國立交通大學

資訊科學與工程研究所

碩 士 論 文

在無線感測網路中建立擴散事件的等高線圖

In-Network Contour Maps Construction for Diffusion Events in Wireless Sensor Networks

研 究 生：賴怡廷

指導教授：曾煋棋 教授

中 華 民 國 九 十 七 年 六 月

在無線感測網路中建立擴散事件的等高線圖
In-Network Contour Maps Construction for Diffusion Events in Wireless Sensor Networks

研 究 生：賴怡廷　　　　Student：Yi-Ting Lai

指導教授：曾煜棋　　　　Advisor：Yu-Chee Tseng

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

在無線感測網路中建立擴散事件的等高線圖

學生：賴怡廷　　　　　　　　　　　　指導教授：曾煜棋

國立交通大學資訊科學與工程研究所碩士班

摘　　　要

在大範圍環境監測的無線感測網路應用中，如何精確且有效率的統整觀測的資料是一大挑戰。在此篇論文中，我們針對在擴散事件中建立感測資料之等高線圖的議題來做探討。有鑒於在無線感測網路當中的精確度與通訊負擔是互相衝突的考量因素，我們提出四種演算法來因應不同的網路分布和應用需求。第一個方法是集中式的演算法，而節點是分布成格網狀。第二個方法是分散式演算法，而節點為亦為格網狀，其希望能夠透過鄰近節點的資訊來減少通訊的負擔。第三個方法是將第一個集中式演算法在格網狀分布之方法推廣至網路隨機分布。最後一個方法則是延伸至第二個分散式計算的方法由格網狀推廣至網路隨機方布，其透過形成區域性的Delaunay 三角形作為分散式計算與通訊的單位，實踐提升能源效率的目標。最後，我們透過效能的分析與模擬的結果來驗證此篇論文的有效性。

關鍵字：電腦圖學，等高線圖，擴散事件，遍布式計算，無線感測網路。

Abstract

One challenge for environmental monitoring in a large-scale wireless sensor network is how to accurately and efficiently summarize the target observation area. This paper considers the construction of contour maps of a sensing field with diffusion events. Observing that accuracy and communication overhead are conflicting concerns, we propose four algorithms for different network deployments and requirements. The first scheme is a centralized one when sensor nodes are deployed in a grid manner. The second scheme is a localized one under the same grid deployment; neighbor information is exploited to reduce communication overheads. The third scheme is extended from the first one and is for random deployment. The last scheme is extended from the second one and is for random deployment. Through localized Delaunay triangulations, it achieves energy efficiency with distributed computation and communication. Performance analysis and simulation results are presented to verify the effectiveness of these results.

Keywords: computer geometry, contour map, diffusion event, wireless communication, wireless sensor networks.

致　　謝

本論文能夠順利地完成，仰賴於許多人的指導與協助。在此，獻上我最誠摯地感謝。

首先，由衷地感謝曾煜棋教授兩年來的指導與鼓勵，並提供良好的研究環境，讓我得以順利的完成此篇論文並取得碩士學位。對於論文的方法，總是能夠提出精闢的見解，讓我學會以更寬廣的視野去看事情，其做學問認真的態度尤其令人感佩，值得作為我終身處事的榜樣。

此外，感謝指導我的潘孟鉉學長和葉倫武學長，在論文的研究上提供了許多寶貴的建議與指導。另外，感謝 HSCC 實驗室全體的成員，在我碩士班兩年中的鼓勵與幫助。

感謝我的好朋友們及關心我的人，謝謝你們聆聽我的難處、幫我解決問題，也帶來給我歡樂，因為有你們的友情支柱，我不怕困難！

最後，感謝我的家人，在我求學生涯中，不斷地對我付出關懷以及永無止盡的愛，總是支持我做的決定，這份體諒很包容，讓我可以恣意地享受自己所選擇的生活，無虞地完成我的學業。

賴怡廷　謹於

國立交通大學資訊工程與科學所 碩士班
中華民國九十七年六月

Contents

List of Figures

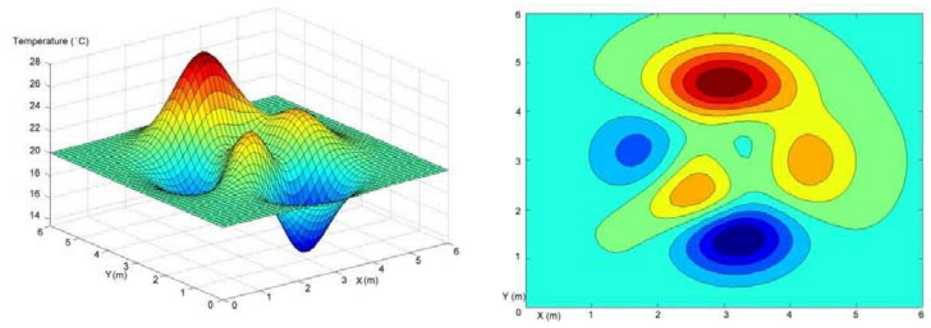
Chapter 1

Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks* (WSNs) possible. A WSN normally consists of many inexpensive wireless nodes, each capable of collecting, processing, and storing environmental information, and communicating with neighboring nodes. Research efforts have been dedicated to several topics, such as smart living space [12][16], localization [6][11], and environmental monitoring [9][17].

This work considers the construction of contour maps of a sensing field covered by a WSN with diffusion events. A contour map is composed of several *isolines*, each as a curve on the surface connecting points of the same value. It helps understand the spatial characteristic and movement of the diffusion event. Fig. 1.1(a) shows the sensing values mapped to a target area and Fig. 1.1(b) shows its contour map. Examples of diffusion events include intensity of light, temperature, and humidity.

Several works [5][8][13][14] have discussed contour maps construction in WSNs. Reference [5] extends the query engine of TinyDB [1] to support visualizing the sensing field. Sensor nodes are deployed into a grid manner and each node builds a small representation of its local area to construct a contour map. However, the shape of the map is axis-dependence. Iso-Map [8] builds contour maps based on selecting some representation nodes (called isoline nodes) around each isoline



(a) (b)

Figure 1.1: (a) Sensing values in a target area. (b) A contour map of (a).

to reduce traffics. It applies a linear regression model for efficient gradient estimation to reduce the reporting packets. However, the final contour maps may not be accurate since each partial contour isoline must pass isoline nodes and Voronoi edges. The contour maps constructed from the works [5][8] are not accuracy because their contour must pass on the boundary of grid or area and sensor nodes. A quadtree-based approach is proposed in [13] to cluster nodes for energy efficiency. However, the map construction itself is not considered. An energy-efficient data collection algorithm is proposed in [14] by allowing only nodes nearby isolines to report. However, it does not mention how to detect the boundaries of isolines and how to use the sensing values to reconstruct contour maps. Furthermore, both [13][14] assume that the sink is responsible for constructing isolines.

Energy-efficient approaches are proposed in [18][10]. The work [10] considers the spatial suppression at individual nodes to reduce traffics and proposes an interpolation and smoothing algorithm at the sink. Nevertheless, the contour map accuracy highly depends on the data suppression rate. Instead of using a simple threshold-based detection, an event detection mechanism based on matching contour maps to sensing values is proposed in [18]. An in-network linear regression to represent the distributions of sensing values in each region is developed. Since it only considers event pattern matching, produced a rough sketch of data distributions. To summarize, [18][10] focus more on reducing communication overhead, and thus may sacrifice map accuracy. Also, these works may not be scalable to larger WSNs. Reference [3][4] focus on approximating contour maps by a family of k-vertex polygon. It is assumed that contour boundaries are given and the goal is data reduction with a provable quality approximation.

As reviewed above, most existing works only address one of the two concerns, communication efficiency and accuracy, and may sacrifice the others. In this work, we attempt to address both the fidelity of contour maps and energy efficiency. We will exploit in-network computation as much as possible to reduce communication workload. We propose four schemes. The first scheme is a centralized one when sensor nodes are deployed in a grid manner. The second scheme is a localized one under the same grid deployment; neighbor information is exploited to reduce communication overheads. The third scheme is extended from the first one and is for random deployment. The last scheme is extended from the second one and is for random deployment. Through localized Delaunay triangulations, it achieves energy efficiency with distributed computation and communication. Performance analysis and simulation results are presented to verify the effectiveness of these results.

The rest of this paper is organized as follows. Chapter 2 describes our network model. Chapter 3 presents our centralized and distributed algorithms for grid-like WSNs. Chapter 4 extends these algorithms to randomly deployed WSNs. Simulation results are in Chapter 5. Chapter 6 concludes this work.

Chapter 2

Network Model

A contour map consists of multiple *isolines*. An isoline is a curve on a surface that connects points of equal values. The difference between each isoline is called *isoline interval*. The set of values that isolines are formed is called *isoline levels*. Fig. 1.1(b) shows an example with isoline interval $= 2$ and isoline levels $= \{14, 16, 18, 20, 22, 24\}$.

A WSN is modeled as a graph $G = (V, E)$, where V contains all sensor nodes and E contains all communication links between nodes in V. Each node maintains its neighbor information by HELLO messages. We assume that each node knows its own location and the network is dense enough to ensure connectivity. We consider two kinds of network deployment, grid and random. In grid deployment, sensor nodes are regularly separated by a distance of L and each node has up to 8 neighbors. In random deployment, there is no special structure.

Nodes sense the field periodically. Isoline levels are given by query packets in advance. Depending on different models, nodes may perform local computations and send their results to the sink. When a packet travels to the sink, some data aggregation may be done, but this is beyond the scope of this work. Finally, the sink will properly construct a contour map.

Chapter 3

Contour Maps Construction for Grid Networks

Given a grid-like WSN, we will present a centralized and distributed schemes. The first scheme can render a 3D surface, while the second one can render polygon-like isolines on a 2D plane.

3.1 Centralized Grid Surface Approximation Scheme

This scheme exploits the grid structure of nodes to compute isolines. We will not address the aggregation or convergecast mechanism while delivering packets, which is beyond the scope of this work. The sink will apply the *bicubic spline interpolation* [15] to obtain isolines from the collected data.

To understand the scheme, we first explain how the cubic spline interpolation works to approximate a 1-dimensional function. We are given a sequence of reals $x_i, i = 1, \ldots, n$, and their corresponding values y_i. A spline consisting of series of piece-wise polynomial curves will be constructed. It is guaranteed that the spline will pass y_i at each $x_i, i = 1, \ldots, n$. The curve in each interval $[x_i, x_{i+1}]$, $i = 1, \ldots, n-1$, is approximated by a polynomial:

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \tag{3.1}$$

where a_i, b_i, c_i, d_i are coefficients to be determined. To pass y_i and y_{i+1} and to

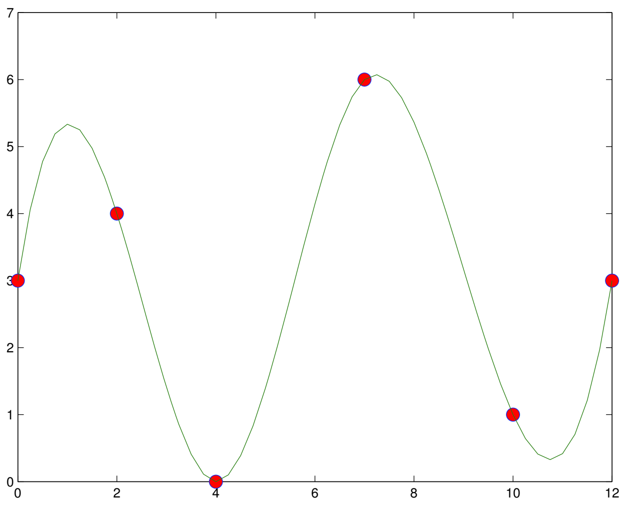


Figure 3.1: An example of the cubic spline interpolation.

satisfy the continuity property, the following equations hold:

$$\begin{cases} S_i(x_i) = y_i \\ S_i(x_{i+1}) = y_{i+1} \\ S_i'(x_i) = S_{i-1}'(x_i) \\ S_i''(x_i) = S_{i-1}''(x_i) \end{cases} \tag{3.2}$$

Since there are four equations and four unknowns, a_i, b_i, c_i, and d_i can be found. Fig. 3.1 shows how this method works given six points $\{(0,3)$, $(2,4)$, $(4,0)$, $(7,6)$, $(10,1)$, $(12,3)\}$.

The bicubic spline interpolation is to interpolate a 2-dimensional function. We are given points $x_i, i = 1,\ldots,n_x$, and $y_i, i = 1,\ldots,n_y$, and their corresponding values $z_{i,j}$. The goal is to construct $(n_x - 1) \times (n_y - 1)$ piece-wise spline curves

$$S_{i,j}(x,y) = \sum_{u=0}^{3}\sum_{v=0}^{3} a_{uv}x^u y^v \tag{3.3}$$

for areas $[x_i, x_{i+1}] \times [y_j, y_{j+1}], i = 1,\ldots,n_x - 1$ and $j = 1,\ldots,n_y - 1$. There are 16 coefficients a_{uv} to be determined for each curve. To guarantee continuity of these piece-wise curves, each curve Eq. (3.4) must satisfy the following 16 equations:

$$\begin{cases} S_{i,j}(x_{i+\delta x}, y_{j+\delta y}) = z_{i+\delta x, j+\delta y} \\ dS_{i,j}(x_a, y_{j+\delta y})/dx = dS_{b,j}(x_a, y_{j+\delta y})/dx \\ dS_{i,j}(x_{i+\delta x}, y_c)/dy = dS_{i,d}(x_{i+\delta x}, y_c)/dy \\ dS_{i,j}(x_a, y_c)/dxy = dS_{b,d}(x_a, y_c)/dxy, \end{cases} \tag{3.4}$$

where $\delta x = 0$ or 1 and $\delta y = 0$ or 1 for $a = i$ or $i + 1$, $b = i - 1$ or $i + 1$, $c = j$ or $j + 1$, and $d = j - 1$ or $j + 1$.

$$\begin{cases} \text{if} \quad a = i, & \text{then} \quad b = i - 1 \\ \text{if} \quad a = i + 1, & \text{then} \quad b = i + 1 \end{cases} \tag{3.5}$$

$$\begin{cases} \text{if} \quad c = j, & \text{then} \quad d = j - 1 \\ \text{if} \quad c = j + 1, & \text{then} \quad d = j + 1 \end{cases} \tag{3.6}$$

While this method can give smooth and accurate surfaces, it relies on all nodes' sensory data. Furthermore, sensor nodes must be deployed in a grid manner, which is practically costly.

3.2 Distributed Grid Plane Approximation Scheme

This scheme aims at reducing the reporting traffics while maintaining acceptable accuracy. First, we will elect some leaders by inhibiting some nodes reporting. Then, leaders will try to construct isoline segments. Finally, isoline segments are sent to the sink to combine them into an isoline. This scheme consists of four steps. Without loss generality, let us consider one isoline level v_i.

Step 1: Candidate Nodes, I-nodes, and O-nodes Selection. We assume that the sensor reading of two neighboring grid nodes would not exceed a threshold ε. Therefore, a node is called a *candidate node* if its reading is in the interval $[v_i - \varepsilon, v_i + \varepsilon]$. To save energy, a non-candidate node does not need to check its neighbors' values, but a candidate node does. If a candidate node with a value less than v_i sees a neighbor with a value larger than v_i, it is called an *i-node* (*inner-boundary node*). If a candidate node with a value larger than v_i sees a neighbor with a value less than v_i, it is called an *o-node* (*outer-boundary node*). An example is shown in Fig. 3.2.

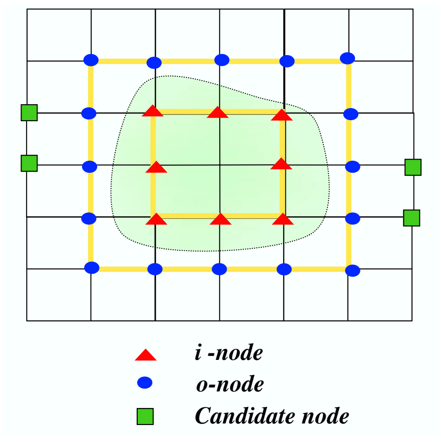


Figure 3.2: The candidate nodes, i-nodes, and o-nodes

Step 2: Grid Formation. For each grid unit, if its four member nodes contain both i-nodes and o-nodes, we will elect one nodes as the leader. For load balance, we can use a random bidding process to elect leaders. Each leader should collect the values of the other three member nodes and compute isoline segments within its grid. Without loss of generality, letting the upper-right node of the grid be an i-node and traversing the grid members in the clockwise direction, there are four combinations: I-O-O-O, I-O-O-I, I-O-I-O, and I-I-I-O, where I means an i-node and O means an o-node. These are illustrated in Fig. 3.3.

Step 3: Partial Isolines Estimation. In each grid, the leader will divide its grid into two triangles. Depending on cases the triangulations are as shown in Fig. 3.3. Traversing a triangle in a clockwise direction, there are two cases, I-O-O and I-I-O. Each triangle must have two *cross edges*, which is I-O or O-I edge, and one *non-cross edge*, which is I-I or O-O edge. We can do a linear interpolation to find a point in each cross edge as an estimated point passed by an isoline. Connecting the two points on the cross edges by a line segment, we would consider this line segment as an isoline segment. Fig. 3.3 shows different combinations of isoline segments for the above

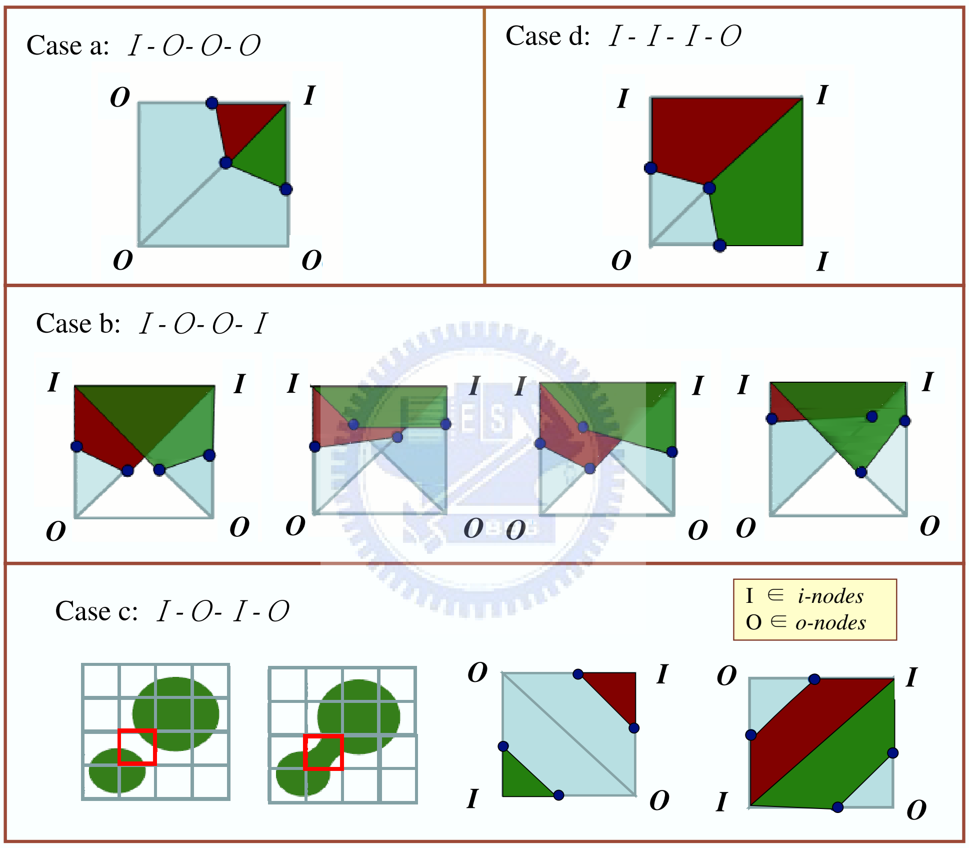


Figure 3.3: Four cases of grid formation and isoline segments estimation.

four cases. Cases (a) and (d) have quite clear boundaries. Case (b), we can consider the union of the gray areas as the area with values less than v_i. Four case (c), there are actually two ways to interpret the areas with values less than v_i (it should be clear from the illustration.).

Step 4: Isolines Combination. Then each leader can report its isoline segments to the sink. Each isoline can be represented by its two endpoints. Since some segments may share one endpoint, each report will contain three or four endpoints. Then the sink can combine them into an isoline. Note that the ambiguous case (c) should be solved at the sink.

Chapter 4

Contour Maps Construction for Random Networks

Next, we relax our grid constraint by allowing sensor nodes to be randomly deployed. A centralized and a distributed schemes based on triangulation of the sensing field will be derived. Since we focus on diffusion events, the Delaunay triangulation which enforces that the circle circumscribing the endpoints of each triangle does contain any other point is adopted to exploit the spatial correlation of sensing data (i.e. each triangle's endpoints are the nearest to each other).

4.1 Centralized Triangular Surface Approximation Scheme

We will extend our centralized scheme for grid networks to one for random networks. The main idea is to translate sensing data at random points to ones at grid points. Then we can apply the same bicubic spline interpolation to construct a contour map. The scheme has three steps:

Step 1: Delaunay triangulation on the locations of sensor nodes.

Step 2: Then, on the sensing field, form a virtual $n_x \times n_y$ grid network, where the grid interval is a user-specific parameter. For each grid location, we will approximate its value as follows. Let $\triangle ABC$ be the triangle that contains location L. The sensing value at location L will be approximated by:

$$z(L) = \frac{z(A) \cdot \overline{LA}}{\overline{LA} + \overline{LB} + \overline{LC}} + \frac{z(B) \cdot \overline{LB}}{\overline{LA} + \overline{LB} + \overline{LC}} + \frac{z(C) \cdot \overline{LC}}{\overline{LA} + \overline{LB} + \overline{LC}}, \quad (4.1)$$

where $z(L)$ is the sensor reading at location L.

Step 3: Finally, apply the bicubic spline interpolation are in Chapter 3.1 on these $n_x \times n_y$ data points to obtain a contour map.

4.2 Distributed Triangular Plane Approximation Scheme

For grid networks, rectangle structures can be easily found in a distributed way. For random networks, distributed triangulation schemes have been proposed in [2]. However, such works aim at forming triangles of the whole network. Here, we are only interested in forming triangles around the potential isolines. We will modify the localized Delaunay triangulation in [7], which can partition a network into planar graph in a distributed manner, to form several triangles around the potential isolines.

The localized communication based on the Delaunay triangulation might not possible, because it can contain links longer than communication range of wireless sensor nodes. We make some assumptions to avoid incompleteness network partition according to a potential isoline: given a graph G, a spanning subgraph H of G is a t-spanner if the length of the shortest path connecting any two nodes in H is no more than t times of the shortest path connecting two nodes in G. Localized Delaunay triangulation [7] that constructs a planar 2.5-spanner of unit-disk graph as a network topology and partitions the network into the Delaunay triangulation. Based on the localized Delaunay triangulation, we propose a distributed triangular plane approximation scheme. This scheme consists of five steps.

Step 1: Candidate Node Selection. We design a localized approximation using neighborhood information to do isolines estimation. For energy saving, instead of collecting overall sensory reading, we first select candidate nodes as describe in Chapter 3.2. Then, we will employ localized Delaunay triangulation with these candidate nodes.

Step 2: Localized Delaunay Triangulation. In [7], the authors propose the k-localized Delaunay triangle $\triangle uvw$ satisfies k-localized property if the interior of disk (u,v,w) does not contain any k-neighbor of u, v, or w and all edges of $\triangle uvw$ are in the communication range. The algorithm of localized Delaunay triangulation is composed of two steps: firstly construct 1-localized Delaunay triangulation, and then make it a planar graph efficiently. We select candidate nodes to be applied to following localized Delaunay triangulation:

Step 2a: The 1-Localized Delaunay Triangulation Formation. Each candidate node broadcasts a $advertisement$ message containing its identity and location to its one-hop neighbors, and then listens to the messages from other nodes. When the candidate node gather all its one-hop neighbor $advertisement$ messages, it compute the Delaunay triangulation through local information. Each candidate node finds all triangles form Delaunay triangulation such that all three edges within communication edge by checking the angle between each edge, and then broadcasts a $proposal$ message that contains eligible triangles information. After sending the proposal message, double-check whether another proposal message sent by its neighbor matching the same triangle or not. Broadcast an $accept$ with matched triangle information, otherwise sent a $reject$ message. Finally, all accepted triangles form the 1-localized Delaunay triangulation.

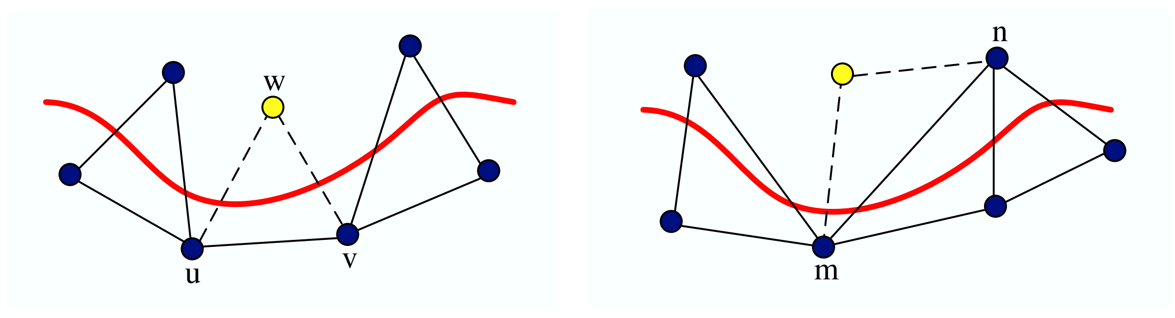
Step 2b: The Planarize 1-Localized Delaunay Triangulation. 1-localized Delaunay triangle does not prevent intersection of two triangles and it may happen when the radius of circumcircle of triangle large than the communication range of sensor nodes. We remove edges of triangles to make 1-localized Delaunay triangle to be an planar in the planarize phase. Initially, each node of 1-localized Delaunay triangles formed by previous phase detect that if any nodes inside of the circumcircle of each triangle. When such a triangle is found, we remove the triangle if its circumcircle contains a nodes from other triangle. Then update the remaining non-removed triangles until no nodes inside any circumcircle of triangles.

Step 3: Boundary Node Selection and Well-Subdividing into Triangular Blocks. The triangular blocks between isolines are well-subdivision if all cross edges belong to two different triangles. After localized Delaunay triangulation is done with candidate nodes, we cannot ensure our candidate nodes are well-subdivision into triangular blocks. That is because we only use a part of nodes of our random networks, that is candidate nodes selected with it own sensory values without considering neighbor nodes' correlation. So that after each candidate node exchanges sensory values with its one-hop neighbors, those candidate and non-boundary nodes will be deleted and those non-candidate and boundary nodes will be added as the following procedure.

Step 3a: Delete Candidate and Non-boundary Nodes. Once all three candidate nodes of the same triangular block belong to all i-nodes or all o-nodes, we can sure that the potential isoline do not pass through the triangular block. Those three nodes can not provide any information of approximate the potential isoline in our localized computing scheme, so that we delete some of the three nodes which do not belong to any other triangular blocks. The deleted node is candidate and non-boundary node, that is the sensory value of the deleted candidate node is with $[v_i - \varepsilon, v_i + \varepsilon]$ but has no neighbors with cross edges across the potential isoline.

Step 3b: Add Non-candidate and Boundary Nodes. We add non-candidate and boundary nodes into our boundary nodes set if there exists any two candidate nodes u and v, both nodes are either i-nodes or o-nodes with a non-cross edge and the edge does not belong to any triangles. These two nodes exist a communication edge between each other but they belong to different triangles. Then, these two nodes will broadcast REQUEST message to their one-hop neighbor. With the 2.5-spanner assumption, we can find that these two nodes must contains the same neighbor nodes in the another side of the potential isoline. The nearest neighbor w will replay JOINT message to these two nodes and construct two cross edges. By adding the nearest common neighbor of these two nodes, the three nodes u, v and w can form a triangles. After this step, there is no isolated non-cross edges without belonging to any triangles. Fig. 4.1(a) shows an example of this step.

When there is no isolated non-cross edges, we will check that whether each cross edge belongs to two different triangles or not. A node m has two cross edges but each cross edge only belongs to one triangles. Then node m will select the smaller ID's node n within one cross edge to trigger adding nodes step. The node m and n will send the REQUEST message to find a common neighbor. The common neighbor will reply the JOIN message to these two nodes and construct one cross edge and one non-cross edge. If multiple nodes reply the JOIN message to these two nodes, the node already belongs to the triangular blocks will be chosen first. If no such node can be found, choose the farthest node from these REQUEST senders. Redo this step until all cross edges belong to two different triangles. Fig. 4.1(b) shows an example of this step.



(a) (b)

Figure 4.1: The step of add non-candidate and boundary nodes.

Step 4: Partial Isoline Estimation. After constructing the well-subdividing triangular blocks, each triangle select a leader node to estimate the partial isoline. There are two kinds of triangles, which are $\triangle IOO$ and $\triangle OII$, in well-subdividing triangular blocks. The $\triangle IOO$ and $\triangle OII$ choose node I and O as the leader node, respectively. Then, each leader node applies the linear model of partial isolines estimation which is already mentioned in Chapter 3.2.

Step 5: Reporting and Combine partial Isolines. The leader of each triangle transmits computing results which contains the parameters of each partial isolines to the sink. Then sink combines adjacency partial isolines to form a complete contour map.

In this scheme, we need only those sensory reading related to the potential isolines construction for energy saving instead of collecting overall sensory data. Base on the localized Delaunay triangulation, we construct the well-subdividing, which means that the network is partition into piece-wise non-overlapping triangles, triangular blocks. Each leader node compute the partial isoline independently. Hence, the localized scheme is scalable, even in large-scale sensor networks, leaders of sensor nodes can approximate potential isolines quickly.

Chapter 5

Performance Analysis and Simulation Results

Several metrics have been proposed to measure the quality of a polygonal approximation. A widely used choice is the Hausdorff error metric. Consider a polygonal curve A and an approximation curve B. Given m points $(a_1, a_2, ..., a_m)$ of A, n points $(b_1, b_2, ..., b_n)$ of B and the notation $d(p, Q)$ is the minimum Euclidean distance from a point p to a polyline Q. The Hausdorff metric is defines as below.

$$H(A, B) = \max(\max_{0 \le i \le m} d(a_i, B), \max_{0 \le j \le n} d(A, b_j)) \tag{5.1}$$

In these simulations, we consider the accuracy of contour map, network traffic and computation workload for grid and random deployment. For grid deployment, we compare our centralized grid surface approximation scheme (denote by CGS) and distributed grid plane approximation scheme (denote by DGS) against the TinyDB, INLR. For random deployment, we compare our centralized triangular surface approximation scheme (denote by CTS) and distributed triangular plane approximation scheme (denote by DTS) against Iso-Map. The accuracy of contour map measures the effectiveness of our algorithm, and the network traffic and computation workload reflects the power efficiency. Both performance metrics are important for environmental monitoring in wireless sensor networks.

5.1 Contour Map Accuracy

We utilize a function in Eq. (5.2) to simulate the relation of sensory location and sensory value (refer to Fig. 1.1 for the measurement and its contour map) as our test data.

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(x/5 - x^3 - y^5)e^{-x^2-y^2} - (1/3)e^{-(x+1)^2-y^2} \tag{5.2}$$

In the past, the accuracy of TinyDB in grid deployment were thought to be the best accuracy compared with all other existing scheme, since it collects sensory values in whole observation area without losing any detail information. INLR is a regression and aggregation-based approach, so that the criteria of regression and aggregation highly effect on accuracy of a contour map. In our simulations, we use the best case of INLR in accuracy, that is supposed that no regression and no aggregation of INLR to construct a contour map to compare with ours.

The key factor of contour map construction accuracy is the density of sensor nodes deployment. In this simulation, we fix the deployment area and vary the number of sensor nodes. When the number of sensor nodes increase, the nodes density also increases. Fig. 5.1 shows the simulation results for different network density in grid deployment. The CGS can achieve much more accuracy than TinyDB and the best case of INLR for different number of sensor nodes in the same area. The CGS performs well in accurate approximation contour isolines with mush lower error compared with TinyDB and INLR in any node density circumstances. The DGS is more accuracy in dense sensor networks but TinyDB and INLR are advantageous in sparse sensor networks. Fig. 5.2 shows the simulation results for different network density in random deployment. The contour map accuracy constructed by Iso-Map has much more Hausdorff error because CTS and DTS mainly rely on the precise definition influence area of sensor nodes.

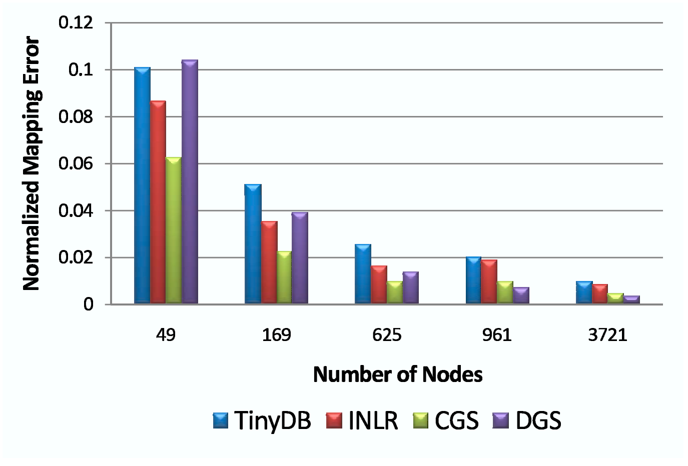


Figure 5.1: Simulation results in grid deployment.

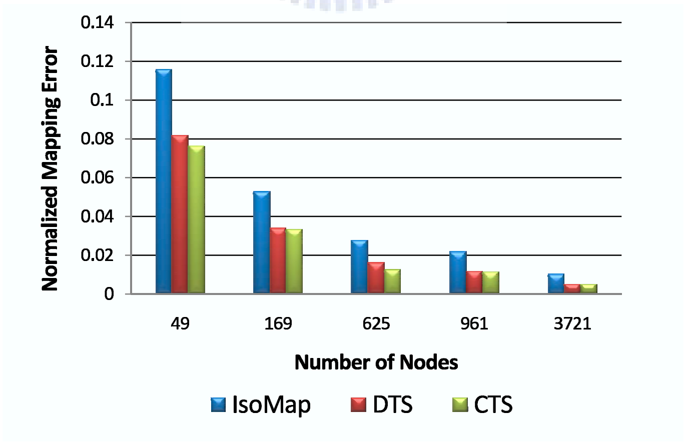


Figure 5.2: Simulation results in random deployment.

5.2 Network Traffic and Computation Overhead

The traffic generation is based on the number of reporting nodes. The work [8] has been proven that the number of isoline nodes is $O(n^{1/2})$ for any constant number K contour regions within a square area of n sensor nodes. TinyDB, INLR, CGS and CTS need to collection all sensory data from whole network, so that traffic generation of these schemes is $O(n)$. Iso-Map, DGS and DTS select some nodes correlated to the isoline to report to the sink, and the traffic generation is thus limited to $O(n^{1/2})$. Specifically, INLR only save slight traffic by aggregating the same reporting packets based on tree network structure. The traffic generation of DGS and DTS are at least one-third of Iso-Map through forming triangular blocks and reporting through leaders. The energy consumption of communication is the most costly in wireless sensor network. In DGS and DTS, we aim to reduce the network traffic in contour map construction. In order to achieve cost effectiveness of energy consumption, we utilize the local information to approximate the contour map. Only leader nodes generate traffic which is reported to the sink to reduce communication overhead.

There are three kinds of computation workload of DGS and DTS: 1) Initiation: the in-networks approach of selecting boundary nodes; 2) Computing: the boundary nodes do localized computing for partial isolines on triangular blocks; 3) Reporting: leaders report the partial isolines in each triangular block. The computation workload of boundary nodes discovery is bound on $O(n^{1/2})$ and partial isolines estimation is also bound on $O(n^{1/2})$ because of only three nodes of a triangular block involved on $O(n^{1/2})$ numbers of triangular blocks. The reporting phase only relays the computation results without additional computation. Thus the computation workload of reporting phase is bounded by $O(n)$. The TinyDB, INLR, and Iso-Map basically rely on the sink to construct the contour map, whereas the DGS and DTS can localized construct the partial isolines. The computation of DGS and DTS are more than TinyDB and INLR, nevertheless, the traffic generation is less and more accuracy in dense sensor networks than others.

Table 5.1: The comparison of network traffic and computation workload.

Approach	Traffic Generation	Network Computation	Sensor Deployment
CGS	$O(n)$	$O(n)$	Grid
DGS	$O(n^{1/2})$	$O(n^{1/2} + n)$	Grid
TinyDB	$O(n)$	$O(n)$	Grid
INLR	$O(n)$	$\Omega(n^{3/2})$	Grid
CTS	$O(n)$	$O(n)$	Random
DTS	$O(n^{1/2})$	$O(n^{1/2} + n)$	Random
Iso-Map	$O(n^{1/2})$	$O(deg(G) \times n^{1/2} + n)$	Random

On the other hands, INLR is a aggregation-based approach, and it can not achieve load balance, because it require processing the similar data by aggregation when reporting packets to the sink. Those nodes close to the sink have heavier computation load than those nodes away form the sink. However, for DGS and DTS, we can balance per node workload by alternating leader nodes selection. Then, the per node computation workload is almost equal in a long run. Table 5.1 summarizes the comparison of network traffic and computation overhead. Here, we define $deg(G)$ is the average degree of each node in the network.

Chapter 6

Conclusions

This paper is the first work to improve contour mapping accuracy for diffusion events in wireless sensor networks. We propose accurate contour maps construction algorithms to approximate the sensing values for diffusion events in an energy efficient way. Since accuracy and communication overhead are conflicting concerns, centralized and localized contour map construction algorithms are proposed for different deployment and environmental requirements. The centralized surface approximation aims at improving accuracy, and the localized plane approximation is to reduce the reporting traffic by localized computing. We redefine the influence area of sensor nodes, and both centralized and localized methods can improve contour mapping accuracy by eliminating the limitation of isoline passing on grid boundary or some sensor nodes. Furthermore, the additional benefits of our contour map construction algorithms is to reduce communication workload and per node computation. In conclusion, the mapping accuracy and scalability of our algorithm is superior, which makes our algorithms feasible for the large-scale deployed sensor networks. In the future, we will extend to spatial and time series contour boundary approximation model.

Bibliography

[1] TinyDB, a declarative database for sensor networks. http://telegraph.cs.berkeley.edu/tinydb.

[2] P. Bose and P. Morin. Online routing in triangulations. In *Proc. of 10th annual Int'l Symp. on Algorithms and Computation (ISAAC)*, 1999.

[3] C. Buragohain, S. Gandhi, J. Hershberger, and S. Suri. Contour approximation in sensor networks. In *Proc. of Int'l Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2006.

[4] S. Gandhi, J. Hershberger, and S. Suri. Approximate isocontours and spatial summaries for sensor networks. In *Proc. of Int'l Symposium on Information Processing in Sensor Networks (IPSN)*, 2007.

[5] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *Proc. of Int'l Symposium on Information Processing in Sensor Networks (IPSN)*, 2003.

[6] S.-P. Kuo and Y.-C. Tseng. A scrambling method for fingerprint positioning based on temporal diversity and spatial dependency. *IEEE Trans. on Knowledge and Data Engineering*, 2008.

[7] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed construction of a planar spanner and routing for ad hoc wireless netwroks. In *Proc. of IEEE INFOCOM*, 2002.

[8] Y. Liu and M. Li. Iso-Map: Energy-efficient contour mapping in wireless sensor networks. In *Proc. of Int'l Conference on Distributed Computing Systems (ICDCS)*, 2007.

[9] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of ACM Int'l Workshop on Wireless sensor networks and applications (WSNA)*, 2002.

[10] X. Meng, T. Nandagopal, L. Li, and S. Lu. Contour maps: Monitoring and diagnosis in sensor networks. *Computer Networks*, 50(15), 2006.

[11] D. Niculescu and B. Nath. Ad hoc positioning system (aps) using aoa. In *Proc. of IEEE INFOCOM*, 2003.

[12] M.-S. Pan, L.-W. Yeh, Y.-A. Chen, Y.-H. Lin, and Y.-C. Tseng. A WSN-based intelligent light control system considering user activities and profiles. *IEEE Sensors*, to appear.

[13] M. Singh, A. Bakshi, and V. K. Prasanna. Constructing topographic maps in networked sensor system. In *Proc. of Workshop on Algorithms for Wireless and Mobile Networks (ASWAN)*, 2004.

[14] I. Solis and K. Obraczka. Efficient continuous mapping in sensor networks using isolines. In *Proc. of Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, 2005.

[15] H. Spath. *Two Dimensional Spline Interpolation Algorithms*. AK Peters, Ltd., 1995.

[16] X. Wang, J. S. Dong, C. Chin, S. Hettiarachchi, and D. Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 2004.

[17] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Proc. of European Workshop on Sensor Networks (EWSN)*, 2005.

[18] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. In *Proc. of ACM Int'l Conference on Special Interest Group on Management Of Data (SIGMOD)*, 2006.