Generating Contours in a Sensor Network Using Isovector Aggregation

Cheng Zhong
National Center for Geographic
Information and Analysis
University of Maine, Orono, ME, 04469, USA
Email: cheng.zhong@umit.maine.edu

Michael Worboys
National Center for Geographic
Information and Analysis
University of Maine, Orono, ME, 04469, USA
Email: worboys@spatial.maine.edu

***Abstract*—Contour maps provide an efficient way to visualize fields sensed by wireless sensor networks. In this paper, we discuss an energy-efficient technique, Isovector aggregation, for generating such contours, using a distributed, in-network approach. Our technique achieves energy efficiency in two main ways. Firstly, only a selection of nodes near contours are chosen to report in each sampling round, and each report message contains information about a part of or the whole contour rather than any single node's ID and value pair. Secondly, contours are progressively merged and simplified along the data routing tree, which eliminates many unnecessary contour points from contour vectors before they are transmitted back to the base station. By Isovector aggregation, the base station receives complete representations of the contours, and these require no further processing. Experimental results using simulations show that the Isovector aggregation approach sends much less data compared to the no aggregation approach and a well-known existing spatial-temporal aggregation technique. The Isovector aggregation approach achieves this energy efficiency without compromising on the accuracy of representations of the baseline maps. Indeed, in many cases our approach produces more accurate representations.**

I. Introduction

The rapid development of wireless sensor technology enables researchers to monitor regions in the physical world. In order to do this, we can use the value threshold approach [1], [2], [3]. When a node's value is higher than the threshold, we can deem this node to be in a high activity region and it should report to the base station. Though the threshold-based approach is simple for implementation, sometimes it is difficult for users to specify suitable threshold values because such values are based on different application environments. In this paper, we address the problem of high activity region monitoring in sensor networks from another approach, *contour map*, which provides an alternative way to visualize the entire monitored field monitored by sensor networks [4], [5], [2].

In wireless sensor networks, one of the most significant issues is the energy problem. Node-to-node data transmission consumes most of the energy and dwarfs all other energy consumption factors [6], [7]. In-network aggregation has been employed successfully as one of the best ways to save network energy [4], [2]. Most previous work does not consider how to generate contours in the network, but rather the base station has to process all the data received to generate the contour map.

In this paper, we describe a novel technique, Isovector aggregation, that targets energy efficient data collection from sensors to generate contours in the network. Energy efficiency is achieved by choosing only a few reporting nodes and employing in-network contour generation and simplification. Our major contributions include the following:

- A ring data structure that is used to generate local contour vectors between a node and its neighbors.
- Isovector aggregation algorithm for in-network contour merging and simplification. Each reporting message contains information about a part of or all the contours rather than any single node's ID and value pair. The total report data size is greatly reduced leading to significant energy savings and the base station does not have to do any further interpolation to generate a contour map.
- Comparison of Isovector aggregation with existing methods, including no aggregation and Isolines aggregation [4]. The results show that Isovector aggregation greatly out-performs these methods.

The rest of this paper is organized as follows. We briefly describe related work in section II. The preliminaries of our work are given in section III. We then present our in-network Isovector aggregation in section IV. After evaluating performance in section V, we conclude the paper in section VI.

II. Related work

In-network aggregation is extensively researched in wireless sensor networks, and many approaches have been proposed for different application scenarios. Most previous papers for contour reporting do not consider how to generate contours using in-network approaches. These methods have to relay all or some node information back to the base station, and later the base station uses such information to interpolate values to other nodes and then generate contours. Event Contour[5], Isolines [4] and COUCH [8] all belong to this category.

Papers that mention how to generate contours using in-network approaches include Isobar [2], EScan [9] and Contour Matching [10]. They all belong to polygon aggregation, aggregating nodes with the same values into polygons as the data flows towards the base station. Each node has to participate

in the aggregation by sending not only ID and value but also location information, which makes polygon aggregation perform worse than Isolines aggregation [4].

III. Preliminaries

In this section, we give an overview of Isovector aggregation. Then preliminaries of our work are presented. The assumptions we made in this paper are that each node has a unique ID and knows its own location through either a GPS or some GPS-less techniques [11], [12]. Then each node knows its neighboring nodes locations and IDs by simple node-to-neighbor communication. We set the default contour scale by 10 if not illustrated specifically. We also call contours contour vectors because each generated contour is composed of a series of points and it has a start point and an end point.

A. Aggregation through contours

Contours are defined by assigning different value ranges. Once the value ranges are defined, we use nodes' values to detect contours. If two adjacent nodes are in different value ranges, there is at least one contour between them and we want the two nodes to detect a contour through communication. These nodes are called *contour nodes*. In order to introduce our aggregation technique, we begin with a simple straight line example. More details will be discussed in the following sections.

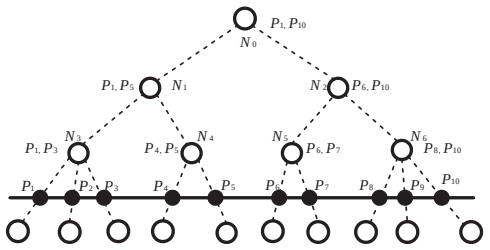

Fig. 1. A straight line contour in a sub-routing tree

Consider a sub-routing tree of the network. Suppose a subset of nodes detect a horizontal contour (figure 1) between them. A few reporting nodes N_3, N_4, N_5 and N_6 are chosen to report. Isovector aggregation works as following: N_3 detects the contour and generate three contour points P_1, P_2 and P_3 which are the mid-points between N_3 and the corresponding neighboring nodes. Based on the three contour points, node N_3 then produces a contour vector (P_1, P_3) and reports it to node N_1. P_2 is not contained in the vector because it is removed by N_3 as a redundant point. Nodes N_4, N_5 and N_6 work in the same way as N_3. Such process will be repeated along the data routing tree. Finally, node N_0 will merge vector (P_1, P_5) and vector (P_6, P_{10}) and generate the vector (P_1, P_{10}). This vector is sufficient to represent the straight line contour. N_0 then only has to report vector (P_1, P_{10}) to the base station. In this aggregation process, because many redundant data are removed, the total data transmitted is significantly reduced, which decreases network energy consumption.

Type	Description
$Negotiation$	Node broadcasting ID, value and location
$Query$	Base station initiating fetching contours
$Notification$	Node broadcasting ID and value
$Vector$	Response message to query

TABLE I
Message type

B. Contour neighborhood ring

We denote the value at node u by $R(u)$ and its value range by $Range(u)$. For any two nodes u and v, if u and v are in the same predefined value range, we have $Range(u) = Range(v)$, otherwise, $Range(u) \neq Range(v)$ (either $Range(u) > Range(v)$ or $Range(u) < Range(v)$). For example: if $R(u) = 42$ and $R(v) = 45$, u and v are both in the value range 40-49 and $Range(u) = Range(v)$.

Definition 1 (Contour neighborhood ring): Let u be a node and $v_1, v_2, ..., v_n$ be the one-hop neighbors of u, sequenced in counterclockwise cyclic order around u, where a start node v_1 is randomly assigned in advance. The contour neighborhood ring associated with u is a ring data structure $[CN_1, CN_2, ..., CN_n]$ starting from v_1 where

$$CN_i = \begin{cases} R(v_i), & \text{if } Range(u) \neq Range(v_i) \\ null, & \text{if } Range(u) = Range(v_i) \end{cases}$$

for $1 \leq i \leq n$.

Figure 2 shows an example of a node representation with its corresponding contour neighborhood ring.

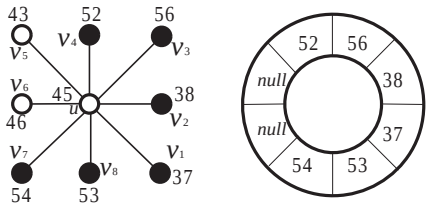

Fig. 2. A node representation of u and the contour neighborhood ring (numeric numbers are values of nodes)

C. Message types and time model

We defines four message types (table I). $Query$ message is broadcasted by the root at the network initialization phase. It contains query information and helps build the routing tree. $Negotiation$ message tells the node's neighbors its ID, initial value and location, and is only broadcast after receiving $Query$ message. $Negotiation$ message will be sent when data changes cause a contour to appear or disappear. The reporting message $Vector$ contains the reporting node's ID, contour vectors and the values of contours.

The time model we use in this paper is based on the cascading timer [13] designed by Solis and Obraczka. Using the cascading timer, a node merges contour vectors after it receives all $Vector$ messages from its children.

IV. ISOVECTOR AGGREGATION

In this section, we present our in-network Isovector aggregation which includes four parts: local contour generation, reporting node selection, contour simplification and contour merging. We assume the query for continuous mapping is processed repeatedly over a series of rounds, where each node generates a value in each round.

A. Local contour generation

Let the $scale$ denote the contour scale we defined. The contour $scale$ value together with the $null$ values in the neighborhood ring divide the ring into several partitions. Then a local contour vector will be generated for each partition by counterclockwise order. Each element in a contour vector is the mid-point between the reporting node and the corresponding contour neighbor. If a node u is chosen to report and it is in a higher value range than corresponding neighbors, the value of the reporting contour vector V between this node and those neighbors is set as: $Value(V) = (R(u)/scale) * scale$. Otherwise, if u is chosen to report and it is in a lower value range, the value of the reporting contour vector V between this node and corresponding neighbors is set as: $Value(V) = (R(u)/scale + 1) * scale$. Besides distinguishing contours of different values, the partition can be used to recognize narrow contours. More details about narrow contours are discussed in [14].

Figure 3 shows an example of a neighborhood ring of a node and the corresponding local contour vectors. Three partitions $\{37, 38\}$, $\{56, 52\}$ and $\{53, 54\}$ exist in the neighborhood ring. Then local contour vectors (P_1, P_2), (P_3, P_4) and (P_7, P_8) will be generated for the three partitions. Each generated vector has a headID (hID) and a tailID (tID) which are equal to the reporting node ID.

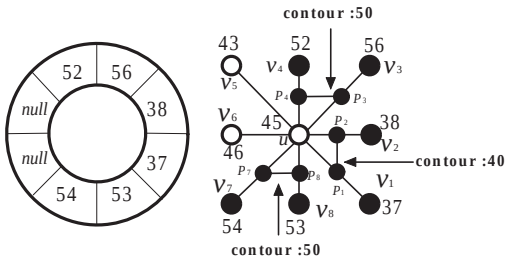

Fig. 3. u's contour neighborhood ring and the corresponding local contour vectors

B. Reporting node selection

Local contour detection in each round is based on neighborhood information received. If the ranges of sensed values are changed, nodes broadcast $Notifications$ to neighbors in the beginning of that sampling round. Each node that receives $Notifications$ compares its own value with neighbors' values. Some neighbors' values might be in the same value range as the reporting node. As specified before, such corresponding entries will be set to $null$ and they, together with the $scale$ value, act as separators to partition the contour neighborhood ring. If some neighbor values are on different sides of a contour, then at least one contour exists. When the reporting timer expires, this node will check if it should report. If it does, a $Vector$ message will be constructed and transmitted to the parent.

Contour detection is symmetric. We only choose contour nodes in the higher value ranges than neighbors to report. Each vector has a headID and a tailID which are equal to the ID(s) of the node(s) that report(s) them. Therefore, by maintaining such information of a contour, the base station can know which side of the contour is in higher value ranges and which side of the contour is in lower value ranges. In some cases, two adjacent nodes may not exist in consecutive value ranges. In this situation, both node in the lower value range and the higher value range will report. Figure 4 gives an illustration.

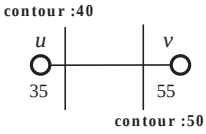

Fig. 4. u will report a contour with value 40 between u and v and v will report a contour with value 50 between u and v.

C. In-network contour simplification

Although a contour vector only contains important contour point information, such information may be still quite large. Without losing contour fidelity greatly, we consider how to weed out redundant points from a contour vector (figure 5). This problem is also called poly-line simplification which has been researched over the years. In this paper, we use the Douglas-Peucker line simplification algorithm for vector simplification in that it was best at choosing critical points when compared with others [15]. We should point out that, although the Douglas-Peucker approach is chosen, any other poly-line simplification method can be adopted if it can retain the shape of the original line.

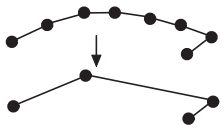

Fig. 5. Simplify a contour vector

D. In-network contour merging

After a node receives $Vector$ messages from its children, it saves them in a local contour Vector-Array ($vArr$). If this node is also a reporting node, local generated vectors will also be added to the array. When the reporting time comes, the node merges saved contour vectors and removes redundant points from all merged contour vectors.

Let $V.h$ (short name of vector head) denote the first point in the vector and $V.t$ (short name of vector tail) denote the last point in a vector V. We define the distance $D(V_1, V_2)$ between two vectors V_1 and V_2 as the minimal value of $D(V_1.h, V_2.h)$, $D(V_1.h, V_2.t)$, $D(V_1.t, V_2.h)$ and $D(V_1.t, V_2.t)$. If the distance $D(V_1, V_2)$ is shorter than a predefined threshold, they should be connected together. The node continuously merges each pair of adjacent contours until no two contour vectors are near enough to each other. In the merging process, any two vectors sent by the same child node will not be merged no matter how near the distance is between them. This restriction is used to avoid generating wrong shapes for narrow contour cases. More detail is discussed in [14].

Combining local contour generation, reporting node selection, contour simplification and contour merging together, our algorithm for Isovector aggregation is shown in algorithm 1. Each node in the network will run the algorithm.

Algorithm 1 Isovector aggregation

```
1:  while 1 do
2:    gets sensed value;
3:    if (value range changes) then
4:      broadcasts Notification;
5:    while (monitor events) do
6:      if (receives Notification) then
7:        updates information;
8:      if (receives children's Vector) then
9:        saves Vector;
10:       needReport = true;
11:     if (reporting timeout) then
12:       if (in a higher value range) then
13:         generates and saves local contours;
14:         needReport = true;
15:       else if (in non-consecutive value ranges) then
16:         generates and saves local contours;
17:         needReport = true;
18:       if (needReport) then
19:         merges vectors;
20:         simplifies vectors (Douglas-Peucker);
21:         composes a Vector and reports;
```

E. The base station

At the end of each sampling round, the base station will receive some contour vectors. It does similar merging operations to the inner nodes, except that contour vectors sent by the same node can be merged together. In this way, we generated an integrated contour map.

V. Evaluation

Isolines aggregation performs significantly better than polygon aggregation [4]. In this section, we evaluate Isovector aggregation by simulation in the NS2 [16] environment and compare it with Isolines aggregation. Isovector aggregation is also compared with no aggregation method in which nodes simply send their ID and values to the base station thought the routing tree. Negotiations with neighbors are not required. In the following simulation, Node ID, location information and value are all 2 bytes long.

A. Simulation setup

We use a sensor network consisting of 16x16 nodes arranged in a $400m^2$ evenly spaced grid to monitor temperature. The base station is placed in the center of the network. The routing tree is built in the network initialization phase by broadcasting $Query$ from the base station. We should point out that, although, in these experiments, nodes are placed according to a grid pattern, similar to Isolines, Isovector is not specific to grid placement. For medium access control, nodes use CSMA at 196Kbps. Their transmission range is set to 40m so each node has 8 neighbors except the outer layer nodes. FLIP [17] was used as the network protocol. The distance tolerance used for the Douglas-Peucker algorithm is 6m and the distance threshold for connecting 2 contour vectors is 14m which is less than the distance between any 2 adjacent nodes. The contour scale is set to 10.

There are two simulation scenarios for temperature monitoring. The first case is to detect static contours to form a contour map snapshot. In the second scenario, we focus on continuous contour monitoring.

We have two criteria for measurement. (1) Data transmission size which reflects the energy consumption by different methods. (2) Contour map similarity which maps to the query precision by different methods. The contour map similarity is calculated as the percentage of points (80*50 points are placed on the map) that are actually in correct value ranges when compared to the baseline map. ArcView GIS is used for interpolation and visualization. We should point out that only the no aggregation method and Isolines aggregation need interpolation by ArcView GIS. Isovector does not need to do this since contour generation is part of the algorithm.

B. Static contours

An irregular contour map, in which 41 percent of nodes are contour nodes, is used for evaluation. Figure 6 is the baseline map generated by all sensor values. Figure 7, 8 and 9 are examples of maps generated by reporting data of the no aggregation method, the Isolines and the Isovector aggregation respectively. Table II gives the simulation results. As we can see from the table, after drawing contours generated by Isovector aggregation, we get a contour map which is highly similar to the baseline map. Isovector sends much less data than Isolines aggregation. For no aggregation method, we find that, due to the huge network traffic caused by no aggregation method, nearly 40 percent of reports, including many reports sent by contour nodes, are automatically dropped by the network. Hence, the no aggregation method does not achieve significantly better map similarity than the other approaches.

C. Moving contours

In our continuous monitoring scenario, we simulate a front moving in from left to right. Temperature increases from the

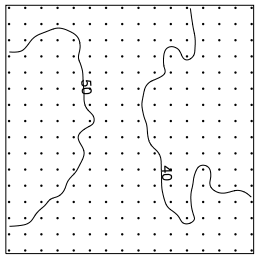

Fig. 6. Baseline map snapshot

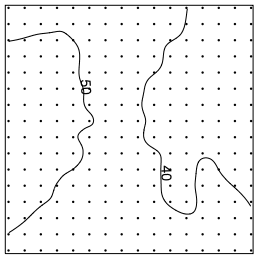

Fig. 7. Map with no Agg.

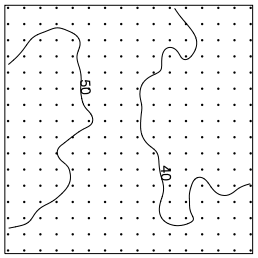

Fig. 8. Map with Isolines

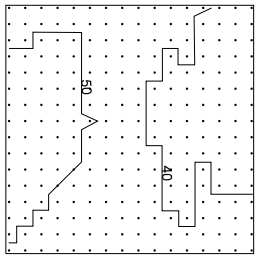

Fig. 9. Map with Isovector

	Similarity	Data sent (Bytes)
no Agg.	96.64% (sd 0.49%)	12111 (sd 534)
Isolines	94.94% (sd 1.16%)	6177 (sd 345)
Isovector	96.03% (sd 0.51%)	4175 (sd 263)

TABLE II
CONTOUR MAP SNAPSHOT

	Similarity (9s)	Data sent (Bytes)
no Agg.	98.0% (sd 0.72%)	110994 (sd 3074)
Isolines	97.7% (sd 0.48%)	26855 (sd 839)
Isovector	98.5% (sd 0.2%)	16617 (sd 807)

TABLE III
MOVING CONTOURS

thirties to the fifties in about 50 meters. The front moves to the right in 9 seconds. The starting value of all points is centered at 35 degrees. The base station, which is placed at the center of the map, starts by initializing the network at time 1s. From time 3s to 11s, nodes report their temperature values in each second. The simulation is stopped at time 12s.

We count the total data sent and also take a map snapshot at 9s for comparison. Table III gives the simulation results. From the table we know that, benefiting from contour vector merging and simplification, Isovector aggregation also sends much less data than Isolines aggregation in the continuous monitoring scenario. The contour map similarity by Isovector is slightly better than Isolines. No aggregation technique sends much more data than Isovector and Isolines. More results about this scenario and other scenarios can be found in [14].

VI. CONCLUSION AND FUTURE WORK

Isovector aggregation is a new approach to achieve energy conservation in wireless sensor networks. Its main advantages are its in-network contour generation and simplification which make Isovector aggregation energy efficient. Our simulation results suggest that Isovector achieves good performance and energy saving compared to other approaches.

There is a dense contour case that Isovector aggregation cannot handle properly. Between two neighboring nodes, Isovector aggregation reports at most two different contours. If two neighboring nodes have large value differences, then it is possible that some contours between them will not be reported. This is the subject of future work.

ACKNOWLEDGMENT

This work is supported by the US National Science Foundation under grant number IIS-0534429. Mike Worboys' work is also supported by the US National Science Foundation under grant numbers IIS-0429644 and DGE-0504494. He is also grateful for the support of the Ordnance Survey of Great Britain.

REFERENCES

[1] D. J. Abadi, S. Madden, and W. Lindner, "Reed: Robust, efficient filtering and event detection in sensor networks." in *VLDB*, 2005, pp. 769–780.

[2] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: Toward sophisticated sensing with queries." in *IPSN*, 2003, pp. 63–79.

[3] Y. Yao and J. Gehrke, "Query processing in sensor networks." in *CIDR*, 2003.

[4] I. Solis and K. Obraczka, "Efficient continuous mapping in sensor networks using isolines." in *Proc. of the 2005 MobiQuitous*, 2005, pp. 325–332.

[5] X. Meng, L. Li, T. Nandagopal, and S. Lu, "Event contour:an efficient and robust mechanism for tasks in sensor networks," *Technical Report, UCLA*, 2004.

[6] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors." *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[7] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications." in *SenSys*, 2004, pp. 188–200.

[8] A. Silberstein, R. Braynard, and J. Yang, "Constraint chaining: on energy-efficient continuous monitoring in sensor networks." in *Proc. of the 2006 ACM SIGMOD Intl. Conf. on Management of Data*, 2006, pp. 157–168.

[9] J. Zhao, R. Govindan, and D. Estrin, "Residual energy scans for monitoring wireless sensor networks." in *IEEE Wireless Communications and Networking Conference*, 2002, pp. 145–156.

[10] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks." in *SIGMOD Conference*, 2006, pp. 145–156.

[11] X. Cheng, A. Thaeler, G. Xue, and D. Chen, "Tps: A time-based positioning scheme for outdoor wireless sensor networks." in *INFOCOM*, 2004.

[12] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity." in *MobiHoc*, 2003, pp. 201–212.

[13] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," in *Proc. of the IEEE intl. Conf. on Communication*, 2004.

[14] C. Zhong, "Isovector: Generating contours in sensor networks," *Technical Report, University of Maine, 2007.5, http://www.spatial.maine.edu/ czhong/tr0702.pdf*, 2007.

[15] E. White, "Assessment of line-generalization algorithms using characteristic points." *The American Cartographer*, vol. 12, pp. 17–27, 1985.

[16] "Ns 2," *http://www.isi.edu/nsnam/ns/*.

[17] I. Solis and K. Obraczka, "Flip: A flexible interconnection protocol for heterogeneous internetworking." *MONET*, vol. 9, no. 4, pp. 347–361, 2004.