NORTHWESTERN UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
Number: NU-EECS-16-05

April, 2016

Title

Efficient Tracking of Uncertain Evolving Shapes with Probabilistic Spatio-Temporal Bounds in Sensor Networks

Authors

Besim Avci, Goce Trajcevski, and Peter Scheuermann
Dept. 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 focus on scenarios where, in addition to simple measurements one is also interested in the boundaries of a shape in which all the sensors' readings satisfy a certain criteria. Given the evolution of the underlying phenomenon being tracked/monitored, the changes of the corresponding sensed values imply that the boundaries of the shape(s) will also evolve over time. To avoid sending constant updates of the individual sensor-readings to a dedicated sink { and yet provide certain guarantees on the answer to user's queries posed to the server which stores the spatial data pertaining to the evolving shapes { we propose a methodology where the accuracy of the answer is guaranteed within certain probabilistic bounds. Towards that, we devised both linguistic constructs for the user to express the desired probabilistic guarantees as part of the query's syntax, along with the corresponding algorithmic solutions and system aspects of their implementation. Our experiments demonstrate that the proposed methodology provides over 25% savings in energy spent on communication in the WSN.

Keywords

sensor networks, probabilistic measurements, evolving shapes

Efficient Tracking of Uncertain Evolving Shapes with Probabilistic Spatio-Temporal Bounds in Sensor Networks

Besim Avci[1], Goce Trajcevski[1⋆], and Peter Scheuermann[1⋆]

Dept. 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 focus on scenarios where, in addition to simple measurements one is also interested in the boundaries of a shape in which all the sensors' readings satisfy a certain criteria. Given the evolution of the underlying phenomenon being tracked/monitored, the changes of the corresponding sensed values imply that the boundaries of the shape(s) will also evolve over time. To avoid sending constant updates of the individual sensor-readings to a dedicated sink – and yet provide certain guarantees on the answer to user's queries posed to the server which stores the spatial data pertaining to the evolving shapes – we propose a methodology where the accuracy of the answer is guaranteed within certain probabilistic bounds. Towards that, we devised both linguistic constructs for the user to express the desired probabilistic guarantees as part of the query's syntax, along with the corresponding algorithmic solutions and system aspects of their implementation. Our experiments demonstrate that the proposed methodology provides over 25% savings in energy spent on communication in the WSN.

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

Efficient Tracking of Evolving Shapes with Probabilistic Spatio-Temporal Bounds

No Author Given

No Institute Given

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 focus on scenarios where, in addition to simple measurements one is also interested in the boundaries of a shape in which all the sensors' readings satisfy a certain criteria. Given the evolution of the underlying phenomenon being tracked/monitored, the changes of the corresponding sensed values imply that the boundaries of the shape(s) will also evolve over time. To avoid sending constant updates of the individual sensor-readings to a dedicated sink – and yet provide certain guarantees on the answer to user's queries posed to the server which stores the spatial data pertaining to the evolving shapes – we propose a methodology where the accuracy of the answer is guaranteed within certain probabilistic bounds. Towards that, we devised both linguistic constructs for the user to express the desired probabilistic guarantees as part of the query's syntax, along with the corresponding algorithmic solutions and system aspects of their implementation. 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, which are 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 [4, 15, 17, 23], and the distinct context of WSNs had its impact on what energy-efficient processing of such queries is about [16, 24]. 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 [3,9,25], however, although there is a 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 shape.

When users pose continuous queries to a system where data source is a WSN, answering the query with the maximum precision while using least amount of system resources is a paramount. In traditional TinySQL 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 [16]. 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 [21].

Numerous works have tried to tackle the problem of efficient incorporation and management of uncertainty in WSN queries [6, 8], along with the continuity aspect of the changes in the monitored phenomena [16, 17]. 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 [5, 7, 12]. 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 preliminary notation and introduces the syntactic elements of the 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, Section 5 gives the conclusion and outlines the possible directions for future work.

2 Preliminaries

We now present the data collection scheme for the WSN that monitors the environment, where the number of phenomena that are being simultaneously sensed can vary (depending on a particular application). Then, we introduce a query language that enables user to pose the queries to the WSN with explicit specification of the probabilistic bounds.

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]. spatial queries.

2.1 Query Model

Several aspects of spatial queries pertaining to 2D shapes boundary detection have been tackled in the literature: boundary detection [7], isocontour construction [5], hole detection [12], etc. Our main focus is on detecting the boundary of "important events" spanning a 2D region, where parameters of the event is specified by the users queries. 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 methodology for SQL-like querying in WSNs is provided by the TinySQL [16] and it caters to two basic scenarios: (1) periodic sampling – as indicated in line #5 in Listing 2.1; 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 2.2.

Listing 2.1. Continuous Query

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

Listing 2.2. 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, there is the orthogonal aspect of the frequency of changes in the monitored phenomenon. For instance, the information gain from reporting that the temperature readings are 20 +/- $0.5°C$ every 10 seconds for 10 minutes – if the acceptable level of uncertainty is $+/-3°C$ – is same as sending only two readings – at the beginning and the end of the 10 minutes 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.

We re-iterate that our objective is to efficiently detect contiguous regions in which the measurements exceed certain threshold, and efficiently track their evolution in response to events corresponding to change of the sensed values. In our earlier work, we proposed predicates pertaining to shapes and objects trajectories along with their in-network detection [3, 20]. In a similar fashion, our focal point in this work is spatial events that are covering

a two dimensional regions. The set of queries we handle refer to these types of events under the consideration of uncertainty. At this stage, one can observe that a query language that is closest to our desiderata is the Event Query Language (EQL) [2], defined by separating the event query to several statements:

- *Event Statement:* conditions to recognize events and values returned by the event
- *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 2.3, 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: Firsly, 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 2.4.

Listing 2.3. 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 2.4. Modified Version

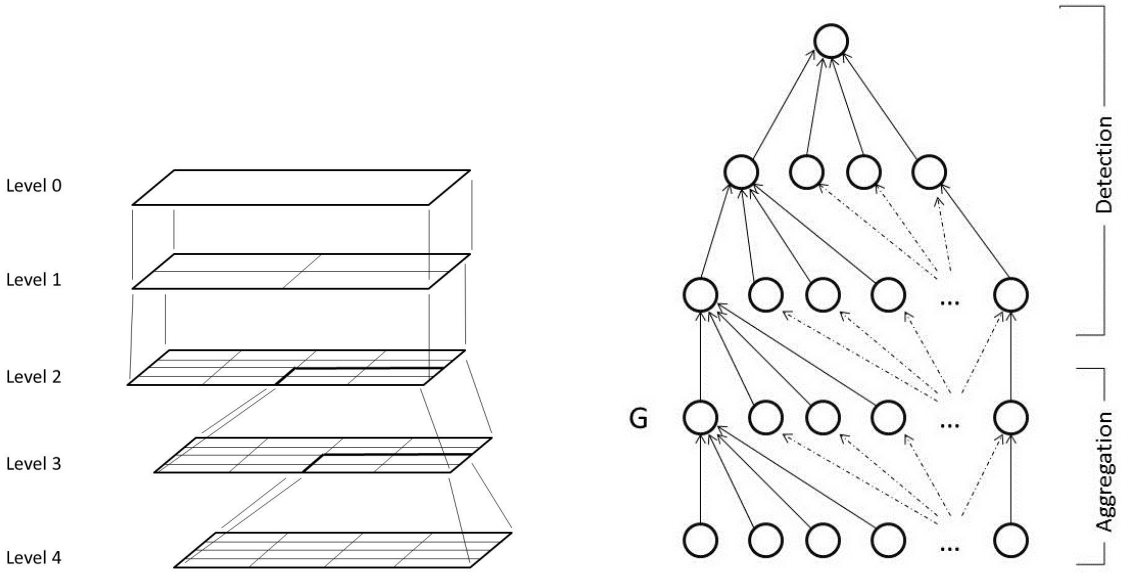
```
DEFINE EVENT  Fire
SIZE  500
AS Min(Probability) as MinCellProbability
WHERE  Temperature > 200
 AND Probability > 0.7

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 2.4 defines a fire event with the parameters being: size of the event is $500ft^2$, temperature readings for each unit cell above $200°F$, and the probability of each cell readings being above $200°F$ is 0.7. In other words, 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 $\geq 200°F$ in an infinitesimally small region within that cell is > 0.75. Then the detection scheme is defined for the *Fire* event as the detection will be carried out on the whole field with data granularity of 256 cells. Afterwards, reporting interval is specified as 60 seconds. Next, the evolution parameter is given as 20%, which will instruct system to report to update the answer to the query either regularly within 60-second 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 structure that can be converted to binary image), from the fire regions where the minimum probability in a unit cell is at most 0.75.

Now the challenge becomes how to identify what constitutes a *significant change* (evolution) in an event. As sensor networks sample the environment and communicate in discrete time intervals called epochs [16], evolution of the shapes between epochs are also discrete. Significant change, or evolution, can be attributed to several aspects of an existing shape: its probability, its size (area), or a combination of both. First, the evolution in the probability of a shape is the positive or negative change in its certainty. If a shape becomes more certain in average than its last-reported version by queried amount, then it means that it has evolved. Another source of a significant change is



(a) Quad-tree Decomposition (b) Granularity Level and Separation of Leaders

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

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 Section 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, since data collection/processing are discrete in terms of time, and are done in predefined intervals. However, two subsequent calculations of a 2D shape brings 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 Section 3.

2.2 Data and Network Model

Before we introduce our network model, we will briefly mention the data decomposition since it couples with the network model as well. We discretize the space into cells and split the whole monitored field to hierarchical raster-like structure, decompose it into n by n grid, and each cell is recursively decomposed further into equally-sized grids. One of the most popular way to do this is by using a quadtree [18], illustrated in Figure 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 [19] to distribute the load among every node in the network. Therefore, all nodes in the network form a tree as in Figure 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 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.

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. The most straightforward technique would be that each sensor senses the phenomena and sends their data to their cell leader. Recursively, that leader would aggregate the data in an uncertainty frame and forward it to the leader of the higher-level cell in quad tree, while maintaining the highest possible data granularity. However, this approach does not entail any communication savings. What we propose is event-based propagation of data while taking advantage of evolution, granularity, probability and threshold parameters.

Nodes will have different tasks depending on the parameters of query, such as query area, granularity etc. and each task requires different set of parameters. Mainly nodes carry out *aggregating* or *detecting* in addition to sensing depending on their level in the tree structure. To re-iterate, we have following parameters for an event:

- γ: threshold for sensing values for event detection.
- p: probability threshold for cell.
- A: area parameter for a connected shape.

and following parameters for their detection:

- P: time period for update frequency.
- c: parameter defining significant change (evolution).
- g: number of unit cells in desired granularity level.
- R: query area.

We discretize the space into equally-sized cells, and the size of the unit cell is specified by the user query via granularity parameter. Then, the level of unit cells in the quad-tree is identified. For example, if g is 256, then the unit cells are at level 4, because $4^4 = 256$. Unit cell, as the name suggests, is the smallest piece of a grid that collectively makes up the shape with its neighboring cells. Hence, sensing 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 tree without merging cells any further. The cell leaders in the quad-tree are horizontally split into two set of nodes in terms of their participation role for handling the query: aggregation nodes, and shape detection nodes as in Figure 1(b).

When a query is posed and the level in the quad-tree satisfying g cells is identified, which portions of the tree has an intersection with the query region are calculated. Because, users can specify a query region that is different than the whole sensing field. The management of any arbitrary query region is very straightforward: if a cell intersects with the query region, then it is included in the detection/reporting. Moreover, all parameters are pushed down to the leaders in the tree structure until the query reaches the desired granularity level, call G. Since any leader below level G governs an area smaller than unit cell, they don't need to receive detection parameters. Therefore, all nodes below G are only equipped with the task of uncertain data aggregation. Figure 1(b) shows the tree hierarchy and the separation of nodes with respect to their duties.

2.3 Aggregation

Uncertainty in the data is a fact of life and there are numerous reasons causing the data uncertainty such as imprecise or malfunctioning sensors, mis-calibration etc. Handling the uncertainty in WSNs has been tackled in different aspects for various applications. However, uncertainty in spatial data, where the aim is to detect/track two-dimensional shapes brings its own set of challenges and has not been addressed yet.

One of the main challenges in WSNs is the discrete nature of the data. When constructing spatial summaries and responding to spatial queries, set of discrete-in-space samplings should usually be converted to 2D regions. Converting discrete data points to continuous regions brings an intermediate approximation step in the representation of spatial data. There are mainly 2 methods to achieve *filling* between discrete data points: (1) converting

data to rectangular cells as raster data type, (2) approximating a polygon as a set of vectors. As mentioned in our data model discussion, we maintain a quad-tree structure with square grids. Next, the challenge becomes how to aggregate the cell-wide uncertain data.

In order to aggregate uncertain sensor readings within a cell, we assume each node senses the *same* phenomenon instance. In other words, each sensor reading is treated as different samplings of the same true phenomenon value, which is assumed to be the same through the whole cell. Now the problem becomes similar to the problem of sensor fusion. Fusing a set of uncertain data sources can be achieved with means of Central Limit Theorem (CLT), Kalman filter, Dempster-Shafer methods etc. CLT states that the arithmetic mean of a large number of samplings – each sensor reading is a sampling of the phenomenon value – follows a Gaussian distribution regardless of the distribution of random variables. However, each cell in the network may not consist of *large* number of sensors, where a safe assumption for the *large* considered to be ≥ 30. However, the number of nodes in a grid cell may be smaller, hindering CLT. Moreover, evidential belief reasoning methods, such as Dempster-Shafer, rely on a set of probability masses and weighted prior beliefs, which can be quite expensive to store for sensor network nodes. Dempster-Shafer methods can leverage from heterogeneous information, which we wish to explore in the future. Hence, we decided to apply Bayes filter [10] for our fusion technique since it is the simpler version of univariate Kalman filter without the control system and our noise is assumed to be normal and mean around 0.

$$Z = X + \mathcal{N}(0, \sigma)$$

where Z is the observed random variable (sensor reading), X is the phenomenon value and $\mathcal{N}(0, \sigma)$ is the noise factor.

Bayes' rule provides a means to make inferences about the true state of the environment x and the observation z. In our framework, true state is the true phenomenon value and observation is the sampling of the phenomenon by a sensor node. The terms true state and phenomenon value, observation and sensed value can be used interchangeably throughout the rest of the text.

$$\Pr(x|z) = \frac{\Pr(z|x)\Pr(x)}{\Pr(z)} \tag{1}$$

The interpretation of each term provides the purpose of using Bayes' theorem. Our main aim is to determine the true state x, which is the phenomenon value for the cell, given observations z, sensed values by the nodes within that cell. Since our true state is a continuous random variable, we would like to get the probability density for it and this 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, and these observations are modeled with $\Pr(z|x)$. Given an observation, z, we would like to get the true value, x, that resulted in this observation. This is called **the likelihood** and can be represented as $\Lambda(x)$. 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:

$$\Pr(x|Z^n) = \frac{\Pr(Z^n|x)\Pr(x)}{P(Z^n)} = \frac{\Pr(z_1, z_2, ..., z_n|x)\Pr(x)}{\Pr(z_1, z_2, ..., z_n)} \tag{2}$$

Instead of calculating and/or storing the joint distribution $\Pr(z_1, z_2, ..., z_n|x)$, we apply recursive Bayes updating. Updating the posterior after a set of observations ensures that all the past information is contained under the Markov assumption, only if previous posterior takes the place of the prior in the next iteration when the next observation arrives. Formally,

$$\Pr(x|Z^n) = \frac{\Pr(z_n|x)\Pr(x|Z^{n-1})}{\Pr(z_n|Z^{n-1})} \tag{3}$$

Note that $\Pr(x|Z^{n-1})$ is the posterior from the previous iteration, $\Pr(z_n|x)$ is the likelihood for the new sensing observation and $\Pr(z_n|Z^{n-1})$ is the normalization term. Thus, when a new observation comes, probability distribution for the true value can be calculated with the formula:

$$\Pr(x|Z^n) = \alpha\Lambda_n(x)\Pr(x|Z^{n-1}) \tag{4}$$

where α is the normalization factor to make $\int \Pr(x|Z^n)dx = 1$.

Having established the structure for the probability densities for the true value of the sensed phenomenon, we now explore the calculation of the pdf and the probability of the true value being above the queried threshold. First

of all, the likelihood function, $\Lambda(x)$ can also be interpreted as sensor model. Another translation of the function would be: "given the actual value of the phenomenon, what is the probability that this node will sense the value z?". It basically means probability distribution for the range of the values a sensor can generate. To this end, we first identify the distribution for the likelihood function as a Gaussian distribution around the true value as per $Z = X + \mathcal{N}(0, \sigma)$ and as previously explained in the literature [14]. Posterior, then, can be calculated as:

$$\Pr(z_n|x) = \Lambda(x) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{1}{2}\frac{(z_n - x)^2}{\sigma^2}\right) \tag{5}$$

$$\Pr(x|Z^{n-1}) = \frac{1}{\sqrt{2\pi}\sigma_{prio}^2} \exp\left(-\frac{1}{2}\frac{(x_{prio} - x)^2}{\sigma_{n-1}^2}\right) \tag{6}$$

$$\Pr(x|Z^n) = \alpha \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{1}{2}\frac{(z_n - x)^2}{\sigma^2}\right) . \frac{1}{\sqrt{2\pi}\sigma_{prio}^2} \exp\left(-\frac{1}{2}\frac{(x_{prio} - x)^2}{\sigma_{n-1}^2}\right) \tag{7}$$

$$= \frac{1}{\sqrt{2\pi}\sigma_{post}^2} \exp\left(-\frac{1}{2}\frac{(x_{post} - x)^2}{\sigma_n^2}\right) \tag{8}$$

The final probability distribution is also a Gaussian with mean x_{post} and standard deviation σ_{post}, since the product of two Gaussians is also another Gaussian. The resulting pdf can easily be calculated by substituting these terms as following:

$$x_{post} = \frac{\sigma_{prio}^2}{\sigma_{prio}^2 + \sigma^2} z_n + \frac{\sigma^2}{\sigma_{prio}^2 + \sigma^2} x_{prio} \tag{9}$$

$$\sigma_{post} = \frac{\sigma^2 \sigma_{prio}^2}{\sigma^2 + \sigma_{prio}^2} \tag{10}$$

Finally, the probability of the phenomenon value being above the queried threshold should be calculated, given the pdf. When the final posterior (pdf for the true value) is calculated after merging all of the readings in a given sensing epoch, the calculation of $\Pr(x > threshold)$ is very easy since posterior pdf is a Gaussian. As can be seen in Figure 2, the area under the curve where left side is bounded by the threshold line represents the cumulative probability that phenomenon value is above the threshold.

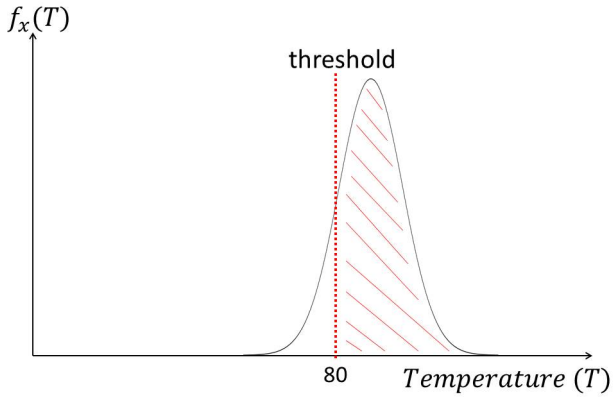
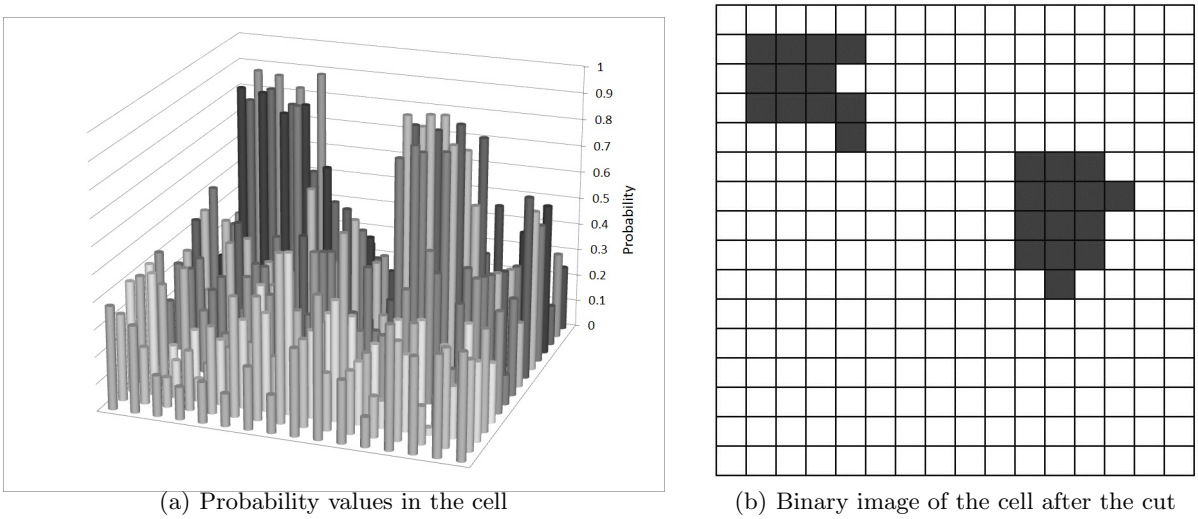


Fig. 2. Normal Distribution with a Threshold

The area under the curve can be calculated by subtracting the cumulative probability density of the threshold from 1, formally $1 - \phi(threshold)$. Hence, the probability can be calculated easily by:

$$1 - \frac{1}{2}[1 + erf(\frac{\gamma - x_{post}}{\sigma\sqrt{2}})] \tag{11}$$



(a) Probability values in the cell

(b) Binary image of the cell after the cut

Fig. 3. Taking a horizontal slice of the probabilities

where *erf* represents the error function.

Moreover, data aggregation is a recursive process in the quad-tree, which requires each parent in the tree to fuse the data coming from each child. This process is called distributed Bayes updating and can be handled in 2 different ways: (1) sending the likelihood functions from the children and having parent apply recursive Bayes filter, (2) sending the local posteriors and calculating the global posterior by dividing each local posterior with global prior. We follow the first approach since synchronization of global priors and posteriors among the nodes creates extra communication overhead. Basically, each node sends their likelihood function to be fused to the aggregation point, which is the cell leader or the parent in the quad-tree hierarchy.

2.4 Detection

In this section, we first establish a formal definition for event shapes. As outlined in our previous work [3], a spatial event can be represented with a predicate:

Definition 1. *A predicate $Q(A, p, \gamma, t)$ holds if there exists a connected 2D shape when:*

- *Readings for each part of the shape is $> \gamma$ with at least p probability.*
- *Area of the shape is $> A$.*
- *Time of occurrence of shape is $> t$.*

And predicates are part of the query along with other parameters such as granularity and evolution etc. 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 several challenges: emph'when the data transmission should be avoided?', *'which nodes should detect the shape?'* etc.

Before we explain shape detection procedure within the network, we state how detection can be done in a centralized manner if data were accumulated at the sink. When cells calculate the probability density function of the phenomenon value and the probability of phenomenon being 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 Figure 3(a), where the bars represent the probability values. If we take a horizontal slice of this whole map with the queried probability parameter, p, then we would be getting a binary image, which can be seen in Figure 3(b). Each dark unit cell represents a region satisfying the query parameters. Using a simple breadth-first search algorithm, we can successfully calculate the shapes S_1 and S_2.

However, shape detection process is distributed within the network, where each leader node in the levels higher than the granularity level can participate. As mentioned before, shape detection follows the data aggregation step and it is assumed that the data has been aggregated until the desired granularity. This implies that all the ancestral

leaders in the quad-tree do not aggregate the data any further but rather try to detect a shape in its region of governance and maintain the data granularity. In addition, since quad-tree splits the field into 4 equally-sized grids at each step, cell leaders govern more sensing field as data propagates towards the root. For example, leader of a unit cell would be responsible for only a single cell, while its grandparent would be responsible for detecting a shape in 16 unit cells. Finally, root would be able to see the whole field and can detect a shape in g number of cells. At each level, the group(s) of connected cells are calculated and areas of these shapes are computed. Then, A parameter is checked for each shape. If total area of any individual shape exceeds A, then it is reported to the querying user. Otherwise, the data transmission is halted from this cell, since there is nothing to report with respect to the query.

Note that when shape-related data are propagated towards the root, each leader recursively sends their data to their parent. If any leader in the hierarchy detects a shape that is touching the boundary of its governance region, then it forwards all of its data to its parent without concluding that shape to be detected since the event causing that partial shape may be split into neighboring cells. Leaders will be able to see 4 times the area its child can see, having the ability to detect a shape if it is split between its children. This way, no valid shape is discarded prematurely for not satisfying the A parameter.

In summary, a detection scenario starts with nodes sensing a phenomenon and sending their data to their cell leader. Cell leader, then, aggregates the data using Bayesian filter mentioned above and calculates the probability density function of phenomenon value of the cell. At the consecutive levels, posterior pdf is calculated using distributed Bayesian updating. At the leaders at G, if the area under the probability density curve, which is left-bounded by the threshold, is greater than p parameter, then it is concluded that this cell satisfies the predicate. Afterwards, cell leader forwards the detection notification – with probability – to its parent cell leader. For the cells sensing below the γ, data transmission is ceased. Basically, only the nodes satisfying both γ and p parameters send their information in the quad tree hierarchy. The information they send can be seen as a binary image representing their cell. At the higher levels, pixels are merged to make up a bigger image and shape detection scheme is initiated. If there exists regions not satisfying A parameter, then they are pruned. When all cell leaders forward their part of the image, a full image with desired resolution is created at the root and returned to the querying user.

2.5 Temporal Detection

The area of effect for the events exhibits variation over time. This variation can be attributed to *shrinking*, *expansion*, *moving* etc. As we discussed earlier, the problem of handling temporal aspect of systems in WSNs has its own set of challenges, especially in terms of limited network resources. Our proposed system enables users to specify their evolution-based adaptive update interval via the queries they pose so that the updates for the temporal variation of shapes can be expressed with a user-defined frequency. This section first describes how the data for each unit cell is updated at each epoch, then lays out the techniques for temporal management of shapes.

Unit Cell Update Since data aggregation is being performed until the data reaches to granularity level as illustrated in Figure 1, all sensing data are supposed to be regularly transmitted in the quad-tree hierarchy. For each epoch, all nodes do their regular sensing and send their data to their respective leaders. Then, Bayesian filter is applied at the leaders in the quad-tree and probability densities are propagated in the network hierarchy until the granularity level nodes. This means that a new probability value is calculated for each unit cell every epoch.

Although absence of data contradicts with Bayesian update, there is still a chance to reduce communication overhead. In the literature, there have been works proposing to block transmission of a newly sensed data because of temporal, spatial, and spatio-temporal correlations [22]. Since we are treating each sensor value within the same cell as sampling the same phenomenon, blocking data transmission based on the spatial correlation hurts the certainty of the data and each sensor may have different probability models. Having more data reduces the uncertainty. For the same reason, blocking because of spatio-temporal correlation hampers the data certainty as well. However, we can utilize from temporal correlation because sensing values remains to be the same in a steady environment. Therefore, if nodes make the exact same sampling, then they do not need to send their new data as it is assumed to be the same by the parent unless noted otherwise. Similarly, if aggregated value of any cell exhibits the same probability distribution – having the same mean and variance, then the transmission is blocked in the next epoch.

Query Response Update When shapes are detected, tracking their evolution is essential as per query requirements. Evolution of a shape may refer to change in its probability of occurrence, its area of effect, or a combination of both. The challenge now becomes how to calculate the evolution for each defined metric. Since unit cells are

updated regularly in each sensing epoch, it is rather straightforward to calculate the new shapes and send the information to querying user without calculating the evolution treating the data from consecutive epochs independent. However, this approach does not entail any communication savings. What we propose is to send updates upon satisfying the evolution parameter. In this sub-section we explore what kind of temporal changes can happen to the shape, then give techniques for the calculation of evolution based on all 3 metrics: area, probability, and combination of both.

Events change their location in both spacial and probabilistic dimensions. A single event can move to any arbitrary direction while maintaining its boundary, shrinking, or expanding. 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 the desired metric. For example, calculating the evolution between two shapes S_{new} and S_{old} can be done as follows:

- **Area Evolution:** There are a few possible metrics for comparing two shapes: Jaccard and Sorensen, etc. Since it is relatively simple and satisfies the triangle inequality, we rely on the Jaccard similarity coefficient for the previously-reported shape and the current one. Namely, two shapes are compared 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, the statement $\left(J(S_{old}, S_{new}) = \frac{|S_{new} \cap S_{old}|}{|S_{new} \cup S_{old}|}\right) < 1 - c$ must be true in order to satisfy the queried evolution.
- **Probability Evolution:** Each unit cell that makes up the shape has a probability value associated with it. Therefore, 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:** In order for our system to combine two different aspects of a shape, we first define a property for the shapes called *Presence*. Presence combines the area and probability of a shape S in a single value and can be calculated as follows:

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

  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 the union $S_{new} \cup S_{old}$, total presence value is calculated and this is denoted as *presence change*. Afterwards, presence change is compared against the presence of S_{old}. If $Presence(S_{net})/Presence(S_{old})$ is greater than queried c, then it means that the shape has evolved.

However, one cannot enforce on a sensing field to have only a single event at any point in time. There can be multiple events spanning the portions of monitored are. Moreover, these events also move and change their shape over time. The problem of tracking the changes in multi-shape settings is more challenging than the basic case mentioned above, since shapes may take place of each other, merge, split etc. Also, ground knowledge about the actual transformation is unknown because of discrete sensing and calculation. Hence, we derived a heuristic-based technique for calculating the evolution of shapes between snapshots for all three metrics.

First, we briefly re-iterate the data propagation model for continuous tracking of shapes. When new data is aggregated by the leaders below the granularity level G and the detection initiated at the leaders at G and up. As also explained in Section 3, a leader above G basically delegates its shape detection duties to its parent since it cannot fully detect a shape touching borders. When a cell leader governs an area that is large enough to detect the shape, it may also consist of multiple shapes in the same area. In addition, a cell leader sends a message only if there is enough evolution has happened in this sampling interval, or propagating the detected shapes from the lower levels. This means that if any one of the shapes evolve, all shape related data is sent since our data format is raster-like and sending the updated shape for a cell leader means sending the whole governed map (of sub-cells). Sending everything or not sending at all allows us to speed up evolution calculation process which is explained next.

After updating the pdfs and data transmission to a leader that detects a shape, leader then tries to calculate the evolution based on the current map and the last sent map. New map contains a set of shapes, all of which may have evolved. In order to calculate the evolution for all the shapes individually, the challenge is to identify which shape it has evolved from. To this end, we apply a shape matching scheme to elect a candidate shape in the last-sent map. In summary, for all shapes in the new map, a matching shape is found from the last-sent map, then the evolution is calculated via between the new shape and matched shape.

Shape Matching: Our matching method relies on a very straightforward heuristics:

$$S_{match} = \underset{S_x}{\text{argmax}}(|Presence(S_x \cap S_{new})|)$$

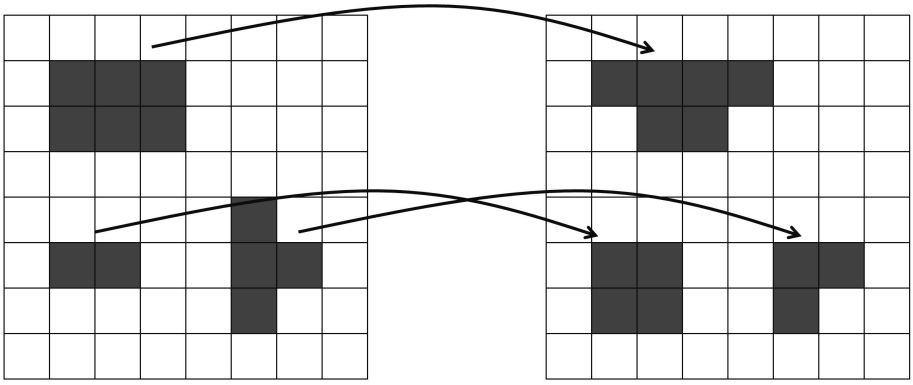


Fig. 4. Shape Matching

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 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. In the very same way, we can calculate the evolution between all of the new shapes and their matching counterpart in the last-reported map. An example matching can be seen in Figure 4.

The property of the bipartite graph we form in the matching stage is that some shapes on the old set may connect to more than 1 shapes in the new set, even though the reverse of this property is not possible. Also, a number of shapes in the old set may not be connected to any of the shapes in the new set. Likewise, a number of shapes in the new set may not be connected to any shape in the old set. All of these properties of our bipartite graph imply a spatio-temporal relationship between two regions [11]. These relationships entail several consequences in terms of evolution. If there is a node on the old set that is connected to two shapes in the new set, this implies that the shape represented by this node has been *split* in this epoch. If there is a disconnected node on the old set, it means either it has merged with another shape or the shape has disappeared. Similarly if there is a disconnected node in the new set, then it means the new shape represents a newly occurring event or a shape that has moved enough that it completely shifted its boundary. All of these spatio-temporal events indicate report-worthy evolution regardless of the c parameter since all of them indicate 100% evolution for at least one of the shapes.

Now we present techniques to detect the evolution implied by these special cases of spatio-temporal relationships, where Presence-based method mentioned above comes short.

Split: In the case of split, there will be more than 1 new shapes and a corresponding old shape. If both of the new shapes have substantial overlap with the old shape, there is chance that for them to individually not satisfy the evolution parameter.

Merge: If multiple shapes merge in the next epoch, only one of them will be matched to the new one as per our shape matching method. 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 with the method because our methods goes through the new shapes and calculates the evolution based on the new shape and its corresponding shape in the old set.

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.

These undetected evolution events can be detected via processing the bipartite graph and identifying whether graph contains any:

- Disconnected node on the new set (Appearance).
- Disconnected node on the old set (Disappearance, Merge).
- A node on the old set connecting to two nodes in the new set (Split).

Our overall temporal management of shapes consists of several steps and at the end of each epoch, cell leaders decide whether or not to send the updated data. To re-iterate, if a leader decides that *any* of the shapes in its cell exhibit evolution, then the leader sends the whole cell information since our data model is raster-like and sending a single shape is as costly as sending the whole map. This gives us a leverage in terms of computation. If evolution is detected at any point in the calculation process, then the rest of the operations are omitted and the new cell data is sent to the parent. Therefore, our evolution detection starts by updating the unit cells and detecting the queried shapes along the hierarchy. As soon as a shape is detected at any leader, it initiates the shape matching process for all the shapes. Then for all the $[S_{match}, S_{new}]$ pairs, evolution is calculated with respective methods for single shape cases. If no evolution is detected for any of the pairs, bipartite graph is analyzed for special spatio-temporal cases. Finally, if none of the special cases are detected, time elapsed since the last update is checked whether it is greater than frequency parameter P or not. If not, then the leader does not send anything at this sampling interval. Otherwise, sends the cell it is managing and updates the last-sent map with the new map.

3 Experimental Analysis

In this section we present the experimental evaluations of our methods. We analyze the effects of in-network continuous query processing for various aspects.

Experiment Setup: We conducted our experiments on a WSN simulator, called SID-net Swans [13]. We formed a WSN consisting of 800 homogeneous nodes, having capabilities to sense the phenomenon at its location with a discrepancy-controlled random placement. Nodes report with a sampling frequency of 10 seconds. Sensing field is set to be 1000 meters by 1000 meters, and each node having the communication range as 50 meters. We used synthetic phenomenon for the experiments. Synthetic phenomenon is built by generating 8 by 8 grids, where each grid cell is assigned a random temperature between $0°C$ and $100°C$, for every 20 minutes and linearly morphing the old grid to the new grid. Also, sensing value for any point in the field is calculated via bilinear interpolation. Moreover, each sensor readings is perturbed with white Gaussian noise with mean being equal to 0 and standard deviation is a random number between 0 and 20. Finally, all of the results we present are the average of 3 independent runs.

First set of experiments were aimed to highlight the communication expenditure difference between in-network and centralized query processing. The queries posed in this set of experiments are below:

```
DEFINE EVENT        Heat
SIZE                300
AS                  Min(Probability) as MinCellProbability
WHERE               Temperature > 80 AND Probability > 0.7

DEFINE DETECTION for    Heat
ON REGION               All
WITH GRANULARITY        256
EVERY                   60

SELECT                  EventImage
FROM                    Heat
```

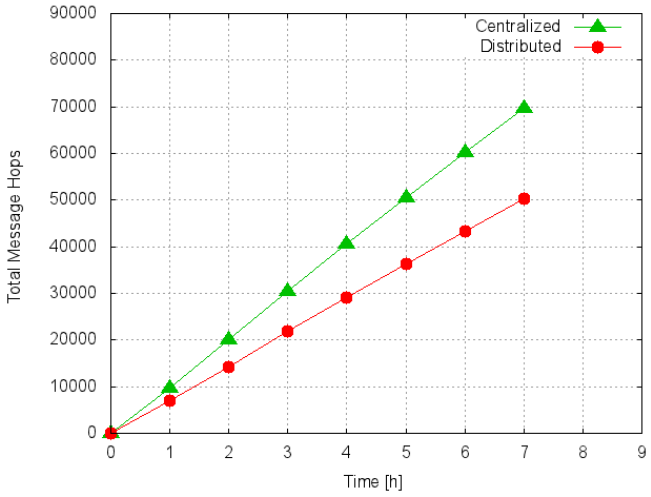
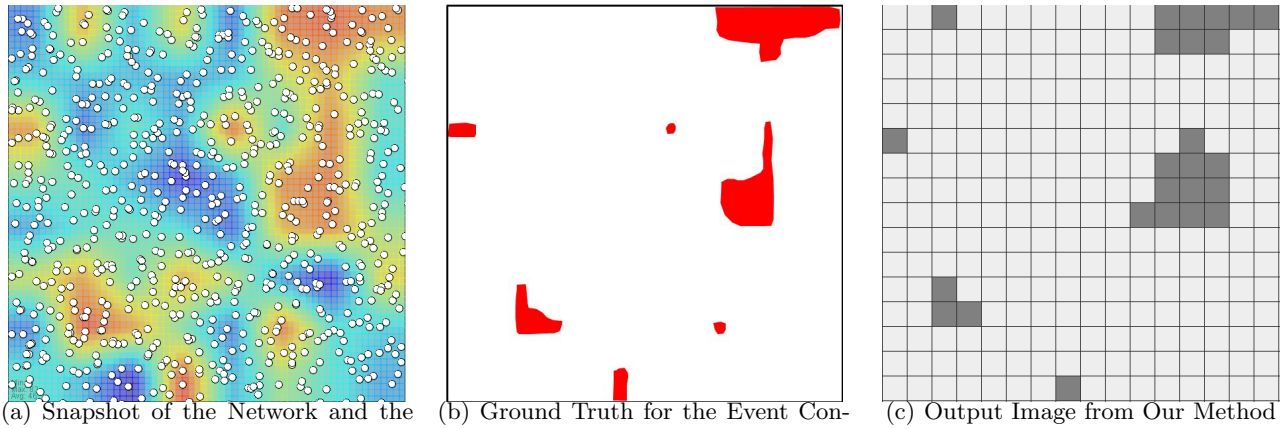


Fig. 5. Communication Expenditure (Centralized vs In-network)



(a) Snapshot of the Network and the Phenomenon

(b) Ground Truth for the Event Contours

(c) Output Image from Our Method

Fig. 6. Shape Approximation

For our first setup, we evaluated the query at the sink in a post-processing manner. The query was evaluated within the network in the second setup. There was no evolution involved in this set of experiments in order to purely identify the impact of in-network processing. Figure 5 compares the communication overhead between the two in terms of total message hops exchanged in the network.

Next, we evaluated the effectiveness of the Bayes filter and our shape detection scheme. Figure 6(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 Figure 6(b). Lastly, the output of our scheme can be seen in Figure 6.

Our last set of experiments illustrate the impact of the evolution property on the network resources. When the evolution parameter is set to 0.3 – 30% – *AREA EVOLUTION*, the communication expenditure difference between the constant and evolution-based reporting is shown in Figure 7. The red line indicates the total message hops over time for evolution-based reporting and the gold line shows the scheme with constant reporting. At the start of the experiments, both techniques need to report the detected shapes, however, evolution-based reporting reduces substantial communication overhead after the initial detection reports.

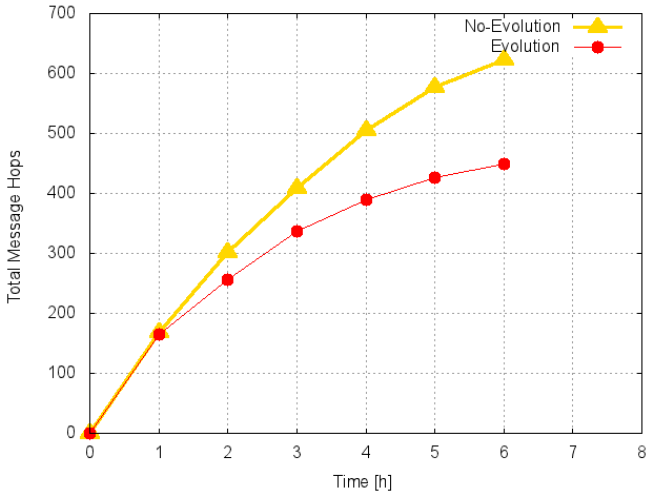


Fig. 7. Communication Expenditure (Constant (No Evolution) vs Evolution-based Reporting)

4 Conclusion and Future Work

In this paper we proposed a shape detection scheme with probabilistic bounds for Wireless Sensor Networks. In addition, we enhance users' control over the network by allowing them to define their update frequency and data granularity. To the best of our knowledge, our shape detection in presence of data uncertainty and adaptive update frequency are new in the literature. As our experiments indicate, our approach is effective for the detection of the event and saves significant energy in comparison to centralized processing at the same time.

For future work, we are aiming 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. Finally, we would like to explore co-occurrence of shapes with probabilistic bounds.

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4), 2002.
2. Giuseppe Amato, Stefano Chessa, Claudio Gennaro, and Claudio Vairo. Querying moving events in wireless sensor networks. *Pervasive Mob. Comput.*, 16(PA):51–75, January 2015.
3. Besim Avci, Goce Trajcevski, and Peter Scheuermann. Managing evolving shapes in sensor networks. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, SSDBM '14, pages 22:1–22:12, New York, NY, USA, 2014. ACM.
4. Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, September 2001.
5. Chiranjeeb Buragohain, Sorabh Gandhi, John Hershberger, and Subhash Suri. Contour approximation in sensor networks. In *Proceedings of the Second IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'06, pages 356–371, Berlin, Heidelberg, 2006. Springer-Verlag.
6. David Chu, Amol Deshpande, Joseph M. Hellerstein, and Wei Hong. Approximate data collection in sensor networks using probabilistic models. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 48, 2006.
7. Min Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*, pages 902–913, 2005.
8. Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui. Convex optimization methods for sensor node position estimation. In *Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Twenty years into the communications odyssey, Anchorage, Alaska, USA, April 22-26, 2001*, pages 1655–1663, 2001.

9. Matt Duckham, Myeong Hun Jeong, Sanjiang Li, and Jochen Renz. Decentralized querying of topological relations between regions without using localization. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 414–417, New York, NY, USA, 2010. ACM.
10. Hugh Durrant-Whyte. Multi sensor data fusion. Technical report, Australian Centre for Field Robotics The University of Sydney, 2001.
11. Martin Erwig and Markus Schneider. Spatio-temporal predicates. *IEEE Trans. Knowl. Data Eng.*, 14(4):881–901, 2002.
12. Qing Fang, Jie Gao, and Leonidas J. Guibas. Locating and bypassing holes in sensor networks. *Mob. Netw. Appl.*, 11(2):187–200, April 2006.
13. Oliviu Ghica, Goce Trajcevski, Peter Scheuermann, Zachary Bischoff, and Nikolay Valtchanov. Sidnet-swans: A simulator and integrated development platform for sensor networks applications. In *SenSys*, pages 385–386, 2008.
14. Soummya Kar and José M. F. Moura. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. *Trans. Sig. Proc.*, 57(1):355–369, January 2009.
15. Samuel Madden, Mehul Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously adaptive continuous queries over streams. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 49–60, New York, NY, USA, 2002. ACM.
16. Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, March 2005.
17. Chris Olston, Jing Jiang, and Jennifer Widom. Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 563–574, New York, NY, USA, 2003. ACM.
18. Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kauffmann, 2006.
19. Ma Chaw Mon Thein and Thandar Thein. An energy efficient cluster-head selection for wireless sensor networks. In *Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation*, ISMS '10, pages 287–291, Washington, DC, USA, 2010. IEEE Computer Society.
20. Goce Trajcevski, Besim Avci, Fan Zhou, Roberto Tamassia, Peter Scheuermann, Lauren Miller, and Adam Barber. Motion trends detection in wireless sensor networks. In *MDM*, 2012.
21. Muhammad Umer, Lars Kulik, and Egemen Tanin. Spatial interpolation in wireless sensor networks: localized algorithms for variogram modeling and kriging. *GeoInformatica*, 14(1):101–134, 2010.
22. Mehmet C. Vuran, Özg¨r B. Akan, and Ian F. Akyildiz. Spatio-temporal correlation: Theory and applications for wireless sensor networks. *Comput. Netw.*, 45(3):245–259, June 2004.
23. Minji Wu, Jianliang Xu, Xueyan Tang, and Wang-Chien Lee. Top-k monitoring in wireless sensor networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(7):962–976, July 2007.
24. Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, September 2002.
25. Xianjin Zhu, Rik Sarkar, Jie Gao, and Joseph S. B. Mitchell. Light-weight contour tracking in wireless sensor networks. In *INFOCOM*, pages 1175–1183, 2008.