. A combined view of this figure and Fig. 7c suggests an optimal choice of the error bound ε at about 0.2, which provides perfect detection recall with comprehensive traffic cost. We adopt this optimal setting of ε for later experiments. Compared with the DIR AGG, the UPD AGG achieves almost the same detection recall but

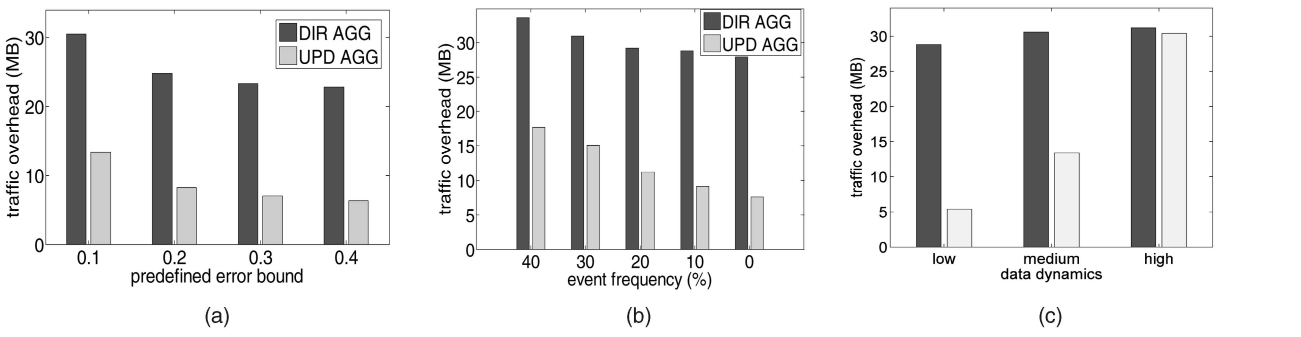


Fig. 8. Network traffic overhead of DIR AGG and UPD AGG under different network settings. (a) Different predefined error bounds. (b) Different event frequencies. (c) Different data dynamics.

with 50 percent less traffic cost than with DIR AGG for all error bound settings.

Fig. 8b shows the traffic saving in UPD AGG method for various event frequencies, from which we observe that a lower event frequency leads to a larger percentage of saving. That is because the data map updating technique adopted in UPD AGG method reduces more unnecessary traffic overhead when events are rare and historic data are aggressively utilized.

Fig. 8c further shows the traffic saving in UPD AGG method under various data dynamics. We group our data set into three different categories: low data dynamics, medium data dynamics, and high data dynamics. The three different groups account for 73 percent, 21 percent, and 6 percent of the data set separately. From the figure, we observe that the UPD AGG method saves more than 80 percent network traffic under low data dynamics and 54 percent network traffic under medium data dynamics. The two situations account for 94 percent of the whole data set.

5.2.2 Comparative Study

We broaden the comparative study among the three different approaches AGG, SAT, and SAM under various parameter settings. Fig. 9 plots the event detection recall of the three approaches. In all cases, our AGG approach achieves the best performance.

As shown in Fig. 9a, as the link loss rate increases, the recall of SAT rapidly drops below 40 percent and tends to 0. SAM approach also bears a bad detection recall. SAM suffers data loss from link loss as well as packet drops and collisions due to the heavy traffic flows. Even when the link loss rate is set to 0, a large amount of data still get lost and lead to a much lower recall of 87 percent, while the other two approaches both achieve nearly 100 percent recall. Our AGG approach demonstrates good tolerability to the network quality. The recall is kept above 60 percent even in a lossy network with up to 40 percent link loss rate.

In Fig. 9b, the network diameter is enlarged to investigate the scalability of three approaches. Again, the SAT approach leads to unacceptable detection recall rate. Our AGG approach keeps high recall rate all along, while the recall of SAM approach drops linearly as the network diameter increases. This is largely because a larger network generates much more network traffics in the SAM approach, leading to more packet drops and collisions.

Fig. 9c plots the detection recall of the three approaches in cases with different event frequencies. We observe that the parameter of event frequency hardly influences the recall rate of the three approaches. All three approaches provide relatively stable recall rates when event frequency is varied, and our AGG approach outperforms the other two. The incremental update in AGG approach provides slightly improved performance when the event frequency decreases.

Fig. 10 plots the network traffic overhead in three approaches. In Fig. 10a, with the increase of the link loss rate, the traffic overhead of all three approaches decreases because of the loss of packets. The SAM approach experiences a faster decrease on the traffic overhead, but the total amount is much larger than the SAT and AGG approaches. The traffic overhead of the SAT approach has a skip decrease when the link loss rate changes from 0 percent to 10 percent; however, from Fig. 9a, we know that this is

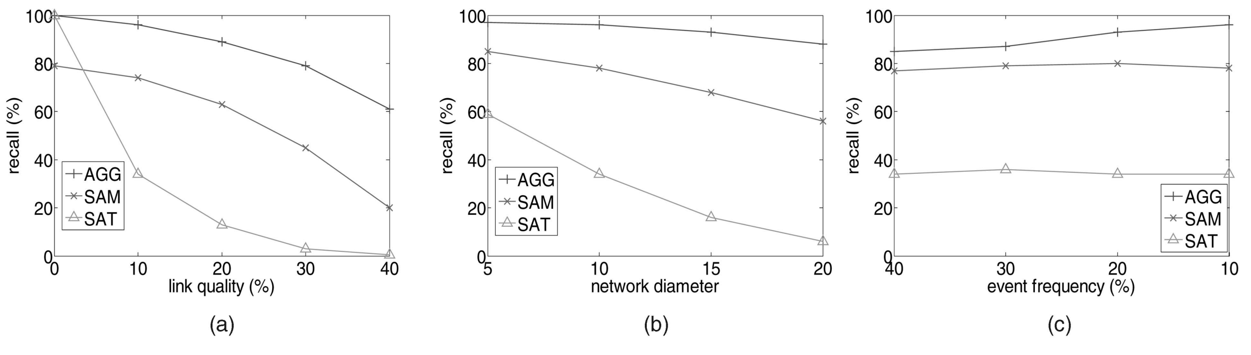


Fig. 9. Event detection recall of the three approaches with different parameter settings. (a) Link quality. (b) Network diameter. (c) Event frequency.

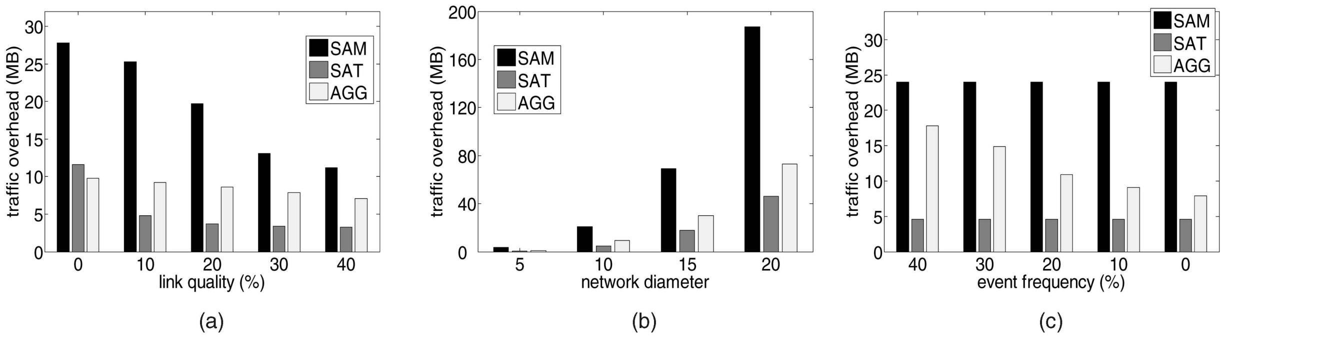


Fig. 10. Network traffic overhead of the three approaches with different parameter settings. (a) Link quality. (b) Network diameter. (c) Event frequency.

because most of the useful information gets lost due to the packet loss. Thus, although SAT has the lowest traffic overhead, it provides nearly unacceptable event detection recall. Fig. 10b shows how network traffic overhead grows as the network size increases. We note that while the SAT and AGG approaches maintain comparatively low traffic overhead against the increase of network diameter, the SAM approach has a dramatic increase of traffic overhead, which greatly constrains its scalability. Fig. 10c shows how the parameter of event frequency affects the three approaches. While the event frequency has little influence on SAM and SAT approaches, the traffic overhead of our AGG approach is reduced as the event frequency decreases, benefiting from the data map updating technique. According to our field investigation in the coal mine, generally, the event frequency in the real world remains low, benefiting the application of our AGG approach.

To summarize, among the three possible approaches, the SAT approach introduces the least traffic overhead but provides the worst, totally unacceptable event detection accuracy; the SAM approach provides somewhat tolerable event detection accuracy but with the most traffic overhead and the worst scalability; our AGG approach provides the best event detection performance with relatively small traffic overhead. We also achieve the best scalability to the network size and tolerability to the network quality in the AGG approach.

6 Conclusions and Future Work

In this paper, we have proposed a nonthreshold-based approach for complex event detection in 3D environment monitoring applications. Other than threshold-based approaches, we have proposed event feature patterns to specify complex events and developed a pattern-based event detection method on the obtained 3D gradient data map. We employ multipath routing architectures to provide robust data delivery and perform in-network aggregation on it to efficiently construct the data map. Space OP model is proposed to describe the environment data distributions. Partial data maps are aggregated by merging OP regions with similar environmental data. The incremental update for the gradient data map explores the usability of historic data while the environment is stable with low frequency of events and further reduces unnecessary data delivery. Our experimental results show the performance gain of our energy-efficient techniques. Moreover, the comparative study with two alternative approaches exhibits that our approach achieves great event detection accuracy with small network traffic overhead.

The future work includes implementing a working system in real-world environment. To carry on, pattern recognition on the obtained gradient data maps with machine learning techniques on historic data samples may provide more efficient detection methods, which will also be included in our future work.

Acknowledgments

This work was supported in part by the Hong Kong RGC Grants HKUST6169/07E and HKUST 6119/07E, by the National High Technology Research and Development Program of China (863 Program) under Grant 2007AA01Z180, by the National Basic Research Program of China (973 Program) under Grant 2006CB303000, by the HKUST Nansha Research Fund NRC06/07.EG01, and by the NSFC Key Project Grants 60533110, 60736016, and 60736013.

References

[1] A. Aguilera and D. Ayala, "Orthogonal Polyhedra as Geometric Bounds in Constructive Solid Geometry," *Proc. Solid and Physical Modeling Symp. (SPM)*, 1997.
[2] R. Baeza-Yates and R.N. Berthier, *Modern Information Retrieval.* Addison-Wesley, 1999.
[3] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the Physical World," *IEEE Personal Comm.*, vol. 7, 2000.
[4] R. Burns, A. Terzis, and M. Franklin, "Design Tools for Sensor-Based Science," *Proc. Third Workshop Embedded Networked Sensors (EmNets)*, 2006.
[5] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using Predictable Mobility for Power Reduction in Sensor Networks," *Proc. Second IEEE/ACM Int'l Workshop Information Processing in Sensor Networks (IPSN)*, 2003.
[6] W. Choi, P. Shah, and S.K. Das, "A Framework for Energy-Saving Data Gathering Using Two-Phase Clustering in Wireless Sensor Networks," *Proc. First IEEE Ann. Int'l Conf. Mobile and Ubiquitous Systems (MobiQuitous)*, 2004.
[7] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate Aggregation Techniques for Sensor Databases," *Proc. 20th IEEE Int'l Conf. Data Eng. (ICDE)*, 2004.
[8] D. Donjerkovic, Y. Loannidis, and R. Ramakrishnan, "Dynamic Histograms: Capturing Evolving Data Sets," *Proc. 16th IEEE Int'l Conf. Data Eng. (ICDE)*, 2000.
[9] 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," *Proc. Fourth IEEE/ACM Int'l Symp. Information Processing in Sensor Networks (IPSN)*, 2005.

[10] F. Furfaro, G.M. Mazzeo, and C. Sriangelo, "Exploiting Cluster Analysis for Constructing Multi-Dimensional Histograms on Both Static and Evolving Data," *Proc. 10th Int'l Conf. Extending Database Technology (EDBT),* 2006.

[11] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Micro-sensor Networks," *Proc. 33rd Hawaii Int'l Conf. System Sciences (HICSS),* 2000.

[12] J.M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond Average: Toward Sophisticated Sensing with Queries," *Proc. Second IEEE/ACM Int'l Workshop Information Processing in Sensor Networks (IPSN),* 2003.

[13] J. Hill and D. Culler, "Mica: A Wireless Platform for Deeply Embedded Networks," *IEEE Micro,* vol. 22, pp. 12-24, 2002.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom,* 2000.

[15] M. Li and Y. Liu, "Underground Structure Monitoring with Wireless Sensor Networks," *Proc. Sixth IEEE/ACM Int'l Symp. Information Processing in Sensor Networks (IPSN),* 2007.

[16] S. Li, Y. Lin, S. Son, J. Stankovic, and Y. Wei, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks," *Telecomm. Systems J.,* vol. 26, 2004.

[17] Z. Li and B. Li, "Loss Inference in Wireless Sensor Networks Based on Data Aggregation," *Proc. Third IEEE/ACM Int'l Symp. Information Processing in Sensor Networks (IPSN),* 2004.

[18] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data Gathering in Sensor Networks Using Energy Delay Metric," *Proc. IPDPS Workshops,* 2001.

[19] S. Madden, M.J. Franklin, and J.M. Hellerstein, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," *Proc. Fifth Symp. Operating System Design and Implementation (OSDI),* 2002.

[20] X. Meng, T. Nandagopal, L. Li, and S. Lu, "Contour Maps: Monitoring and Diagnosis in Sensor Networks," *Computer Networks,* 2006.

[21] V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N.B. Shroff, "Design of Surveillance Sensor Grids with a Lifetime Constraint," *Proc. First European Workshop Wireless Sensor Networks (EWSN),* 2004.

[22] S. Nath, P.B. Gibbons, S. Seshan, and Z.R. Anderson, "Synopsis Diffusion for Robust Aggregation in Sensor Networks," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2004.

[23] S.-J. Park, R. Vedantham, R. Sivakumar, and I.F. Akyildiz, "A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks," *Proc. ACM MobiHoc,* 2004.

[24] I. Solis and K. Obraczka, "Efficient Continuous Mapping in Sensor Networks Using Isolines," *Proc. Second IEEE Ann. Int'l Conf. Mobile and Ubiquitous Systems (MobiQuitous),* 2005.

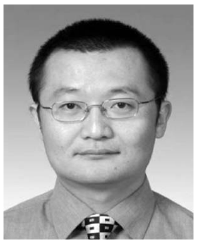
[25] Y. Tian, E. Ekici, and F. Ozguner, "Energy-Constrained Task Mapping and Scheduling in Wireless Sensor Networks," *Proc. Workshop Resource Provisioning and Management in Sensor Networks (RPMSN),* 2005.

[26] N. Xu et al., "A Wireless Sensor Network for Structural Monitoring," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2004.

[27] M. Younis, P. Munshi, and E. Al-Shaer, "Architecture for Efficient Monitoring and Management of Sensor Networks," *Proc. Sixth IFIP/IEEE Int'l Conf. Management of Multimedia Networks and Services (MMNS),* 2003.

Mo Li received the BS degree in computer science and technology from Tsinghua University, Beijing, in 2004. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include wireless sensor networks, pervasive computing, network security, and peer-to-peer computing. He is a member of the IEEE.

Yunhao Liu received the BS degree in automation from Tsinghua University, Beijing, in 1995, the MA degree from Beijing Foreign Studies University, Beijing, in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University in 2003 and 2004, respectively. He is currently with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He is also an adjunct professor at Xi'an Jiaotong University, Jilin University, and Ocean University of China. His research interests include wireless sensor network, peer-to-peer computing, and pervasive computing. He is a member of the ACM and a senior member of the IEEE.



Lei Chen received the BS degree in computer science and engineering from Tianjin University, Tianjin, China, in 1994, the MA degree from the Asian Institute of Technology, Bangkok, Thailand in 1997, and the PhD degree in computer science from the University of Waterloo, Ontario, Canada, in 2005. He is currently an assistant professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include multimedia databases, sensor databases, peer-to-peer databases, and probabilistic databases. He is a member of the ACM and the IEEE.