12-2008

Generating Contour Maps for Dynamic Fields Monitored by Sensor Networks

Cheng Zhong

GENERATING CONTOUR MAPS FOR DYNAMIC FIELDS MONITORED BY SENSOR NETWORKS

By

Cheng Zhong

B.A. Beijing Institute of Technology, 2001

M.S. Peking University, 2004

A THESIS

Submitted in partial fulfillment of the

Requirements for the Degree of

Master of Science

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

December 2008

Advisory Committee:

Michael F. Worboys, Professor of Spatial Information Science and Engineering, Advisor

Silvia Nittel, Associate Professor of Spatial Information Science and Engineering

Kate Beard-Tisdale, Professor of Spatial Information Science and Engineering

LIBRARY RIGHTS STATEMENT

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of Maine, I agree that the library shall make it freely available for inspection. I further agree that permission for “fair use” copying of this thesis for scholarly purposes may be granted by the Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature: _____

Date: _____

GENERATING CONTOUR MAPS FOR DYNAMIC FIELDS MONITORED BY SENSOR NETWORKS

By Cheng Zhong

Thesis Advisor: Dr. Michael F. Worboys

An Abstract of the Thesis Presented
In Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Spatial Information Science and Engineering)
December, 2008

Wireless sensor networks provide a new tool that enables researchers and scientists to efficiently monitor dynamic fields. In order to extend the lifetime of the network, it is important for us to minimize network data transmission as much as possible. Previous work proposed many useful aggregation techniques to answer max, min and average questions, and some of them have been employed in real applications. But we cannot get spatial information from these aggregation techniques.

This thesis presents an efficient aggregation technique for continuous generation of contour maps for a dynamic field monitored by a wireless sensor network. A contour map is a useful data representation schema that provides an efficient way to visualize an approximation to the monitored field. In this thesis, we discuss an energy-efficient technique, which we call *Isovector Aggregation*, for generating such contours using an in-network approach. Our technique achieves energy efficiency in two principal ways.

Firstly, only a selection of nodes close to contours are chosen to report, and each reported message contains information about a part or all of the contours, 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. Using Isovector Aggregation, the base station receives a complete representation of the contours that requires no further processing. Analysis shows that for region-related monitoring, Isovector Aggregation is the only technique that has $O(\sqrt{n})$ traffic generation and that considers in-network traffic reduction at the same time. These two factors make Isovector Aggregation highly scalable. Experimental results using simulations also show that Isovector Aggregation involves considerably less data transmission compared to other approaches, such as the no-aggregation approach and the *Isolines Aggregation* technique, without compromising the accuracy of representations of the baseline maps.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Michael F. Worboys for his encouragement, guidance, and support for my research at all times. I also wish to acknowledge the support and advice from other members of my thesis advisory committee, Dr. Silva Nittel and Dr. Kate Beard.

I want to thank all fellow graduates in the Department of Spatial Information Science and Engineering, especially to Jixiang Jiang, Qinghan Liang, Danqing Xiao who gave me useful feedback on my thesis. I would like to acknowledge Dr. Ignacio Solis at Palo Alto Research Center who gave me sample code and critical suggestions. Based on his code, I designed and developed my work.

I would also thank Ms. Ellen Huff, Ms. Jane Morse and Ms. Hope Duncanson for the English proof reading of this thesis.

Finally, I would like to thank all my family members and friends. This thesis could not be finished without their continuous support and care.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The rapid development of electronic and engineering technology has made possible the deployment of small, inexpensive, low-power, distributed devices, which are capable of local processing and wireless communication. Such nodes are called *sensor nodes* (Culler et al. 2004). Sensor nodes sense physical data of the area to be monitored. The continual analog signal sensed by the sensors is digitized by an Analog-to-Digital converter and sent to controllers for further processing. One of the most famous sensor nodes is the Berkeley Mote (Zhao and Guibas 2004). It is a well-designed tiny, self-contained, battery-powered computer with radio links, which enable the mote to communicate and exchange data with one another.

Each sensor node is only capable of sampling in a limited range and has limited processing abilities. But when collaborating with other nodes, we can construct a sensor network that is able to measure a given physical environment in great detail.

In this chapter, we present some useful background of this thesis. First, wireless sensor networks will be introduced and we discuss typical applications of wireless sensor networks. Then we present existing research challenges of wireless sensor networks, in which reducing data transmission is one of the most crucial topics in this area. Data aggregation, as a useful approach to reduce data transmission, is described later. The

research goal and hypothesis of the thesis are also presented. At the end of this chapter, we list the contributions and the structure of the whole thesis.

1.1 Wireless Sensor Networks

Wireless sensor networks are one of the driving forces of current network research. A wireless sensor network is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations.

It is expected that wireless sensor networks will play a key role in future research work, offering researchers and application specialists unprecedented opportunities in environmental sensing, monitoring and analysis. In contrast to other sensing technologies (e.g., LIDAR), a wireless sensor network is not restricted to particular types of phenomena: a wireless sensor network can detect any physical parameters, such as light intensity or temperature or ozone – so long as the corresponding sensor has been developed, and the device can withstand the deployment region's environmental conditions. Wireless sensor networks have a variety of applications (Xu 2004):

1) *Habitat monitoring application:* Cerpa et al. (2001) describe habitat monitoring as an application for wireless sensor networks-habitat sensing for bio-complexity mapping. As an example, in August 2002, researchers and scientists from University of Berkeley and Intel Research Laboratory deployed a sensor network on Great Duck Island, Maine, to monitor the behavior of storm petrels (Mainwaring et al. 2002).

2) *Environment forecasting system:* A wireless sensor network could be used to construct an environment observation and forecasting system. For example, in June 1991, Mount Pinatubo on the island of Luzon in the Philippines erupted catastrophically after about 600 years of dormancy. Layers of ash surround the crater and the effect of mudflows in this previously heavily forested and agricultural region can be traced as ribbons flowing downhill. Now, as shown schematically in figure 1.1, we have the capability to cover such an area with a network of sensors, with the intention of collecting data and monitoring changes in critical variables such as temperature.

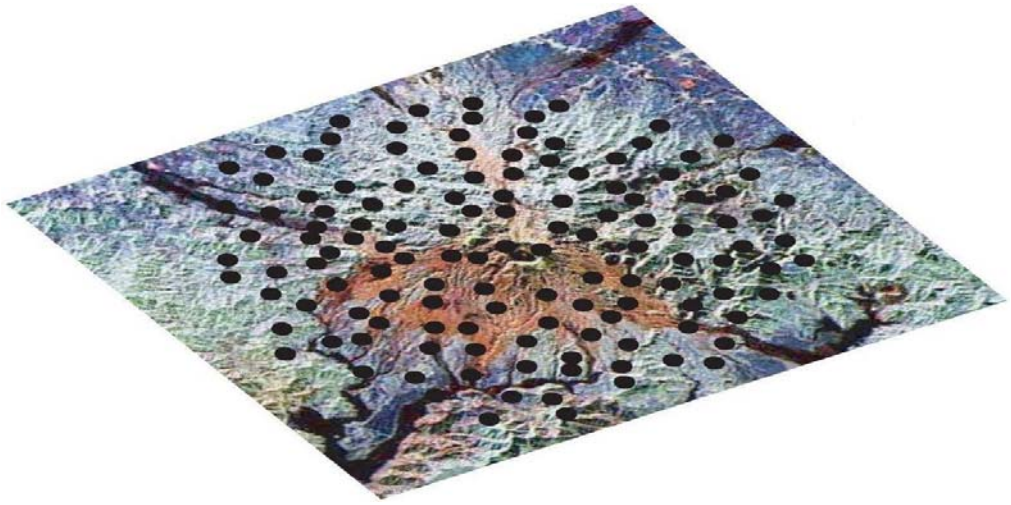

Figure 1.1. Sensor deployment around a volcano

3) *Health application:* We can use a sensor network to tele-monitor human physiological data, tracking and monitoring of doctors, patients and drug administrators inside a hospital (Akyildiz et al. 2002). The idea of putting biomedical sensors into the human body is a promising research area, although many challenges exist. Similar applications include glucose level monitoring, organ monitoring and cancer detectors.

1.2 Research Challenges in Wireless Sensor Networks

Though wireless sensor networks have already shown a promising future, there are many obvious challenges in this research field (Tubaishat and Madria 2003):

1) *Deployment:* Most sensor nodes are deployed in regions, such as in forests which have no infrastructure at all. We may have to toss sensor nodes from a plane to the monitored area. In this situation, it is up to the sensor nodes to identify their connectivity and distribution. Such deployments may also create topological holes and some regions may not be monitored if there are no sensor nodes in such places.

2) *Dynamic changes:* Sensor nodes are small electronic devices that can easily fail. For example, a heavy rain may cause some nodes deployed in the forest to be unworkable. In some applications, it may be required that a sensor network should be able to change connectivity and the routing structure dynamically.

3) *Untethered:* Each sensor node is not connected to any energy source. In other words, sensor nodes are battery powered and the energy is limited. We must utilize such energy efficiently in order to prolong the network lifetime. From some previous research, node to node data transmission and communication consumes most of the energy and dwarfs all other energy consumption factors (Pottie and Kaiser 2000, Shnayder and Hempstead 2004). In order to make optimal use of energy, it is important to minimize data transmission between sensor nodes as much as possible.

1.3 Data Aggregation

In wireless sensor networks, data is transmitted by wireless radio. Following a special routing schema, each sensor node transmits its sensed value hop by hop until the base

station receives all reports. A simple approach to monitoring a given physical field is to continuously transmit all sensor values all the time. Another possible approach is the value threshold approach (Abadi et al. 2005, Hellerstein et al. 2003a, Yao and Gehrke 2003). When a node's value is higher than the predefined threshold, we deem this node to be in a high activity region, and this is reported to the base station. Although the continuous approach and the threshold-based approach are simple to implement, sometimes they have to transmit too much data and may not be able to provide sufficient information about the field. As discussed in section 1.2, one of the most important characteristics of sensors is that each sensor is battery powered. Energy saving for sensor networks is therefore one of the primary goals in wireless sensor network research.

While many researchers focus on designing new sensor hardware and efficient protocols for communications, there is other work on the efficient algorithms for data processing in the network. In most applications, we do not need every discrete value of each sensor all the time. It is possible that we can compress the same or similar sensor readings in the network. This technology is called *data aggregation* in wireless sensor networks. The common goal of different in-network approaches is the same: reducing the total data transmission so as to prolong the lifetime of the entire sensor network. The basic idea of data aggregation is that rather than sending individual data items from sensors to the base station, multiple data items are aggregated as they are forwarded by the sensor network (Solis and Obraczka 2005a). Data aggregation is application dependent, i.e., depending on the target application, the appropriate data aggregation operator/aggregator will be employed. For example, suppose that in a controlled temperature environment, the average temperature needs to be monitored. As sensors

generate temperature readings periodically, internal nodes in the data collection tree rooted at the base station average data received from downstream nodes and forward the result toward the base station. Though data aggregation may decrease the accuracy, by transmitting less data units, the net effect is that considerable energy savings can be achieved.

1.4 Research Goal and Hypothesis

From the viewpoint of databases, standard queries, such as querying maximum minimum, and average values from sensor nodes, are often required by database users. These queries could be quickly answered using existing aggregation techniques (Madden et al. 2002, Yao and Gehrke 2002). In addition to these standard queries, many users are also interested in the region related questions: “Which region has values between 50 and 60?” and “What is the boundary of the phenomenon?” Traditional aggregation techniques are not able to answer these questions easily. In many application scenarios, contour maps can provide enough information to answer such questions, as they provide an efficient way to visualize the field monitored by wireless sensor networks (Meng et al. 2004, Hellerstein et al. 2003b).

A *contour* or *isoline* is a line along which values of the field are equal to a constant. This thesis is about generating contour maps for the dynamic field monitored by the wireless sensor networks. The goal is to design and implement an aggregation technique, *Isovector Aggregation*, which can continuously generate contour maps for the monitored dynamic field. This thesis has the following hypothesis:

It is possible to design an aggregation technique that generates contour maps with better precision and less network communication cost than a well-known aggregation technique-Isolines Aggregation (Solis and Obraczka 2005a, Solis and Obraczka 2005b).

1.5 Contributions and Thesis Structure

In this thesis, we describe a novel technique, Isovector Aggregation, which utilizes energy efficient data collection from sensor nodes to generate contours in the network. Energy efficiency is achieved by two approaches:

1) Instead of having all sensor nodes send their values to the base station, only a few nodes near the contours need to report. Reporting data contains contours rather than discrete values.

2) Instead of having all data transmitted directly, we remove redundant data, generate contours in the network and transmit them to the base station.

Our major contributions include the following:

1) A data structure-neighborhood ring that is used to generate local contour vectors between a node and its neighbors. The neighborhood ring designed in this thesis fully utilizes the node neighbor cyclic information and it saves all neighbor values of a specific node. Later, by comparing with the node values, each neighborhood ring will be used to generate local partial contours between the node and its neighbors.

2) Isovector Aggregation technique for in-network contour merging and simplification. In each sampling round, we generate local partial contours between nodes. Each reporting message contains information about a part of or all the contours rather than any single node's ID and value pair. In the process of reporting, contours with the

same values will be merged if they are near to each other. An in-network polyline simplification algorithm is also applied to remove redundant points from contours. Through all these techniques, the total report data size is greatly reduced, which leads to significant energy saving, and the base station does not have to do any further interpolation to generate contour maps.

3) Theoretical analysis which shows that for region related monitoring Isovector Aggregation is the only technique that incurs $O(\sqrt{n})$ traffic and that considers in-network traffic reduction at the same time. We analyze Isovector Aggregation for both traffic generation and traffic reduction. From Iso-map (Liu and Li 2007), we know that, Isovector Aggregation only requires $O(\sqrt{n})$ nodes to report, which promises less data generation when the sensor network is huge. We analyze the traffic reduction in a simplified binary tree routing structure. In the worst case, Isovector Aggregation still performs better than a well-known aggregation technique-Isolines Aggregation. These two factors make Isovector Aggregation very scalable for large applications in wireless sensor networks.

4) Experimental comparison of Isovector Aggregation with existing methods, including the no-aggregation technique and Isolines Aggregation**.** We simulate Isovector Aggregation using the Network Simulator 2 (NS2) network simulator and compare it with some baseline techniques. Data transmission and contour map accuracy are used as the evaluation metrics. The simulations include detecting static contours, monitoring moving contours and probing the relations between node densities and the data transmission. In all simulations, Isovector Aggregation out-performs these methods by

sending much less data than others. This energy efficiency is achieved without compromising on the accuracy of representations of the baseline maps.

The rest of this thesis is organized as follows. In Chapter 2, we discuss related work, which includes: general data aggregation techniques in wireless sensor networks, boundary/contour detection and reporting techniques in wireless sensor networks, polyline simplification techniques, and the CCM (Zhong and Worboys 2008) technique which also utilizes neighbor cyclic order information. In Chapter 3, we describe the core idea of Isovector Aggregation using a simple example. We give preliminary definitions, including that of the contour neighborhood ring, message types that will be used, and the data aggregation time model. In Chapter 4, we present Isovector Aggregation in detail. Chapter 5 gives the theoretical analysis of Isovector Aggregation from the perspective of network traffic generation and in-network traffic reduction. In order to verify the usability of Isovector Aggregation, Chapter 5 shows experimental analysis of Isovector Aggregation through extensive simulations. Finally, we conclude this thesis and discuss future work.

CHAPTER 2

RELATED WORK

In this chapter, we discuss previous work that uses suppression/aggregation techniques. General aggregation techniques are discussed first and we then discuss contour related aggregation techniques. One of the techniques (Isolines Aggregation, Solis and Obraczka 2005a, Solis and Obraczka 2005b) will be used for comparison later. Polyline simplification technology, as an important technology used in this thesis, is the next topic that will be examined. At the end of this chapter, we look at the Continuous Contour Mapping (CCM, Zhong and Worboys 2008) technique that is closely related to this thesis.

In-network aggregation is extensively researched for data reporting, and many aggregation approaches are proposed for different application scenarios. They all use temporal suppression, spatial suppression, or the combination of both suppressions. Silberstein et al. (2006) summarize the definitions of spatial and temporal suppression and their possible combination:

1) *Spatial suppression:* A node suppresses its reading if it is identical to those of its neighboring nodes.

2) *Temporal suppression:* If a node's reading is not changed since the last transmission, the node does not have to report to the base station. The base station can use its previous reading as the current reading.

3) *Spatial-temporal suppression:* It is possible to combine spatial and temporal suppression. If readings do not change, they should not be reported. In addition, if they do change, but the relationship between neighboring nodes remains the same, some reports may be suppressed.

In the case of continuous contour mapping, one possible combination of spatial and temporal suppression is that we only collect contour node information for reporting. We also use such technology in this thesis. More details will be discussed in Chapter 4.

2.1 General Aggregation Techniques

Many environmental phenomena can be represented as dynamic spatial fields, in that they are viewed as "signal" landscapes or surfaces (Gandhi et al. 2007). For example, distributions of temperature, humidity and pollutant levels naturally fit this viewpoint. As we discussed in Chapter 1, the lifetime of the wireless sensor network is determined by the energy remaining in all nodes, and the wireless communication between nodes consumes the most energy. Thus, algorithms that can reduce the data transmission in the network are highly desirable.

Motivated by these considerations, there has been a great interest within the wireless sensor network research area in the different data aggregation methods. It is reasonable to view a wireless sensor network as a distributed database system. From the viewpoint of databases, the most used aggregation operators are min (get the minimum value), max (get the maximum value) and avg (get the average value). These traditional aggregation operators are already efficiently implemented in the wireless sensor network (Madden et al. 2002, Yao and Gehrke 2002). Moreover, some robust statistics, such as medians or

quantiles (Greenwald and Khanna 2004, Shrivastava 2004) are also implemented. Clearly, all these aggregation techniques focus on numerical statistics, not the spatial shape of the phenomena. In many cases, users may want spatial-related information of the field, but cannot get any spatial information using the above aggregation techniques. Some good examples are: Which region has values between 50 and 60? What is the boundary of the phenomenon? These questions could be answered by generating boundaries or contours.

2.2 Contour Detection in the Network

Local boundary/contour detection is the first step for contour generation. Many previous works propose different solutions. We select some good ones and discuss their approaches and drawbacks.

Chintalapudi and Govindan (2003) discuss three qualitative approaches for localized edge detection namely statistical, image processing based and classifier based approaches. These approaches mark some nodes near the boundary as boundary nodes. They do not consider how to transmit boundary information back efficiently, and all data has to be returned to the base station to construct an accurate boundary. In this situation, a great deal of network energy will be consumed. Their edge detection approaches are limited and may be only used for boundary estimation applications.

Ding et al. (2005) propose localized fault-tolerant boundary and fault sensor detection using spatial data mining techniques. Like Chintalapudi and Govindan's three approaches, Ding et al. do not describe how to report these output nodes to the base station and do not consider continuous boundary changes. They just locally calculate boundary nodes and

fault-tolerant factors are considered in the computation process. If all boundary nodes have to report, a large amount of data will be transmitted.

2.3 Generating Contours at the Base Station

After contours are detected, we need to generate and transmit them back to the base station. Most previous papers on contour reporting do not consider how to generate contours during the contour reporting process. These methods have to relay all or some node information back to the base station, and later the base station uses such information to generate contours with the help of external software packages. They are not energy efficient because of the large data transmission and they cannot directly generate contour maps.

The *event contour* is a method discussed by Meng et al. (2004). In this work, spatial suppression and temporal suppression are combined. The event contour method does spatial suppression based on averages, in which all nodes attempt to report their values at different time slots during a logical timestamp. Nodes overhear reports from their neighbors. When a node's slot comes up, it first computes the average of all values overheard so far. If its value equals this average, its report is suppressed. The base station fills in missing values with the average of the neighboring values. In addition, this paper demonstrates that contour maps are useful for sensor network applications. There are some drawbacks of the event contour method. First, this approach does introduce some complications. To accurately derive a node's value, the base station must know which neighbors were averaged, when suppression was triggered; the average of this subset may be different from the average of all neighbors. Resolving this discrepancy requires a

global reporting order of all nodes, maintained by the base station. Second, some nodes that are far away from the contours need to report. The event contour method can never suppress all readings in an area. Third, if their reading difference is above the average, both nodes need to report, even though they are direct neighbors. Fourth, they still need the help of external tools to generate contour maps.

Solis and Obraczka propose the Isolines Aggregation method. Energy efficiency is achieved by having each node only report to the base station if it detects a contour between itself and its one-hop neighbors; otherwise, no report is generated. Because the detection is symmetric, Isolines Aggregation chooses half of contour nodes to report. The drawback of Isolines Aggregation is that, in a sparse wireless sensor network, nearly half of the nodes along both sides of each contour will report in each sampling round. This may cause a great communication overhead. In order to reduce the data transmission, Isolines Aggregation only reports contours that nodes detected between themselves and their children or parents according to the existing data routing tree when nodes are densely deployed. In other words, if some parts of a contour are between two sub-routing trees, then the contour cannot be detected precisely. In figure 2.1, only nodes u and v can detect the vertical contour. Nodes a and b cannot detect the contour even a is an one-hop neighbor of b.

Isolines Aggregation is a good technique, which combines spatial suppression and temporal suppression to a certain degree. Only some nodes near the isolines are chosen to report. In the experiments, the Isolines Aggregation method is compared with the Isobar Aggregation method (Hellerstein et al. 2003b). The contour map accuracy achieved by the Isolines Aggregation method is better than the Isobar Aggregation approach and the

bytes sent to the base station by the Isolines Aggregation method are also less than the Isobar Aggregation method. For this reason, we will use it as one of the two baseline techniques in Chapter 5.

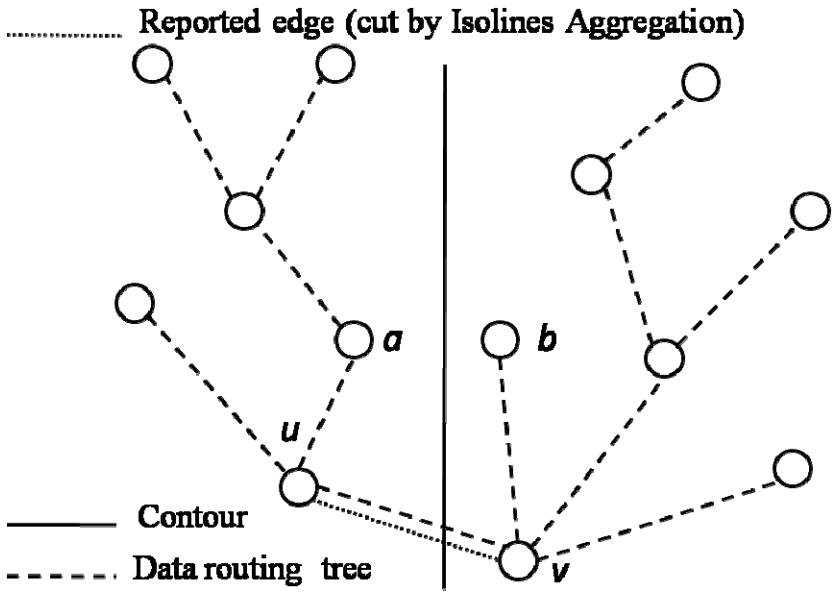

Figure 2.1. The drawback of Isolines Aggregation

Silberstein et al. (2006) propose a method called *COUCH* that attempts to make full use of spatial suppression and temporal suppression. The COUCH method collects history and statistics for minimal cost spanning forest construction and continuous optimization. Ideally, only one node on each side of the contour needs to report a change. The COUCH approach shows good performance if the contours move regularly or the contours seldom change. If the contours change irregularly, historical information may become less useful for the min-cost forest construction and optimization. In the worst case (a vertical contour changes to a horizontal contour), most contour nodes have to report in the next round. What is more, the spanning forest has often to be updated, which

will consume large amounts of energy. Again, the COUCH method has to generate other nodes' values at the base station.

In a recent paper, Liu and Li (2007) propose the *Iso-map* method, which shares similar ideas with Isolines Aggregation. They argue that all previous approaches (except Isolines Aggregation) require all nodes to report to the base station, leading to large amount of traffic generation. Their Iso-map approach, an energy efficient protocol for contour mapping, builds contour maps based solely on the reports collected from intelligently selected "isoline nodes" near contours in wireless sensor networks. They prove that only letting such contour nodes report will decrease network traffic generation from $O(n)$ to $O(\sqrt{n})$. They also conduct comprehensive trace-driven simulations to verify this protocol, and demonstrate that the Iso-Map method outperforms previous approaches in the sense that it produces accurate contour maps with significantly reduced cost. Iso-map is similar to Isolines Aggregation, though only a few contour nodes are required to report in each round; the authors do not consider how to reduce data transmission in the network, and they still have to generate contours at the base station using external software packages.

2.4 Generating Contours in the Network

Few papers consider how to generate contours using in-network approaches. They all are associated with the polygon aggregation approach.

Hellerstein et al. propose a method called *Isobar Aggregation* (Hellerstein et al. 2003) that uses spatially correlated data aggregation for mapping purposes. They realized that contours could provide an important way to visualize sensor fields and may have

applications in a variety of biological and environmental monitoring scenarios (Estrin 2002). Their method aggregates nodes with the same values into polygons, as the data flows towards the base station. An approximate, lossy approach is also presented to produce approximate contour maps. In short, the Isobar Aggregation approach tries to reduce data transmission in the sense of in-network traffic reduction instead of traffic generation. Each node has to participate in the aggregation by sending not only its ID and value but also the location information, which makes the Isobar Aggregation method send nearly 20% more data than the Isolines Aggregation approach. At the same time, the Isobar Aggregation method does not achieve a better map accuracy than the Isolines Aggregation approach.

Zhao et al. (2002) propose the *EScan* approach which uses the same approach as Isobar Aggregation and has the same problems as Isobar Aggregation. The difference is that this method constructs contour maps to monitor sensor residual energy instead of monitoring physical events in the network. The resulting contour maps show the residual energy distribution of the sensor network.

Xu et al. (2006) mention in-network contour generation for contour map matching, which actually is an extension of polygon aggregation. After generating contours, they use contour matching to detect events happening in the network and it is more reliable than the threshold based approaches. In this approach, local location information from each node still has to be transmitted before the aggregation, which causes huge data transmission. Their method can generate regular polygons in the network.

Unlike all these approaches, Isovector Aggregation only focuses on the boundaries of polygons and is not confined to regular polygons. In Isovector aggregation, such

boundaries are called contours and a contour map is composed of many polygons. We only use information on nodes that can detect contours to generate contour maps.

2.5 Polyline Simplification

Often a polyline has too high a resolution for an application, such as visual displays of geographic map boundaries or detailed animated figures in games or movies. That is, the points on the polylines representing the object boundaries are too close together for the resolution of the application. In geographic information science, technology for solving this problem is called *polyline simplification* and it has been an active research area in the past.

Polyline simplification technology could bring many benefits. For example, by reducing coordinates pairs, the vector time plotting speed is increased and the storage space required is decreased. For the same reason, both the time needed for vector to raster conversion and the time needed for processing vectors could be reduced.

The Douglas-Peucker algorithm (Douglas and Peucker 1973) is one of the well-known and efficient algorithms for polyline simplification (White 1985). The Douglas-Peucker algorithm defines a tolerance ε, and uses a recursive approach as follows:

1) The two extreme endpoints of the polyline are first connected with a straight line.

2) Distances from all intermediate polyline vertices to the straight line are calculated.

3) If all these distance are less than the tolerance ε, the approximation is good and all intermediate points are removed. The algorithm ends here.

4) If any of these distances is greater than the tolerance ε, the point that is furthest away from the straight line is chosen as a vertex that divides the original polyline into two shorter polylines (figure 2.2). Then the algorithm goes to step 1) using recursion.

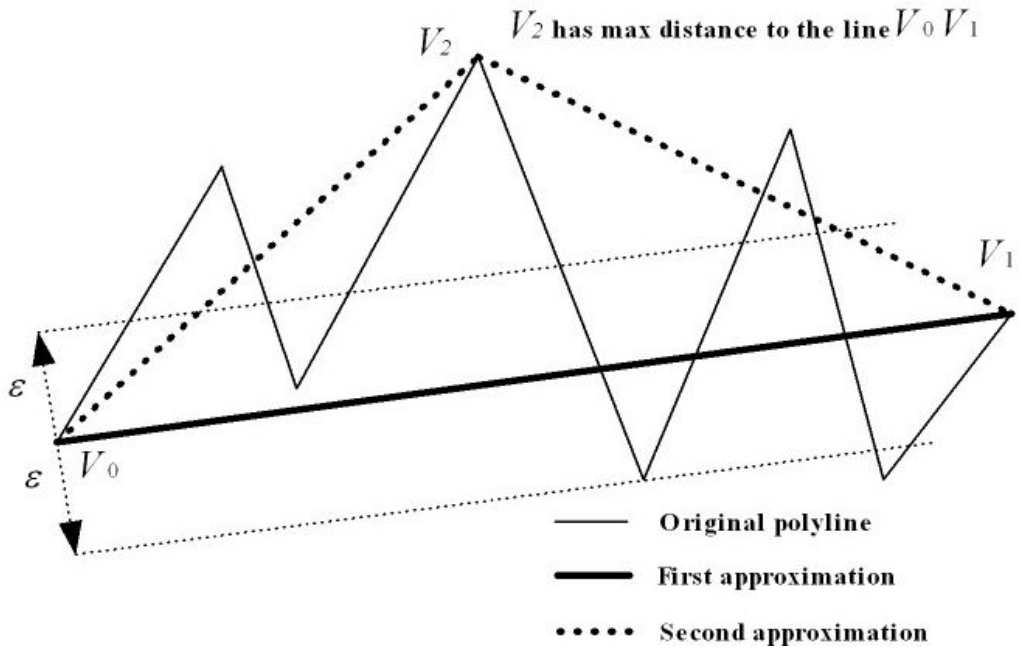

Figure 2.2. Illustration of the Douglas-Peucker algorithm

2.6 Continuous Contour Mapping (CCM)

Zhong and Worboys propose the Continuous Contour Mapping (CCM) method (Zhong and Worboys 2008) for continuous contour monitoring. A ring data structure is designed to suppress neighborhood information that utilizes the cyclic order information of neighbors. We will discuss the CCM contour monitoring first and then give the drawbacks of this technique.

The ring data structure-contour neighbor array, in short, is a small bit array to store one-hop neighbor information of a node. If a node reports to the base station, the message

contains all one-hop neighbor information by using this structure. The definition of the array is as follow:

Definition 2.1: Contour Neighbor Array (CN-array): 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 the starting node v_1 is randomly assigned in advance. The CN-array associated with u is an array of bits [b_1, b_2 ,..., b_n, h] where

$$b_i = \begin{cases} 0 & if \quad Range(u) = Range(v_i) \\ 1 & if \quad Range(u) \neq Range(v_i) \end{cases}$$

for $1 \le i \le n$.

h is set to 1 if all contour neighbors of u are in a higher value range than u. Otherwise, if all contour neighbors of u are in a lower value range than u, h is set to 0.

From the value of h, the base station can know the contour value (higher or lower) between a reporting node and its contour neighbors. For example, suppose the contour values are defined using a contour interval of 10 (40, 50 etc.). If $R(u) = 47$ and h of u is 1, then u detects a contour with value 50. Figure 2.3 is a contour node representation with its corresponding CN-array.

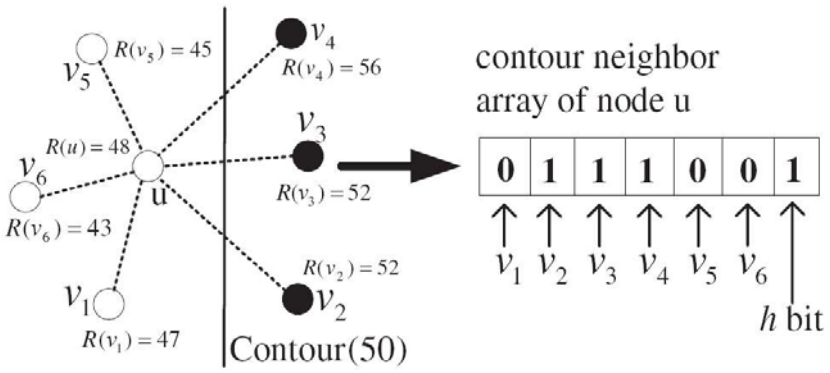

Figure 2.3. A CN-array representation

In the network initialization phase, after receiving a query message from the base station, each node randomly chooses a neighboring node as its starting node and initializes its CN-array by communicating with its neighbors. Then each node sends its own ID, location, CN-array and corresponding NI-array (neighbor ID array) back along the data routing tree. In this way, the base station knows the whole network topology and the mapping information between each node's CN-array and neighbors. Later the base station can decompress each received CN-array correctly. A randomized timer algorithm is designed and it only chooses a few nodes near contours to report and each reporting message only includes the node ID, reading and the corresponding CN-array. Other nodes are all suppressed. For each node, u performs an algorithm with the following three steps in a sampling round:

Step 1: u decides if it should participate in the reporting node selection in this round. If it should not (u is not a contour node), u skips the following steps. Step 1 makes sure that only nodes that can detect contours will participate in the reporting.

Step 2: If u is a contour node and $Range(u)$ and $u's$ contour neighbors have not changed since the last transmission, u only sends its ID back, broadcasts a 'report sent' message and skips step 3. Step 2 provides additional temporal suppression in each sampling round. Only nodes with changed information are required to send all new information back.

Step 3: u randomizes a timer for a time which is much shorter than the sampling period. If u does not receive any 'report sent' message from its neighbors after the timeout, u sends its ID, reading and CN-array back, and broadcasts a 'report sent'

message. This is a randomized algorithm for reporting node selection. All selected reporting nodes are randomly distributed along each contour.

The above steps explain how the CCM approach reduces total data transmission and saves the network energy: randomly selecting a few reporting nodes and each node encoding other contour nodes information. The CCM algorithm is also optimized using a probability based approach. Simulation results reported show that, depending on node density, CCM provides an average reduction of 60% on the total amount of report data, compared to the baseline spatial suppression algorithm.

Though the improvement is clear, the CCM approach has some obvious drawbacks:

1) A bit array is used to compress neighborhood data and suppress neighbors. Because each bit only has two possible values: 0 and 1, many may be reduced when we compose the array. The consequences are that, in many situations, contours are dense and it is possible that some neighbors of a node are in a higher value range than the node and others are in a lower value range than the node. The CN-array alone cannot deal with such cases correctly. Figure 2.4 gives a good example. In short, CCM is good for boundary monitoring, in which each sensor node only has two possible values: 0 and 1. But it does not provide a generic solution for contour mapping.

2) Similar to the COACH and Isolines Aggregation methods, the CCM approach only considers reducing data transmission from the traffic generation aspect. Although it uses a bit array for data compressing, it does not consider traffic generation and in-network traffic reduction at the same time. It has to interpolate values to other nodes at the base station and then uses external tools to generate contour maps.

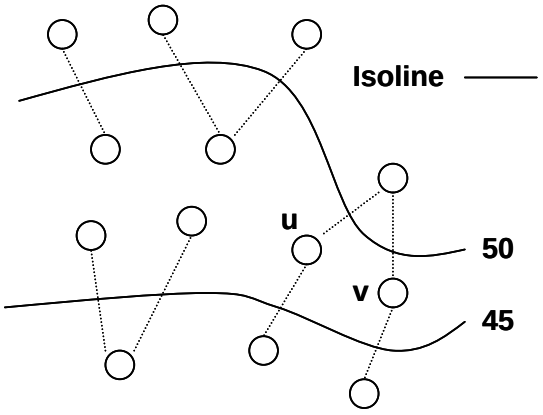

Figure 2.4. A dense contour case that the CCM approach cannot handle correctly

2.7 Summary

In this chapter, we list some previous work that is related to this thesis. We first discuss general aggregation techniques used in wireless sensor networks. We present previous works for contour detection and contour generation, including generating contours at the base station side and generating contours within the network. We then focus on polyline simplification because it is directly related to this thesis. In order to remove redundant data and reduce data transmission, the Douglas-Peucker polyline simplification algorithm, will be used in this thesis.

The CCM approach is an energy-efficient technique for continuous mapping that utilizes neighboring node cyclic order information to design the data structure for data storage and data compressing. Though CN-array in CCM is different from the neighborhood ring we propose in this thesis, CCM is the only previous work that unitizes neighboring node cyclic order information, and the CN-array also saves some neighbor information. The CN-array is used to compress neighbor information; whereas, the neighborhood ring used in this thesis is used to generate local partial contours.

CHAPTER 3

OVERVIEW AND PRELIMINARIES

Contour maps are used as the data representation tools of our data aggregation method. In this chapter, we first briefly discuss contours and contour maps. After giving the definition and characteristics of dynamic fields that will be monitored by wireless sensor networks, we use a simple example to explain the basic idea of our Isovector Aggregation method. In addition, we discuss other preliminaries of the thesis, including the definition of neighborhood ring, the types of messages and the time model used by Isovector Aggregation. The assumptions we make 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 (Shang et al. 2003, Cheng et al. 2004). Then each node knows its neighboring nodes locations and IDs by simple node-to-neighbor communication. We set the default contour scale to 10 and the initial scale is 0 if not specified. We also call contours *contour vectors* because each generated contour is composed of a series of points and it has a starting point and end point. If the starting point and end point of a contour are the same, this contour is called a *ring*. In the following chapters of this thesis, contour, vector, and contour vector are interchangeable terms.

3.1 Scalar Dynamic Fields

A *scalar field* (Duckham et al. 2005) is a spatial domain R, such that for each point $p \in R$, there is a unique scalar value, s_p, assigned to p. In this work, it is assumed that a sensor node associated with point p detects s_p with complete precision. For example, in the case of forest fire monitoring, each point in the region of network deployment resides in a scalar field where temperature is the scalar value.

A spatial scalar field represents the variation of some scalar property over a region of space. Examples of the scalar properties include temperature, wind-speed, or the concentration of a gas pollutant in the air. A spatial field is defined as a function from space to a scalar property. In a dynamic field, phenomena change over time. Hence, it is not adequate for us to only give one snapshot of the monitored field. This thesis also takes such a requirement into consideration.

3.2 Contour Maps

A *contour line* in a scalar field (Meng et al. 2004) is a curve connecting points of the same particular values. A *contour map* uses contour lines (often just called a "contour") to join points of equal values. Contours are often given specific names starting with the prefix "iso-" according to the nature of the variable being mapped. Contour maps have a scale value which separates two adjacent lines. For example, given a scale of 10, a temperature contour map indicates regions that differ by 10 units of temperature. The region between two adjacent lines is in the same temperature range. "Contour maps present a simple way of fine tuning the trade-off between information and the cost of obtaining it by adjusting the step values to suit situational requirements" (Meng et al.

2004). There can be many contour maps, each corresponding to a different parameter, (e.g., temperature, pressure and wind speed), with different scale values for each parameter, thereby giving a clearer picture of the monitored field. In this thesis, temperature monitoring will be used as an example.

In many research areas, especially in geographic information science, the contour map is a widely used mechanism for data representation. As an example, figure 3.1 is a ground temperature contour map of the USA (UNISYS, 2002).

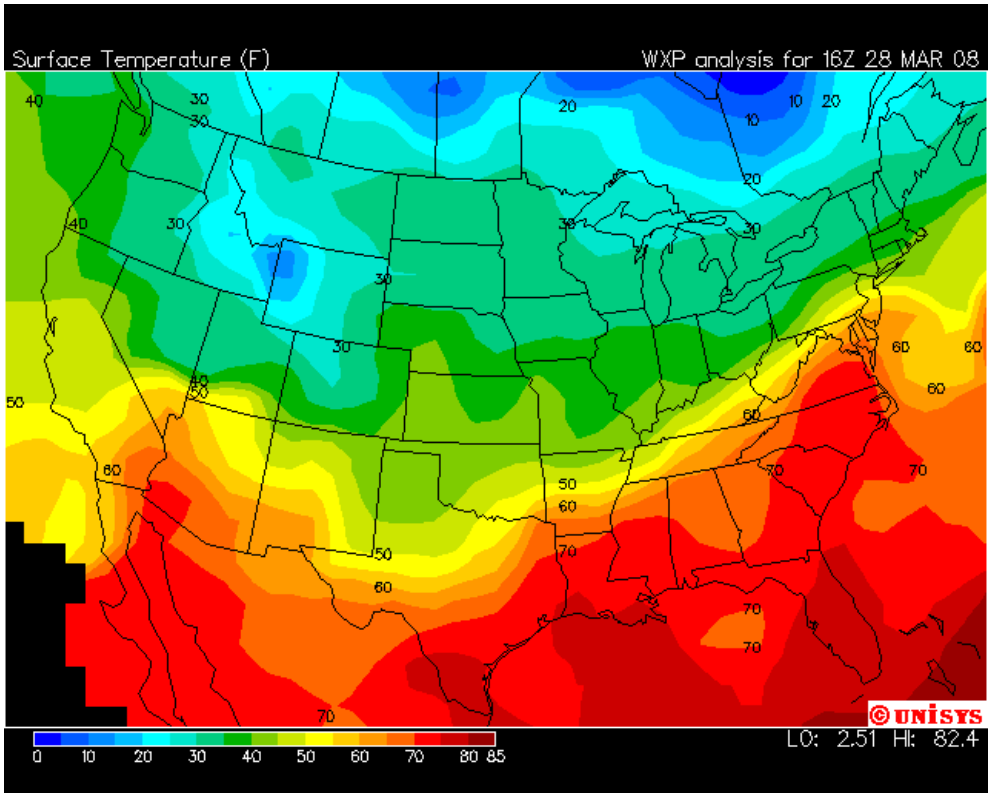

Figure 3.1. A temperature contour example of USA

3.3 Aggregation through Contours

In order to generate a contour map, we have to define a suitable contour scale for each application. The criterion is how much detail we require from maps. If we want

detailed maps, a fine contour scale should be used. If we only require less detail maps and energy is limited, we can set a coarser contour scale. It is a trade-off between map accuracy and energy cost.

Once the contour scale is defined, we use neighboring node value differences to detect contours at each sampling round. 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 the contour through communication. Our goal is to achieve energy efficiency and therefore to minimize the number and size of messages sent in the network.

In order to introduce our aggregation technique, we begin with a simple straight contour example. Consider a sub-routing tree of the network. Suppose a subset of nodes attempt to detect a straight line contour (figure 3.2) between them. A few reporting nodes N_3, N_4, N_5 and N_6 are chosen to report. Isovector Aggregation follows steps shown in figure 3.3: 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 following the data routing tree. 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. After N_1 receives the contour vector (P_1, P_3) from N_3 and (P_4, P_5) from N_4, it merges the two vectors to a new vector (P_1, P_{10}). In this merging process, contour points P_3 and P_4 are removed as redundant points by node N_1. N_1 then reports

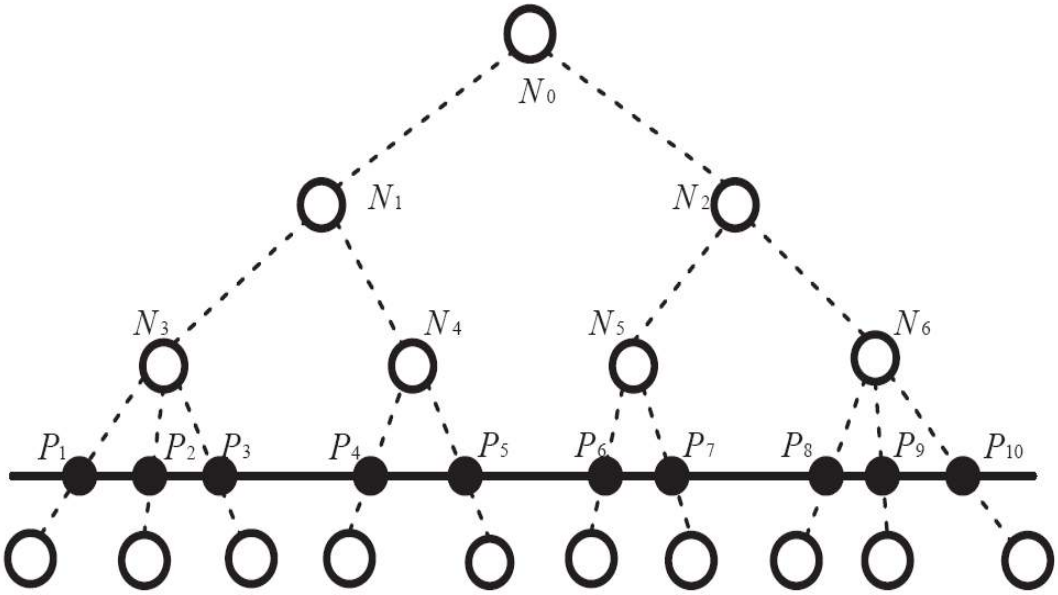

Figure 3.2. A straight contour in a sub-routing tree

the vector (P_1, P_5) to N_0 and N_2 reports the vector (P_6, P_{10}) to node N_0. Such a process will be repeated along the data routing tree. Finally, node N_0 will merge the vectors (P_1, P_5) and (P_6, P_{10}) and generate the vector (P_1, P_{10}) (figure 3.3). This vector is sufficient to represent the straight line contour. N_0 then only reports (P_1, P_{10}) to the base station. In this aggregation process, because redundant data is removed, the total data transmitted is reduced, which decreases network energy consumption.

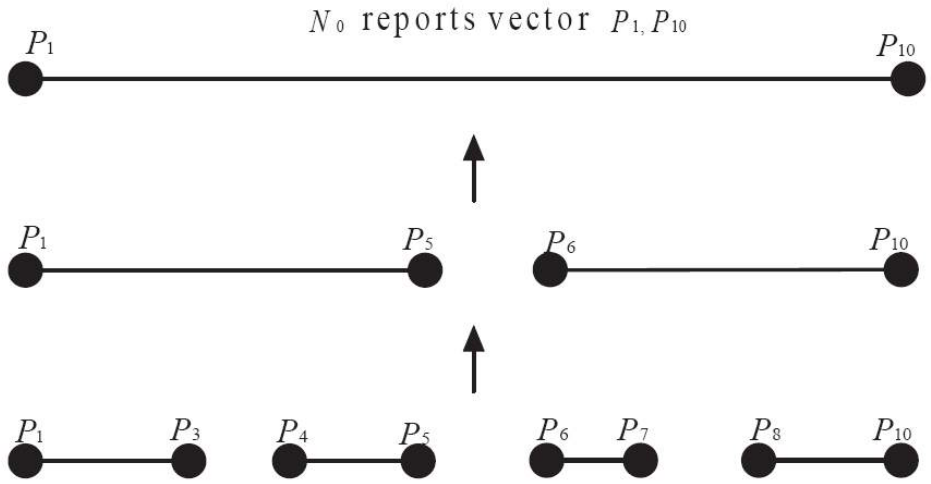

Figure 3.3. An Isovector Aggregation example

3.4 Definitions

Definition 3.1: Network Model. The *communication topology* of the wireless sensor network can be represented by a simple graph $G = (V(G), E(G))$ in the plane, where $V(G) = \{ v_1, v_2, ..., v_n \}$ is the set of sensor nodes in the wireless sensor network. An edge e_{ij} ($e_{ij} = e_{ji}$) exists for each pair of nodes v_i and v_j, if they are within communication range of each other. $E(G)$ is the set of all edges.

Definition 3.2: Neighborhood. Let u be a node. The *neighborhood* N_u of u contains as elements all nodes v such that there exists an edge e_{uv}. We call v a *neighbor* of u.

As specified above, some predefined thresholds are set which partition the value into different value ranges for contour mapping. If a node and one of its neighbors are in different value ranges, there exists contour(s) between them. Nodes in the same value range have similar properties.

We denote the value at node u by $R(u)$ and its value range by $Range(u)$. Starting from the initial scale 0, the ranges used in thesis are continuous intervals ([0,9), [10,19), [20,29), etc.). 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 3.3: Contour Neighborhood. Let u be a node. The *contour neighborhood* CN_u of u is the subset of N_u where for any node v in CN_u, $Range(u) \neq Range(v)$. For simplicity, we call each member of CN_u a *contour neighbor* of u. For any $v \in CN_u$, we have $u \in CN_v$. $|CN_u|$ represents the number of element in CN_u.

Definition 3.4: Contour Node. A *contour node u* is a node that has at least one contour neighbor. It immediately follows that $|CN_u| > 0$ if and only if u is a contour node.

Definition 3.5: 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 starting node v_1 is randomly assigned in advance. The *contour neighborhood ring* associated with u is a ring data structure $[n_1, n_2, ..., n_n]$ starting from v_1 where

$$n_i = \begin{cases} R(v_i) & if \quad Range(u) \neq Range(v_i) \\ null & if \quad Range(u) = Range(v_i) \end{cases}$$

for $1 \leq i \leq n$. Figure 3.4 shows an example of a node representation with its corresponding contour neighborhood ring. In figure 3.4, numeric numbers are values of

nodes. The white nodes are neighbors that have the same value range of u ([40,49)) and the black ones are neighbors with different value ranges from u.

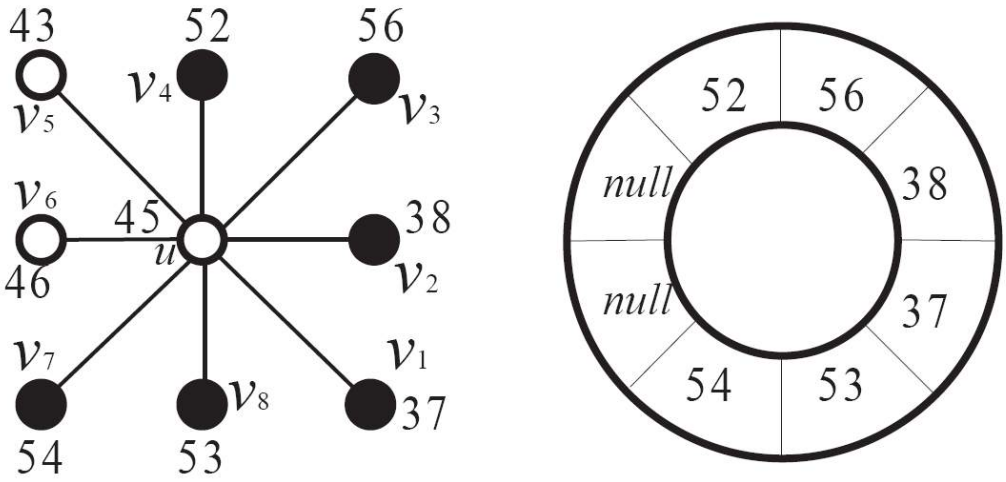

Figure 3.4. Node u's representation and the neighborhood ring

In this thesis, we use an array to represent each neighborhood ring. It does not matter which node is chosen as the starting node because we connect the head and the tail of the array when we want to use it. For example, both the sequence of the form [37, 38, 56, 52, null, null, 54, 53] and the sequence of the form [56, 52, null, null, 54, 53, 37, 38] represent the same array associated with figure 3.4 where v_1 is the starting node for the first sequence and v_7 is the starting node for the second sequence.

3.5 Message Types

Each message transmitted in a wireless sensor network consists of two fields; the *message header* contains required information for communication, including the source address, the destination address and the message type, and the *data field*. In this thesis, all

messages have the same message header but contain different message type information. The real differences among messages lie in data fields.

Type	Description
Query	Base station initiating fetching contours
Negotiation	Node broadcasting ID, value and location
Notification	Node broadcasting ID and value
Vector	Response message to query
Updating	Response message without contour information
Acknowledgement	Response message to children

Table 1. Message types

We define six types of messages (table 1) as follows:

1) A *query* message is broadcast by the root at the network setup phase. The data field includes the contour scale and sampling interval information that should be delivered to every node in the network.

2) A *negotiation* message tells neighbors the node ID, initial value, and location of the owner node. It is only broadcast after a node receives a query message. The data field contains the node ID, the contour scale and the node location. This message will be sent only once by each node during the network initialization phase. Each node saves neighboring location information after receiving these messages from neighbors.

3) *Notification* messages will be sent when data changes cause a contour to appear or disappear. The data field is composed of the node ID and value. Because each node already has neighbors' location information after network initialization, using the notification message, we can save additional bytes when compared with a negotiation message.

4) A *vector* message contains the reporting node's ID, contour vectors, and contour values. This is the most frequently used message type in Isovector Aggregation and it is a message type with variable length. Different vector messages may have different lengths because they may contain different numbers of contour vectors.

5) An *updating* message is a kind of reporting message that only contains the sending node's ID. If a node receives an updating message from one of its children, it will use the contours previously reported by that child as the current ones. In this way, nodes do not have to send large amounts of contour vector data back at each round if some contours do not change.

6) An *acknowledgement* message is used to deal with packet loss. A node can send this message back after receiving a reporting message (updating message or vector message) from a child.

3.6 Aggregation Time Model

There are many aggregation time models (Solis and Obraczka 2004(b)), including:

1) *Periodic simple:* nodes wait a predefined amount of time, aggregate all data received in that period, and send out a single reporting message.

2) *Periodic per-hop:* a node produces an aggregate message and sends it out after it receives all reporting messages from its children.

3) *Cascading timers:* similar to periodic per-hop, a node produces a single packet and sends it out after it receives all reporting messages from its children. The difference is that in each sampling round the timeout of each node is set based on the node's

position in the data routing tree. A child node's timeout will happen before its parent's timeout.

The time model we use in this thesis is based on cascading timers. Cascading timers is a mechanism designed to fit periodic data generation applications in which nodes produce data at regular periods. It has an important property (Solis and Obraczka 2004b): the cascading timers model does not require clock synchronization among nodes. In a wireless sensor network, no matter how efficient clock synchronization mechanisms become, they will require additional message exchange among nodes and thus incur additional energy consumption. In Chapter 1, we stated that saving sensor network energy is one of the most crucial goals of all wireless sensor network applications. Additional message exchange should be avoided if possible.

The cascading timers model starts by having the base station broadcast a query message to all nodes hop by hop. Upon receiving the query message, nodes send a reply back to their parent to build the routing tree. According to the cascading timers model, in each sampling round, a node's reporting time is set based on the node's position in the data routing tree. The node that has most hops from the base station is the first one that reports and all children of the base station report lastly. Thus, a node sends its report right before its parent does. A given node aggregates data received from its children into a single message, which is then forwarded to its parent. Using the cascading timers mechanism, a node merges contour vectors after it receives all vector messages from its children. In order to handle packet loss, we modify the time model slightly and set a relatively longer time-out for those nodes near the base station.

3.7 Summary

This chapter gives preliminaries of the thesis. The definition of a contour map is explained and we show the core idea of Isovector Aggregation though a straight contour line example. The thesis has two assumptions. The first assumption is that each node has a unique ID. This is reasonable for most wireless sensor network applications. We also assume that each node knows its location though a certain technology or by pre-input. This assumption is the prerequisite of our work because contour points are calculated based on nodes' locations and values.

We give the definition of a contour neighborhood ring with explanations using an example. This neighborhood ring is the most important data structure used to store neighbor information and to generate contours in Isovector Aggregation. We define six types of messages in the network for different purposes. The vector message is the most important one and it has variable length. In order to do the aggregation in a wireless sensor network, we choose the cascading timers mechanism as the aggregation time model.

CHAPTER 4

ISOVECTOR AGGREGATION

In this chapter we present our in-network Isovector Aggregation method which includes four parts: local contour generation, reporting node selection, contour simplification and contour merging. Additional temporal suppression for reducing reporting message size is also discussed for phenomena whose rate of change is slow. We assume the query for continuous contours monitoring is processed repeatedly over a series of rounds, where each node generates a value at the beginning of each round. All nodes collaborate together using the Isovector Aggregation in that round. At the end of each sampling round, we generate a contour map snapshot at the base station that is complete in-network.

As the first step of Isovector Aggregation, following the negotiation between all neighboring node pairs, local contour generation refers to the process that generates initial contours through the comparison between node values and the values saved in the neighborhood ring. As we discussed in Chapter 2, the comparison is simple and straightforward. If a node and one of its neighbors are not in the same value ranges, then at least there is one contour between them and the contour value is between the two nodes' values. Though many nodes near the contours can generate such local partial contours, in most cases, we only choose half of the nodes near contours to report.

In order to achieve energy efficiency, such generated contours are progressively merged and simplified in the network along the fixed data routing tree built in the network initialization phase. If two contour lines with the same value are near to each other (distance is less than a predefined tolerance), we connect them using a straight line. In order to remove redundant contour points in a contour, the Douglas-Peucker polyline simplification algorithm is also applied in each level of the data routing tree. In the best cases, we only need two contour points to represent a straight contour. The repeated in-network contour merging and simplification procedures significantly reduce total data transmission.

After the base station receives all reports at the last step of each sampling round, it runs the similar contour merging and simplification algorithms and then plots all final contours.

This chapter develops several algorithms for the Isovector Aggregation method. As an aid to the reader, table 2 lists important variables and their descriptions.

Name	**Description**
Scale	The contour scale value (the default value is 10)
headID (hID)	The head ID of a contour vector
tailID (tID)	The tail ID of a contour vector
vArr	The array that saves contour vectors
tol	The distance threshold for connecting two contour vectors
needReport	The reporting flag of a node

Table 2. Important variables and descriptions

4.1 Local Contour Generation

In order to report contours for a monitored dynamic field, first we have to detect contours in each sampling round. The method we used for contour detection is straightforward: a contour exists between any two neighboring nodes if they are in different value ranges. In each round, right after a node senses a value from the physical world, it broadcasts such information to neighbors using a notification message. Each node, after receiving such messages, updates its neighborhood ring accordingly. Then the node can compare its own value with the neighbors' values stored in the neighborhood ring. If the node and one of its neighbors are in different predefined value ranges, a local contour is detected between them. We call the middle point between these two nodes a *contour point*. Each node has saved all neighbors' location information in the initialization phase and it knows its own location. Let u and v be two neighboring nodes and they are in different value ranges. Let (u_x, u_y) and (v_x, v_y) be the coordinate locations of nodes u and v. We can easily calculate the coordinates of the middle point location using the following equation.

$$MidPoint(x, y) = \left(\frac{u_x + v_x}{2}, \frac{u_y + v_y}{2}\right) \qquad \text{Equation (1)}$$

As we discussed in Chapter 3, if a data entry in the neighborhood ring of a node is set to null, the corresponding neighbor must be in the same value range as the node and no contour exists between them.

The null values saved in the neighborhood ring and the predefined contour scale work as separators dividing the ring into several partitions. A local contour vector will be generated for each partition by collecting all contour points in counterclockwise order in that partition. Let *Scale* denote the contour scale a user defines. 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 the neighbors is set as:

$$Value(v) = (R(u) \mod Scale) * Scale \qquad \text{Equation (2)}$$

where mod is the modulus operator. As an example, if $R(u)$, the value of the node u, is 45 and the *Scale* is 10, $Value(v)$ will be 40.

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:

$$Value(v) = (R(u) \mod Scale + 1) * Scale \qquad \text{Equation (3)}$$

For example, if $R(u)$ is 45 and *Scale* is 10, $Value(v)$ will be 50.

Figure 4 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 and local contour vectors (P_1, P_2), (P_3, P_4) and (P_7, P_8) will be generated for the three partitions. Here vector (P_1, P_2) is a contour with value 40, vector (P_3, P_4) is a contour with value 50 and vector (P_7, P_8) is a contour with value 50. Each generated vector has a *headID* (*hID*) and a *tailID* (*tID*) which are equal to the reporting node ID. Assume the ID of the node u is 100. Then in our example, vector (P_1, P_2)'s *hID* and *tID* are both 100. Vectors (P_3, P_4) and (P_7, P_8) have the same *hID* and *tID as* the vector (P_1, P_2). We can also see from figure 4.1, although contour point P_2 is near to contour point P_3, we do not connect P_2 and P_3 because they have different contour values.

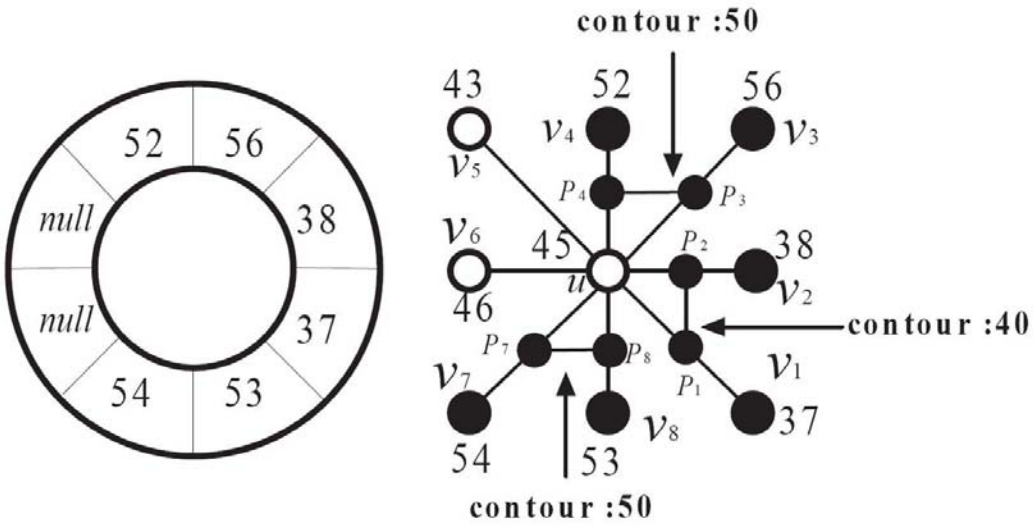

Figure 4.1. Node u 's neighborhood ring and corresponding local contour vectors

In practice, most reporting nodes only have one partition each, and hence each of them only generates one local contour vector. In some rare cases, a reporting node near a narrow contour will generate two or more partitions, which may indicate a *narrow contour* exists. A contour is called a narrow contour if it could be detected by a node's diagonal neighbors. For example, in figure 4.2 the contour could be detected by u 's three

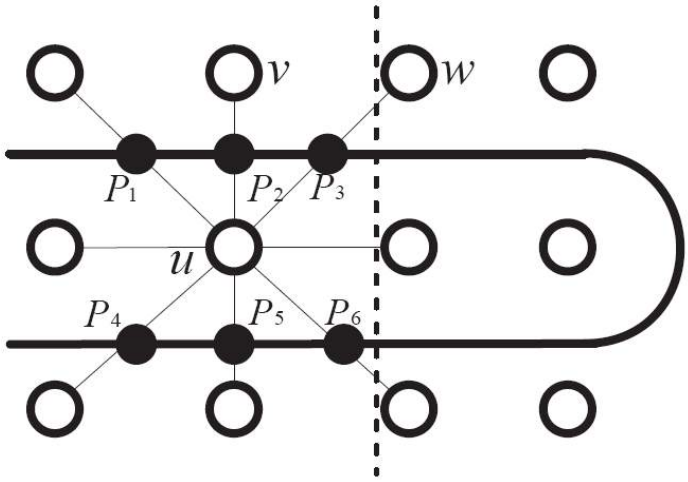

Figure 4.2. Narrow contour 1

pairs of diagonal neighbors, and this contour is a narrow contour. Besides differentiating contours of different values, the partition can be used to identify narrow contours. For the above narrow contour, our aim is that the final contour shape should be similar to the one shown in figure 4.3.

Figure 4.3. Narrow contour 2

Suppose node u generates vector V_1 (P_4, P_6, P_3, P_1) (figure 4.4) and reports it through node v, and the right part of the broken line also generates a vector V_2 which starts at P_i and ends with P_j. Suppose also that V_2 is reported though node w. When V_1 and V_2 meet at a common ancestor of v and w, it is not easy for this ancestor to merge V_1 and

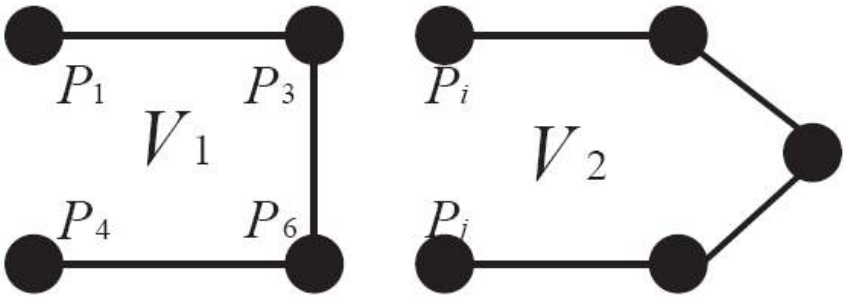

Figure 4.4. Narrow contour 3

V_2 together to form a new vector similar to the one shown in figure 4.3. Suppose u generates two vectors V_0 and V_1 and does not merge them. After the common ancestor receives three vectors, it is straightforward to connect P_3 to P_i, and P_6 to P_j (figure 4.5), because they are near to each other. Then the contour we get will be similar to the one shown in figure 4.3. In order to ensure V_0 and V_1 are not merged before they meet V_2, one simple condition we have to define is that vectors sent by the same node should not be directly merged in the network. This process is a part of the merging algorithm that will be illustrated in detail in section 4.3.

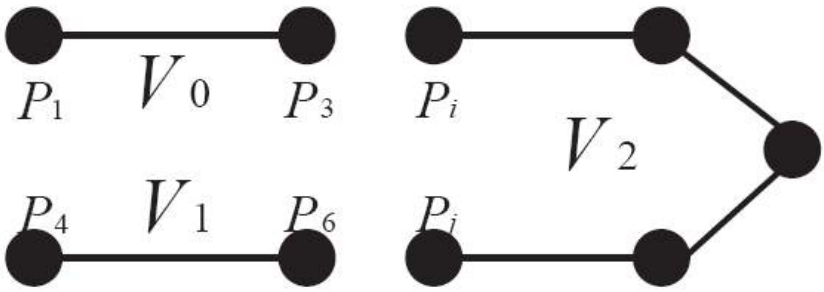

Figure 4.5. Narrow contour 4

4.2 Reporting Node Selection

Local contour detection in each round is based on neighborhood information gathered through node-to-neighbor communication. If the ranges of sensed values are changed, a node broadcasts a notification message to neighbors at the beginning of each sampling round. After receiving all notification messages from its neighbors, each node compares its own value with neighbors' values. Some neighbors' values might be in the same value range as the reporting node. As defined in Definition 3.5, the corresponding entries of the neighborhood ring will be set to null and they, together with the scale value, act as

separators to partition the contour ring. If some neighbor values are on different sides of a contour, 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.

The basic approach is to select all contour nodes to report. Besides bringing an additional communication overhead, this approach will result in a potential problem; two neighboring nodes may report an identical contour. Later when these two identical contours meet, we need to design more complicated merging algorithms to deal with such duplication. In fact, contour detection is symmetric. Not all contour nodes are required to generate contour vectors and report. In most cases, letting contour nodes on one side of the contours report is enough. We choose contour nodes in the higher value ranges to report. Each vector has a *headID* and a *tailID* which are equal to the ID of the node(s) that report(s) them. Therefore, by maintaining such information, the base station can know which side of the contour is in a higher value range and which side of the contour is in a lower range. In some cases, two adjacent nodes may not exist in consecutive value ranges. In this situation, both nodes in the lower value range and the higher value range will report. Figure 4.6 gives an illustration. In figure 4.6, Node u will report a contour with value 40 between u and v and node v will report a contour with value 50 between u and v.

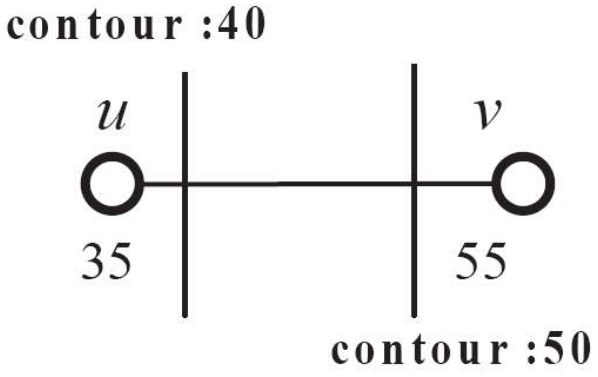

Figure 4.6. Reporting node seletion

4.3 In-network Contour Simplification

Different contour lines have different shapes. In Chapter 3, we use a straight contour line as an example to show the idea of Isovector Aggregation. It can be seen that two contour points (starting point, end point) are enough to represent it without losing any information. If the contour line shape is irregular, we may need more points to represent it. Isovector Aggregation is a progressive process through the data routing tree. After a node receives all vector messages from its children, if some contours heads or tails are near to each other and have the same contour value, they will be merged. Sooner or later, a contour may contain a large amount of contour points and in many cases they are redundant. We then need more bytes to represent them. Figure 4.7 shows two representations of the same contour lines. Suppose each contour point needs 2 bytes to be represented. Contour 1 needs 12 bytes (additional 2 bytes are required to store the contour value), but contour 2 only needs 6 bytes.

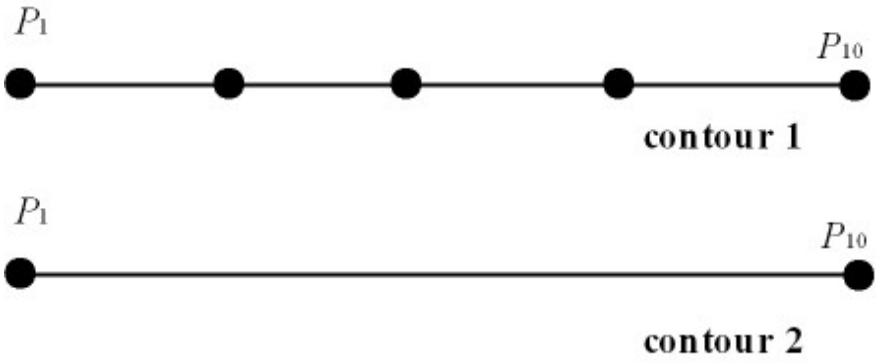

Figure 4.7. Comparison of two contours

Without losing contour fidelity, we consider how to weed out redundant points from a contour vector (figure 4.8). This polyline simplification problem has been researched over the years. In this paper, we use the Douglas-Peucker polyline simplification algorithm for vector simplification because it was best at choosing critical points when compared with others (White 1985). We should point out that although the Douglas-Peucker approach is chosen, any other polyline simplification method can be adopted if it can retain the shape of the original line.

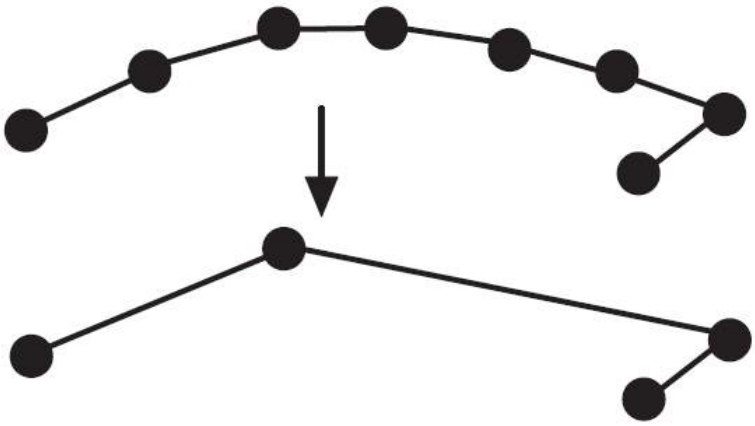

Figure 4.8. Simplify a contour vector

4.4 In-network Contour Merging

In-network contour merging is an important step to reduce the contour count. After a node receives a vector message, it saves it 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 tries to merge saved contour vectors in the *vArr* 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.

Though the merging algorithm is simple, there are two rules we must follow:

1) The rule for choosing the distance threshold is that it should be less than the distance between two adjacent nodes. Each contour point in a contour is a middle point between two adjacent contour nodes. When we define the distance threshold for merging, the neighboring node distance is an essential factor. For simplicity, we only discuss the situation in an evenly distributed network. Figure 4.9 illustrates this rule.

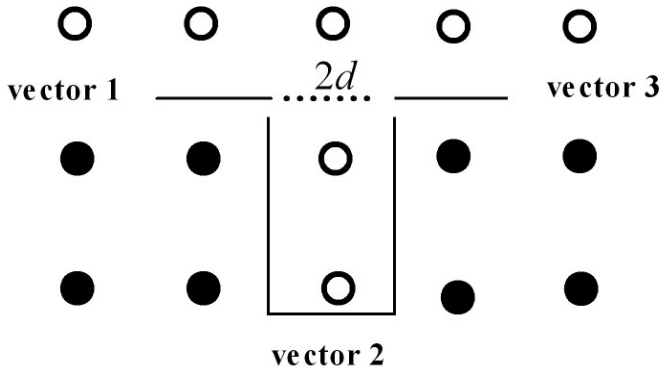

Figure 4.9. Before merging

In figure 4.9, there are three partial contour vectors, V_1, V_2 and V_3, with the same contour value. By our local contour generation approach, the distance between V_1 and V_2 will be less than d which is half of the distance between two adjacent nodes. The distance between V_2 and V_3 is also less than d and the distance between V_1 and V_3 is $2d$ which is the distance between two adjacent nodes. Ideally, we hope to connect V_1 with V_2 and connect V_2 with V_3. The final contour should be the one shown in Figure 4.10.

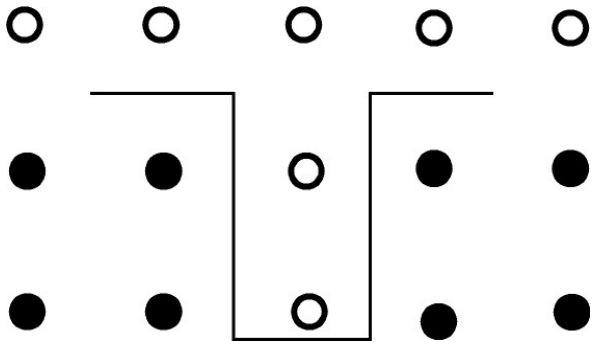

Figure 4.10. Correct merging

If the distance tolerance is equal to or larger than $2d$, it is possible that V_1 and V_3 are merged first. Suppose the merging result of V_1 and V_3 is V_4, Later the merging algorithm cannot merge V_2 and V_4 correctly and we cannot get the correct contour (Figure 4.11).

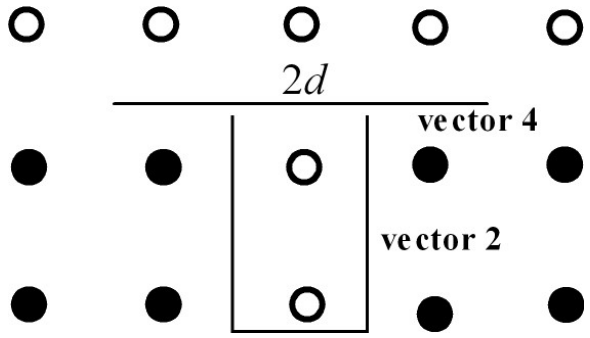

Figure 4.11. Wrong merging

2) 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 also used to avoid generating wrong contour shapes of narrow contour cases. More precisely, a vector head (tail) cannot be connected to another vector head (tail) if they have the same *headID* (*tailID*). From our merging algorithm, if two vectors sent by the same child should be merged, they will be merged at that child node according to the distance threshold-based algorithm (Algorithm 3, figure 4.14). The child node will transmit one merged contour rather than two unmerged contour vectors to the parent node.

By calculating the distance between two contour vectors, Algorithm 1 (figure 4.12) checks merging conditions.

Algorithm 1 ckMerge(V_1, V_2, tol)

1: **if** (V_1 or V_2 is a ring) **then**
2: **return** 0;
3: **if** ($V_1.value \neq V_2.value$) **then**
4: **return** 0;
5: calculate $D(V_1, V_2)$;
6: **if** ($D(V_1.h, V_2.h) \equiv D(V_1, V_2)$ and $D(V_1, V_2) < tol$ and $V_2.hID \neq V_2.hID$) **then**
7: **return** 1;
8: **if** ($D(V_1.h, V_2.t) \equiv D(V_1, V_2)$ and $D(V_1, V_2) < tol$ and $V_2.hID \neq V_2.tID$) **then**
9: **return** 2;
10: **if** ($D(V_1.t, V_2.h) \equiv D(V_1, V_2)$ and $D(V_1, V_2) < tol$ and $V_2.tID \neq V_2.hID$) **then**
11: **return** 3;
12: **if** ($D(V_1.t, V_2.t) \equiv D(V_1, V_2)$ and $D(V_1, V_2) < tol$ and $V_2.tID \neq V_2.tID$) **then**
13: **return** 4;
14: **return** 0;

Figure 4.12. Algorithm 1

If the distance is less than a predefined tolerance and the merging parts are not reported by the same node, Algorithm 1 returns a positive value which will be used by Algorithm 2 (figure 4.13). Algorithm 2 merges two specific contour vectors. For example, when *mgValue* is equal to 1, V_1 head and V_2 head will be connected together. The new generated V_3 then replaces V_1 in the *vArr* structure.

Algorithm 2 mgVector(V_1, V_2 $mValue$)

1: **if** ($mValue \equiv 1$) **then**
2: $V_3 \leftarrow Reverse(V_1) + V_2$;
3: $V_3.hID \leftarrow V_1.tID$;
4: $V_3.tID \leftarrow V_2.tID$;
5: **if** ($mValue \equiv 2$) **then**
6: $V_3 \leftarrow Reverse(V_1) + Reverse(V_2)$;
7: $V_3.hID \leftarrow V_1.tID$;
8: $V_3.tID \leftarrow V_2.hID$;
9: **if** ($mValue \equiv 3$) **then**
10: $V_3 \leftarrow V_1 + V_2$;
11: $V_3.hID \leftarrow V_1.hID$;
12: $V_3.tID \leftarrow V_2.tID$;
13: **if** ($mValue \equiv 4$) **then**
14: $V_3 \leftarrow V_1 + Reverse(V_2)$;
15: $V_3.hID \leftarrow V_1.hID$;
16: $V_3.tID \leftarrow V_2.hID$;
17: $V_3.value \leftarrow V_1.value$;
18: $V_1 \leftarrow V_3$;

Figure 4.13. Algorithm 2

By calling Algorithms 1 and 2, algorithm 3 (figure 4.14) continuously checks the merging status of two vectors and merges them if they meet the merging standard.

All new generated vectors are also stored in the local *vArr* (last line of algorithm 2) structure. After the local time-out, Douglas-Peucker algorithm will be called for vector simplification. Those simplified vectors will be added to the vector message and sent to the parent. Each node along the data routing tree performs the same process until the base station receives the final results.

Algorithm 3 mergeVectors($vArr[]$, tol)

```
1: for (i ← 0 to vArr.len − 3) do
2:    for (j ← i + 1 to vArr.len − 3) do
3:       mValue ← ckMerge(vArr[i], vArr[j], tol);
4:       if (mValue ≠ 0) then
5:          mgVector(vArr[i], vArr[j], mValue);
6:          remove vArr[j] from vArr;
7:    if (vArr[i] changed) then
8:       i ← i − 1
```

Figure 4.14. Algorithm 3

4.5 Updating with Temporal Suppression

In continuous contour monitoring, choosing only a few contour nodes to report is already an approach that combines both spatial suppression and temporal suppression. We try to explore more suppression for some rarely changed cases.

In some applications, contours do not change frequently because events are rare, and most nodes' values are unchanged or only slightly changed over time. In such situations, nodes need not transmit vector messages to the base station along the data flow tree. In each sampling round of Isovector Aggregation, each node saves a copy of the detected and received contours before merging. In the next sampling round, if a node finds that such information was not changed, it only sends an updating message to its parent. The parent will use the node's previous contours as the current ones. If a node only receives updating messages from children in a specific round, it transmits an updating message to its parent in the same way.

Combining local contour generation, reporting node selection, contour simplification, contour merging, and temporal suppression together, our algorithm for Isovector

Aggregation is shown as Algorithm 4 (figure 4.15). It follows the following steps in each sampling round. (Note: a node always monitors incoming messages).

Algorithm 4 Isovector Aggregation

```
1:  while 1 do
2:    gets sensed value;
3:    if (value range changes) then
4:      broadcasts notification;
5:    while (monitoring 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 (receives children's updating) then
12:       marks the corresponding existing contour;
13:       needReport = true;
14:     if (reporting timeout) then
15:       if (in a higher value range) then
16:         generates and saves local contours;
17:         needReport = true;
18:       else if (in non-consecutive value ranges) then
19:         generates and saves local contours;
20:         needReport = true;
21:       if (needReport) then
22:         if (all received msgs are updating and no new
            local contours) then
23:           composes an updating and reports;
24:         else
25:           merges contours;
26:           simplifies contours (Douglas-Peucker);
27:           composes a vector and reports;
```

Figure 4.15. Algorithm 4

1) At the beginning of a sampling round, if the value range of a node changes, the node broadcasts a notification message to all neighbors.

2) If a node receives a notification message from its neighbors, it updates its neighborhood ring.

3) If a node receives a vector message from a child, it saves the received contours and sets the *needReport* flag true because it has to forward received information to its parent at least.

4) If a node receives an updating message instead of a vector message, it marks the corresponding contours saved in the last round. Such marked contours will be used for merging. The *needReport* flag is also set to true.

5) After the cascading timers model timeouts the node, the node compares neighboring values saved in the neighborhood ring. If the node is in a higher value range of the generated contours or it is in a non-consecutive value range, it generates local contours and saves them for future merging. The *needReport* flag is set to true because it has to report detected contours back.

6) If a node only receives updating messages and all generated local contours are the same as contours generated in the last sampling round, it only needs to compose an updating message and send it out later. Otherwise, the Isovector Aggregation algorithm calls the merging algorithm (Algorithm 3) to reduce the total contour vectors it saves. Then the Douglas-Peucker line simplification algorithm is applied on all merged contours, and all redundant contour points are removed. The node then composes a vector message that contains all simplified contours in the data field.

The process graph shown in figure 4.16 illustrates steps for the Algorithm 4.

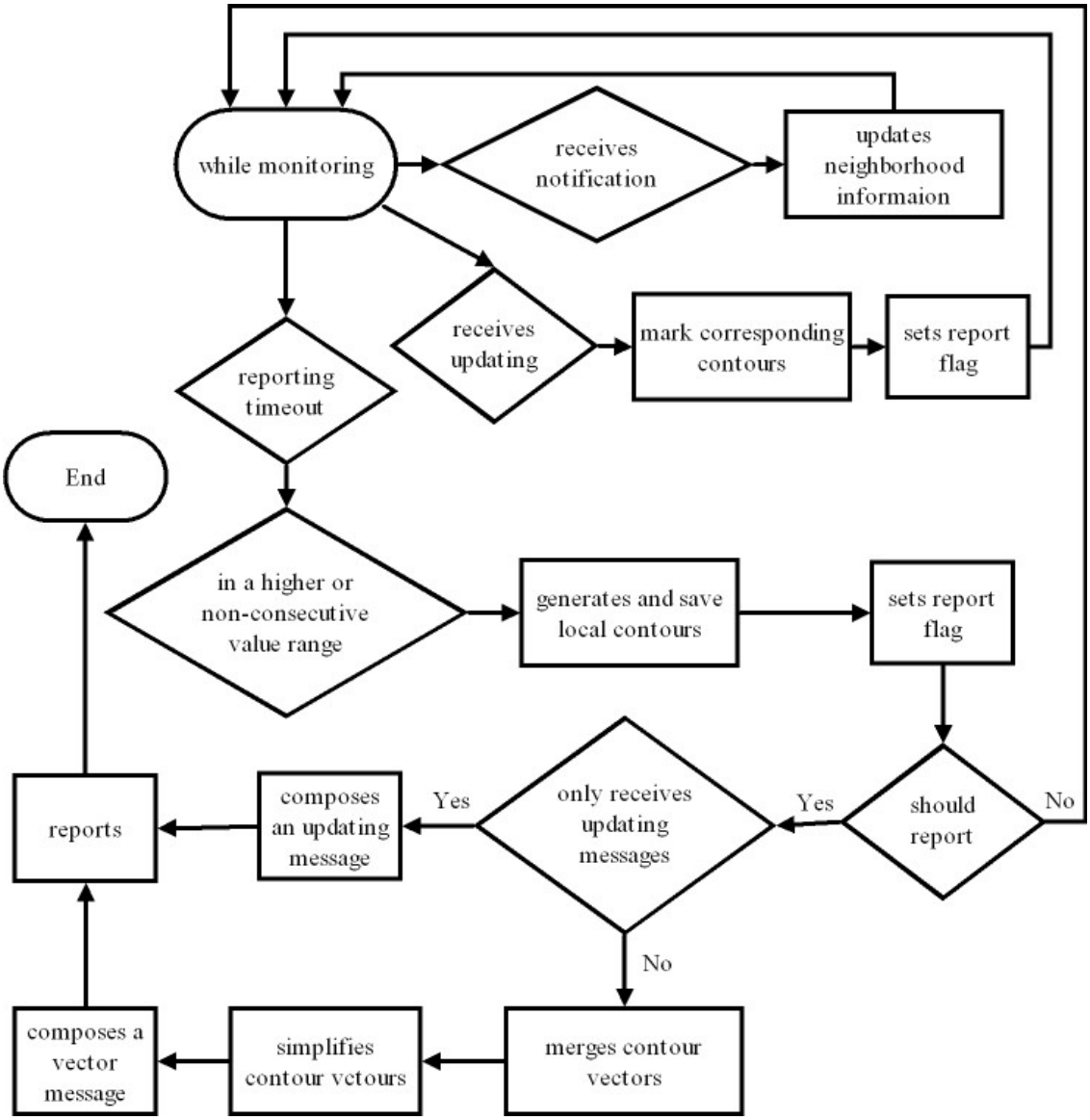

Figure 4.16. Process graph of Isovector Aggregation

4.6 The Base Station

The base station is the root of the data routing tree and it has two functions.

1) In the network initialization phase, the base station broadcasts a query message to build the data routing tree and disseminate such information to all nodes in the network.

2) At the end of each sampling round, it will receive all reported contour vectors from its children. Similar to other nodes in the network, it does merging operations, but there is a slight difference. At the base station, contour vectors sent by the same node can be merged together. Algorithm 2 is changed to Algorithm 5 (figure 4.17) and runs at the base station side. When the base station merges contour vectors, it will not take vector *headID* or *tailedID* into consideration. In this way, we ensure all contours can be merged if they are near to each other to generate integrated contours. After getting these contours, they can be drawn and we get the contour map for this sampling round.

Algorithm 5 BSCkMerge (V_1, V_2, tol)

```
1: if (V1 or V2 is a ring) then
2:     return 0;
3: if (V1.value ≠ V2.value) then
4:     return 0;
5: calculate D(V1, V2);
6: if (D(V1.h, V2.h) ≡ D(V1, V2) and D(V1, V2) < tol)
   then
7:     return 1;
8: if (D(V1.h, V2.t) ≡ D(V1, V2) and D(V1, V2) < tol)
   then
9:     return 2;
10: if (D(V1.t, V2.h) ≡ D(V1, V2) and D(V1, V2) < tol)
   then
11:    return 3;
12: if (D(V1.t, V2.t) ≡ D(V1, V2) and D(V1, V2) < tol) then
13:    return 4;
14: return 0;
```

Figure 4.17. Algorithm 5

4.7 Dealing with Packet Loss

Vector messages sent by nodes near the base station may contain a great deal of contour information and so their size is relatively large. If such a message is dropped by the network, we may not obtain a good contour map at the base station. For the nodes near the base station, we introduce an acknowledgement message which will be sent to children by a parent node after receiving a vector message. Then such children can report again if they do not receive such an acknowledgement message from parents after a short timeout.

4.8 Summary

In this chapter, we describe our in-network Isovector Aggregation method. The Isovector Aggregation algorithm includes four components: local contour generation, reporting node selection, contour simplification and contour merging.

1) *Local contour generation:* In the previous chapter, we designed the neighborhood ring data structure to store all neighboring values. By comparing the value differences between a node and its neighbors, a node generates local contours in each specific sampling round.

2) *Reporting node selection:* Contour detection is symmetric. If a contour exists between two nodes, both nodes are able to detect it. If both of them report, the network will send redundant data and incur undesirable network traffic. We only choose contour nodes in the higher value ranges to report.

3) *Contour simplification:* We use the Douglas-Peucker polyline simplification algorithm for contour vector simplification. The benefit of the contour simplification is

obvious. It reduces the numbers of bytes required to represent contours without significantly decreasing the contour precision.

4) *Contour merging:* A distance tolerance-based algorithm is used to merge contours with the same value. The rule for choosing the correct distance threshold is discussed in detail. The base station runs a similar merging process and then draws final generated contours. No additional processing or interpolation is required. This is one of the most different features between Isovector Aggregation and previous techniques.

In this chapter, we also consider contour updating with temporal suppression. If a vector message is the same as the report of the last round, the node only needs to report a small updating message to its parent. The parent will use its last report as the current one. We use a retransmission mechanism to deal with possible package loss near the base station.

CHAPTER 5

THEORETICAL ANALYSIS

Intuitively, Isovector Aggregation should be more energy efficient than many previous methods mentioned in Chapter 2 because it only chooses a few contour nodes to report and progressively removes unwanted data along the data routing tree. In order to verify this intuition, in this chapter, we do some theoretical analysis of our Isovector Aggregation method, focusing on traffic generation and traffic reduction during transmission.

Traffic generation refers to how many nodes are required to report before the aggregation. For example, if there are n nodes in the wireless sensor network and each node needs to report, the complexity of traffic generation is $O(n)$.

Traffic reduction means that we use any in-network aggregation technique to reduce or compress generated data before the base station receives them or we report all data directly without in-network processing.

5.1 Traffic Generation

As we can see from Chapter 2, previous techniques often require all nodes to generate reports before aggregation. Suppose the network has n nodes, the complexity of traffic generation for those techniques is $O(n)$.

In Iso-map (Liu and Li 2007), Liu and Li report that their method selects $O(\sqrt{n})$ number of contour nodes. Then the total contour length is also $O(\sqrt{n})$ in a wireless sensor network with infinite density ($n \to \infty$ and $d \to 0$, where d is the communication range of a node). Because we use mid-points to represent contours and we only choose contour nodes to report, on average the number of points in contours is between $O(\frac{\sqrt{n}}{2})$ and $O(\sqrt{n})$. Thus the complexity of traffic generation of Isovector Aggregation is also $O(\sqrt{n})$. This result may indicate that, from the perspective of traffic generation, Isovector Aggregation is more scalable than many techniques (COACH and Isobar Aggregation). Isolines Aggregation and Iso-map have the same property.

5.2 In-network Traffic Reduction

In real applications, the amount of communication overhead that can be reduced by Isovector Aggregation depends on contour shapes, contour locations, network topology and the *simplification ratio* (the percentage of data reduced by a simplification method) of each inner node. We analyze Isovector Aggregation and compare it with Isolines Aggregation in a simplified m-level binary tree network topology. A contour exists between leaf nodes and $m-1$ level nodes (figure 5.1). All $m-1$ level nodes are chosen to report. So there are $2m-1$ reporting nodes in total that have to transmit data to the root of this tree where the integrated contour will be generated. Suppose each inner node can simplify the merged contour by a fixed $1-k$ ratio after receiving children' reports, the size of the new generated contour by an inner node is in ratio k to the total contour

size received by the node. In this binary tree case, k is in the range [0.5, 1]. The packet header size is usually a small constant and we omit it in the analysis.

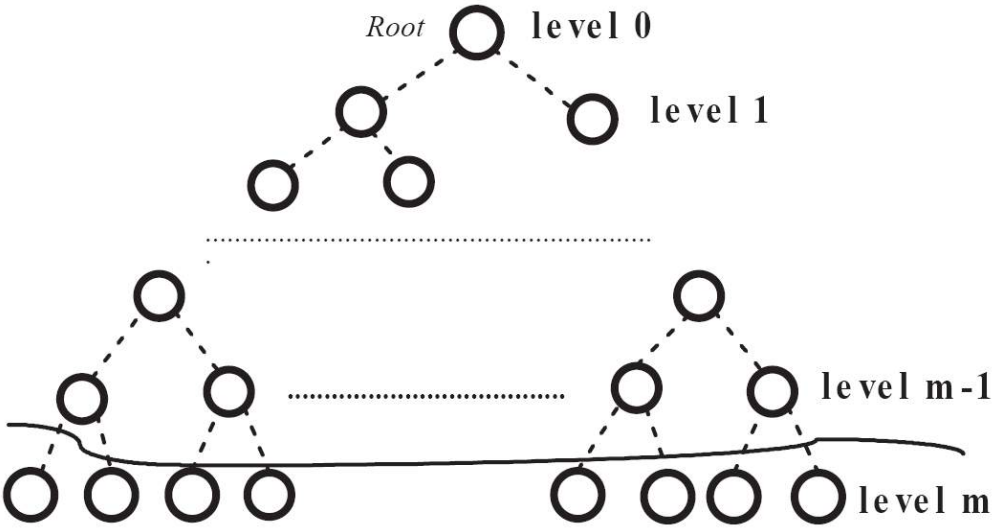

Figure 5.1. Traffic reduction in a binary tree structure

In Isolines Aggregation, each reporting node contains its own data along with that of its two children (ID, and Value) and this requires 12 bytes of storage. In Isovector Aggregation, each reporting node contains a contour vector starting point location, end point location, *headID, tailID* and the contour value, and this requires 10 bytes in total. We can see that even when contour vectors are not simplified, a reporting node of Isovector Aggregation requires fewer bytes than Isolines Aggregation, if there is more than one neighbor on the other side of the contour.

Now we compare how much data are sent by Isolines Aggregation and Isovector Aggregation. There is no in-network data reduction mechanism in Isolines Aggregation, and each reporting message in Isolines Aggregation has to travel $m-1$ hops before

reaching the root. So the total size of data transmission is: $S_{isolines} = 12 \cdot 2^{m-1} \cdot (m-1)$. The complexity of data transmission of Isolines Aggregation is always $O(m \cdot 2^m)$.

In Isovector Aggregation, after a parent receives two reports from its children, it merges them into one. Let the received contour size be s. After merging and simplification, the new contour size will be $2sk$. So, the total size of data transmission is:

$$S_{isovector} = 10 \cdot (2^{m-1} + 2k \cdot 2^{m-2} + ... + 2^{m-1} \cdot k^{m-2})$$

$$= 10.2^{m-1} \cdot (1 + k + k^2 + ... + k^{m-2})$$

When $k = 1$, all contours are not simplified after merging and we reach the upper bound of Isovector Aggregation. This situation occurs when we want to keep all contour points, in this case, we have $S_{isovector} = 10 \cdot 2^{m-1} \cdot (m-1)$. We can see that data transmission by Isovector is less than Isolines Aggregation but the complexity is also $O(m \cdot 2^m)$. When $0.5 \leq k < 1$, contours are simplified and we have $S_{isovector} = 10 \cdot 2^{m-1} \cdot \frac{1 - 0.5^{m-2}}{1-k}$.

In particular, when $k = 0.5$, we have $S_{isovector} = 10 \cdot 2^{m-1} \cdot \frac{1 - 0.5^{m-2}}{1-k} = 10 \cdot (2^m - 2)$ and this is the lower bound of the Isovector Aggregation. The complexity is $O(2^m)$. The total data transmission is reduced by a factor of m compared to Isolines Aggregation. When we remove all inner contour points of a contour, we can reach this bound.

What we show in the above is only a simplified situation. In a real application, contour lines will not always be straight lines and they may not all exist in the lowest level of the data routing tree. Depending on the different structures of data routing trees and the real situation in the dynamic field, we may get different results. But Isovector Aggregation has the same performance as Isolines Aggregation in the worst situation.

5.3 Summary

In this chapter, we have developed theoretical analyses for Isovector Aggregation. We have shown that Isovector Aggregation only incurs $O(\sqrt{n})$ network traffic. The in-network traffic reduction is also considered.

Comparison with other techniques given in table 3 shows the performance of Isovector Aggregation.

Method	**Traffic generation**	**Traffic reduction**
No Aggregation	$O(n)$	No
IsoBar	$O(n)$	Yes
Iso-map	$O(\sqrt{n})$	No
Isolines	$O(\sqrt{n})$	No
Isovector	$O(\sqrt{n})$	**Yes**

Table 3. Comparisons of different techniques

CHAPTER 6

EXPERIMENTAL ANALYSIS

In this chapter, we do experimental analyses of Isovector Aggregation through simulation in the NS2 (Network Simulator 2) environment, and we make a comparison with Isolines Aggregation. We do not implement Isobar Aggregation for comparison because it has been shown that Isolines Aggregation performs significantly better than Isobar Aggregation (Solis and Obraczka 2005). The simulation results are also compared with the no-aggregation method in which nodes simply send their values to the base station through the routing tree, and negotiations between different nodes are not required. Basically, with the no-aggregation method the base station will get a value from each node and generate the most accurate map. However, in most real wireless sensor network applications the considerable network traffic caused by having all nodes report will bring considerable packet loss, which means the base station cannot receive all reporting messages. In the following simulations, Node ID and location information are all two bytes long. Temperature information is also two bytes. The contour scale is set to 10 and the starting value is zero. Though we choose temperature as the measurand, Isovector Aggregation can measure any sensed variable.

6.1 Simulation Setup

We use a wireless sensor network consisting of 16×16 (256) nodes arranged in a $400m^2$ evenly spaced grid to monitor temperature. The distance between each adjacent node pair is 25m, and the base station is placed in the center of the network. We should point out that although in these simulations, nodes are placed according to a grid pattern, similar to Isolines Aggregation, Isovector Aggregation 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 except outer layer nodes has 8 neighbors. FLIP (Solis and Obraczka 2004a) is used as the network protocol. The distance tolerance used for the Douglas-Peucker algorithm is 6m, and the distance threshold for connecting two contour vectors is 14m that is less than the distance between any two adjacent nodes.

The tree-base routing scheme (Madden et al. 2002) is used for simulations. After the startup of the network, the base station broadcasts a query message on its radio. All nodes that hear the query message process it and rebroadcast it on to their neighbors. They keep on broadcasting until all nodes in the network have heard the query message. In this process, a routing tree is built. Then, each node broadcasts the negotiation message to neighbors. Each node initializes its contour ring by means of the negotiation messages it receives. The neighbor ID and location pair information are saved. Next time, nodes only have to broadcast notification messages. Finally, the network is fully initialized, and each node then starts reporting according to the query information it receives.

6.2 Scenarios and Evaluation Metrics

There are three simulation scenarios for temperature monitoring. The first scenario includes detecting two straight line contours with value 40 and 50 respectively, and detecting irregular contours. This simulation scenario is used to see if the Isovector Aggregation is adaptable to different contour shapes. In the second scenario we change the contour node densities instead of increasing the node count of the network. The total data transmission in different contour node ratios is recorded. This scenario will give us an alternative view of the scalability of Isovector Aggregation. In a dynamic field, phenomena change over time, and it is important that Isovector Aggregation should be suitable for the continuous contour mapping. In order to verify the suitability for generating contour maps in a continuous manner, we focus on continuous contour monitoring in the third simulation scenario.

There are two criteria for evaluation:

1) *Data transmission size:* This reflects the energy consumption by different approaches. For Isovector Aggregation and Isolines Aggregation, data transmission for negotiation between nodes is also included. We have proved that Isovector Aggregation is more scalable than many previous works because of the $O(\sqrt{n})$ traffic generation and the in-network traffic reduction. Counting the data transmitted will give us a clearer view of the network traffic of different aggregation techniques. While this might seem to not take energy consumption for receiving data into consideration, we should note that the cascading timers model is well suited to any (MAC) protocols or external control algorithms that switch idle nodes off to low-power radio mode because communication is

not taking place (Solis 2005). This approach can save nodes' receiving power. For the no-aggregation method this mechanism is not applicable without additional modification.

2) *Contour map accuracy:* This corresponds to the query precision by different approaches. The contour map accuracy is calculated as the percentage of points (80×50 points placed on the map) that are actually in correct value ranges when compared to the baseline map generated using all node values. We use the ArcView GIS software package (ArcView, ESRI) as the external tool for interpolation and visualization at the base station side. 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 the contour generation is a part of the aggregation

6.3 Detecting Static Contours

We use a regular contour map for evaluation. This map includes two vertical contours with value 40 and 50 respectively that go across the wireless sensor network from the top to the bottom. The contour node ratio in this case is 25%. In other words, 256×0.25 = 64 nodes can detect contours.

By Isovector Aggregation, the base station only produces two contour vectors and each contour vector only consists of two points. This implies that Isovector Aggregation only uses a starting point and an end point to represent a straight line. An example of received contour vector is 40:(99, 387):(99, 12). Here 40 refers to the contour value. (99, 387) is the start point coordinate and (99, 12) is the end point coordinate. In all 10 run simulations, Isovector Aggregation produced exact contours. The data size transmitted by

Isovector Aggregation is 3102 bytes with standard deviation (sd) 199 bytes; whereas, 5025 bytes (sd 494) are transmitted by Isolines Aggregation; and 12331 bytes (sd 429) are sent by the no-aggregation method. Compared to Isolines Aggregation, Isovector Aggregation sends 38% less data. Compared to the no-aggregation method, Isovector Aggregation sends 74% less data. Since our Isovector Aggregation approach produces the exact contour map all the time, it is not meaningful to compare the map accuracies with other methods in this regular contour map case.

An irregular contour map is also used for evaluation. Because Isovector Aggregation is compared with the Isolines Aggregation, the map we used in this scenario is similar to the one used in Isolines Aggregation (Solis and Obraczka 2005a) and this ensures that the simulation results are reliable. Figure 6.1 shows the baseline map generated from all nodes' values. Figure 6.2 shows the example of maps generated using the no-aggregation

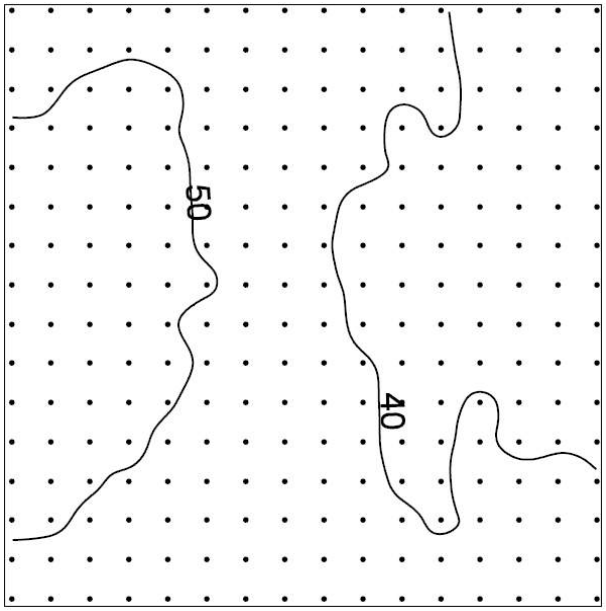

Figure 6.1. Baseline map snapshot

method. Figure 6.3 shows the example of maps generated using the Isolines Aggregation method and figure 6.4 shows the example of maps generated using the Isovector Aggregation method. Table 3 gives the simulation results.

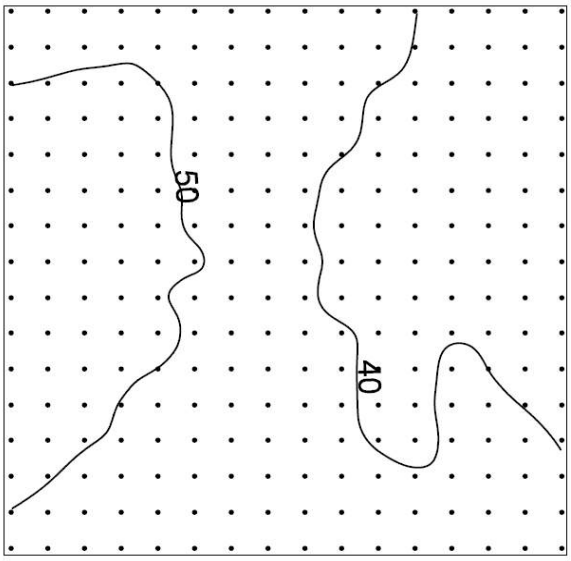

Figure 6.2. Map with no-aggregation

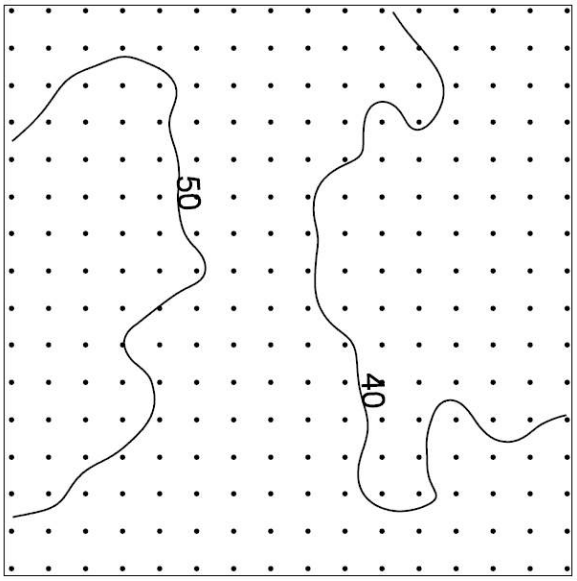

Figure 6.3. Map with Isolines Aggregation

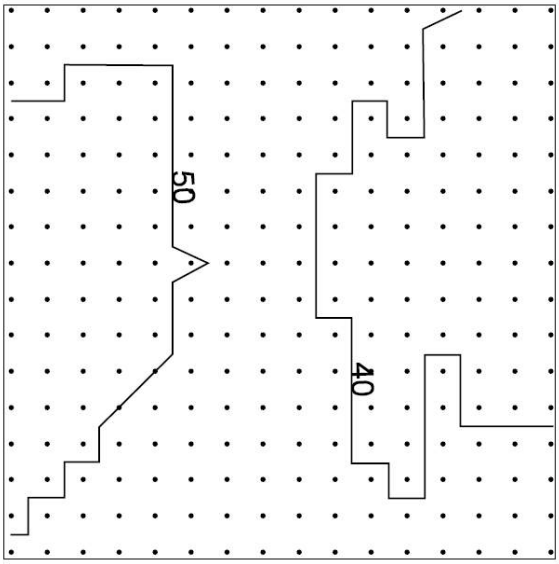

Figure 6.4. Map with Isovector Aggregation

Method	**Accuracy**	**Data sent (bytes)**
No Aggregation	96.64 (sd 0.49%)	12111 (sd 534)
Isolines	94.94 (sd 1.16%)	6177 (sd 345)
Isovector	**96.03 (sd 0.51%)**	**4775 (sd 263)**

Table 4. Contour map snapshot

As we can see from table 4, after drawing contours generated by Isovector Aggregation, we get a contour map similar to the baseline map. Isovector Aggregation also sends much less data than Isolines Aggregation. For the no-aggregation method, we find that nearly 30% of messages, including many messages sent by contour nodes are automatically dropped by the network. Hence, the no-aggregation method does not achieve 100% map accuracy.

6.4 The Impact of Contour Node Ratios

In this scenario, we change the contour node percentage by changing the number of contours. We query all existing contours and measure the total data sent by the no-aggregation method, the Isolines Aggregation method, and the Isovector Aggregation method. Figure 6.5 shows the result. From the figure we know that both aggregation methods send more data as the contour node ratio increases, because more nodes can detect contours when the contour node ratio is high. As more redundant points are removed from the reports, Isovector Aggregation sends much less data in the case of a high contour node ratio when compared to Isolines Aggregation. This implies that Isovector Aggregation is scalable and it is applicable for mapping dense contours in wireless sensor networks. Because the no-aggregation method lets all nodes report, the contour node ratio has no impact on it.

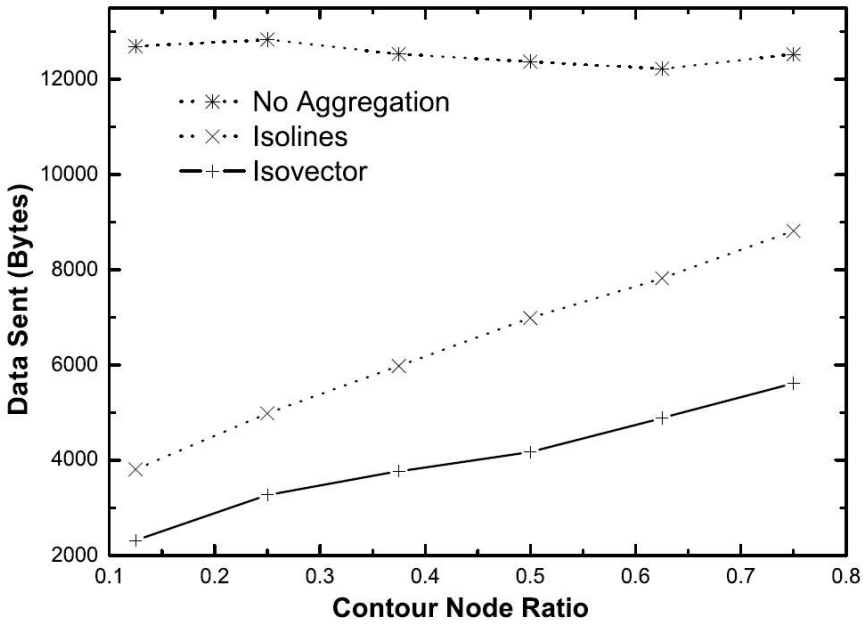

Figure 6.5. Data sent at different contour node ratios

6.5 Monitoring Moving Contours

In this dynamic mapping scenario, we simulate a front moving in from left to right. Temperature increases from the thirties to the fifties in about 50 meters. The front moves to the right in 9s (second). The starting value of all nodes 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.

Method	**Accuracy (9s)**	**Data sent (bytes)**
No Aggregation	98.0 (sd 0.72%)	110994 (sd 03074)
Isolines	97.7 (sd 0.48%)	26855 (sd839)
Isovector	**98.5 (sd 0.2%)**	**17417 (sd 807)**

Table 5. Moving contours

We count the total data sent in this moving scenario and take map snapshots at the 9s point for comparison. Figures 6.6, 6.7, 6.8 and 6.9 show examples of contour map snapshots at 9s and table 5 gives the simulation result. From table 5 we know that benefiting from contour vector merging and simplification, Isovector Aggregation sends less data than Isolines Aggregation. Also, the contour map accuracy achieved by Isovector Aggregation is better than Isolines Aggregation. The no-aggregation technique sends much more data than Isovector Aggregation and Isolines Aggregation. This result implies that Isovector Aggregation is appropriate for continuous contour mapping.

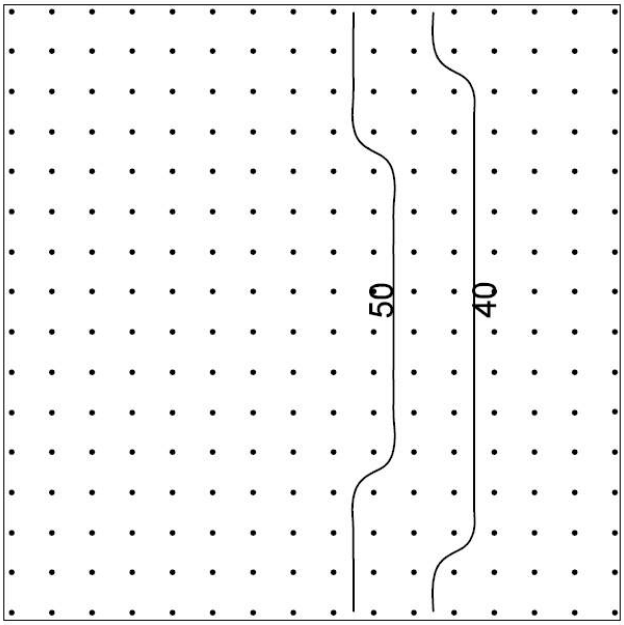

Figure 6.6. Baseline map at 9s for moving contours

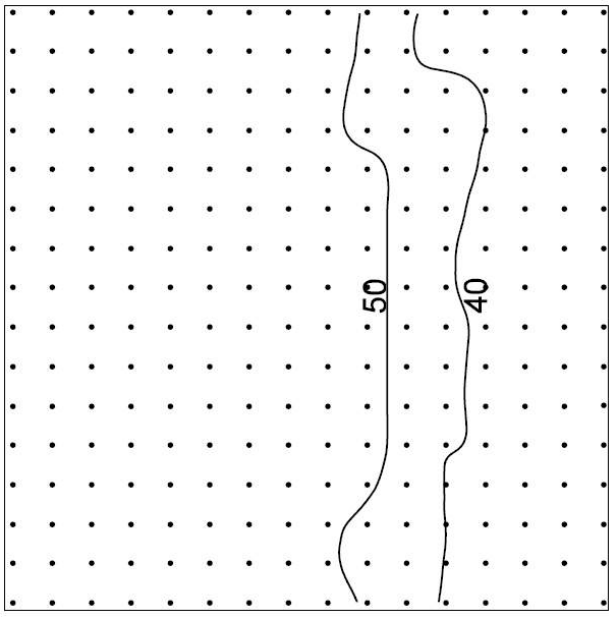

Figure 6.7. Map with no-aggregation for moving contours

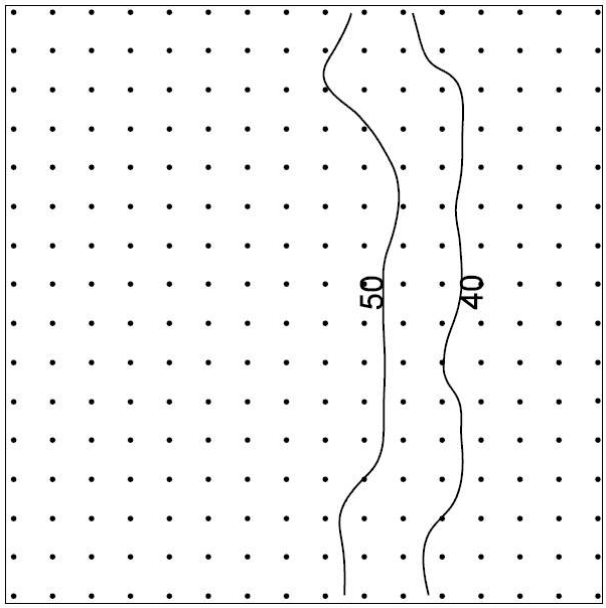

Figure 6.8. Map with Isolines Aggregation for moving contours

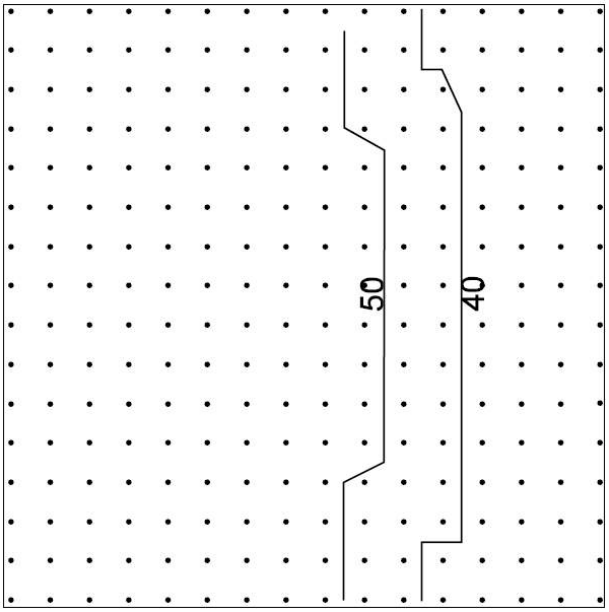

Figure 6.9. Map with Isovector Aggregation for moving contours

6.6 Summary

In this chapter, we do experimental analysis for Isovector Aggregation though simulations using the NS2 network simulator. The simulation results are compared with the results of both the Isovector Aggregation method and the no-aggregation method. The results report that Isovector Aggregation exhibits good map accuracy with significant advantage in energy efficiency.

With the $O(\sqrt{n})$ traffic generation and in-network traffic reduction mechanism, Isovector Aggregation has better scalability than other methods. In the simulation, we manually change the contour node rate by setting different contour numbers and measuring the total data transmitted. Isovector Aggregation sends the least data when the contours are dense in the network. This result accords well with the theoretical analysis.

We evaluate Isovector Aggregation using a dynamic mapping case. Contour map snapshots are captured in the simulation. Similar to the results we get from the static contour detection scenario, the total data transmitted by Isovector Aggregation in this scenario is less than the other two baseline techniques. As well as this, it achieves the best contour map accuracy.

It is important to mention that only the lossless no-aggregation method can achieve total map accuracy under our map accuracy metric. In a real sensor network, especially in a dense network the large amount of traffic caused by the no-aggregation method makes perfect network transmission unrealistic. Because we use mid-points between nodes to represent contours and the Douglas-Peucker polyline simplification algorithm is applied to each contour, Isovector Aggregation will only rarely achieve 100% percent map accuracy, even in a dense network.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The rapid development of wireless sensor networks enables researchers and scientists to monitor the physical world more efficiently. The continuous transmission approach and the threshold based approach are the two basic monitoring approaches. These approaches are simple and easy to implement, but they have an obvious problem. All or most nodes will die quickly because each sensor node is battery powered and the energy is limited. For this reason, the essential problem we have to solve is to save the network energy as much as possible without sacrificing accuracy of the data. In-network data aggregation holds promise as a technique that saves energy and does not significantly lower accuracy.

7.1 Conclusions

This thesis describes a novel in-network data aggregation method, Isovector Aggregation, which aggregates data into contours. We use a simple approach to detect contours. If two adjacent nodes are in different value ranges, there must be at least one contour between them. In order to detect contours, a ring data structure that fully utilizes the neighbor cyclic order information is designed to store neighbors' values. For different communication purposes, we design six different types of messages. Because contours in the dynamic physical field change over time, we want to apply dynamic monitoring and

mapping instead of static contour detection. In order to achieve our purpose, we use cascading timers as the data aggregation time model, which has many advantages compared with others.

Isovector Aggregation consists of four components: local contour generation, reporting node selection, in-network contour merging, and in-network contour simplification. Contours are locally generated between nodes that are in different value ranges. In most cases, we only choose half of the contour nodes to report, and this only results in $O(\sqrt{n})$ traffic, where n is the total number of sensor nodes. In-network contour merging and simplification techniques are employed at each inner node to reduce data transmission. In addition, we use a retransmission mechanism to deal with packet loss in the network and temporal suppression can save more energy if contours do not change frequently.

Both theoretical analysis and experimental analysis results are consistent with the design purposes of the aggregation technique we proposed.

7.2 Discussion

As discussed in Chapter 1, the main goal of this thesis is to design an aggregation technique that can save network energy without compromising the contour map accuracy. We present Isovector Aggregation as the solution.

Theoretical analysis is conducted in this thesis on both traffic generation and traffic reduction. The results show that Isovector Aggregation is the only technique that has $O(\sqrt{n})$ traffic generation and considers in-network traffic reduction at the same time. These two factors make it scalable in large wireless sensor networks.

Experimental analysis is also conducted in this thesis through simulations. There are three simulation scenarios for temperature monitoring. In the first scenario, we use Isovector Aggregation to detect both a regular contour map and an irregular contour map. In the second scenario, we try to find the impact of contour node densities on the data transmission. We also use a dynamic monitoring case for evaluation in the third scenario. All simulation results show that Isovector Aggregation not only achieves high map accuracies compared with baseline maps, it also sends significantly less data than the Isolines Aggregation method and the no-aggregation method. Both theoretical analysis and simulations show that Isovector Aggregation is scalable and could be applied to many different mapping applications in wireless sensor networks.

In short, the main achievements of the thesis are as follows:

1) An aggregation technique, Isovector Aggregation, to generate contour maps using in-network approaches. All contours are generated in the network.

2) Detailed theoretical analysis and experimental analysis. All results show that Isovector Aggregation can save network energy compared to the Isolines Aggregation and no-aggregation techniques. Also, Isovector Aggregation does not significantly decrease the contour maps accuracy. In fact, in all simulations scenarios, Isovector Aggregation achieves better map accuracy than Isolines Aggregation.

In conclusion, the research goal listed in Chapter 1 has been achieved using Isovector Aggregation.

7.3 Work Published and Related

This section presents work related to and derived from the approach developed in this thesis that have been accepted and presented at conferences and workshops.

We have proposed the Continuous Contour Mapping (CCM) method (Zhong and Worboys 2008) that has been presented at the fifth IEEE Consumer Communications and Networking Conference. We have also proposed the Isovector Aggregation method (Zhong and Worboys 2007) that has been presented at the fifth IEEE Upstate NY Workshop on Communications, Sensors and Networking. These two papers are directly related to the approach developed in this thesis.

With the combination of wireless sensor network technology and geographic information science, many researchers may be interested in detecting different topological changes using wireless sensor networks. The contour maps generated at different specific time stamps tell users different snapshots of the entire monitored field. If we want to know what topological changes happened between two timestamps, users can compare the map snapshots at two different times. Furthermore, based on the thesis work, we have developed a Detecting Topological Change (DTC) approach (Farah et al. 2008) to be presented at the 2008 International Conference on Geographic Information Science. This approach can detect all topological changes in the network. Instead of transmitting contours back to the base station, this approach transmits detected changes to the base station directly.

The DTC approach (Farah et al. 2008) uses a similar data structure, the neighborhood ring, which is developed based on this thesis work. Each neighborhood ring is also partitioned into intervals. In order to determine which topological change has occurred, a

node that has changed status will initiate a series of tests based upon the neighborhood components of its neighborhood ring.

There are also differences between the DTC approach and the method reported in this thesis:

1) The thesis focuses on detecting contours in the network. The proposed Isovector Aggregation method can give a direct view of the monitored field in each sampling round; whereas, the DTC approach is more concerned with detecting topological changes.

2) The approach developed in this thesis does not require any global broadcast; whereas, the DTC approach cannot avoid global broadcast when detecting a self-split.

7.4 Future Work

This thesis presents an aggregation method which aggregates sensor data into contours efficiently. There have been many paths that we have not had time to explore. Some of the future research directions include:

1) 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, some contours between them will not be reported (figure 7.1). In figure 7.1, node u will report a contour with value 40 between u and node v and node v will report a contour with value 60. But neither u nor v will report a contour with value 50 between u and v. This is a problem we need to address in the future.

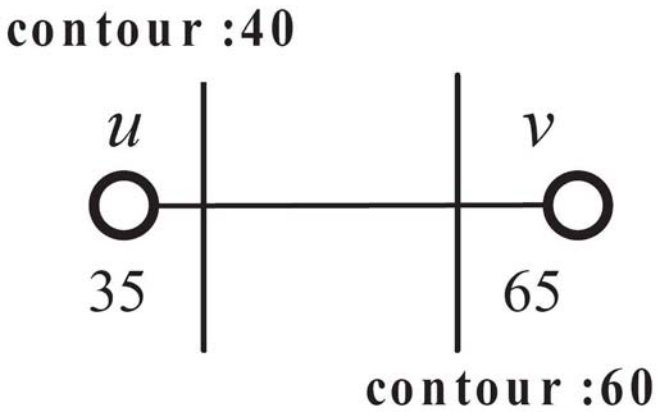

Figure 7.1. A case that Isovector Aggregation cannot deal with

2) We introduce the acknowledgement message to solve packet loss, but we did not fully implement this mechanism in the simulations.

3) As we also emphasized in the thesis, energy saving is one of the most crucial goals for wireless sensor applications. Chen et al. (Chen et al. 2001) show that the energy consumption ratio of idle:receive:transmit is 1: 2: 2.5. Even an idle node consumes a significant amount of energy. Hence, the greatest savings only result from deactivating nodes. Some researchers have already looked into this problem (Xu et al. 2001, Liu et al. 2002, Zhang and Cao 2004). In our work, one possible approach is to make the wireless sensor event-driven. If nothing happens in the network, all sensor nodes could deactivate themselves. If only an event happens in a sub-region of the whole field, only sensor nodes that are around that sub-region activate themselves to monitor the event. The drawback of this event-driven approach is that it may decrease the responsiveness of the network. In other words, events that happen in the network may not be detected promptly. We are required to find a good balance between event detection and resource economization. This is an interesting research direction for the future.

BIBLIOGRAPHY

Abadi, D. J., Madden, S., and Lindner, W. (2005). Reed: Robust, efficient filtering and event detection in sensor networks. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pp. 769–780.

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine*, vol.40, issue 8, pp. 102-114.

ArcView. ESRI. http://www.esri.com/software/arcview/

Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M. and Zhao, J. (2001). Habitat monitoring: Application driver for wireless communications technology. in *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications*, pp. 20-41.

Chen, B., Jamieson, K., Balakrishnan, H. and Morris, R. (2001). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. in *Proceedings of the ACM/IEEE international conference on Mobile Computing and Networking*, pp. 481-494.

Cheng, X., Thaeler, A., Xue, G. and Chen, D. (2004). Tps: A time-based positioning scheme for outdoor wireless sensor networks. in *Proceedings of the 23nd Conference of the IEEE Computer and Communications Societies(INFOCOM)*, pp. 2685-2696.

Chintalapudi, K. and Govindan, R. (2003). Localized edge detection in sensor fields. *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 273-291.

Culler, D., Deborah, E. and Mani, S. (2004). Guest Editors' Introduction: Overview of Sensor Networks. *Computer*, vol. 37, issue 8, pp. 41-49.

Ding, M., Chen, D., Xing, K. and Cheng, X. (2005). Localized fault-tolerant event boundary detection in sensor networks. in *Proceedings of the 24th Conference of the IEEE Computer and Communications Societies(INFOCOM)*, pp. 902-913.

Douglas, D. and Peucker, T. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, vol. 10, pp. 112-122.

Duckham, M., Nittel, S. and Worboys, M. (2005). Monitoring dynamic spatial fields using responsive geosensor network, in *Proceedings of the 13th annual ACM International Workshop on Geographic information systems*, pp. 51-60.

Estrin., D. (2002). Embedded networked sensing for environmental monitoring. Keynote, circuits and systems workshop. http://lecs.cs.ucla.eduestrin/talks/CAS-JPL-Sept02.ppt

Farah, C., Zhong, C., Worboys, M. and Nittel, S. (2008). Detecting topological change using wireless sensor networks, in *Proceedings of Geographic Information Science (GIScience)*, vol. 5266, pp. 55-69.

Gandhi, S., Hershberger, J. and Suri, S. (2007). Approximate Isocontours and Spatial Summaries. in *Proceedings of the Information process in sensor networks(IPSN)*, pp. 400-409.

Greenwald, M. and Khanna, S. (2004). Power-conserving computation of order-statistics over sensor networks. in *Symposium on Principles of Database Systems (PODS)*, pp. 275-285.

Hellerstein, J. M., Hong, W., Madden, S. and Franklin, M.J. (2003) (a). The design of an acquisitional query processor for sensor networks. in *Proceedings Of the 2003 ACM International. Conference on Management of Data (SIGMOD)*, 2003, pp. 491-502.

Hellerstein, J. M., Hong, W., Madden, S. and Stanek, K. (2003) (b). Beyond average: Toward sophisticated sensing with queries. in *Proceedings of the Information process in sensor networks (IPSN)*, pp. 63-79.

Intanagonwiwat, C., Govindan, R. and Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. in *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, pp. 56-67.

Liu, J., Cheung, P., Zhao, F. and Guibas, L. J. (2002). A Dual-Space Approach to Tracking and Sensor Management in Wireless Sensor Networks.in *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSN*A), pp. 131-139.

Liu, Y., and Li, M. (2007). Iso-map: Energy-efficient contour mapping in wireless sensor networks. in *Proceedings of the 27th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 36-48.

Madden, S., Franklin, M. J., Hellerstein, J. M. and Hong W. (2002). Tag: A tiny aggregation service for ad-hoc sensor networks. in *Proceedings of the 2002 USENIX Symposium on Operating Sysems Design and Implementation (OSDI)*, pp. 131-146.

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D. and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pp. 88-97.

Meng, X., Li, L., Nandagopal, T. and Lu, S. (2004). Event contour: an efficient and robust mechanism for tasks in sensor networks. Technical Report, UCLA, 2004.

Network Simulator 2. http://www.isi.edu/nsnam/ns/

Pottie, G. J. and Kaiser W. J. (2000). Wireless integrated network sensors. *Communications of the ACM*, vol. 43, no. 5, pp. 51-58.

Shang, Y., Ruml, W., Zhang, Y. and Fromherz M. P. J. (2003). Localization from mere connectivity. in *MobiHoc*, pp. 201-212.

Shnayder, V., Hempstead, M., Chen, B. R., Allen, G. W. and Welsh, M. (2004). Simulating the power consumption of large-scale sensor network applications. in *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pp. 188-200.

Shrivastava, N., Buragohain, C., Agrawal, D. and Suri, S. (2004). Medians and beyond: New aggregation techniques for sensor networks. in *Proceedings of the 2nd international conference on Embedded networked sensor systems (SENSYS)*, pp. 239-249.

Silberstein, A., Braynard, R. and Yang, J. (2006). Constraint chaining: on energy-efficient continuous monitoring in sensor networks. in *Proceedings Of the 2006 ACM International. Conference on Management of Data (SIGMOD)*, pp. 157-168.

Sinha, A. and Chandrakasan A. (2001). Dynamic Power Management in Wireless Sensor Networks, *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 62-75.

Solis I. (2005). Efficient Protocols for Power-Constrained Heterogeneous Wireless Ad-hoc Networks, Ph.D. dissertation, University of California Santa Cruz.

Solis, I. and Obraczka, K. (2005) (a). Efficient continuous mapping in sensor networks using isolines. *in Proceedings of the 2005 MobiQuitous*, pp. 325-332.

Solis, I. and Obraczka, K. (2005) (b). Isolines: Energy Efficient mapping in sensor networks" in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, pp. 379-385.

Solis, I. and Obraczka, K. (2004) (a). I. Solis and K. Obraczka. Flip: A flexible interconnection protocol for heterogeneous internetworking. *ACM/Kluwer Mobile Networking and Applications (MONET) Special on Integration of Heterogeneous Wireless Technologies*, vol. 9, no. 4, pp. 347-361.

Solis, I. and Obraczka, K. (2004) (b). The impact of timing in data aggregation for sensor networks. in *Proceedings of the IEEE International Conference on Communication*, vol. 6, pp. 3640-3645.

Tubaishat M. and Madria S. (2003). Sensor Networks : An Overview. *IEEE Potentials*, vol. 22, pp. 20-23.

UNISYS. (2002). http://weather.unisys.com/surface/sfc_con_temp.html

White, E. (1985). Assessment of line-generalization algorithms using characteristic points. *The American Cartographer*, vol. 12, pp. 17-27.

Xu, N. (2004). A survey of Sensor network Application. Technical report, Computer Science Department, University of Southern California.

Xu, Y., Heidemann, J. and Estrin, D. (2001). Geography-informed energy conservation for ad-hoc routing. in *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 70-84.

Xue, W., Luo, Q., Chen, L. and Liu, Y. Liu. (2006). Contour map matching for event detection in sensor networks. in *Proceedings Of the ACM International Conference on Management of Data (SIGMOD)*, pp. 145-156.

Yao, Y. and Gehrke, J. (2003). Query processing in sensor networks. in *Online proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, pp. 201-212.

Yao, Y. and Gehrke, J. (2002). The Cougar approach to in-network query processing in sensor networks. in *SIGMOD Record*, vol. 31, pp. 9-18.

Zhao, F., Guibas J. L. (2004). Wireless Sensor Networks: An information processing approach. Morgan Kaufmann Press.

Zhao, J., Govindan, R. and Estrin, D. (2002). Residual energy scans for monitoring wireless sensor networks. in *IEEE Wireless Communications and Networking Conference*, 2002, pp. 145-156.

Zhang, W. and Cao, G. (2004). DCTC: Dynamic Convoy Tree-Based Collaboration for Mobile Target Tracking. *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689-1701.

Zhong, C. and Worboys, M. (2007). Generating contours in a sensor network using isovector aggregation. in *Proc. of the 5th IEEE Upstate NY Workshop on Communications, Sensors and Networking*, pp.131-135.

Zhong, C. and Worboys, M. (2008). Continuous contour mapping in sensor networks. in *Proc. of the 5th IEEE Consumer Communications and Networking Conference (CCNC)*, pp.152-156.

BIOGRAPHY OF THE AUTHOR

Cheng Zhong was born in Sangzhi, Hunan, P.R.China on June 24, 1980. He attended school in Sanzhi, and graduated from Sangzhi Number One High School.

He obtained a B.S. in Economics from Beijing Institute of Technology, P.R.China in 2001 and a M.S. in Computer Software and Theory from Peking University, P.R.China in 2004. During this time, he was a member of the Operating System Laboratory at Peking University. His research interests included information retrieval, system component repertory. He was involved in a national 863 project and invented a method patented “A Correlation Based Query Method for Operating System Component Repertory” for Peking University. His master thesis at Peking University was titled “The Design and Implementation of the Sub-retrieval System for the Embedded Operating System Component Repertory”. He also worked part-time at Microsoft Research Asia when he was studying for his master degree. From 2004 to 2006, he worked for the Oracle Corporation, China Development Center as a software developer. He finished Oracle Japan Location Based Services 2.0 project and the Oracle MapViewer 11g Beta version with other coworkers during that period.

After moving to Maine in August 2006, he was enrolled for graduate study at the University of Maine and served as a graduate research assistant in the Department of Spatial Information Science and Engineering. He is a candidate for the Master of Science Degree in Spatial Information Science and Engineering from the University of Maine in December 2008.