Tracking Uncertain Shapes with Probabilistic Bounds in Sensor Networks

Besim Avci, Goce Trajcevski(✉), and Peter Scheuermann

Department of Electrical Engineering and Computer Science,
Northwestern University, Evanston, USA
{besim,goce,peters}@eecs.northwestern.edu

Abstract. We address the problem of balancing trade-off between the (im)precision of the answer to evolving spatial queries and efficiency of their processing in Wireless Sensor Networks (WSN). Specifically, we are interested in the boundaries of a shape in which all the sensors' readings satisfy a certain criteria. Given the evolution of the underlying sensed phenomenon, the boundaries of the shape(s) will also evolve over time. To avoid constantly updating the individual sensor-readings to a dedicated sink, we propose a distributed methodology where the accuracy of the answer is guaranteed within probabilistic bounds. We present linguistic constructs for the user to express the desired probabilistic guarantees in the query's syntax, along with the corresponding implementations. Our experiments demonstrate that the proposed methodology provides over 25 % savings in energy spent on communication in the WSN.

1 Introduction

A Wireless Sensor Network (WSN) consists of hundreds, even thousands of tiny devices, called nodes, capable of sensing a particular environmental value (temperature, humidity, etc.), performing basic computations and communicating with other nodes via wireless medium [1]. WSNs have become an enabling technology for applications in various domains of societal relevance, e.g., environmental monitoring, health care, structural safety assurances, tracking – to name but a few. Given that the nodes (also called motes) may be deployed in harsh or inaccessible environments, the efficient use of their battery power is one of the major objectives in every application/protocol design, in order to prolong the operational lifetime of the WSN.

The problem of efficient processing of continuous queries has been addressed in the database literature [5,16,18,24], and the distinct context of WSNs had its impact on what energy-efficient processing of such queries is about [17,25]. However, previous research attempts trying to tackle spatial queries pertaining to two-dimensional evolving shapes are underwhelming. A few research attempts propose temporal boundary detection schemes [3,10,26] and, although there is a

G. Trajcevski—Research supported by NSF – CNS 0910952 and III 1213038, and ONR – N00014-14-1-0215.

J. Pokorný et al. (Eds.): ADBIS 2016, LNCS 9809, pp. 338–351, 2016.
DOI: 10.1007/978-3-319-44039-2_23

consensus that one needs to be aware of the uncertainty – there are no systematic approaches that will capture the notion of uncertainty and couple it with the (energy) efficient processing of detecting/tracking evolving spatial shapes.

In traditional TinySQL-like systems, users indicate with the query-syntax what kind of data they would like to fetch, what sort of functions to apply on the data and, most importantly, how frequently they would like to retrieve the relevant information [17]. If query is responded too frequently, network resources are drained quicker – but if query responses are returned infrequently, then the user's view of the (evolution of the) phenomenon may be obsolete. In addition, quite often the users are interested to know the "map" of the spatial distribution of the underlying phenomenon, instead of a collection of individual sensor readings at selected locations [22]. Numerous works have tried to tackle the problem of efficient incorporation and management of uncertainty in WSN queries [7,9], along with the continuity aspect of the changes in the monitored phenomena [17,18]. Complementary to these, there are works related to 2D boundary detection, both from the perspective of iso-contour of values read, as well as communication holes [6,8,13]. The main motivation for this work is based on the observation that, to the best of our knowledge, there has been no work that would seamlessly fuse the probabilistic aspects of the sensed data and the boundary of the evolving shapes representing contiguous regions in which sensors reading exceed a desired threshold with a certain probability. Towards that, our main contribution can be summarized as follows:

- We develop a shape detection scheme for spatial data summaries with probabilistic bounds by discretizing the space and applying Bayesian filtering.
- We provide both linguistic constructs and efficient in-network algorithmic implementation for processing the novel predicates. We enable users to choose adaptive update frequencies and data granularity in our query model.
- We present a query management scheme that achieves a balance between responding to queries more frequently if underlying phenomena are changing rapidly or by responding with a predefined interval, where query answer is valid for a longer period of time.

The rest of the paper is structured as follows. Section 2 lays out the preliminaries, notation and introduces the syntactic elements of our proposed query language. Section 3 explains the details of the system design and provides the methodology for detecting the boundaries that is amenable to efficient probabilistic updates. Section 4 presents the experimental evaluation of our work. Finally, Sect. 5 gives the conclusion and outlines the possible directions for future work.

2 Basic Queries and Data Model

We assume that a WSN consists of a collection $SN = \{sn_1, sn_2, \ldots, sn_k\}$ of k nodes, each of which is aware of its location in a suitably selected coordinate system [1].

Query Model: Several aspects of spatial queries pertaining to 2D shapes detection have been tackled in the WSN literature: boundary detection [8], isocontour construction [6], hole detection [13], etc. Our focus is on detecting the boundary of "important events" spanning a 2D region, with user-specified parameters of the events of interest. Given the energy limitations of the sensor motes, no WSN query is truly continuous in the absolute sense, but is rather a sequence of discrete snapshots over time. When users pose a query to a WSN, they specify a certain sampling period for the desired frequency. The basic SQL-like querying in WSNs is provided by the TinySQL [17] and it caters to two basic scenarios: (1) periodic sampling – as indicated in line #5 in Listing 1.1; and (2) event-based queries, provided by the TinyDB approach for more efficient query processing, when the code that generates the events is compiled into the sensor nodes beforehand – shown in Listing 1.2.

Listing 1.1. Continuous Query

```
SELECT count(*)
FROM sensors, rlight
WHERE sensors.nid=rlight.nid
AND sensors.light<rlight.light
PERIOD 2 s
```

Listing 1.2. Event-based Query

```
ON EVENT radiation-leak(loc)
SELECT Sensor.value, Sensor.loc
FROM Sensors
WHERE Sensors.value>1200
PERIOD 100 s
```

The sampling period imposes a natural trade-off: more frequent samplings (and reporting) deplete the energy faster, while less frequent ones may render the data obsolete and miss some significant changes. However, the information gain from reporting that the temperature readings are $20 \pm 0.5°\text{C}$ every 10 s for 10 min – if the acceptable level of uncertainty is $\pm 3°\text{C}$ – is same as sending only two readings – at the beginning and the end of the 10 min interval, thereby saving 598 transmissions. Thus, by incorporating an extra, explicitly specified parameter of a (relative) "significant change", our approach dynamically adapts the consumption of resources to the fluctuations in the sensed values.

In our earlier work, we proposed predicates pertaining to shapes and objects trajectories along with their in-network detection [3,21]. In a similar fashion, our focal point in this work is spatial events that are covering two dimensional regions, with a consideration of uncertainty. A query language that is closest to our desiderata is the Event Query Language (EQL) [2], defined by separating the events into several statements:

- *Event Statement:* conditions to recognize (parameterized) events
- *Detection Statement:* rules specifying how and where to detect an event
- *Tracking Statement:* rules specifying how to track an event
- *Query Statement:* syntax for expressing queries on events.

An example of EQL syntax is shown in Listing 1.3, corresponding to a scenario for tracking a gas cloud, initiated by detecting a composite event corresponding to three phenomena (light gas, temperature and oxygen). In this work we provide a few modifications and propose the language Evolving Shapes Event Query Language (ES-EQL). The main modifications are two-fold: Firstly, ES-EQL does not use an explicit *Tracking Statement* since, by default, we make the WSN track the events of interest. Moreover, our detection methodology differs from what is proposed in [2] significantly enough so that we cannot adopt the tracking statement component as such. Secondly, we augment the *Detection Statement* with a clause called *EVOLUTION*, which defines the update interval in conjunction with *EVERY* clause, and a *WITH GRANULARITY* clause for users to specify the data granularity. An exemplary ES-EQL query that can be compared EQL syntax is shown in Listing 1.4.

Listing 1.3. Sample EQL Statement

```
DEFINE EVENT GasCloud
SIZE: 3hops
AS: Avg(Light) as lightGasAvg,
WHERE: lightGasAvg < 50
 AND tempAvg >40
 AND oxygenAvg < 60

DEFINE DETECTION for GasCloud
ON REGION:          Explosion
EVERY:              1000

DEFINE TRACKING for GasCloud
EVOLUTION:          1hop
EVERY:              1000
TIMEOUT:            5m

SELECT   Position, Speed,
   oxygenAvg
FROM     GasCloud
WHERE    oxygenAvg >50
```

Listing 1.4. Modified ES-SQL Version

```
DEFINE EVENT   Fire
SIZE   500
AS Min(Probability) as
    MinCellProbability
WHERE   Temperature > 200
 AND Probability > 0.7

DEFINE DETECTION for      Fire
ON REGION           All
WITH GRANULARITY   256
EVERY              60
EVOLUTION          0.2

SELECT   EventImage
FROM     Fire
WHERE    MinCellProbability <0.75
```

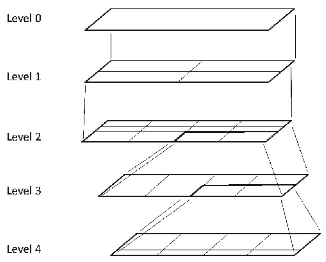
The first statement in Listing 1.4 defines a fire event with the parameters being: size of the event is 500 ft^2, temperature readings for each unit cell above 200°F, and the probability of each cell readings being above 200°F is 0.7. If multiple sensors are located within a particular cell (for a given granularity of the division of space of interest) then the probability of the temperature value being ≥200°F in an infinitesimally small region within that cell is >0.75. Then the detection scheme for the *Fire* event will be carried out on the whole field with data granularity of 256 cells. Afterwards, reporting interval is specified as 60 s and the evolution parameter of 20 % – instructing the system to update the answer either regularly within 60-s intervals or in case of occurrence of 20 % change in the event. Finally a query statement is issued, with fetching an "image" of the event (in fact a 2D data grid-structure that can be converted to an image), from the fire regions where the minimum probability in a unit cell is ≥75 %.

Now the challenge becomes how to identify what constitutes a *significant change* (evolution) in an event. WSNs sample the environment and communicate in discrete time intervals called epochs [17] so, the evolution of the shapes between epochs is also discrete. Significant change, or evolution, can be attributed to several aspects of changes in an existing shape: – its probability; – its size (area); – or a combination of both. The evolution in the probability of a shape is the positive or negative change in its certainty. If a shape becomes more certain than its last-reported version by queried amount, then it means that it has evolved. Another source of a significant change is the *area evolution*. When the area of the shape (regardless of its probability-value) changes by a certain percentage – stated in the respective query – then that shape is considered to have significantly changed. Lastly, over time both the boundaries of the shape as well as the confidence in their existence may change, so the evolution would be progressing on two aspects simultaneously. Implementation details of all 3 schemes are discussed in Sect. 3. In terms of ES-EQL query syntax, the change in the area can be specified with *AREA EVOLUTION*, the change in the certainty of the shapes with *PROBABILITY EVOLUTION*, and the combined/overall change with only *EVOLUTION* clause.

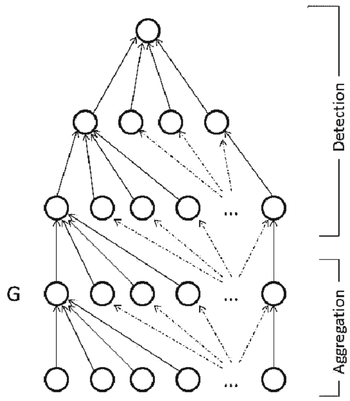
When comparing two shapes for evolution, the problem of shape identification arises, due to discrete data collection/processing. predefined Two subsequent calculations of a 2D shape bring another level of uncertainty: do these two shapes really refer to the same event? One may resort to defining possible worlds and exploring all the possibilities for identification of shapes is a way to handle this aspect of a problem. However, this ready-made approach makes the evolution calculations computationally expensive, and its investigation is beyond the scope of this work. Instead, we explore spatio-temporal relationships such as *split* and *merge*, the details of which (i.e. comparing last-reported shape and the new shape) are discussed in Sect. 3.

Data and Network Model: We discretize the space into cells and split the monitored field to hierarchical raster-like structure, decomposed into n by n grid, recursively continuing the decomposition One of the most popular way to do this is by using a quadtree [19], illustrated in Fig. 1(a). At the top level we have a single cell which represents the whole sensing field, then we build the quadtree by splitting the sensing field into 4 sub-fields of equal size. We note that the depth of the quadtree in our current implementation (although it can handle any arbitrary depth), is 4 – thereby providing 256 leaf-level cells.

At any given level, each cell has a designated/elected leader for data collection and processing. Depending on the queries, these leaders collect data from the sensor nodes in their cell and relay the processed data to their parent, which is the leader of the parent cell in the quadtree. However, electing a dedicated leader for data collection creates unbalanced energy drainage in the network, reducing the network lifetime. Therefore, we apply rotating leader scheme [20] to distribute the load among every node in the network. Therefore, all nodes in the network form a tree as in Fig. 1(b). With different levels, data can be defined in different granularity. When continuous spatial queries are posted to the system, the sensor nodes start sensing the environment and send their sensed value to their cell leaders. Then, at each level of the hierarchy, sensed data is aggregated to lower granularities if need be. Finally, the sink (root of the tree) streams the query update from the network to the querying users. In order for the system to respond to the queries that are based on certain thresholds, each cell at each level aggregates its data with respect to the given threshold(s) and



(a) Quad-tree Decomposition



(b) Granularity Level and Separation of Leaders

Fig. 1. Separation of the sensing field and quad-tree hierarchy

forwards it for shape detection in the higher levels. When data is aggregated enough, in other words, data granularity has been lowered to user needs, shape detection schemes start on elected leaders.

3 Aggregation and Shape Detection

When queries are posed to the system, the task for each sensor node may be different. Since WSNs have very limited energy budget, it is important to minimize the communication overhead and ensure the execution of the query in the meantime. Thus, the most straightforward approach of each sensor sensing the phenomena and sending their data to the respective cell leader who, in turn, would aggregate the data in an uncertainty frame and forward it to the leader of the higher-level cell in quadtree – may not be efficient in terms of energy expenditure. Our proposed event-based propagation of data taking advantage of evolution, granularity, probability and threshold parameters, is explained in the sequel.

In addition to sensing (and transmitting), a sensor node may have the tasks of *detection* of an event of interest and *aggregation*. We have following parameters for an event: – γ: sensing threshold; – p: probability threshold per cell; – and A: min-area for a connected shape. As part of the query, the elements that decide the detection are: – T: time period for update frequency; – c: relative change in value, defining a significant change (evolution); – g: number of unit cells in desired granularity level; – and R: query area.

The position and the size of a unit cells – i.e., cells which there is a single cell-leader (i.e., cluster-head) and all the other nodes are leaves in the quadtree – is uniquely identified when user specifies the desired granularity g. For example, if $g = 256$, then the unit cells are at level $L = 4 (= \log_4 256)$ and the addressing scheme for a cell $c_{i,j}$ denotes simply the location in the 2D array representing the row number (i) and the column number (j) corresponding to the distance from the bottom-left (i.e., south-east) corner of the region of interest. However, there is also a semantic role of the unit cell: it is the smallest piece of the resolution of the grid that collectively makes up the interior and/or boundary of a 2D shape (with its neighboring cells).

The sensed data is aggregated until the phenomenon can be represented with a collection of unit cells. Following the data aggregation, shape detection scheme is executed in the higher levels of the quadtree, without merging cells any further. The cell leaders in the quadtree can be horizontally split into two sets of nodes in terms of their participation-role for handling a given query: *aggregation* nodes, and *shape detection nodes* – as illustrated in Fig. 1(b).

We note that the parameter R above (i.e., the area of interest for a given query) need not be identical with the entire area covered by the WSN. Thus, when a query q with a set of instantiated parameters is posed and the level in the quadtree satisfying g cells is identified – if a particular cell intersects with the query region, then it is included in the detection/reporting. All the parameters are pushed down the quadtree structure until the query reaches the desired granularity level $L_{g,q}$. Since the leaders at L_q calculate the query-related properties of a unit cell, the nodes below this level do not receive any of the query parameters. The nodes at higher levels $L < L_{g,q}$ are only tasked with uncertain data aggregation, not the shape detection (cf. Fig. 1(b)).

Uncertainty in the data values in WSN are a fact of life, due to factors such as: – imprecise or malfunctioning sensors; – mis-calibration; – physical limits of precision based on the distance between the sensor and the phenomenon-source; etc. When tracking 2D shapes, a particular challenge is due to discrete nature of the data sampling [22] and its "conversion" to continuous regions. Aggregating the cell-wide uncertain data makes the problem becomes similar to the problem of sensor fusion, for which there are variety of theoretical approaches in the literature (e.g., based on Central Limit Theorem (CLT), Kalman filter, Dempster-Shafer methods, etc.). CLT states that the arithmetic mean of a large number of samplings follows a Gaussian distribution regardless of the distribution of random variables – sensor readings in our case. However, each cell in the network may not consist of *large* number of sensors, where a safe number for *large* is ≥ 30 – thus, the number of nodes in a grid cell may hinder the applicability of CLT. Evidential belief reasoning methods, such as Dempster-Shafer theory, rely on a set of probability masses and weighted prior beliefs, which can be quite expensive to store within the network. Hence, throughout this work we applied Bayesian filters [11] (i.e., a simpler version of univariate Kalman filters without the control system), making inferences about the true state of the environment x (i.e., phenomenon value) and the observation z (i.e., sampling by sensor).

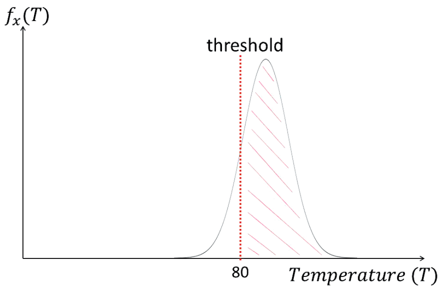


Fig. 2. Probability and a threshold

The "true state" is a continuous random variable, and its probability density function (pdf) is encoded in **the posterior**: $\Pr(x|z)$. Our prior beliefs about the phenomenon is encoded in **the prior**: $\Pr(x)$. Observations are made to obtain the true state x, modeled via $\Pr(z|x)$ – called **the likelihood** and denoted $\Lambda(x)$. Finally, marginal probability $\Pr(z)$ serves as normalization factor for the posterior. With multiple sources of sensing data, $Z^n = z_1, z_2, ..., z_n$, the posterior probability becomes[1] $\Pr(x|Z^n) = \alpha \Lambda_n(x) \Pr(x|Z^{n-1})$, where α is the normalization factor to make $\int \Pr(x|Z^n)dx = 1$ [15]. Note that the likelihood function, $\Lambda(x)$ can also be interpreted as sensor model – alternatively: "given the actual value of the phenomenon, what is the probability that this node will sense the value z?"

When the final posterior pdf is calculated after merging all of the readings in a given sensing epoch, the calculation of $\Pr(x > \mathit{threshold})$ is straightforward (cf. Fig. 2).

Data aggregation is done recursively along the quadtree, each parent fusing the children data – a distributed Bayes updating based on sending the likelihood functions from the children and having the parent apply recursive Bayes filter. Basically, each node sends their likelihood function to be fused to the aggregation point, which is the cell leader or the parent in the quadtree hierarchy.

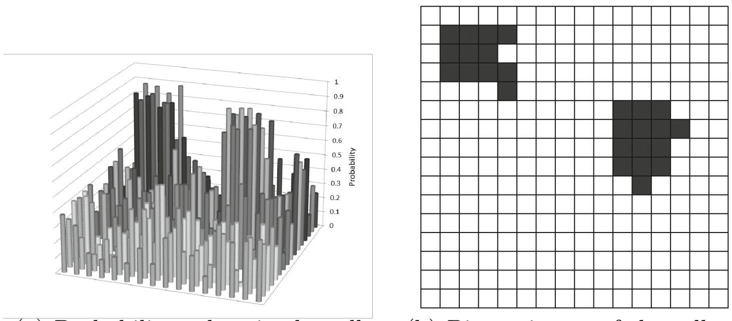
To detect a shape, we rely on results in [3], and we define a spatial event via predicate $Q(A, p, \gamma, t)$, which holds if there exists a connected 2D shape such that: (a) Readings for each part of the shape are $> \gamma$ with probability $\geq p$; (b) Area of the shape is $> A$; and (c) Time of occurrence of shape is $> t$.

Cell leaders gather the data from the nodes in their vicinity to aggregate and to forward it to their parent in the tree hierarchy. However, propagating probabilistic data poses new challenges: *"when the data transmission should be avoided?"* and *"which nodes should detect the shape?"*

In centralized settings, when cells calculate the probability density function of the phenomenon value and it is above the threshold, each unit cell can be represented as a single value in $[0, 1]$ interval pertaining to the given query (predicate) parameters. When all cells are represented with a single probability value, the whole map can be plotted as Fig. 3(a), where the bars represent the probability values. Taking a horizontal "slice" of this map with the queried probability parameter, p, would yield the binary image shown in Fig. 3(b), dark cells representing a region satisfying the query parameters. Using a simple breadth-first search algorithm, we can successfully calculate the shapes S_1 and S_2.

In distributed WSN, shape detection follows the data aggregation step, and we assume that the data has been aggregated at desired granularity.

[1] Due to a lack of space, we present the full derivations at [4].



(a) Probability values in the cell (b) Binary image of the cell after the cut

Fig. 3. Taking a horizontal slice of the probabilities

Thus, ancestor-leaders do not aggregate any further, but rather try to detect a shape in its region of governance and maintain the data granularity. As data propagates towards the root, cell leaders govern larger sensing areas (e.g., a "grandparent" would be responsible for detecting a shape 16 times larger than "grandsons"). At each level, the areas of group(s) of connected cells are computed and A parameter is checked for each shape – reporting when the total area $\geq A$. Otherwise, the data transmission is halted from this cell, since there is nothing to report with respect to the query. If any leader in the hierarchy detects a shape that is touching the boundary of its governance region, it forwards all of its data to its parent, since the shape may be split into neighboring cells. We note that, while tempting – halting data transmission from the cells whose probability of being above the threshold is below the p may cause potential problems when aggregating for a bigger cell in the higher level. If 3 sub-cells report being above the threshold with at least 0.7 probability, and one cell sending no data – implying sensing below the specified γ and p thresholds. The aggregation of these 4 sub-cells into a single value is invalid since applying Bayes' filter on 4 random variables to a generate single pdf in the absence of one does not actually represent the true value.

3.1 Temporal Evolution and Updates

The area of events may *shrink*, *expand* and/or *move* over time. Given the query-syntax, since data aggregation is performed until desired granularity level (cf. Fig. 1), all sensing data could be regularly transmitted in the quadtree hierarchy. Blocking transmission of a newly sensed data because of temporal, spatial, and spatio-temporal correlations is analyzed in detail in [23] and, in our solution, we capitalize from temporal correlation when sensing values remain steady – i.e., nodes do not need to send their new data which has not significantly changed from the previous transmission; similarly if the aggregated value of any cell exhibits the same probability distribution (i.e., mean and variance).

Evolution of a detected shape may refer to change in its probability of occurrence, its area of effect, or a combination of both evolution for each defined metric. Events change their location in both spatial and probabilistic dimensions and tracking the evolution of a single event can be achieved via calculating the boundary and probabilities of each unit cell for the shape in each epoch, then comparing the new shape against the last-reported shape based on a desired metric:

- **Area Evolution:** Since it satisfies the triangle inequality, we rely on the Jaccard similarity coefficient for the previously-reported shape and the current one. using Jaccard index and if the new shape is less than $1 - c$, where c is the evolution parameter, similar to the old shape, then it means it evolved. Formally, $\left(J(S_{old}, S_{new}) = \frac{|S_{new} \cap S_{old}|}{|S_{new} \cup S_{old}|}\right) < 1 - c$ must be true to detect area evolution.
- **Probability Evolution:** Each unit cell that makes up the shape has a probability value associated with it, and the probability of a shape is calculated via taking the average of all unit cell probability values. As the event becomes more certain, average probability values increase, consecutively decreasing the uncertainty, $1 - p$. The evolution in probability refers to change in the average uncertainty. Given new and last-reported shapes, if the uncertainty of new shape is c or $1 - c$ times the last-reported shape, then the new shape is considered *evolved.*
- **Area-&-Probability Evolution:** We first define a property called *Presence* which combines the area and probability of a shape S in a single value P_{AP}, calculated as:

$$P_{AP}(S) = \sum_{i \in S} p_i \times A_i$$

where i is a cell that is part of S, p_i is the probability in that cell, and A_i is the area of the cell. Therefore, when the new shape is calculated at the most recent epoch, all parts of the shape may indicate a probability change from its last-reported version. First, we calculate the net change per cell in the new shape. For each cell in the intersection, net probability change is calculated via $|p_{new} - p_{old}|$. For the parts of the new shape that was not part of the old shape (or vice versa) $(S_{new} \setminus S_{old})$, net probability change is defined as p_{new} (or p_{old}) treating the p_{old} (or p_{new}) as 0. After calculating the net probability change for each part of $S_{new} \cup S_{old}$, total presence value is calculated – denoted as *presence change* (P^c_{AP}). If $P_{AP}(S_{new})/P_{AP}(S_{old}) \geq c$, then shape has evolved.

To calculate the evolution for multiple shapes, the challenge is to identify which prior shape a given current shape has evolved from. To this end, we apply a shape matching scheme to elect a candidate shape in the last-sent map. Our matching method relies on a very straightforward heuristics:

$$S_{match} = \underset{S_x}{\operatorname{argmax}}(|P_{AP}(S_x \cap S_{new})|)$$

The shape in the previous epoch that shows the biggest intersection presence is regarded as the matching shape. Given a set of old shapes and a set of new

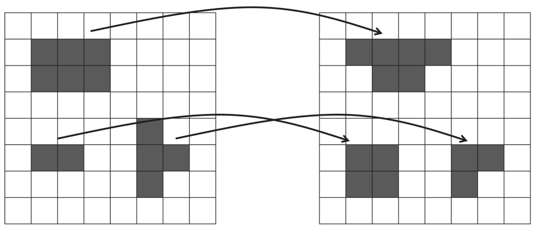


Fig. 4. Shape matching

shapes, we can form a bipartite graph where nodes represent the shapes and edges represent the matchings between new and old shapes. Matching each shape in the new map to another shape in the last-reported map enables us to track shapes between epochs (cf. Fig. 4).

Some shapes on the old set may connect to more than one shape in the new set (the reverse of is not true). Also, a number of shapes in the old set may not be connected to any of the shapes in the new set; and a number of shapes in the new set may not be connected to any shape in the old set. All of these properties of the bipartite graph imply evolving spatio-temporal relationship between two regions [12]:

Split: In the case of split, there will be more than 1 new shapes corresponding an old shape. Note that if both of the new shapes have substantial overlap with the old shape, individually they may not satisfy the evolution parameter.

Merge: If multiple shapes merge in a subsequent epoch, only one of them will be matched to the new shape. Even though the new shape had absorbed another shape, there is a possibility that it may fail to satisfy c parameter.

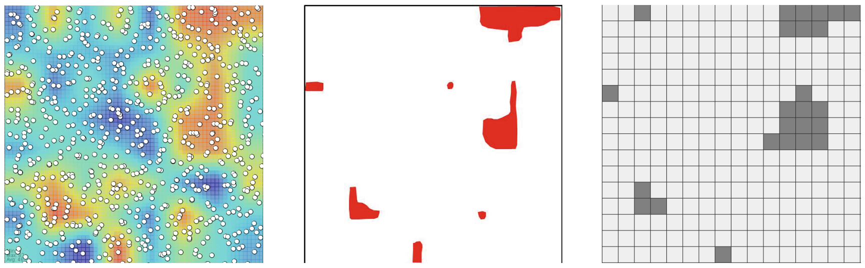
Disappearance: If a shape disappears from the map, then it is not detected in the current epoch.

Appearance: The case where a new event happens and a new shape occurs in the next epoch is not handled with the above method either.

For this, we detect whether the bipartite graph contains any: – disconnected node on the new set *(Appearance)*; – disconnected node on the old set *(Disappearance, or Merge)*; – a node on the old set connecting to two nodes in the new set *(Split)*.

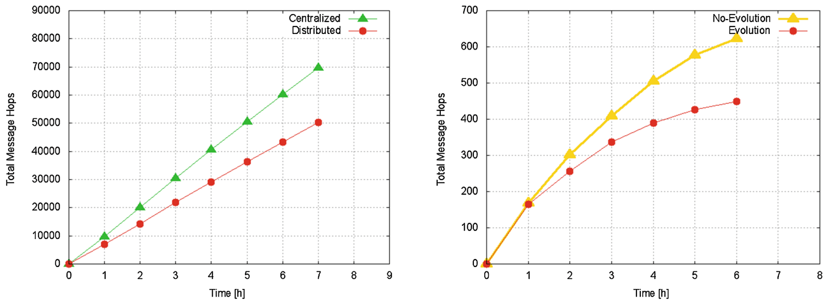
4 Experimental Analysis

Our experimental setup on SID-net Swans [14] simulator is a WSN consisting of 800 homogeneous nodes, capable to sense the phenomenon at its location with a discrepancy-controlled random placement, reporting every 10 s. Sensing field is set to be $1000 \times 1000\,\text{m}^2$, and each node had a communication range as 50 m. We used synthetic phenomenon for the experiments, built by generating 8 by 8



(a) Snapshot of the Network and the Phenomenon (b) Ground Truth for the Event Contours (c) Output Image from Our Method

Fig. 5. Shape approximation



(a) Centralized vs. In-network (b) Constant (No Evolution) vs. Evolution-basedReporting

Fig. 6. Communication expenditures (Color figure online)

grids and each cell assigned a random temperature between 0°C and 100°C, for every 20 min, linearly morphing the old grid to the new grid. Sensing value for any point in the field are calculated via bilinear interpolation. Sensor readings are perturbed with white Gaussian noise with $mean = 0$ and standard deviation a random number between 0 and 20. We processed the query with following parameters: R (query area) = 300; Temperature >80 F and $p = 0.7$ (70 %); Granularity = 256 unit-cells; Sampling frequency = 60 s; Change = 10 % – and the results we present were averaged over 3 independent runs.

Firstly, we evaluated the effectiveness of the Bayes filter and our shape detection scheme. Figure 5(a) shows the snapshot of the heatmap generated by our simulator with the location of the nodes (white disks) interpolated on top. For our query, event contours are extracted as ground truth in Fig. 5(b). Lastly, the output of our scheme can be seen in Fig. 5(c).

The second set of experiments highlight the difference in communication expenditure between in-network and centralized approach. Figure 6(a) illustrates the communication overhead of the centralized (vs. in-network) shape detection in terms of total message hops exchanged in the network. Our second set of experiments illustrate the impact of the evolution. When *AREA EVOLUTION = 30 %*, the communication expenditure difference between the constant and evolution-based reporting is shown in Fig. 6(b). The red line indicates the total messages over time for evolution-based reporting and the gold line shows the scheme with constant reporting. Note that at the start of the execution both techniques need to report the detected shapes.

5 Conclusion and Future Work

We proposed a 2D shape detection and tracking scheme with probabilistic bounds in WSNs, and enhanced users' control over the network by allowing a selection of update frequency and data granularity as part of query's syntax. As our experiments indicate, our approach is effective for the detection such events and is more energy-efficient in comparison to centralized processing. In the future, we plan to incorporate mobile nodes where nodes move freely, join and leave the network at will. Besides, we would like to extend our framework to capture belief-based data fusion methods with a semi-supervised belief updating protocol, and adapt them to approximate representations [22] in sparse WSN.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Comput. Netw. **38**(4), 393–422 (2002)
2. Amato, G., Chessa, S., Gennaro, C., Vairo, C.: Querying moving events in wireless sensor networks. Pervasive Mob. Comput. **16**(PA), 51–75 (2015)
3. Avci, B., Trajcevski, G., Scheuermann, P.: Managing evolving shapes in sensor networks. In: SSDBM (2014)
4. Avci, B., Trajcevski, G., Scheuermann, P.: Efficient tracking of uncertain evolving shapes with probabilistic spatio-temporal bounds in sensor networks. Technical report 2016–06, EECS Dept., Northwestern University (2016)
5. Babu, S., Widom, J.: Continuous queries over data streams. SIGMOD Rec. **30**(3), 109–120 (2001)
6. Buragohain, C., Gandhi, S., Hershberger, J., Suri, S.: Contour approximation in sensor networks. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 356–371. Springer, Heidelberg (2006)
7. Chu, D., Deshpande, A., Hellerstein, J.M., Hong, W.: Approximate data collection in sensor networks using probabilistic models. In: ICDE (2006)
8. Ding, M., Chen, D., Xing, K., Cheng, X.: Localized fault-tolerant event boundary detection in sensor networks. In: INFOCOM (2005)
9. Doherty, L., Pister, K.S.J., El Ghaoui, L.: Convex optimization methods for sensor node position estimation. In: INFOCOM (2001)
10. Duckham, M., Jeong, M.H., Li, S., Renz, J.: Decentralized querying of topological relations between regions without using localization. In: ACM-GIS (2010)
11. Durrant-Whyte, H.: Multi sensor data fusion. Technical report, Australian Centre for Field Robotics The University of Sydney (2001)

12. Erwig, M., Schneider, M.: Spatio-temporal predicates. IEEE Trans. Knowl. Data Eng. **14**(4), 881–901 (2002)
13. Fang, Q., Gao, J., Guibas, L.J.: Locating and bypassing holes in sensor networks. Mob. Netw. Appl. **11**(2), 187–200 (2006)
14. Ghica, O., Trajcevski, G., Scheuermann, P., Bischoff, Z., Valtchanov, N.: Sidnet-swans: a simulator and integrated development platform forsensor networks applications. In: SenSys, pp. 385–386 (2008)
15. Kar, S., Moura, J.M.F.: Distributed consensus algorithms in sensor networks with imperfectcommunication: link failures and channel noise. Trans. Sig. Proc. **57**(1), 355–369 (2009)
16. Madden, S., Shah, M., Hellerstein, J.M., Raman, V.: Continuously adaptive continuous queries over streams. In: ACM SIGMOD (2002)
17. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. **30**(1), 122–173 (2005)
18. Olston, C., Jiang, J., Widom, J.: Adaptive filters for continuous queries over distributed data streams. In: ACM SIGMOD (2003)
19. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kauffmann, San Francisco (2006)
20. Thein, M.C.M., Thein, T.: An energy efficient cluster-head selection for wireless sensor networks. In: ISMS (2010)
21. Trajcevski, G., Avci, B., Zhou, F., Tamassia, R., Scheuermann, P., Miller, L., Barber, A.: Motion trends detection in wireless sensor networks. In: MDM (2012)
22. Umer, M., Kulik, L., Tanin, E.: Spatial interpolation in wireless sensor networks: localized algorithms for variogram modeling and kriging. GeoInformatica **14**(1), 101–134 (2010)
23. Vuran, M.C., Akan, Ö.B., Akyildiz, I.F.: Spatio-temporal correlation: theory and applications for wireless sensor networks. Comput. Netw. **45**(3), 245–259 (2004)
24. Wu, M., Xu, J., Tang, X., Lee, W.-C.: Top-k monitoring in wireless sensor networks. IEEE Trans. Knowl. Data Eng. **19**(7), 962–976 (2007)
25. Yao, Y., Gehrke, J.: The cougar approach to in-network query processing in sensor networks. SIGMOD Rec. **31**(3), 9–18 (2002)
26. Zhu, X., Sarkar, R., Gao, J., Mitchell, J.S.B.: Light-weight contour tracking in wireless sensor networks. In: INFOCOM, pp. 1175–1183 (2008)