5-2009

Towards Spatial Queries over Phenomena in Sensor Networks

Guang Jin

TOWARDS SPATIAL QUERIES OVER PHENOMENA IN SENSOR NETWORKS

By

Guang Jin

B.A. Huazhong University of Science and Technology, 2000

M.S. Huazhong University of Science and Technology, 2003

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

May, 2009

Advisory Committee:

Silvia E. Nittel, Associate Professor of Spatial Information Science and Engineering, Advisor

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

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

Clayton M. Wheeler, Associate Professor of Chemical Engineering

Matt Duckham, Senior Lecturer of Geographic Information Science, University of Melbourne, Australia

LIBRARY RIGHTS STATEMENT

TOWARDS SPATIAL QUERIES OVER PHENOMENA IN SENSOR NETWORKS

By Guang Jin

Thesis Advisor: Dr. Silvia E. Nittel

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
(in Spatial Information Science and Engineering)
May, 2009

Today, technology developments enable inexpensive production and deployment of tiny sensing and computing nodes. Networked through wireless radio, such senor nodes form a new platform, wireless sensor networks, which provide novel ability to monitor spatiotemporally continuous phenomena. By treating a wireless sensor network as a database system, users can pose SQL-based queries over phenomena without needing to program detailed sensor node operations. DBMS-internally, intelligent and energy-efficient data collection and processing algorithms have to be implemented to support spatial query processing over sensor networks. This dissertation proposes spatial query support for two views of continuous phenomena: field-based and object-based.

A field-based view of continuous phenomena depicts them as a value distribution over a geographical area. However, due to the discrete and comparatively sparse distribution of sensor nodes, estimation methods are necessary to generate a field-based query result, and it has to be computed collaboratively 'in-the-network' due to energy constraints. This dissertation proposes SWOP, an in-network algorithm using Gaussian

Kernel estimation. The key contribution is the use of a small number of Hermite coefficients to approximate the Gaussian Kernel function for sub-clustered sensor nodes, and processes the estimation result efficiently.

An object-based view of continuous phenomena is interested in aspects such as the boundary of an 'interesting region' (e.g. toxic plume). This dissertation presents NED, which provides object boundary detection in sensor networks. NED encodes partial event estimation results based on confidence levels into optimized, variable length messages exchanged locally among neighboring sensor nodes to save communication cost. Therefore, sensor nodes detect objects and boundaries based on moving averages to eliminate noise effects and enhance detection quality. Furthermore, the dissertation proposes the SNAKE-based approach, which uses deformable curves to track the spatiotemporal changes of such objects incrementally in sensor networks. In the proposed algorithm, only neighboring nodes exchange messages to maintain the curve structures. Based on in-network tracking of deformable curves, other types of spatial and spatiotemporal properties of objects, such as area, can be provided by the sensor network. The experimental results proved that our approaches are resource friendly within the constrained sensor networks, while providing high quality query results.

ACKNOWLEDGMENTS

First of all, I need to thank my parents, Chaoyang Jin and Jinyu Ma, my wife, Jiaojiao Gu. Without their love and support, I could not finish my Ph.D. study.

I appreciate the guidance and encouragement from my advisor, Dr. Silvia Nittel. I also need to thank the other members of my advisory committee, Dr. Kate Beard, Dr. Michael Worboys, Dr. Clayton Wheeler and Dr. Matt Duckham. This dissertation could not be finished without their support and advices.

I thank all members of the Laboratory of GeoSensor Networks, Arda Nural, Qinghan Liang and Danqing Xiao for their suggestions and help on my dissertation and dissertation presentation.

I gratefully acknowledge the financial support for my Ph.D. study. My financial support includes the NSF Award No.0448183, the NSF Award No.CTS-0428341, and the University Graduate Research Award (UGRA 2007-2008).

The last but not the least, I want to thank the members of the Department of Spatial Information Science and Engineering, and all fellows of the Office of International Program. Because of their friendship and hospitality, I have had the wonderful and precious time in Maine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The advancements of device manufacture have extended our abilities to measure and record phenomena occurring in the world, in parallel with the development of information processing techniques assisting us to understand the phenomena. The continuing miniaturization of microchips has enabled micro devices to be integrated with micro sensors, computing units, local storage and wireless radios.

With integrated sensing, computing and communicating, these tiny devices can be interconnected to form a new platform, *Wireless Sensor Network* (WSN). A WSN can be distributed over a geographic area. Environmental activities can be observed, estimated and understood at high spatial and temporal resolutions. We are particularly interested in efficient approaches to collect and process spatial information in WSNs, such as monitoring a continuous phenomenon within a geographic region.

First, a brief introduction to the enabling technology of WSN is necessary.

1.1 Wireless Sensor Network

This section describes the technology aspects and gives an overview of current WSN applications.

1.1.1 Sensor, Sensor Node and Networked Sensor Nodes

Recent developments in micro-scale sensor technologies have dramatically decreased the size of sensors. The miniaturization of sensors brings several advantages such as increased portability, low-power operation, and improved selectivity. At the University of Maine, research programs are underway to develop novel sensors in the *Laboratory for Surface Science and Technology* (LASST). In LASST, new sensing materials, engineering and fabrication technologies are being explored to develop advanced tiny sensors. For example, the project led by C. Wheeler integrates micro-scale hotplates with thinfilm sensors to improve the selectivity and sensitivity of sensors to toxic gases [WTW+01].

Tiny devices consisting of micro-scale sensors, wireless communication and computing units are known as *sensor nodes*. Among today's available general sensor node platforms, the MICA series and Telos series are both designed by the University of California at Berkeley. Figure 1.1 shows two sensor node models of the MICA series, MICA2 and MICADOT manufactured by Crossbow. Both platforms originated in the Smart Dust project

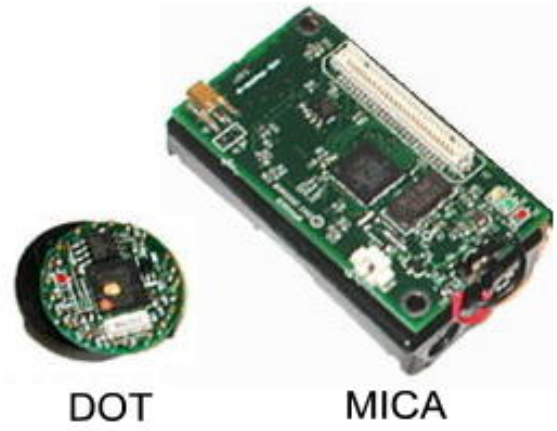


Figure 1.1. MICA2 and MICADOT

[WLLP01], which introduced the first prototype, the "COTS mote" [Hol00]. Later commercially available sensor node platforms have brought more advanced features. For example, the Telos motes provide the hardware write protection to fight against the malicious code infringement in the wirelessly programmable environment [PSC05]. The *Sun Small Programmable Object Technology* (SPOT), provided by SUN, supports the *Java 2 Micro Edition Virtual Machine* (J2ME VM).

The architecture of general sensor nodes mainly comprises three parts, the computing and storage unit, the sensor unit and the wireless communication unit. Via the attached sensors, sensor nodes collect readings about the local phenomena such as the environmental temperature or the concentration of a toxic chemical [JMGRP09]. The on-board computing and storage unit enables sensor nodes to run customized program codes and process the sensor data. Through the integrated radio devices, sensor nodes are able to communicate

and collaborate with each other to monitor complex environmental activities. The novel features allow a WSN to run with little human maintenance. Compared to traditional sensing platforms, WSNs provide a higher resolution, more precise, faster and more economical solution to observe the physical world at a novel scale in real time.

1.1.2 WSN Applications

In this dissertation, we focus on geo-sensor networks (i.e., WSNs that are generally used to automatically monitor environmental activities). Traditionally, investigators have used expensive, large-sized sensors connected through cables for power and communication, such as ocean buoys or windmills. The sample rate (over the temporal space) and density (over the geo-graphical space) usually is low due to technical and economical difficulties (e.g., network deployment and connectivity). For some applications, the traditional infrastructure is sufficient. For example, weather stations are spread out miles away from each other; and weather data are often collected hourly. WSNs, however, enable novel applications, which need highly detailed and real-time measurements from the physical world.

Seabirds are an interesting research topic to marine ornithologists. Seabird colonies, however, are sensitive to human disturbances. Even a 15-minute visit to a cormorant colony can cause 15% mortality in eggs and chicks in a breeding year [And95]. The human disturbance is more serious for a small island environment, since some seabirds cannot emigrate to other lands. WSNs provide ornithologists a less disruptive way to observe the seabird

habitat. In 2002, an interdisciplinary group consisting of computer scientists, computer engineers and ornithologists conducted a survey on the Great Duck Island, a 237 acre island located south of Mount Desert Island in Maine [MCP+02]. A set of specific sensors, including pressure sensors, infrared sensors, and highly sensitive humidity sensors, were attached to MICA nodes. Without significantly changing the hardware design of MICA nodes, the survey provided high quality data and produced a new habitat monitoring kit for ornithologists.

Novel WSNs are best used to monitor phenomena, which cannot be observed by using traditional sensing platforms. For example, WSN make it possible to observe the microclimate of the plants in an orchard, vineyard or other precision agriculture areas, which cannot be observed by remote satellites. Redwood trees are known to be the largest and oldest trees in the world. Some trees are more than 360 feet in height. As one can image, the microclimate varies significantly over the height of a redwood tree and has substantial spatiotemporal variations. Humidity fronts also move along the giant trees as the trees move water from the earth into the air. Before the availability of WSN, botanists had to climb a giant redwood tree to attach a winch at the top, and vertically haul an instrument set connected through a long cable to a battery powered data logger. In such a way, it is almost impossible to collect detailed information about the variations of the microclimate along and around the tree. In 2004, a test application used MICA2 and MICA2DOT nodes to record the 44-day life of a 70-meter tall redwood tree [TPS+05]. The sensor data were

collected at a remarkably high sampling rate at every 5 minutes and a high density of 2 meters in space, which successfully illustrated a high-resolution view of the microclimate over the tree. Although WSNs can provide higher resolution than traditional sensing platforms, the sensed samples are still discrete points, and need intelligent data collection and processing mechanisms to compile a smooth view of the microclimate around the tree or in specific, interesting areas of the microclimate.

Monitoring and controlling systems play one of the most important roles in agriculture and industry. WSNs can run over vineyards and provide real-time monitoring results of temperature, soil humidity, sunlight and fertilizer [BBB04]. Combined with actuator nodes, an automatically monitored vineyard can use local heaters, defoggers, or watering to optimize the growing conditions for grapes. WSNs enable novel levels of precision agriculture today.

Sensor nodes can be used to observe the contamination in wide-area environments [JMGRP09]. In this type of application, users are mostly interested in the contaminated regions that pose a health hazard to humans, for example, the chemical contamination in a battlefield with regard to chemical warfare agents. In most cases, the boundary of contaminated regions is sufficient to describe the regions. However, the information about contamination has to be computed and available in real-time to notify humans located in the area. We can use a WSN to monitor such an environment, and detect the boundary of

contaminated regions. A WSN is able to provide real-time reports about the spatiotemporal changes of the contaminated regions, which can hardly be done by traditional sensing platforms.

WSNs are used to collect real-time sensor readings from the physical world. The real-time sensor readings provide a new perspective to model and understand environmental activities. If the Moore theorem remains true, sensor nodes will continue to become smaller, more powerful, and more economical. The WSN solution will be more economical and more efficient in the near future.

1.2 Modeling WSN

Before presenting our research questions, we present a conceptual model with regard to WSNs, their deployment, the underlying phenomena to be monitored and the collected sensor readings.

1.2.1 Phenomenon

A phenomenon is "a particular (kind of) fact, occurrence, or change as perceived through the senses or known intellectually" as defined by the online Oxford English Dictionary [Oxf08]. In environmental monitoring, phenomena are the subject of observation. A phenomenon is a material thing occurring in the physical world. Examples of phenomenon are temperature, wind-speed, or the concentration of a gas pollutant in the air. As illustrated by

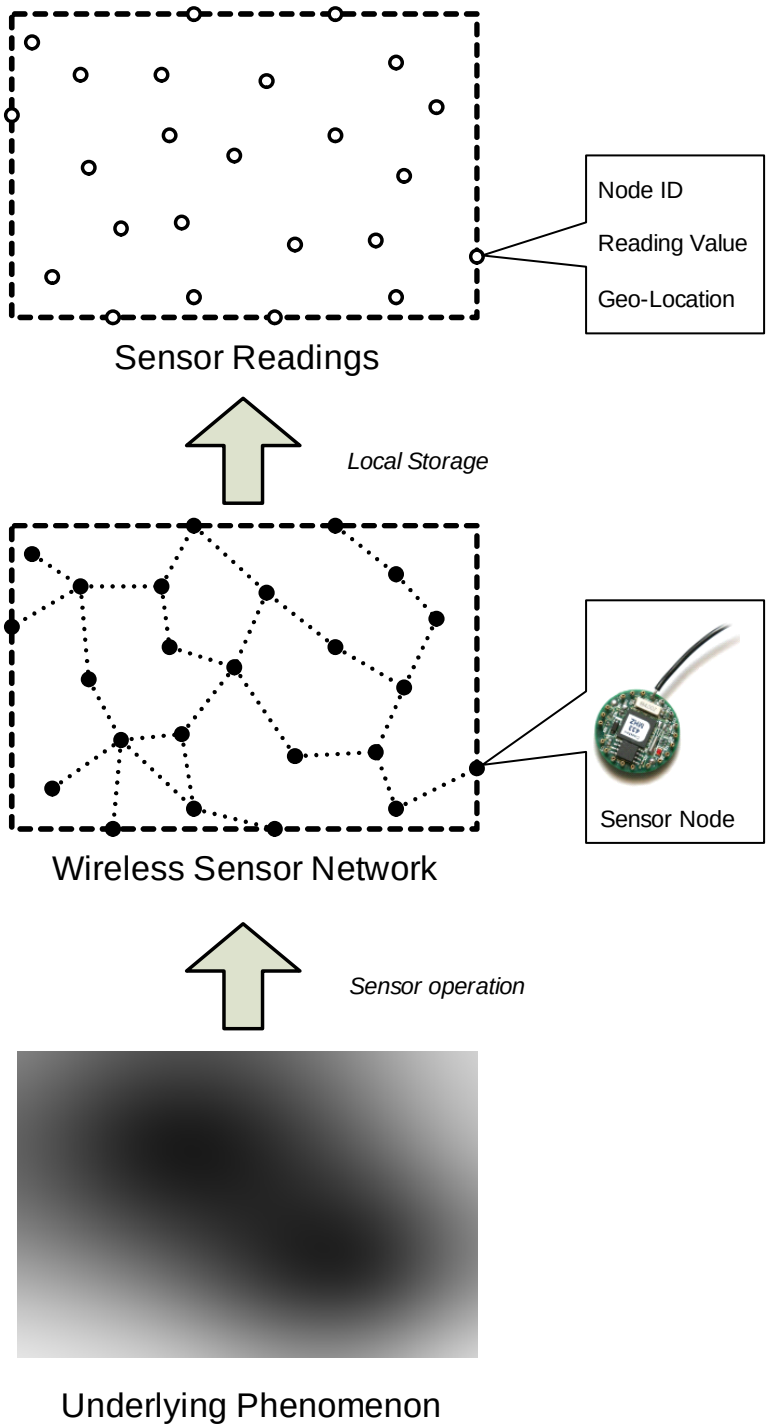


Figure 1.2. A model of wireless sensor networks

Figure 1.2, a WSN measures one or more underlying phenomena through the distributed sensors. A conceptual model is needed to interpret sensor readings. Different models, however, may interpret the same phenomenon in different ways.

1.2.2 Monitored Region

A monitored region, $\mathbb{M}$, is a subregion of the geographical space. Within the boundary of $\mathbb{M}$, a WSN is installed to observe the phenomena inside by attached sensors. In this dissertation, we assume that $\mathbb{M}$ is a single contiguous 2D Euclidian space defined by,

$$\mathbb{M} \subset \mathbb{R}^2. \tag{1.1}$$

1.2.3 Sensor Reading

Sensors return local readings through physical or chemical interactions with the underlying phenomena. A sensor node can measure local phenomenon properties directly through attached sensors. For example, temperature sensors can measure the environmental temperature. Through the readings from on-board sensors, a sensor node can also process indirect information. For example, the battery voltage readings can provide indirect temperature estimation results, since the battery voltage and the environmental temperature are highly correlated [DGM+05]. Some sensors can detect phenomena far away from where the sensors are located. For example, a camera can return 2D photographic readings about

a faraway place through a telephoto lens. Most tiny sensors, however, only produce readings about the local phenomenon properties. In this dissertation, we use s_i to identify an individual sensor node and its spatial location. In current WSNs, sensor readings are geo-referenced and time-stamped to indicate where and when the readings are collected.

1.3 Field-based and Object-based Models

We now need to clarify several basic concepts. In this dissertation, we use the term "*space*" to refer to "geographic space". Spatial data represent the structure and properties of phenomena over locations at the Earth's surface [WD04]. Since sensor nodes are embedded in the physical world and collect geo-referenced sensor readings, WSNs provide spatial and spatiotemporal data directly.

A fundamental question of this dissertation is how we model and represent the spatial information of an underlying phenomenon. There are two general and distinct types of models, *field*-based and *object*-based models [WD04].

1.3.1 Field

A field-based model views an underlying phenomenon as a set of locations with properties. Typically, a field has no boundary. For example, one can imagine the temperature field over all geographic locations in the State of Maine, or the University of Maine Campus. In a field-based model, a phenomenon is formalized as a function from a spatial framework to

an attribute domain [WD04]. The attribute domain can consist of simple labels or ordered labels (i.e., *nominal* attribute and *ordinal* attribute). In this dissertation, we focus on the attribute domain consisting of quantities on an *interval* attribute or *ratio* attribute scale. Consequently, in this dissertation, the attribute domain is represented by real numbers.

Formally, given a geographical space S and a class of scalar values V, a field is a function $\mathbb{Y}$ whose domain is S and codomain is V [DNW05]. Although the geographical space is a 3D space, we restrict S as a 2D space here.

Definition 1 *A field is defined as follows.*

$$\mathbb{Y} : \mathbb{R}^2 \rightarrow \mathbb{R}. \tag{1.2}$$

The function $\mathbb{Y}()$ can be continuous or discontinuous. In this dissertation, we are only interested in spatiotemporally continuous phenomena. For many practical reasons, we need the phenomena to be smooth. More specifically, our approaches target phenomena that can be represented by fields with first derivatives existing everywhere.

In this dissertation, for each point p in $\mathbb{M}$, we define the field $\mathbb{Y}()$'s value as $\mathbb{Y}(p)$. $\mathbb{Y}(p)$ is a 1D scalar value in V. $\mathbb{Y}(s_i)$ is the field value at the node s_i's location, and also indicates the accurate and noise-free raw sensor reading at s_i.

1.3.2 Object

In an object-based model, the underlying phenomena are viewed as collections of objects with properties. Different from fields, an object usually covers a discrete region in space,

and has a closed boundary. An object must be identifiable, relevant (be of interest) and describable (have properties) [WD04]. The object boundary provides important geometric information to describe the object. Non-spatial properties (such as name, identifier or owner) can be "virtually" attached to an object. In a field-based model, the spatial framework is a fixed reference. A field measures the distribution in attribute values with respect to the reference. In an object-based model, the spatial framework is not a distinguished reference. The geo-reference is provided by objects themselves (e.g., via the object boundaries) [WD04].

There are many areas in spatial information science, in which fields and objects are used as different representations for the same phenomenon. For example, a field representation is used as a basis to extract "features" (i.e., objects). Feature extraction and change detection are established research domains to automatically extract objects, especially from remote sensing images. Similarly, we can use WSNs to observe and monitor a continuous phenomenon,(e.g., the NO_2 distribution over the coastal region of Downeast Maine), but be interested in the regions of dangerous levels, (e.g., a toxic NO_2 cloud). These regions and their boundaries can be extracted, and represented as objects. Thus, a second type of query for continuous phenomena is established in extracting and tracking boundaries representing as objects.

2D Object

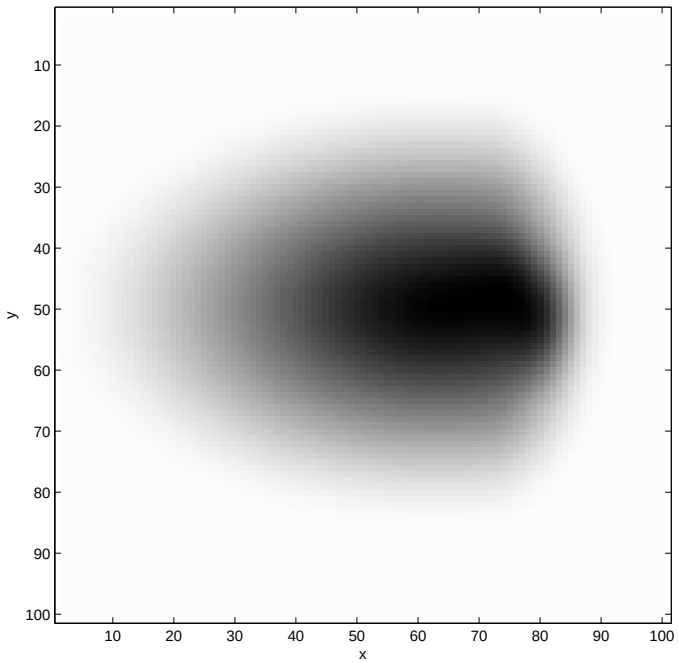
We model spatial objects as 2D objects in this dissertation, since the monitored region normally is 2D. We assume that a 2D object can be identified based on user defined threshold values, such as a threshold value = toxic level for human.

Definition 2 *For a given point, p, we define a 2D object, $\mathbb{O}()$, as a local object status given by a user-defined threshold, T.*

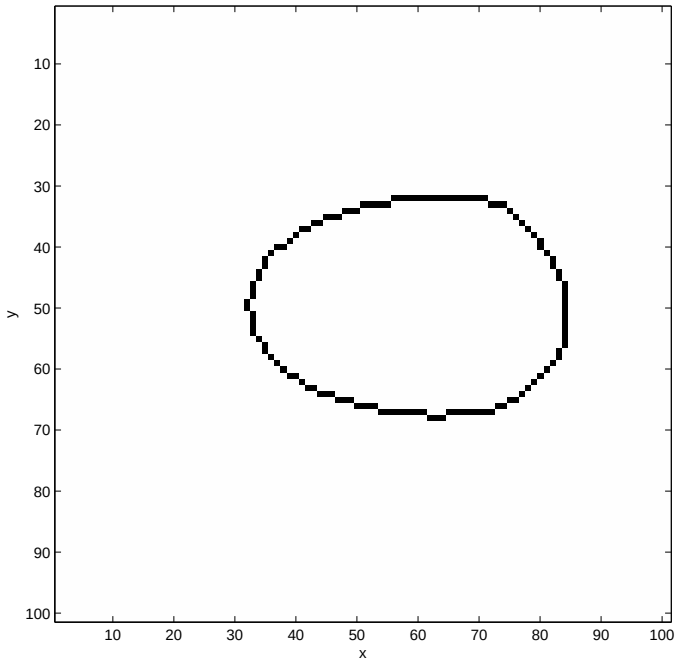
$$\mathbb{O}(p) = \begin{cases} 1, \textit{if } \mathbb{Y}(p) \geqslant T \\ \\ 0, \textit{else} \end{cases} \tag{1.3}$$

Equation 1.3 provides a simple but useful model to derive objects based on quantitative sensor readings. If $\mathbb{Y}()$ represents temperature, we can use $\mathbb{O}()$ based on $T = 200$°C to define a fire. Equation 1.3 can be extended to derive complex objects. For example, a cozy place, $Cozy()$, indicates where the temperature is between 20°C and 25°C. $Cozy()$ can be defined as, $Cozy(p) = AND(O_1(p), NOT(O_2(p)))$. Here O_1 is defined by the threshold, $T_1 = 20$°C; O_2 is defined by $T_2 = 25$°C.

Equation 1.3 provides the definition of local object status at individual spatial points. To understand the spatial properties of 2D objects, we need to find the 2D regions covered by the spatial objects. The *boundary* of a 2D object separates the object region from the non-object region. The object boundary provides important geometric information about the object, such as its shape, size and location. For example, based on the field representation



(a)



(b)

Figure 1.3. A phenomenon and the boundary of a 2D object
(a) the field representation of the phenomenon, and (b) the boundary based on $T = 0.5$

shown in Figure 1.3(a), the region covered by the 2D object based on $T = 0.5$ is inside the boundary as shown by Figure 1.3(b). Since the underlying phenomenon is spatially continuous, the threshold, T, provides a natural choice to define the object boundary.

Definition 3 *For a given point, p, the boundary status, $\mathbb{B}()$, of an object $\mathbb{O}()$ defined by the threshold T, is,*

$$\mathbb{B}(p) = \begin{cases} 1, \textit{if } \mathbb{Y}(p) = T \\ \\ 0, \textit{else} \end{cases} \tag{1.4}$$

Both Equation 1.3 and Equation 1.4 describe 2D objects in a field-based way [Gal01]. Based on Equation 1.3 and Equation 1.4, distributed sensor nodes can generate point reports about the 2D objects and object boundaries. Tracking 2D objects temporally provides the foundation to extract spatiotemporal properties of 2D objects.

Spatiotemporal Qualitative Changes

Abstract spatiotemporal properties of 2D objects are useful to describe 2D objects' change in time and space. Several properties are especially useful in qualitative approaches.

A qualitative approach models 2D objects as two types of entities, *continuants* and *occurrents*. Continuant entities endure over time. In this dissertation, continuants are 2D spatial objects, such as boundaries, holes, and regions. Occurrent entities "comprise what are variously called events, processes, happenings" [Sim87], such as *splitting* and *merging* events.

By combining continuants with occurrents, we can present the qualitative changes of 2D objects within the spatiotemporal space. According to A. Galton, spatiotemporal qualitative changes can be classified into eight different classes, namely *Changes in dimension*, *Changes in connectivity*, *Changes in location*, *Changes in orientation*, *Changes in area (size)*, *Changes in shape*, *Changes in posture*, and *Changes in non-geometric spatial attributes* [Gal00]. Changes in dimension and connectivity are purely topological changes. Changes in location, orientation and area changes are metrical changes. In general, location changes describe motion in spatiotemporal spaces. Changes in orientation involve turning or rotation. Changes in shape describe the temporal shape change of a spatial object. Changes in posture are complex changes involving all of the above changes. Changes in non-geometric spatial attributes describe the spatiotemporal changes that are not only based on the geometric representation. For example, the "front" and "back" notions about some objects, such as machines, need more information than only geometric information.

Let's take the wildfires that occurred in Southern California in October 2007 as an example of qualitative spatiotemporal changes. We consider the individual wildfires as objects and observe their qualitative changes. The *Witch Fire*, the largest wild fire, started in Witch Creek Canyon near Santa Ysabel and spread quickly to Ramona, Rancho Bernardo, Poway and Escondido. The *Poomacha Fire* began on the La Jolla Indian Reservation, and established itself on the Palomar Mountain. On October 24, the *Witch Fire* and the *Poomacha Fire* merged and entered the Agua Tibia Wilderness. On October 31, the *Witch*

Fire was declared fully contained. The above description of wildfires only reports the qualitative spatiotemporal properties, such as the area, location, and topology information, about the 2D objects.

Sensor nodes have been deployed to detect the toxic contamination in wide-area environments [JMGRP09]. Similarly to the wildfire example, we can use WSNs to identify toxic clouds as objects from the contaminated region, and track the spatiotemporal movements of the clouds. By tracking 2D objects, a WSN is able to monitor abstract spatiotemporal changes of 2D objects. An abstract spatiotemporal change, such as "a toxic cloud is splitting at location X", can usually be transmitted in a more compact form rather than the temperature values of all nodes. Furthermore, the spatial and spatiotemporal qualitative information is more helpful for people to understand environmental activities than some quantitative spatial data.

1.4 Intelligent Data Collection using WSN

With integrated wireless communication, tiny sensor nodes provide attractive features, such as the high portability and autonomy. These advantages, however, also bring new challenges. Our objective is to design resource-efficient and robust intelligent data collection strategies using WSN. Therefore, we need to assess the constraints of WSN first and how to design our approaches efficiently.

1.4.1 Constraints of WSN

WSNs are a constrained environment with the following characteristics [CES04, ASSC02].

Limited energy sources Today's sensor nodes are powered by batteries. A WSN needs to run with little human maintenance and interference. It is often inconvenient and uneconomical to replace or recharge a sensor node's battery in practice. Thus, minimizing the energy consumption of sensor nodes is a main consideration in current research.

Limited processing capability and memory space Microprocessors, or *Micro Processing Units* (MPU), on sensor nodes are designed to run energy-efficiently. A typical MPU on sensor nodes needs around one $milliwatt$ while running at about $10\ MHz$ today. An MPU can switch into the sleep mode to preserve more energy, but at the expense of decreased computational power. The memory space of microprocessors is also limited. Typically, only a few $Kbytes$ RAM and a few $Mbytes$ ROM are available on a sensor node. Due to the power and size limitation, both processing and memory capabilities of sensor nodes are fairly limited. Software programs running on sensor nodes have to run within the constraints of the MPU and limited memory space.

Constrained wireless communication Compared with current wired Ethernets, the wireless communication of WSN is deliberately simple and has a compact protocol stack

to reduce the overhead. For example, the bandwidth over the data link layer of MICA2 nodes is around $56\ Kbps$ under the best condition. The communication range of wireless radios on sensor nodes is also limited, typically to tens of meters. Furthermore, the energy consumption of wireless radios increases rapidly with respect to the target distance and transmitted message size [Ett98]. For example, broadcasting one bit of data on a Berkeley $MICA2$ node consumes the same amount of energy as computing 800 or more instructions on-board. To cover a large area, a WSN has to relay messages by different sensor nodes. The wireless communication consumption significantly influences the WSN performance. Sensor data need to be collected intelligently. The in-network data processing helps to understand the collected data as well as to minimize the overall communication in the entire network. The wireless communication also faces more difficulties, such as the higher packet loss rate, compared with wired communication connections. Due to the unavoidable hop-by-hop communication, the wireless communication contention is also a challenge to WSN.

Failure prone nature WSNs are volatile in many ways. The first volatility is the imprecise and inaccurate sensor readings. Information processing techniques must be able to handle faulty and noisy sensor readings in a WSN. Due to the limited battery life, uneven communication load and unpredictable environmental factors, some sensor nodes may stop functioning while other nodes are still alive. To avoid a crash of the whole system, WSNs need to automatically recover from node failures and

adapt their processing, which brings challenges to message routing and information processing algorithms. The wireless communication is not as reliable as the current wired communication due to the nature of wireless channels. For example, the MICA2 nodes are using the $900\ MHz$ band that conflicts with some cordless phones. Handling the package loss and increasing the information processing quality is an important issue for WSNs.

1.4.2 Sensor Network Database Management System

Although programming languages [GLB+03], operating systems [LLWC03, HKS+05] and high-level programming modules [WM04] for WSNs are available, application programming for WSNs is highly tedious, and needs computer science expertise, due to the constraints and highly parallel nature of WSN. Early WSN applications simply collected raw sensor readings to a central base station as a proof of concept [MCP+02]. This simple centralized approach is not feasible since the expensive communication drains the battery energy fast, and shortens a WSN's lifetime to a small fraction of the potential lifetime. A WSN can be viewed as a special *DataBase Management System* (DBMS). We name this special DBMS as *Sensor network Database Management System* (SDMS). From the perspective of SDMS, a WSN is a virtually single, but in reality a distributed database system. Each sensor node acts as a mini database system, which participates in the distributed query execution. The programming complexity of WSN applications is greatly reduced for the

user by SDMSs. Users can use the standard declarative query language, which is the *Structured Query Language* (SQL), to interact with SDMSs without knowing how to operate sensor nodes, route query results or optimize the query execution.

Today several SDMSs prototypes are already available, such as TinyDB [MFHH05, HHMS03, MFHH02] and Cougar [BGS01, DGR+03, BGS00]. As said, the important idea of using SDBMs for intelligent data collection is the simple user interface in the form of declarative SQL queries. For example, a user can pose a query

> "SELECT light, temp FROM sensors WHERE light > 400 AND node.id $= 43$ SAMPLE PERIOD $= 1024ms$ LIFETIME $= 30days$"

in TinyDB [MFHH05]. This SQL statement requests the light and temperature readings of the $No.43$ sensor node only when the light reading is above 400. The predicate, "SAMPLE PERIOD", defines the sampling rate, while "LIFETIME" indicates how long the query is active in the SDMS. TinyDB has been applied in the redwood tree project [TPS+05].

Albeit providing a simplified user interface, SDMSs are still responsible to generate and optimize the code to actually execute the data collection and processing in WSNs. Users, on the other hand, are shielded from the programming details. A paradigm to generate the query execution and optimize the resource consumption of WSN is the in-network aggregation query processing. The *Tiny AGgregation service* (TAG) is a widely applied framework for processing aggregation queries in SDMSs [MFHH02]. Based on a tree-structured routing protocol [AWSBL99, WTC03], all sensor nodes are connected to the

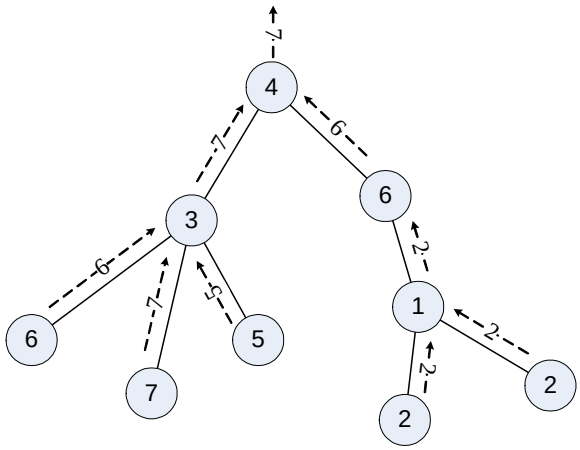


Figure 1.4. Processing a MAX query in TAG

root node in TAG. In this network topology, the root node is usually connected to the base station and works as a gateway to the network. The time is divided in to *epochs*, and parent-children nodes are synchronized through epochs. Each node is allowed to send a single message during an epoch. For example, in Figure 1.4, the circles indicate the sensor nodes, and the numbers inside circles are local sensor readings. A node aggregates the partial results from its children, computes a new partial result by comparing the locally sensed information with the partial results from the children nodes, and transmits the new partial result to its parent in the next epoch. For example, to process a query that reports the maximum value collected in the WSN, a node computes the local maximal value based on its local sensor readings and the partial results from its children, and sends the local partial maximal result to its parent node, as illustrated by Figure 1.4. In TAG, all nodes participate in the aggregation processing. The message size can be kept constant during

the processing. It is provable that the wireless consumption is minimal in this distributed aggregation processing.

By using SDMSs, users interact with WSNs through standard declarative query languages. Sensor nodes participate in the query execution in SDMSs through efficient query processing algorithms to reduce the wireless consumption.

1.5 Research Challenges

Today's SDMSs are still in their infancy. In this dissertation, we focus on the SDMS support of spatial queries for continuous phenomena in environmental applications. To understand the phenomenon monitored by a WSN, we need to process readings from different sensor nodes and find the relationships among the nodal sensor readings to represent the underlying phenomenon. Collecting raw sensor readings and analyzing them at a central base station can be implemented. However, collecting raw sensor readings from a large number of nodes at a high rate depletes the network resources quickly, especially in the multi-hop communication topology. The challenge is to process spatial queries and collect information within a WSN as much as possible, and find distributed algorithms to support our approaches.

Although field-based and object-based models are represented differently, both types of models are necessary and have their own application areas. "The field-based and object-based views of the world are not separate monolithic systems but intimately related by an

intricate network of interconnections [Gal03]". An SDMS should support field-based as well as object-based models, and act differently in different models.

1.5.1 Querying Phenomena as Fields

Due to the fact of discrete node distribution in WSNs and the fact of sensing noise, an SDMS can only estimate an observed field, and provide an estimated function, $\hat{Y}()$, to users. The first challenge is that only nodal sensor readings are available in WSNs, but we are interested in the field over continuous subregions. Thus, additional estimation techniques are necessary to analyze further spatial information and relationships from sensor readings.

In this dissertation, we name this type of query "*quantitative spatial window queries*". A quantitative spatial window query returns a fine-grained, but estimated value distribution of a phenomenon within a user-specified geo-referenced rectangular query region. Among different ways to represent a phenomenon's value distribution, the image-based representation is the most general and intuitive, and can be used to represent such query results.

We distinguish two types of spatial queries that use estimation: spatial *point* queries and spatial *window* queries. A spatial point query requests the estimated value of any geographic point within the monitored region. In a simple case, this point location can overlap with a sensor node location, and the SDMS returns the sensor readings collected by the node. In most cases, the requested point value does not coincide with any sensor node

location; the SDMS has to estimate the value based on readings from several sensor nodes surrounding the requested point. In the case of spatial window queries, an estimation of all points located within a user specified rectangular region is requested. Thus, the SDMS needs to return a rectangle-shaped image of the underlying phenomenon distribution within the query region at a user-specified resolution. By plotting the image results over time, the SDMS can provide a video-like representation about the spatiotemporal distribution of the underlying phenomenon.

Traditional spatial DBMSs process spatial window queries on stored data. In WSNs, however, the sensor data are acquired on demand from the sensors; the estimation results should be computed in real time. Due to the discrete node distribution in WSNs, sensor readings are point samples with regard to the underlying phenomenon. We need to execute additional estimation techniques to interpolate phenomenon values in-between sensor nodes. The challenge is to provide an energy-efficient, collaborative algorithm for the field estimation. The algorithm needs to run in the collaboration between sensor nodes, by exploiting the fact that the co-located values are spatially related.

1.5.2 Querying Phenomena as Objects

The second perspective focuses on continuous phenomena from an object perspective. Here, we are interested in the spatial properties of objects, such as the boundary of a sub-region of the continuous phenomena with similar characteristics (e.g., a wildfire region). Efficient algorithms are needed to detect and track the objects.

Objects in an object-based model must be identifiable and describable [WD04]. To apply an object-based model, an SDMS needs to identify objects from raw sensor readings. Since objects are often derived entities, they can be identified in different ways. For example in an SDMS, an object can be found using the combined results from different types of sensor readings [AML05]. An abstract object, fire, can be identified if the temperature reading is over $100°\mathrm{C}$ and the humidity reading is below 10%. Similarly, the fire could be identified based on the temperature reading alone, however, using different thresholds.

To understand the evolution of a complex object such as wildfires, we need to extract the boundary first, and provide further processing based on the boundary, such as the boundary tracking. A quantitative result from a field-based model, however, can hardly provide such abstract and complex spatial information. Furthermore, focusing on the object boundary can provide the opportunity to save energy resources.

The challenge here is that a WSN can only provide geo-referenced scalar values of underlying phenomenon. We need to provide efficient approaches to identify 2D objects and object boundaries in WSNs based on local sensor readings. Furthermore, a WSN needs to

efficiently track 2D objects over time, based on which an SDMS can extract spatiotemporal properties about the objects.

Foremost, users are interested in global spatial information, whereas constrained WSNs favor localized information processing. An efficient approach needs to return high-quality results about the underlying phenomena while still processing the information locally within a WSN. Thus, we seek for distributed approaches to process spatial information and queries in SDMSs. We also need collaborative algorithms of estimation techniques and information extraction techniques to generate spatial information and query results from raw sensor readings within a WSN. Our approaches must meet the resource requirements of the constrained environment. In general, our approaches need to keep a graceful balance between the WSN resource consumption and the result quality.

1.6 Contributions

This dissertation presents several approaches to process spatial queries and information over underlying phenomena in WSNs. We developed efficient approaches for both the field-based and object-based models.

1.6.1 SWOP

In a field-based model, an SDMS needs to return quantitative values to represent the estimated field function. We provide an efficient approach, *SWOP*, to support quantitative

spatial window queries. The SWOP approach returns the estimated spatial distribution of an underlying phenomenon within a continuous spatial window region at a user-specified resolution by using the Gaussian Kernel estimation. Here the window region is a user-defined rectangle. The SWOP approach first groups sensor nodes into sub-clusters according to node locations. Next, the SWOP approach transforms Gaussian weighted readings into the Hermite series for each cluster representing the detailed information about local sensor nodes. In this way, a large set of readings are represented by a small number of Hermite coefficients. The SWOP approach reduces the communication cost for node IDs and individual sensor readings. The total amount of data transmitted inside the network is reduced by logarithmic order, while the computation cost on individual nodes is kept constant. A microserver deployed in the network with more powerful resources evaluates the transformed data set to generate the final spatial window query result. Because of the fast convergence speed of the Hermite expansion, the estimation difference between SWOP and the traditional Gaussian Kernel estimation is small. As shown by our experimental results, the mean squared errors between the results of SWOP and the centralized approaches are as small as 10^{-4}. We have run SWOP over multiple data sets (real and synthetic data sets), and the experimental results demonstrate that SWOP reduces the communication cost by up to 90% compared with transmitting raw sensor readings to a central base station performing the "out of the network" estimation.

1.6.2 NED

In an object-based model, the object boundary provides useful spatial information to identify and track objects from underlying phenomena. The object boundary usually covers a small subregion of the monitored region. A WSN needs less resources by reporting an object boundary than by reporting an overall image-like result about the monitored region. To detect the local object boundary, a sensor node needs to encode local estimation results into digital messages and exchange messages among neighboring nodes. We propose NED, a distributed approach to detect object and object boundary in WSN. The NED approach uses a variable length encoding mechanism for the object and object boundaries detection results to reduce the communication cost. If a node detects a significant local object (or non-object) reading, the node uses a 2-bit message to encode the local estimation result. If a node is nearby the object boundary, the node tends to detect insignificant object readings and make erroneous estimation results. The NED approach allows the node with low confidence about the local estimation results to encode the local results as 33-bit messages. In this way, the nodes nearby an object boundary make high quality estimation results about the local object status. Based on the statistical models, the NED approach returns high-quality object detection results while reducing the wireless communication consumption. The NED approach provides the constrained WSN a flexible, efficient and high-quality approach for the in-network object and object-boundary detection. Our experimental results illustrate that NED's communication cost varies relatively to different sensing noise levels.

Our experiments show that the NED approach can provide the same quality of object and object boundary detection as achieved by moving-mean-based approaches. Also, the NED approach only uses the similar communication cost as required by majority-voting-based approaches.

1.6.3 Tracking Deformable Curves in WSN

To understand the spatiotemporal properties of a 2D object, we need to use a closed 2D curve to represent the object. In the proposed approach, we assume that an initial boundary is given to define the object, and we observe the deformation of that closed curve to represent and track the underlying 2D object. Sensor nodes only track individual vertices on the closed curve without knowing the global detailed geometric information about the objects. Messages are exchanged locally to deform the tracking curves. By tracking deformable curves, a WSN is able to produce spatiotemporal properties about a 2D object by the aggregated information. In this way, an SDMS does not need to return the detailed geometric representation of a 2D object, and saves energy. For example, we can use the aggregated information to represent the area and area change of a 2D object. In our approach, the deformable curves are breakable. Hence, the representative curves adapt their shapes to track multiple objects, and the topological changes involved. Compared to transmitting boundary points or boundary geometry from a WSN, our approach requires about 25% communication cost to maintain the tracking curve. Our simulation results prove that

our approach further reduces the communication consumption by providing abstract spatiotemporal properties about 2D objects. Reporting the spatiotemporal properties through the in-network aggregation needs even less communication cost than the tracking curve maintenance requires.

Our approaches are in-network distributed approaches in which sensor nodes process information locally and exchange information in a neighboring region. Our approaches require much smaller amount of wireless communication cost than transmitting raw sensor readings from a WSN. The difference between traditional centralized approaches and our approaches, on the other hand, is small. Therefore, our approaches can provide high quality query results.

1.7 Intended Audience

This dissertation is intended for researchers and developers interested in the design of WSN to model underlying phenomena and use SDMSs to process spatial and spatiotemporal information. The intended audience comprises designers and practitioners from the fields of computer science, DBMS, WSN and GIS. This dissertation provides approaches to extract and represent spatial and spatiotemporal information from WSN. This dissertation should be particularly interesting to the designers of next-generation spatial software who plan to develop models and approaches to support real-time observations of the physical world.

1.8 Organization of Remaining Chapters

The following parts of this dissertation start with a background review in Chapter 2, which gives an overview on the state of art about the research related to spatial query processing in SDMSs. In Chapter 3, we present our approach to support spatial window queries in SDMSs. An efficient approach to extract object boundaries is explained and discussed in Chapter 4. Our approach on tracking deformable 2D objects is described in Chapter 5. Chapter 6 presents and analyzes our experiment results. We make the conclusion and propose future plans in Chapter 7.

Chapter 2

BACKGROUND AND RELATED WORK

Since the wireless communication is the most significant energy drain in WSNs, the main research objectives in SDMSs focus on minimizing the communication. The network infrastructure of WSN is an important component to support the distributed query and information processing in SDMSs. Similar to the Internet, today's WSN is based on a simplified OSI reference model [Zim88]. IEEE 802.15.4 and ZigBee provide the standard protocols for the hardware-dependent layers [IEE06, GNC+01, Zig06, Kin04].The network, transport and application layers are often collapsed into one layer in current WSNs to optimize in-network processing as well as communication. A simplified layer model can help WSNs to avoid excessive headers added through different layers. Combining routing protocols with the application-level query processing in SDMSs can help to avoid unnecessary communication cost and achieve high efficiency. Thus, communication strategies and information

processing strategies need to be co-optimized. Many routing protocols are designed for particular applications and network topologies [ASSC02, YF04, KK05].

Designing novel communication protocols is beyond the scope of this dissertation. Although we need to select a matching communication protocol to reduce the wireless communication consumption, our focus is on designing an efficient data collection and processing technique in SDMSs to provide high quality results to users. In the remaining parts of this chapter, we will explore various intelligent spatial data collection and information processing approaches. We will analyze the energy cost and result quality of these approaches.

2.1 Processing Quantitative Spatial Window Queries

In the remainder of this dissertation, the term "quantitative spatial window queries over underlying phenomena" is shortened to the term *spatial window queries*. As mentioned before, quantitative queries with regard to continuous phenomena have the objective to estimate the value distribution of the spatial field in the region specified by the query predicate, usually a window or a point. The following estimation methods have so far been applied to answer quantitative spatial queries in SDMSs. We will analyze the underlying estimation methods, the estimation quality of query results and the execution cost, especially the communication cost, of the distributed implementations.

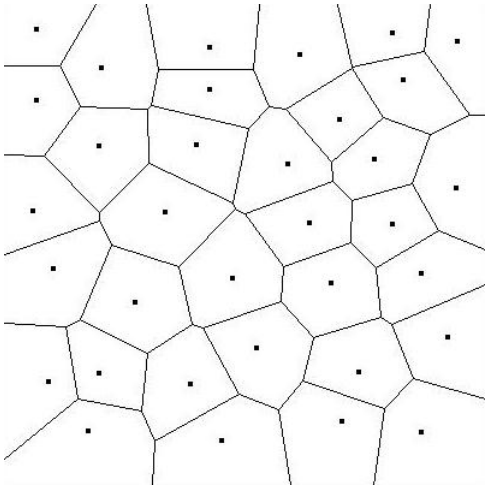


Figure 2.1. An example of a Voronoi diagram

2.1.1 Using Voronoi Diagrams and TIN

Among various estimation techniques, Voronoi-diagram-based techniques are one of the simplest estimation models, and provide a rather coarse estimation result. Nevertheless, this method takes the spatial distribution of available sensor nodes into account when estimating a spatial point or window query. A Voronoi diagram partitions the monitored region into a set of "Voronoi cells" based on the location of sensor nodes, as shown by Figure 2.1. The phenomenon values in a cell are represented by the sensor reading at the cell center. For example, a spatial average query result can be represented as a weighted sum of sensor readings according to the size of Voronoi cells [GHS03, SS04].

By using this estimation technique to answer a spatial window query, an SDMS needs to build the Voronoi diagram and find the Voronoi cells that intersect with the spatial window region. Traditionally, a Voronoi diagram is found through the sweeping algorithm based on the centralized availability of a set of point locations [BKOS00]. This approach works well when sensor nodes are static and no sensor readings are lost. Sharifzadeh et al. proposed a distributed algorithm to build the Voronoi diagram [SS04]. The basic idea of this distributed algorithm is to let a sensor node begin the Voronoi diagram construction with a partial Voronoi diagram based on the location of neighboring nodes. By exchanging the locations of newly found nearby nodes, the sensor node can polish the partial diagram to integrate the new point locations. Through iterations of message exchanging, distributed sensor nodes can find the final Voronoi diagram in a WSN. Each sensor node can find the shape of its local Voronoi cell. Harrington et al. improved this distributed Voronoi diagram construction algorithm by enhancing the message exchange procedure [HH05]. A distributed Voronoi-diagram construction algorithm is useful for mobile WSNs, but needs additional communication cost.

Suppose n ($n > 3$) sensor nodes are located within the window region. The upper bound of the number of edges to represent the Voronoi diagram is $3n - 6$, and the lower bound is $n - 1$ [AK00]. To estimate a spatial window query result using the Voronoi diagram, an SDMS needs a linear amount of communication cost with respect to the number of sensor nodes involved. Apparently, without further optimization, the cost of processing

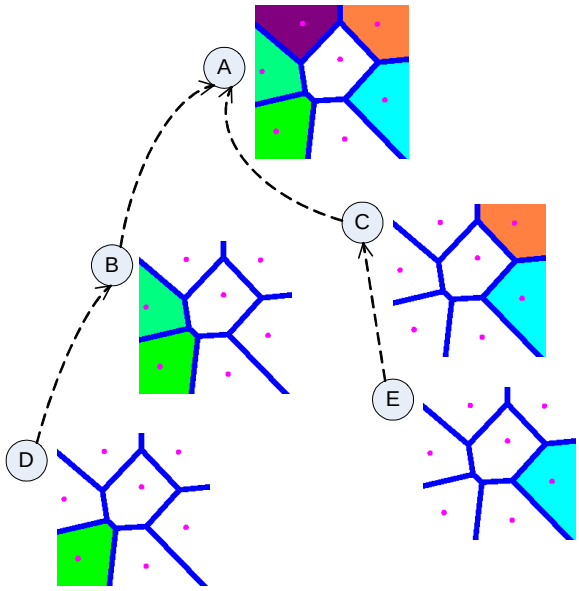


Figure 2.2. Processing Voronoi-diagram-based spatial window queries in TAG

Voronoi-diagram-based spatial window queries would be equivalent to the cost of collecting raw sensor readings to a central base station.

Based on the TAG framework, an SDMS can aggregate different Voronoi cells with sensor readings into a complete Voronoi-diagram-based representation about the underlying phenomenon in a WSN, as shown by Figure 2.2. To reduce the communication cost, Hellerstein et al. proposed to simplify the shape of Voronoi cells [HHMS03]. First, in [HHMS03], the space is partitioned into a set of regular cells. Due to the spatiotemporal continuity of underlying phenomenon, nearby grid cells may contain similar sensor readings. These cells are named as *isobars* in [HHMS03]. During the in-network aggregation,

a node can merge neighboring isobars into a larger continuous isobar. To save communication cost, the number of edges to represent the isobar shape can be reduced during the in-network aggregation. The shape simplification usually causes lossy results, and may neglect some readings from the WSN.

In [SS04], Sharifzadeh et al. also used a *Triangulated Irregular Network* (TIN) based on the Voronoi diagram to represent the underlying phenomenon distribution as a 3D terrain. To calculate and represent a TIN, however, a WSN still needs to consume at least the same amount of resources as required by the Voronoi-diagram-based approach. Harrington et al. presented an in-network surface simplification for TIN-based approaches. Similar to the approach in [HHMS03], the approach in [HH05] chooses to simplify the 3D terrain surface. In [HH05], sensor nodes can calculate the error introduced by removing the local readings from the final TIN surface. Based on locally calculated error values, sensor nodes use a randomized procedure to determine whether the local readings are integrated into the global TIN terrain.

Overall, Voronoi-diagram-based approaches need an expensive algorithm to find and represent the Voronoi cells, especially with respect to the in-network communication consumption. However, as long as the sensor nodes are stationary, the cells need to be defined only once. Over the application lifetime of a WSN, nodes and communication links may fail, and the Voronoi diagram must be recalculated. The second portion of the cost is defined by finding the intersection between the query predicate and the appropriate Voronoi

cells. Since Voronoi-diagram-based models are one of the simplest estimation models, the results based on the Voronoi diagram are rather coarse. To reduce the communication cost, the shape of Voronoi cells needs to be simplified by merging neighboring cells containing similar readings. The in-network shape simplification approaches, however, cause lossy query results and degrade the quality of query results.

2.1.2 Using Spatial Regression Methods

Spatial regression models the underlying phenomenon as a function. The solution of regression can be represented as,

$$\hat{Y}(q) = \sum_{i=1}^{k} [w_i \cdot f_i(q)], \tag{2.1}$$

where the estimated value for any point in the monitored region is a weighted sum of predefined basis functions, $f()$s. In a 2D polynomial regression, similar to the 1D polynomial temporal regression [DKR04], $f()$s are polynomial functions of x and y coordinates. For instance, an underlying phenomenon can be represented by a quadratic polynomial function, $\hat{Y}(q) = w_0 + w_1 \cdot x_q + w_2 \cdot y_q + w_3 \cdot x_q^2 + w_4 \cdot y_q^2 + w_5 \cdot x_q y_q$, where x_q and y_q represent the x and y coordinates of point q. Another form of spatial regression is known as the Kernel regression, where the basis functions are a set of kernel functions, $k()$s [GBT+04]. Each kernel function has a unique kernel center in space. A kernel function is typically a predefined non-increasing function of the Euclidian distance between the kernel center and the input point. Given k basis functions $f()$s and n sensor readings, to minimize the *Mean*

Squared Error (MSE), the weights for basis functions are computed by,

$$W = \left(\sum_{i=1}^{n} (F(s_i)^T F(s_i)) \right)^{-1} \sum_{i=1}^{n} (F(s_i)^T Y(s_i)), \tag{2.2}$$

where W and $F()$ are the vector format of w and $f()$ (i.e., $W = [w_1, w_2, \cdots, w_k]^T$, and $F(s_i) = [f_1(s_i), f_2(s_i), \cdots, f_k(s_i)]$).

If the basis functions are given, the estimated weights, W, are sufficient for an SDMS to represent the underlying phenomenon. Compared with transmitting raw readings, sending the estimated weights consumes much less communication and consequently less energy from a WSN. To compute the weight values, however, the WSN requires additional communication and computation cost.

Guestrin et al. applied a distributed implementation of the Gaussian elimination to solve the linear equations in [GBT+04]. The Gaussian elimination needs to convert the full matrix, $\sum_{i=1}^{n} (F(s_i)^T F(s_i))$, into a triangular matrix by subtracting equality constraints from each other appropriately. In the distributed implementation, sensor nodes maintain two local partial matrixes based on $\sum_{i=1}^{n} (F(s_i)^T F(s_i))$ and $\sum_{i=1}^{n} (F(s_i)^T Y(s_i))$. Sensor nodes begin the Gaussian elimination with an initial guess about the value of W. Sensor nodes then exchange local results and apply the neighbors' results to get a more accurate value of W. Typically, the computation and message exchange need to run several iterations to achieve a satisfactory error tolerance. Delouille et al. presented the *Embedded Polygons Algorithm* (EPA) to solve the general matrix inversion for WSNs [DNB04]. EPA can be used to compute the value of the weight matrix W. In EPA, sensor nodes are connected into

a number of sub node sets. The node sets are also represented geographically as polygons in [DNB04]. Different from the approach in [GBT+04], where each node exchanges messages with neighbors over individual matrix elements, polygon node sets exchanges messages iteratively over matrix blocks in EPA. A small amount of communication cost is required to represent the value of W. Computing the value of W in a WSN, however, is still expensive with regard to the communication cost [GBT+04].

Another way to find the estimated W is to aggregate the two matrices, $\sum_{i=1}^{n}(F(s_i)^T F(s_i))$, and $\sum_{i=1}^{n}(F^T Y(s_i))$ within a WSN. The base station then computes the value of W outside the network. For a more complex spatial phenomenon distribution, more basis functions are required to increase the estimation quality. The communication cost increases exponentially for more basis functions, since each node needs $(k^2 + k) * k_i$ data to aggregate for k basis functions. Here, k_i represents the data length for one element in the matrix. Several types of kernels, such as the Block kernel or Cone kernel chosen by [GBT+04], in the Kernel regression can relax the communication cost for individual nodes to represent the partial matrix. The quality of estimation result, however, is deteriorated due to the discontinuity of these kernel functions. The choice of kernel centers is another relevant issue, since different kernel centers affect the estimation quality.

Although a WSN needs a small amount of communication cost to represent the weights, W, the communication cost to find the solution of W is still expensive in a spatial-regression-based approach. If a WSN is monitoring a dynamic phenomenon at a high temporal rate, a spatial-regression-based approach faces more difficulties.

2.1.3 Kriging

The Kriging model is named after D.G. Krige, a South African mining geologist, who developed the preliminary version [BG96]. The Kriging model is built based on the variogram model, $\gamma()$, a function measuring the covariance between two points in space. If the underlying variogram model is accurate, the results of Kriging are supposed to be the "best" estimation results with regard to the estimation errors. Another advantage of Kriging is that the estimation results comprise two parts: the estimated value of the unknown point p, $\hat{Y}(p)$, and the estimated standard error, S_z.

The underlying phenomena are usually assumed stationary and isotropic here. In other words, the covariance between any two points only depends on the distance between them, not on the direction in which they are separated nor the location where they are located. There are several Kriging variations based on different assumptions about the underlying phenomena. The ordinary Kriging requires an unknown constant *mean* existing over the whole region. The estimated value for the query point p by the ordinary Kriging is the

weighted sum of available samples,

$$\hat{Y}(q) = \sum_{i=1}^{n} w_i Y(s_i). \tag{2.3}$$

The estimated standard error S_z is given by

$$S_z = \sqrt{2\sum_{i=1}^{n} w_i \gamma(|s_i - q|) - \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \gamma(|s_i - s_j|)}. \tag{2.4}$$

with the constraint equation

$$\sum_{i=1}^{n} w_i = 1 \tag{2.5}$$

to maintain the unbiased estimation result. To minimize the estimation error, the ordinary Kriging applies the Lagrange multiplier to get $n + 1$ linear equations to get the root of weights,

$$V_+ W_+ = v_+(q) \tag{2.6}$$

where

$$V_+ = \begin{bmatrix} \gamma(|s_1 - s_1|) & \cdots & \gamma(|s_1 - s_n|) & 1 \\ \gamma(|s_2 - s_1|) & \cdots & \gamma(|s_2 - s_n|) & 1 \\ & \ddots & & \\ \gamma(|s_n - s_1|) & \cdots & \gamma(|s_n - s_n|) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix}, W_+ = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ \lambda \end{bmatrix} \text{ and } v_+(q) = \begin{bmatrix} \gamma(|s_1 - q|) \\ \gamma(|s_2 - q|) \\ \vdots \\ \gamma(|s_n - q|) \\ 1 \end{bmatrix}.$$

Here, the λ is the Lagrange parameter. It can be easily found that the standard error of the estimation can also be defined as,

$$S_z = \sqrt{\sum_{i=1}^{n} w_i \gamma(|s_i - q|) + \lambda}.$$

Theoretically, the estimation error is assumed to be zero-mean Gaussian values, $N(0, S_z)$. Thus, a typical estimation result of Kriging is $\hat{Y}(q) \pm 2S_z$, with 95% confidence that the real phenomenon value falls inside the interval.

In [UKT08], Umer et al. presented a quad-tree-based approach, *Quad Suppress* (QS), to find the appropriate variogram model. In QS, a quad tree is found based on a predetermined grid resolution. In each quad cell, a node is chosen as the cell head. The covariance value among the neighboring sensor readings can be computed using the in-network aggregation over the quad tree. By collecting the covariance values, the base station can find the experimental variogram model. Based on the experimental variogram model, a distributed matrix inversion algorithm is used to solve the linear Kriging equation, Equation 2.6, for any individual point in the monitored region [UKT08].

The computation and communication cost of Kriging is expensive even for a single point evaluation. To perform an evaluation over a spatial window, the cost of Kriging is much more expensive. The estimation of the variogram model also requires an additional cost. Although the Kriging results are fine grained and high quality, the evaluation cost of Kriging limits the application of Kriging in the constrained WSN.

2.1.4 Discussion

Besides the estimation techniques discussed above, there are many other established estimation methods, which can be applied to estimate the phenomenon distribution based on

the point sensor readings. Due to the computation and communication complexity, many methods cannot be applied for spatial window queries in SDMSs directly, but have to be adapted and tied to an energy-efficient in-network, collaborative execution strategy. The domain of processing spatial window queries is still limited today, and at an early stage. Only simple approaches have been proposed so far. The approaches discussed above have the same objective that the estimation results should be good quality while the in-network resource consumption needs to be reduced.

We propose SWOP for spatial window queries in SDMSs. SWOP is based on the Gaussian Kernel estimation, which is generally used to generate smooth estimation results based on point samples. SWOP does not require expensive communication cost to either return raw sensor readings or compute the estimation results in a WSN. Instead, SWOP transforms raw sensor readings and the geo-reference information into a small number of Hermite coefficients. In this way, SWOP keeps a graceful balance between the estimation quality and the resource consumption (especially the communication cost).

2.2 Processing 2D Object-based Queries

Object-based models are interested in the object detection and tracking. In most cases, the object boundary needs to be detected to represent and track objects. Some SDMS research has been done with regard to the in-network object and object boundary detection. Due to the discrete node distribution, the result of object boundary detection is a point set around

the object boundary. Several approaches have also been proposed to link the boundary points into closed curves and to track the change of the geometric shape over time.

2.2.1 Object and Object Boundary Detection

We define an object based on user-specified thresholds, as shown by Equation 1.3. Therefore, the threshold value is used to derive the object boundary. In a naive approach, a sensor node can compare its local sensor readings with the threshold value to estimate the local boundary status, as illustrated by Equation 1.4. If merely based on Equation 1.4, however, sensor nodes may fail to report the boundary location, since sensor nodes are discretely distributed. Sensor nodes may not be located on the exact object boundary. Therefore, a node must compare its local sensor reading with neighbors' readings for better boundary estimation results.

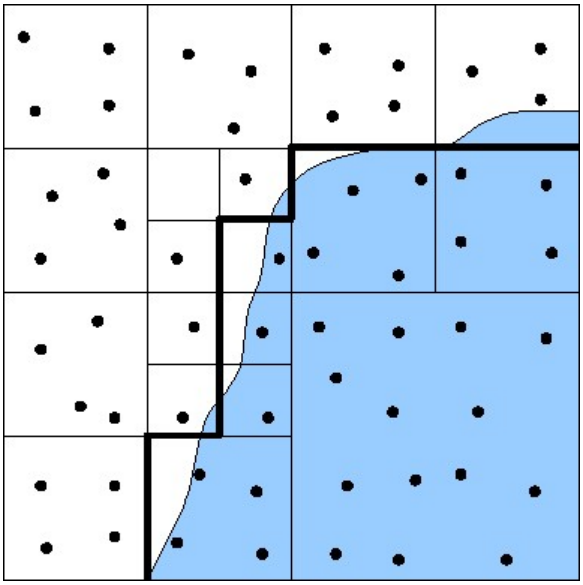


Figure 2.3. Detecting object boundary based on quad tree

Nowak et al. proposed a quad-tree-based approach to find the object boundary [NM03]. A quad tree represents a spatial region as a tree where each internal node has zero or four children, as illustrated by Figure 2.3. The variance value of local boundary estimation results, based on Equation 1.4, within a quad tree cell can be used to control the quad tree structure. If the variance value in a quad tree cell is large, the cell needs to be divided into four smaller cells. In this way, the size of quad tree cells around the object boundary is smaller than other cells. In Figure 2.3, the small cells are connected by the thick lines to represent the boundary. A WSN needs several iterations to collect sensor readings, compute the variance and perform the splitting or merging operations on the quad tree. The small-sized cells can be used to represent the object boundary, whereas additional communication cost is required to maintain the quad-tree structure.

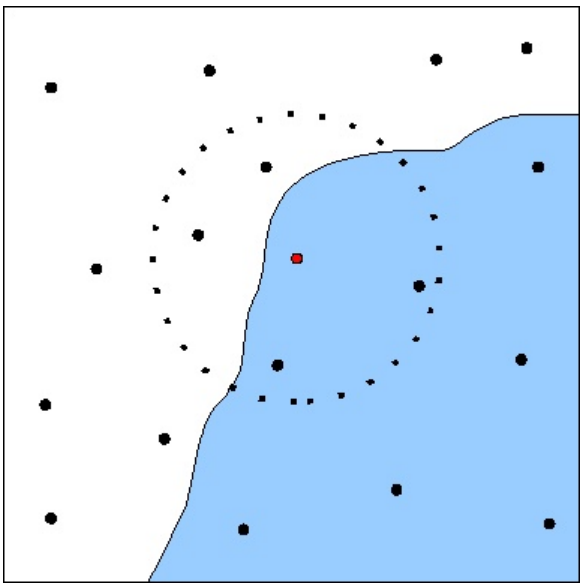


Figure 2.4. Detecting object boundary based on neighboring readings

Several approaches use statistical models to detect the object and object boundary based on local and neighboring sensor readings. For example, in Figure 2.4, the red node can collect its neighbors' readings within the communication range and do the computation locally.

Krishnamachari et al. applied a Bayesian model to estimate the local object status [KI04]. In this approach, sensor nodes encode the local object status estimation by 1-bit boolean value (e.g., 1 for object interior and 0 for object exterior). By considering the possible faulty readings, a sensor node uses the local majority voting result based on neighbors' object reports to determine its local object status.

Based on another statistical model, Ding et al. proposed an approach to detect the object boundary in WSNs [DCXC05]. In this approach, sensor nodes exchange local sensor readings. After receiving neighboring sensor readings, a node uses the median value to represent the local phenomenon value. Based on the hypothesis that the underlying phenomenon values at an object interior are similar to a user-defined value, sensor nodes can use the estimated phenomenon values to determine the local object status. The boundary detection is built on another hypothesis that the underlying phenomenon values on an object boundary should be at the middle of object phenomenon readings and non-object phenomenon readings. By using the statistical models, significant object and object boundary estimation results can be found.

In addition to the statistical models, Chintalapudi et al. presented two additional techniques, an image processing approach and a classifier-based model, for object boundary detection in WSNs [CG03]. The image processing approach uses the Prewitt (difference) filter to compute the gradients of object status along x and y directions. Similar to the local variance values in the quad-tree-based approach, the gradient values nearby an object boundary are usually larger than the gradients at other regions. In this way, the sensor nodes detecting large gradient values can report the local object boundary. The classifier-based approach is based on the assumption that the sensor readings inside object interior are significantly different from the readings from the object exterior. Sensor nodes around the object boundary can therefore partition the neighboring readings into two groups (i.e., object and non-object estimation results). Consequently, the nodes that can find the two groups report the object boundary.

Duckham et al. proposed a conceptual framework to monitor underlying phenomena based on the qualitative properties of 2D objects [DNW05]. In this work, quantitative sensor readings (e.g., real numbers) are converted into qualitative values (e.g., object and non-object status; integer numbers). Sensor nodes are partitioned into two groups (i.e., active and inactive sensor nodes). Active nodes are connected by a triangulation, and exchange messages among each other. A node can activate itself either by detecting a significant local reading change, or receiving a request from a neighboring node. After exchanging messages, active nodes can deactivate themselves or wake up inactive nodes. In such a way,

more nodes are active around the object boundary than the nodes in other regions. Nodes located on the object boundary can consequently report the boundary detection results. Inactive nodes are in the sleep mode to save energy.

In many applications, a single type of sensor may not well represent the underlying object status. For these cases, combining results among multiple types of sensor readings is useful to detect the object and object boundary. For example, a fire object can be defined by the combination of hot temperature readings and low humidity readings. Abadi et al. proposed the *Robust, Efficient Filtering and Event Detection* (REED) to process the object detection based on different types of sensor readings [AML05]. In REED, one type of sensor reading is contained by one sensor table. REED proposes efficient algorithms to optimize the join operations among different sensor tables. The core idea in REED is to minimize the partial join results by reordering the join operations along the hierarchical in-network processing.

Compared to the quad-tree-based approach, sensor nodes can perform the boundary detection only based on neighboring readings by using these approaches, as shown by Figure 2.4. Sensor nodes around the object boundary can prepare the detection results locally. Due to the discrete distribution of sensor nodes, however, the boundary reports can only be points around the 2D object boundary.

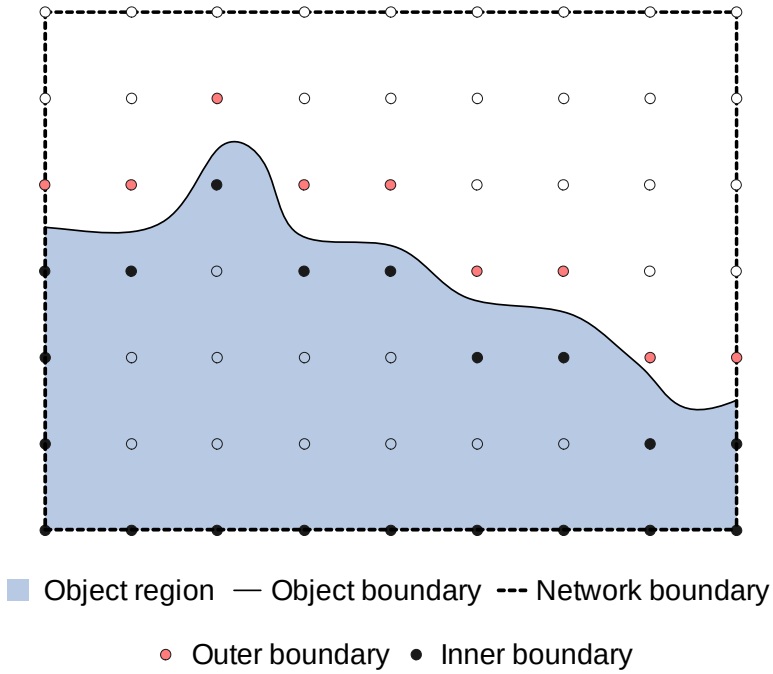


Figure 2.5. Detecting object and object boundary by WSN

2.2.2 Boundary Geometry Formation

From the in-network boundary detection, a set of points can be found around the object boundary. A point set, however, may represent different spatial regions [GD06]. To well represent a 2D object, linking the points along an object boundary is necessary.

In the first type of approach, a WSN reports the boundary points to a central base station. Based on the collected boundary points, the base station can run traditional centralized approaches to find the appropriate geometry for the object boundary. Figure 2.5 illustrates the underlying object and the boundary points found in WSNs. Apparently, the point boundary reports can be simplified as the inner boundary reports (e.g., the black dots

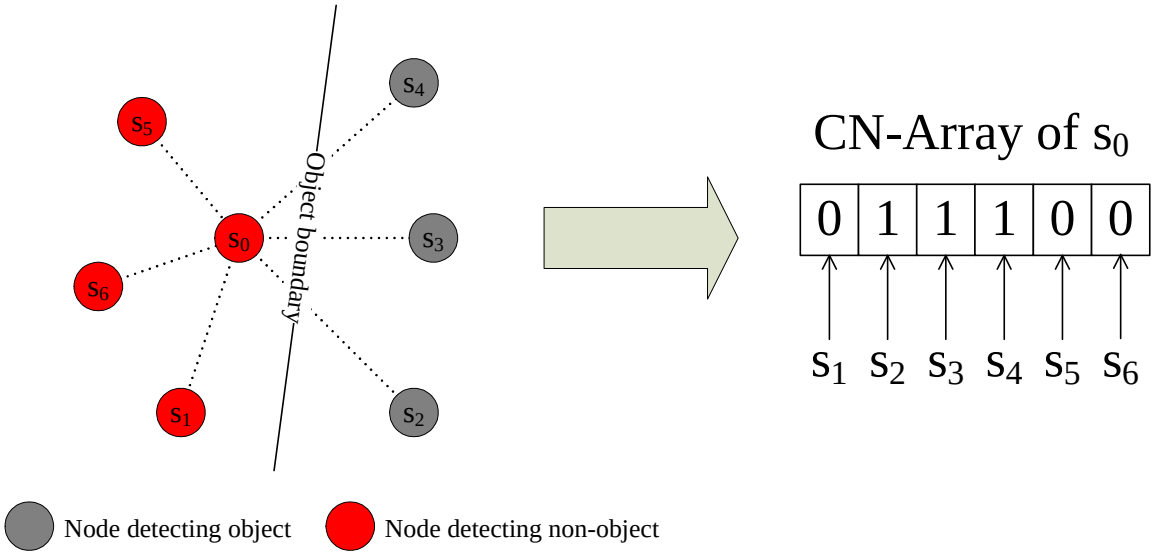


Figure 2.6. A CN-Array representation

in Figure 2.5) or the outer boundary reports (e.g., the red dots in Figure 2.5), which is the basic idea presented by [SO05]. In this way, about half of the nodes, which detect the object boundary, can hold their boundary reports from transmitting back to the base station.

Meng et al. used the spatial and temporal suppression to reduce the boundary point results [MLNL04]. The spatial suppression is similar to the example shown above, in which only either inner or outer boundary reports are transmitted back to the base station. By using the temporal regression, a node can suppress a local boundary report if the node has reported a boundary result a small interval ago. Since the underlying phenomena are usually spatiotemporally continuous, the suppression approaches can reduce the wireless communication cost.

Zhong et al. presented the *Contour Neighbor Array* (CN-Array), which is an efficient way to encode the boundary point results [ZW08]. The CN-Array approach assumes that the WSN is static; a central base station knows the location of sensor nodes and the neighborhood topology. As shown by Figure 2.6, each bit in a CN-Array represents a neighbor and the bit value indicates the neighbor's local object detection result. Except for representing a boundary point result, a CN-Array can reveal the direction of how the object boundary goes through the neighborhood. By using CN-Array, more than half of nodes, which detect the object boundary, can suppress their boundary reports. After receiving CN-Arrays from the WSN, a base station can better generate the final object boundary.

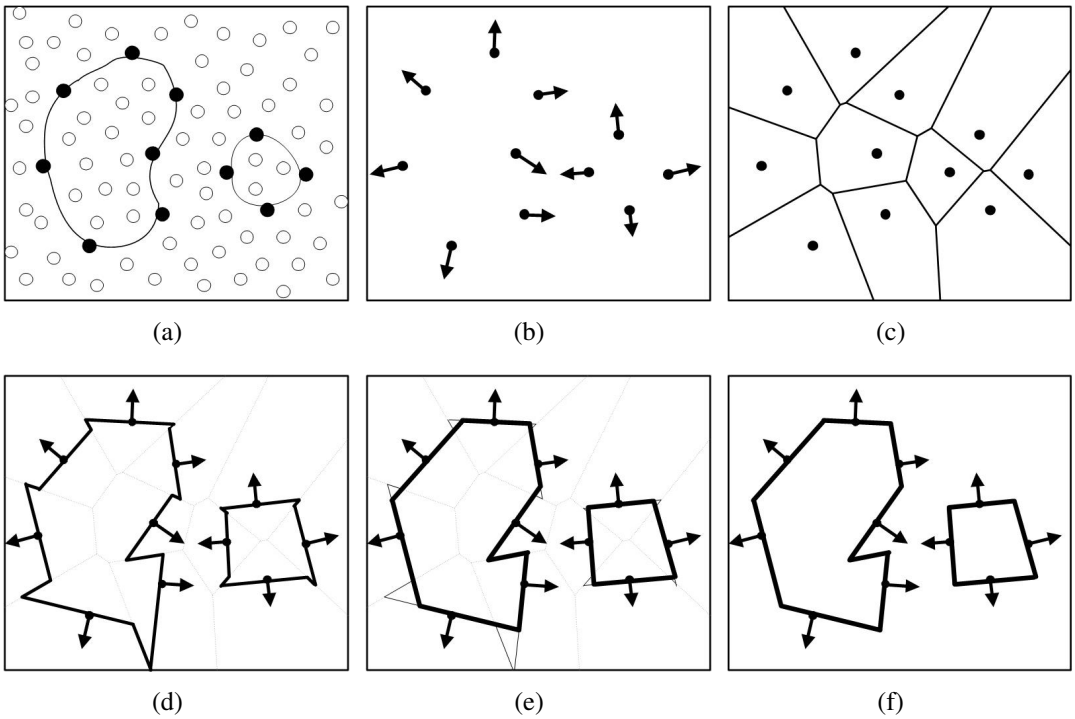


Figure 2.7. Linking boundary points based on Voronoi diagram

As shown by the CN-Array example, the boundary gradient information can be used to link the points. Liu et al. proposed an in-network approach based on the Voronoi diagram [LL07]. By comparing the neighboring nodes' local object and object boundary estimation results, a sensor node can find the direction of how the object boundary goes across the local Voronoi cell, as illustrated by Figure 2.7. In this way, a local boundary point can be extended to a directed partial line. By connecting the partial boundary lines, a WSN can find closed curves to represent 2D objects, as shown by Figure 2.7.

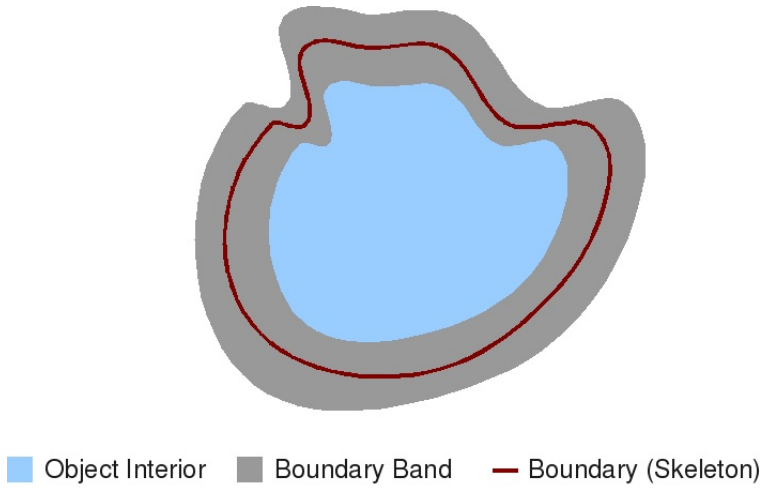


Figure 2.8. Detecting object boundary by using skeleton

Alternatively, Zhu et al. chose to use the *Skeleton* to link the boundary points [ZSGM08]. In the field of computer vision, the skeleton of a region is a set of connected lines to represent the topological information of the region. To find a skeleton to represent the object boundary, the approach in [ZSGM08] first finds a band region covered by the points along

an object boundary. Afterward, the skeleton of the boundary band region can be extracted to represent the object boundary, as shown by Figure 2.8.

To save communication, the shape of the object boundary can also be simplified in WSNs. Given a curve composed of line segments, the curve simplification algorithms need to find a similar curve that contains fewer points. Zhong applied the Douglas-Peucker algorithm to simplify the object boundary in WSNs [Zho08]. The original Douglas-Peucker algorithm is a recursive algorithm [DP73]. The algorithm connects the first line segment by connecting the start and end points of the original curve. For each line segment, the Douglas-Peucker algorithm first scans the points on the original curve that are located in-between the line segment. If the distance between all points and the line segment is smaller than the error threshold, ε, the line segment is kept. Otherwise, the furthest point away from the line segment in the point set is chosen to break the line segment into two line segments, as shown by Figure 2.9. In [Zho08], a binary tree is built to process the pair-wise comparison over the line segments. The original object boundary shape, therefore, can be simplified through the in-network pair-wise comparison. Gandhi et al. used a similar approach, named *stick-fitting*, to simplify the boundary shape [GHS07]. Additionally, the topology consistency between the original curve and the simplified curve is maintained by the *untangling* in [GHS07].

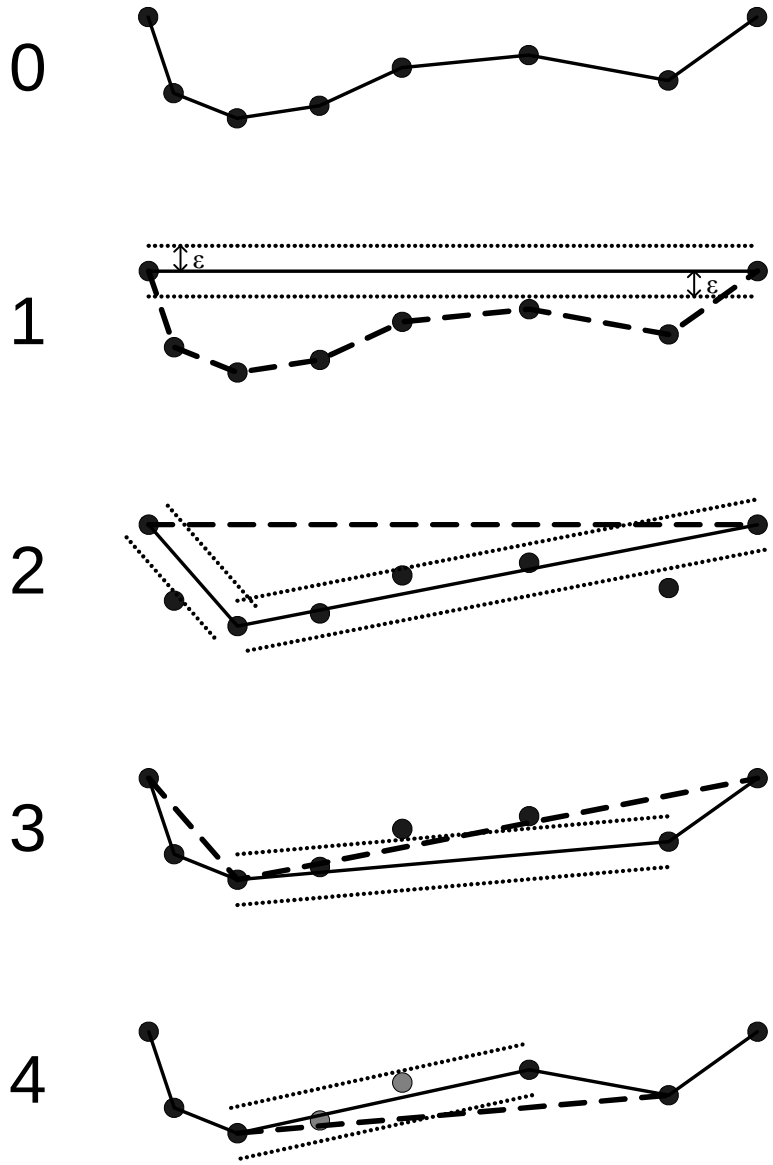


Figure 2.9. Douglas-Peucker Algorithm

2.2.3 Boundary Change Detection and Tracking

After identifying the initial geometry to represent object boundary, we need to provide an efficient algorithm to track 2D objects over time. There are different options. First, we can track the changes of the geometric representation of an object boundary over time. Secondly, we can also derive more abstract and complex spatiotemporal properties using the geometric representation. For example, mentioning the example from Chapter 1 again, users might be interested in the qualitative properties of whether a wildfire is enlarging in area or the fire is moving towards the north.

Overall, we are looking for the most energy-efficient option to report those changes over time. One alternative is to continuously compute a new, updated boundary in each time step, and report the boundary geometry to the base station. The approaches discussed in Section 2.2.2 are classified into this category. The second alternative is to find an efficient algorithm to compute either the incremental changes of the geometric shape of the object instead of computing the entire boundary from scratch in each or to identify the abstract spatiotemporal properties (like splitting, merging, etc) and report these properties.

Jiang et al. presented an in-network approach to detect topological changes involved among multiple 2D objects [JW08]. When 2D objects change their shapes and locations over time, local sensor nodes can detect the change of local object status. The region in which the local object status changes between two consecutive time slots is called a *transition* region in [JW08]. The approach presented by [JW08] is based on detecting the

C-components, which are adjacent to the transition region. Sensor nodes are clustered into groups in this approach. After collecting different types of C-components detected by neighboring nodes, a cluster-head node computes the topological changes by comparing the C-component relationships, and reports the topological changes to a base station

Xue et al. proposed an efficient approach to detect different types of shapes [XLCL06]. In this approach, the monitored region is partitioned into regular grid cells. Sensor nodes merge neighboring cells by comparing the sensor readings in cells. By comparing the relation between different rectangle cells, the in-network aggregation can also return the shape-matching results. Based on a simple rectangle boundary representation, an emerging object can be identified through the shape matching. For example, the pyramid shape is used to identify the emergence of a gas leak. This approach can also be extended to detect shape changes over time.

In this dissertation, we propose an approach based on the incremental tracking of the geometric shape of object boundary based on the deformable curve model. The deformable curve model, also known as the SNAKE model, is used to identify objects and object boundaries in digital images [KWT88]. For example, the deformable curves can identify reconstructed roads from remote sensing images [ASG01]. In SNAKE, a closed curve is used to represent the object boundary. Based on the representative curves, our approach supports the in-network derivation of spatiotemporal properties of 2D objects.

2.2.4 Discussion

An SDMS requires running several steps to support object-based models. The first step is the 2D object and object boundary detection. The second step is the geometry generation based on the boundary point reports. The third step is the 2D object tracking. The next step is the abstract property extraction. All of these steps need to meet the requirements of constrained WSN.

The object and object boundary detection algorithms provide us the foundation to derive 2D objects from underlying phenomena. Connecting the points around an object boundary provides us a geometric representation about the 2D objects. Except for the object-based models, the object boundary is also useful for other issues in WSN. For example, the boundary location can help sensor nodes to suppress their reading reports, since sensor readings around an object boundary are often similar to each other [SBY06].

The proposed NED approach is designed based on statistical models for the efficient in-network object and object boundary detection. In NED, we focus on reducing the neighboring message exchange for the object boundary detection. In practice, sensor readings are often affected by noise. Nodes located around an object boundary tend to make faulty object detection results. In NED, the nodes around object boundary exchange more messages to increase estimation quality while other nodes communicate less to save energy.

Although the SNAKE model is widely used in the field of computer vision, no literature has applied it in WSNs. We found that the SNAKE model is an appropriate model for incremental boundary tracking in the energy and resource constrained WSN. A closed curve is defined by a set of vertices connected with edges in the SNAKE model. Sensor nodes are able to track individual vertices and therefore adjust the curve, without knowing the global shape of the curve. Tracking the geometric changes of object boundary over time allows us to compute abstract and complex spatiotemporal properties about 2D objects.

2.3 Chapter Summary

This chapter presents related approaches for spatial query processing in SDMSs. For field-based models, current approaches have applied simple estimation methods to generate the estimated field results based on raw sensor readings. Most approaches, however, have either limited estimation quality or expensive in-network resource consumption. The SWOP approach is proposed to overcome these drawbacks. We demonstrate that the SWOP approach provides better estimation results and requires less communication cost. For object-based models, most related approaches focus on efficient object and object boundary detection. The NED approach is also proposed for this purpose. Few literatures, however, are available for geometry-based tracking and property extraction of 2D objects in WSNs. We propose an in-network approach to track 2D objects based on the SNAKE model. In addition, efficient property extraction algorithms are also presented by this dissertation.

Starting from the next chapter, we present our approaches to process spatial information and queries in SDMSs. The remaining parts of this dissertation present the detailed algorithms of our approaches, and how our approaches benefit the constrained WSN.

Chapter 3

A QUANTITATIVE WAY TO PROCESS SPATIAL WINDOW QUERIES

In this chapter, we present *Spatial Window Query Over Phenomena* (SWOP), a quantitative approach to process spatial window queries in WSNs. SWOP focuses on estimating the spatial distribution of an underlying phenomenon within the user-defined spatial window, as shown by Figure 3.1. SWOP is based on the Gaussian Kernel estimation to interpolate the sensor readings within the region of interest.

3.1 Kernel Estimation

Different from Kernel regression, Kernel estimation is a nonparametric estimation and can be stated as "total amount of observed values per unit area [BG96]". Kernel estimation is also a spatial moving-average method. The estimation result is robust against the sensing noise. For a point, q, in the monitored region, $\mathbb{M}$, Kernel estimation treats the phenomenon

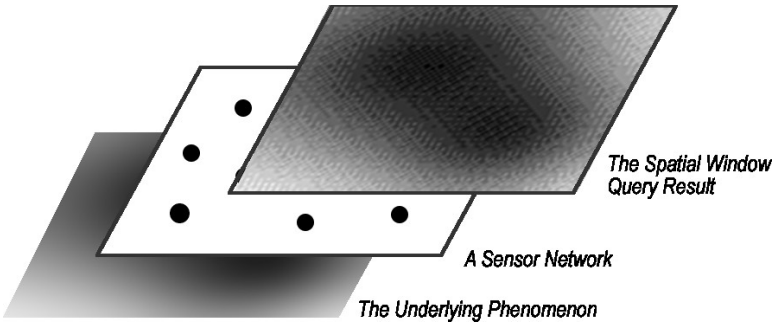


Figure 3.1. Spatial window queries

value at that point as,

$$\hat{Y}(q) = \frac{1}{\tau^2} \sum_{i=1}^{n} \mathbb{Y}(s_i) K\left(\frac{|s_i - q|}{\tau}\right), \text{where } s, q \in \mathbb{M}. \tag{3.1}$$

In Equation 3.1, $\mathbb{Y}(s_i)$ represents the reading of sensor node s_i;$\hat{Y}(q)$ is the estimation result for the point q; $|s_i - q|$ indicates the Euclidian distance between the point q and the sensor node s_i; τ is the band-width (also called the smoothing parameter).

A possible approach to evaluate Kernel estimation in a WSN can apply a simple distributed algorithm, in which every sensor node evaluates its nearby region in $\mathbb{M}$ [Rac02, HHMS03]. In a routing tree based protocol, each node aggregates its local result with the partial results from its children [HHMS03, XLCL06]. In such an approach, the size of total data extracted from a WSN is linearly scaled by the number of points to represent the Kernel estimation result. To answer a high resolution result, we need more estimation points than the number of involved sensor nodes. A simple distributed approach often makes no

significant improvement over the approach based on centrally collected raw sensor readings, especially if the raw readings are compressed in WSNs.

3.2 Gaussian Kernel and Fast Transforms

The main difficulty of applying Kernel estimation in WSNs is the entangled links between estimation points and sensor readings. Neither directly evaluating sensor readings outside the network nor directly evaluating estimation points within the network is resource-efficient. SWOP chooses another way to process the Kernel estimation based on the Gaussian kernel. The Gaussian Kernel estimation has a wide range of applications, such as financial analysis [BY03] and image processing [YDGD03], and estimates the phenomenon value at the point q as,

$$\hat{Y}(q) = \frac{1}{\tau^2} \sum_{i=1}^{n} \mathbb{Y}(s_i) e^{-|s_i - q|^2/\tau^2}, \text{where } s, q \in \mathbb{M}. \tag{3.2}$$

To break the entangled links between estimation points and sensor nodes, SWOP needs to transform the Gaussian kernel. Two fast transforms are available, the *Fast Gaussian Transform* (FGT) [GS91] and the *Improved Fast Gaussian Transform* (IFGT) [YDGD03]. Both fast transforms use an infinite series to approximate the Gaussian kernel and truncate insignificant series terms to accelerate the evaluation speed. The truncated series can be encoded by a small amount of data to represent detailed information about all raw readings including the geo-reference information.

Let's first explain two transforms in the 1D space. The FGT utilizes the Hermite expansion to represent the exponential function as,

$$e^{-|s_i-q|^2/\tau^2} = \sum_{j=0}^{\infty} \frac{1}{j!} \left(\frac{\Delta s_i}{\tau}\right)^j h_j \left(\frac{\Delta q}{\tau}\right), \tag{3.3}$$

where $\Delta s_i = s_i - s_*$, $\Delta q = q - s_*$ and the Hermite functions $h_j(x)$ are defined by

$$h_j(x) = (-1)^j \frac{d^j}{dx^j} \left(e^{-x^2}\right).$$

The FGT needs to group sensor nodes into sub-clusters. Here, s_* is the cluster center which satisfies $|s_i - s_*|/\tau < 1$, so the Hermite coefficients converge to zero and the Gaussian kernel can be safely approximated by the first p terms,

$$\hat{\mathbb{Y}}(q) \approx \frac{1}{\tau^2} \sum_{j=0}^{p} A_j(s) h_j \left(\frac{\Delta q}{\tau}\right), \tag{3.4}$$

where the Hermite coefficients $A_j(s)$ are defined as

$$A_j(s) = \frac{1}{j!} \sum_{i=1}^{n} Y(s_i) \left(\frac{\Delta s_i}{\tau}\right)^j. \tag{3.5}$$

The IFGT(Improved Fast Gaussian transform) [YDGD03] factorizes the Gaussian kernel as

$$e^{-|s_i-q|^2/\tau^2} = e^{-\frac{\Delta s_i{}^2}{\tau^2}} e^{-\frac{\Delta q^2}{\tau^2}} e^{-\frac{2\Delta s_i \Delta q}{\tau^2}} \tag{3.6}$$

and uses the Taylor expansion to approximate,

$$e^{-2\Delta s_i \Delta q/\tau^2} = \sum_{j=0}^{\infty} \frac{2^j}{j!} \left(\frac{\Delta s_i}{\tau}\right)^j \left(\frac{\Delta q}{\tau}\right)^j.$$

In the IFGT, Equation 3.2 is approximated as

$$\hat{\mathbb{Y}}(q) \approx \frac{1}{\tau^2} \sum_{j=0}^{p} C_j(s) e^{-\Delta q^2/\tau^2} \left(\frac{\Delta q}{\tau} \right)^j, \tag{3.7}$$

where the Taylor coefficients $C_j(s)$ are defined as

$$C_j(s) = \frac{2^j}{j!} \sum_{i=1}^{n} Y(s_i) e^{-\Delta {s_i}^2/\tau^2} \left(\frac{\Delta s_i}{\tau} \right)^j. \tag{3.8}$$

Here, the cluster center satisfies $2|\Delta s_i||\Delta q|/\tau^2 < 1$, so the Taylor coefficients converge to zero and terms after the first p terms can be safely truncated.

To safely truncate the series, both the FGT and IFGT group the sensor nodes into sub-clusters with a radius smaller than the required bandwidth. For each cluster, the Hermite coefficients, Equation 3.5, and the Taylor coefficients, Equation 3.8, can represent the detailed sensor node locations and sensor readings. We need to choose one transform that requires a smaller number of expansion terms for the same quality criteria, and apply it to SWOP. Assume $\rho_s = |s_i - s*|/\tau$ is the normalized cluster radius, and $\rho_q = |q - s_*|/\tau$ is the normalized distance between query points and cluster centers. The Hermite expansion requires $\rho_s < 1$, while the Taylor expansion requires $2\rho_s\rho_q < 1$. The Taylor expansion also requires $\rho_q > 1$, or the estimation result ignores some important readings outside the range [YDGD03]. Thus, the Taylor expansion requires smaller cluster radii than does the Hermite expansion. The IFGT also introduces an exponential term, 2^j, to the expansion, which decreases the convergence speed of the expansion. In other words, for the same cluster size, the terms in Hermite expansion converge faster to zero than the terms in the

Taylor expansion do, which can also be proven by the error bound of FGT [BR02] and IFGT [YDGD03]. Thus, we choose the Hermite expansion in SWOP.

3.3 SWOP

3.3.1 Normalized Kernel

Although the Kernel estimation model in Equation 3.1 is useful in many cases, the normalized Kernel estimation model usually performs better, especially when nodes are unevenly distributed. The normalized model estimates the phenomenon value at the point, q, as the estimation value by Equation 3.1 divided by total kernel weights,

$$\hat{\mathbb{Y}}(q) = \frac{\sum_{i=1}^{n} Y(s_i) e^{-|s_i - q|^2/\tau^2}}{\sum_{i=1}^{n} e^{-|s_i - q|^2/\tau^2}}, \text{where } s, q \in R. \tag{3.9}$$

In Equation 3.9, τ^{-2} in the denominator and the numerator is canceled. Based on the normalized model, the Kernel estimation result is robust against the uneven spatial distribution of sensor readings. For example, sensor readings may get lost during the wireless transmission.

3.3.2 Data Reduction in High Dimensional Space

Differentiating different dimensions is important when a phenomenon is directionally different. The Kernel estimation can use different bandwidths for different dimensions for

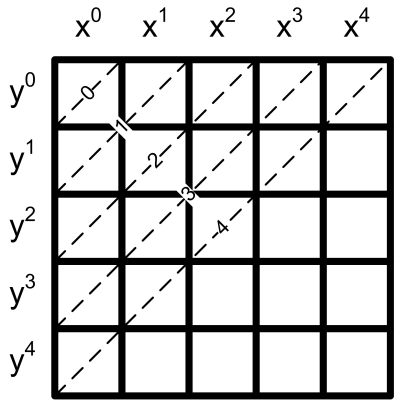


Figure 3.2. Coefficient polynomial order

anisotropic phenomena. In a high dimensional space, the FGT treats Equation 3.3 as a product of p-terms Hermite expansion along each dimension, and requires p^d terms in total for a d-dimensional space [GS91]. In the IFGT, the total number of terms in a d-dimensional space is $\binom{p+1}{d}$ by treating the vector product as a scalar dot-product in Equation 3.6 [GS91]. Therefore, the IFGT outperforms the FGT in high dimensional space [YDGD03].

Sensor node locations are at least 2D in real applications. In other words, if we choose to transform the Gaussian function as the original FGT does, the data requirement grows exponentially for more Hermite coefficient terms along each dimension. The communication channel of a constrained WSN can be easily overwhelmed to achieve smaller errors for the fast transform. Let's reconsider both transforms as indicated by Equation 3.4 and Equation 3.7. Both transforms benefit from the elimination of insignificant series terms

with values close enough to 0. According to the inequality for Hermite functions [Sza51],

$$\frac{1}{n!}\left|h_n(x)\right| \leqslant \frac{2^{n/2}}{\sqrt{n!}}e^{-x^2/2}, \text{ where } n \geqslant 0 \text{ and } x \in \mathbb{R},$$
$$\frac{2^{n/2}}{\sqrt{n!}} < 1, \text{ where } n \geqslant 4,$$

and $\rho_s < 1$, we can conclude that in Equation 3.4, the expansion terms converge to zero. The Hermite function, $h_n()$, targets the query point. The cluster radius, ρ_s, determines the convergence speed. The significance of the Hermite expansion terms is determined by the polynomial order of the coefficients. In the 1D scenario, taking the first p terms means taking the expansion terms with polynomial order lower than $p-1$ in both fast transforms. In a high dimensional space, the traditional coefficient terms ordering strategy chosen by the FGT requires an exponential increase on the resource consumption to achieve smaller errors. SWOP, on the other hand, reorders the series terms based on their significance (i.e., in the polynomial order). For example, in a 2D space, the original Hermite coefficients from the network can be represented as a 2D array as shown by Figure 3.2, where each element is a product of Hermite coefficients in $x-$ and $y-$ dimension. To get the Hermite coefficients less than the quartic order, SWOP only requires the upper-left triangular matrix, since the lower-right triangular elements are much closer to 0 than the upper-left elements. In this way, SWOP relaxes the data requirement of Hermite expansion from p^d to $\binom{p+1}{d}$ based on the $(p-1)$ polynomial order in a d-dimensional space. By ordering Hermite coefficients in this way, SWOP requires the same cost as the IFGT does in the 2D space [YDGD03]. Our experimental results confirm our expectation. By truncating Hermite coefficients based

on the polynomial order, SWOP outperforms the traditional FGT. For the same amount of communication, SWOP returns better estimation results than the original FGT does as compared with the result from traditional centralized approaches.

3.3.3 Clustering in Dynamic Networks

SWOP groups sensor nodes into non-overlapping sub-clusters according to the node locations and transforms raw readings into the Hermite coefficients. For each sensor cluster, SWOP is a special aggregation query. For different types of networks, SWOP uses different processing strategies.

The main difference of processing SWOP in mobile and static networks is the clustering algorithm. The FGT does it by dividing the space into regular grid cells, named "Boxes" [GS91]. This clustering algorithm is simple and may introduce empty boxes due to the uneven distribution of sensor nodes, especially when nodes are mobile. The optimal clustering, however, is known to be *NP* hard [BE97]. Several sub-optimal clustering algorithms, such as K-means, G-means and hierarchical clustering [HK01], are useful for static WSNs. Distributed clustering algorithms, such as HEED [YF04] and LEACH [HCB02], provide efficient clustering approaches for mobile WSNs. In distributed clustering algorithms, each sensor node can be a cluster head or belong to a cluster. A sensor node with more remaining energy and more potential communication links with others more likely announces itself to be a cluster head. Other nodes can join an appropriate cluster by detecting and analyzing

the cluster-head announcements. HEED has several advantages over LEACH, such as supporting multi-hop clustering and different clustering preferences. Thus, we choose HEED as the basic clustering method in SWOP for mobile networks. In SWOP, the cluster radius should be smaller than the Kernel bandwidth, τ, to satisfy the convergence condition, while bigger clusters are favorable to achieve a higher compression rate. Thus, we set the cluster radius to 0.9τ in SWOP. An issue of applying distributed clustering algorithms is that the required cluster size might be larger than the possible communication range of sensor nodes. In this case, SWOP allows small clusters to merge until the clusters grow to the required cluster radius. A similar algorithm can also be found in [JN05].

After the clustering procedure, each sensor cluster in SWOP is identified by its cluster center, s_*, which may not coincide with the location of the cluster head node. The detailed information about individual sensor readings and sensor locations can be transformed into a small number of Hermite coefficients. Because of the fast convergence speed of the Hermite expansion, we expect the difference between the results of SWOP and centralized Kernel estimation to be small. For each cluster in a mobile network, the Hermite coefficients are special aggregation data, and are routed to the central base [YF04].

3.3.4 Description of SWOP Algorithm

Table 3.1 outlines the clustering algorithm that we have developed in [JN05] for SWOP. When the network starts up, all sensor nodes are cluster heads. Each cluster head uses

$to\_announce()$, a HEED like algorithm, to make cluster head announcements. After receiving the cluster head announcements, a cluster head can decide to merge its cluster to another cluster. All non-head member nodes can detect the changes by the notification from their cluster head nodes. Small clusters merge into larger clusters until merging any two neighboring cluster will cause the cluster radius to be larger than the required cluster size. A distributed clustering algorithm is necessary for a mobile WSN in SWOP. In a static WSN, we can apply a centralized, better and more expensive clustering algorithm.

Table 3.1. Algorithm of distributed clustering

Require: A required cluster size for an appropriate Gaussian kernel bandwidth. The radius of the cluster should be smaller than the bandwidth to satisfy the converge condition of Hermite expansion.

Ensure: This algorithm merges small clusters into larger clusters until no merge can be done.

```
new_member ⇐ receive_new_join()
update_my_cluster(new_member)
if to_announce() then
    broadcast_announcement()
else if candidate ⇐ receive_announcement() then
    if satisfy_required_cluster_size(my_cluster, candidate) then
        join_to(candidate)
        resgin_cluster_head()
        notify_memeber_nodes()
    end if
end if
```

After identifying the cluster center and the number of nodes in the cluster, distributed sensor nodes can aggregate the Hermite coefficients as indicated by Equation 3.5. The number of nodes in every cluster determines whether the raw readings are converted into

Table 3.2. Algorithm of preparing Hermite coefficients

Require: The Hermite coefficient polynomial order, n
Ensure: Aggregation of Hermite coefficients from the network

```
msg ⇐ receive_msg()
if from_same_cluster(msg) then
  if my_cluster.number_of_members() > tolerance then
    denominator_coefficients
    ⇐ aggregate_Hermite_coefficient(msg, my_location, 1, n)
    numerator_coeffcients
    ⇐ aggregate_Hermite_coefficient(msg, my_location, my_reading, n)
    my_msg ⇐ pack_message(numerator_coeffcients,
    denominator_coefficients, cluster_center)
  else
    my_msg ⇐ pack_message(msg, my_reading, my_location, cluster_center)
    route_to_central_base(my_msg)
  end if
else
  route_to_central_base(my_msg, msg)
end if
```

Hermite coefficients. Table 3.2 illustrates the algorithm of preparing the Hermite coefficients. For a mobile sensor network, the Hermite coefficients for both denominator and numerator in Equation 3.9 should be prepared, if the number of nodes in a cluster is large enough. The function $aggregate\_Hermite\_coefficient()$ takes the local sensor's location and reading to prepare the Hermite coefficients for required polynomial order according to Equation 3.5. If a cluster contains a small number of nodes, the raw data including sensor readings and locations are returned to the central computer as shown by Table 3.2. The numerator coefficients represent both nodes' locations and sensor readings, while the denominator coefficients represent only nodes' location information. In a static network, we

only need the numerator coefficients since the location of sensor nodes can be cached by the central base.

Table 3.3. Algorithm of preparing final spatial window results at the central base

Require: Specifications for required query window, Kernel bandwidth, cluster radius and polynomial order of Hermite coefficient.
Ensure: Sending query specifications into the network, and generating the spatial window query results

1: $init\_msg \Leftarrow get\_query(specifications)$
2: $send\_to\_sensors(init\_msg)$
3: $(Hermite\_coefficients, cluster\_centers) \Leftarrow receive\_from\_network()$
4: **for** $point \in query\_window$ **do**
5: $\quad generate\_estimation\_result(point, Hermite\_coefficients, cluster\_centers)$
6: **end for**
7: $return(estimation\_result)$

After receiving a spatial window query from users, a central base first invokes the necessary sensor nodes and disseminates initial messages into the network as shown by Table 3.3. After receiving all Hermite coefficients and uncompressed data from all clusters, the central computer needs to reconstruct the weighted readings based on Equation 3.3 from the partial expansion terms and aggregate them with uncompressed readings. After the central base estimates all points within the spatial window region based on a user-defined resolution, a visual image is returned.

3.3.5 Analysis of SWOP

Computation Cost

The computation cost of the central computer is related to the number of points m for a user-defined resolution, the number of clusters k and the chosen polynomial order $p-1$. The computation complexity can be formulated as $O(m*k*\binom{p+1}{2})$. The chosen polynomial order $p-1$ and the number of clusters k are much smaller than the number of invoked sensor nodes n for an acceptable error tolerance. The computation cost for a central computer can be relaxed as $O(m)$, which is linearly relative to the user-defined resolution for a spatial window query result. Further computation acceleration can be achieved by differentiating the Hermite series around the estimation points as shown by [GS91]. In this dissertation, we don't consider it because the computation is done by a central base or a microserver, and the computation cost has no effect on the network.

The computation cost on the distributed sensor nodes is dominated by the clustering procedure, since aggregating Hermite coefficients requires a constant cost as shown by Equation 3.5 and Table 3.2. In a mobile WSN, SWOP chooses a distributed clustering algorithm, which consumes extra resources from the network. If the required cluster radius is smaller than the communication range, SWOP requires a constant amount of resource from the network, which has been proven by [YF04]. If the cluster radius is larger than the communication range, small clusters need to be merged into larger clusters. In the

worst case where all nodes need to be merged into a single cluster, the complexity of the merging operations is $O(log(n))$ for n nodes. Thus, the total computation complexity on a sensor node is $O(log(n))$ in the worst case and $O(1)$ in general cases for a mobile network. For a static WSN, the clustering pattern can be predetermined by the central base. The computation complexity can be further relaxed.

Communication Cost

In SWOP, the size of total data extracted from the network is determined by the number of clusters, c, and the chosen polynomial order, $p-1$. For each cluster, $k_p + 2\binom{p+1}{2}k_i$ bits are needed for the Hermite coefficients, where k_p is the required bit-length to represent a point for the cluster center s_* and k_i is the required bit-length to represent a term of the Hermite series. Whereas, $l(k_p + k_i)$ bits are needed for the raw data if l sensor nodes are in the cluster.

The total communication cost within the network depends on particular communication protocols and the network topology. For a mobile environment, clustering protocols are preferable. Typically, the non-head nodes and their cluster-heads have a direct communication link. In the worst topology, where all cluster-head nodes form a linear structure, receive and relay messages one by one, SWOP requires $0.5(1+c)c(k_p + 2\binom{p+1}{2}k_i)$ data for the total communication between cluster head nodes within the network. If h represents the number of cluster heads before a particular cluster head on the path to the central base,

the cluster head receives $(c-h-1)(k_p+2\binom{p+1}{2}k_i)$ and sends $(c-h)(k_p+2\binom{p+1}{2}k_i)$ to the next hop. In the best case, in which every cluster head can directly send its messages to the central base, the total communication cost within the network is $c(k_p+2\binom{p+1}{2}k_i)$.

For a static environment using routing tree based protocols, SWOP is a set of multiple aggregation queries over non-overlapped spatial clusters. The in-network query processing can be optimized by the algorithm provided by [TYD+05a, TYD+05b]. SWOP can be categorized as a min query in [TYD+05a]. Further cost evaluation on the wireless communication can be found in [TYD+05a, TYD+05b].

3.4 Chapter Summary

This chapter presents an efficient approach, SWOP, to process spatial window queries in a WSN. The SWOP approach utilizes the spatial properties of the sensor nodes. The communication reduction of SWOP results from the spatial clustering. In a large cluster with more than $[k_p+2\binom{p+1}{2}k_i]/(k_p+k_i)$ sensor nodes, the raw readings can be reduced to the first $p-1$ order Hermite coefficients in a 2D space. Since the SWOP approach focuses on the spatial properties of the sensor nodes, the available compression techniques can also be applied on SWOP's transformed data among different clusters in the multi-hop transmission. Due to normalized Kernel estimation's robustness against noisy and lossy samples, SWOP performs well in the noisy and lossy environment of WSN. Furthermore, we do not limit SWOP to static WSNs. The denominator in Equation 3.9 only represents

the spatial properties of invoked sensor nodes. In a static WSN, the spatial properties of sensor nodes are typically cached by the central base station. More than half the amount of the communication can be saved by excluding the denominator in the normalized Kernel estimation from the in-network communication, in a static WSN. A centralized clustering algorithm can also find better clustering patterns and help SWOP to accomplish a higher data compression rate.

The results from SWOP are snapshots about the underlying phenomenon's spatial distribution. Plotting the snapshots over time can provide a video-like representation about the phenomenon. Compression techniques on the temporal data, such as approaches in [DKR04, JCW04], are applicable to the transformed SWOP's aggregated Hermite terms.

Overall, SWOP is an efficient approach to process spatial window queries for both static and mobile WSNs.

Chapter 4

NOISE-TOLERANT OBJECT AND OBJECT BOUNDARY DETECTION

In this chapter, we describe our approach to answer object boundary detection queries in WSNs. The approach is named *Noise-tolerant & Energy-efficient object and object boundary Detection* (NED). The algorithm is an energy-efficient and noise-tolerant approach to detect object and object boundary for subregions in continuous phenomena. NED uses an optimized, variable length encoding strategy to encode local object estimation results, through which the wireless communication cost is reduced. Based on established statistical models, NED can reduce the sensing noise effect, while still providing high-quality object and object boundary detection results.

4.1 Foundation of NED

As indicated by Equation 1.2, a sensor provides the local phenomenon values at the sensor node's location. Sensor readings, however, are usually affected by sensing noise in real world applications. In this chapter, we use $\hat{Y}(s_i)$ to indicate a noisy reading of sensor node s_i,

$$\hat{Y}(s_i) = \mathbb{Y}(s_i) + \epsilon \tag{4.1}$$

The error term, ϵ, in Equation 4.1 indicates the noise effect on the local sensor reading.

We define the immediate neighboring nodes of sensor node, s_i, as a node set,

$$\mathbb{N}(s_i) : \{s_j | s_j \textit{ AND } s_i \text{ can directly communicate}\}, \tag{4.2}$$

and $s_i \in \mathbb{N}(s_i)$.

Assumption 1 *Compared to the traditional powerful wireless radio, the area covered by the direct radio communication range of sensor nodes is small. In today's WSNs, the number of sensor nodes in $\mathbb{N}(s_i)$ is limited. For example, in TinyOS, the maximal value of $\mathbb{N}(s_i)$ is 16. The underlying phenomena, therefore, are assumed stationary and isotropic locally in the direct radio communication region.*

Assumption 1 simplifies our definitions for local object and object boundary detection results.

Assumption 2 *In this chapter, we assume the nodes in $N(s_i)$ are evenly distributed.*

Assumption 3 *In this dissertation, the sensing noise, ϵ is assumed to be independent among different sensor nodes. We use a general model of ϵ, which defines the noise as a white normal random variable with 0 mean and a given variance, σ^2. In other words, sensor readings are unbiased about the underlying phenomenon. In most cases, the sensor specification manual provides the σ value.*

Based on above assumptions, we can present our approach to detect local object and object boundary results based on local and nearby sensor readings.

4.2 Encoding Local Object Detection Results

In a WSN, the wireless communication is the most energy consuming part. To deal with noisy sensor readings and detect objects, sensor nodes have to encode local sensor readings into digital values and exchange messages through the wireless radio. A local sensor reading is usually represented as a real number (32 bits or more) [DCXC05], while a binary variable (1 bit) can represent a local object detection result [KI04]. Exchanging the binary object detection results is resource-efficient to the constrained environment. The noisy sensor readings, however, may cause faulty binary results, and degrade the quality of object and object boundary detection results. Estimation results based on float values are more robust against the sensing noise, but at the expense of increased communication burden. NED needs to keep a graceful balance between the communication consumption and the detection quality.

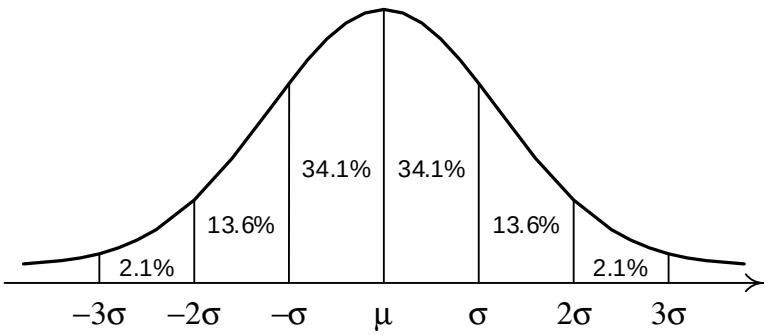


Figure 4.1. Normal probability density distribution

As shown by Figure 4.1, the probability density of a normal random variable concentrates around the mean value. For example, given the mean μ and the variance σ^2, a normal variable has 95% to be within the range $[\mu - 1.96\sigma, \mu + 1.96\sigma]$. Different sensor readings provide different confidence levels on local object estimation results. If a sensor reading is much larger than the threshold T (e.g., the distance to the threshold is larger than 1.96σ), the object estimation should be a significant result (at the confidence level greater than 95%). Similarly, a sensor node may detect significant non-object (e.g., $T - \hat{Y}(s_i) > 1.96\sigma$) and insignificant object readings (e.g., $|T - \hat{Y}(s_i)| \leqslant 1.96\sigma$).

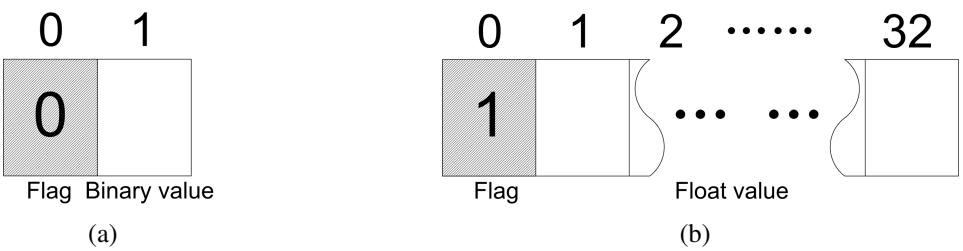


Figure 4.2. Message formats of local object detection
(a) the message format for significant object detection, and (b) the message format for insignificant object detection

To balance the communication cost and the detection quality, NED uses a variable length encoding mechanism to represent local estimation results of individual sensor nodes as shown by Figure 4.2. First, we use Δ_1 to indicate a distance from the threshold, T. Δ_1 also indicates a belief level of local object estimation. For example, $\Delta_1 = 1.96\sigma$ means the distance is away from the threshold T with the 95% confidence. A sensor node can detect a significant object or non-object reading based on the confidence level of Δ_1, and use a 2-bit message to represent its reading. As explained by Figure 4.2(a), the first bit in a message is a flag to indicate if the following bits represent significant or insignificant sensor readings. The flag is 0 to indicate local significant object or non-object readings. If the first bit is 0, the second bit is 1 for a significant object reading or 0 for a non-object reading. If a sensor node observes an insignificant object (i.e., the local sensor reading falls within the $[T - \Delta_1, T + \Delta_1]$ range), the node changes the flag to 1 and requires additional 32 bits to represent the original sensor reading, as shown by Figure 4.2.

After receiving messages from its neighbors, a sensor node needs to recover the binary results into regular real numbers to do further analysis. A significant object reading is represented by $T + \Delta_1$; a significant non-object reading is $T - \Delta_1$.

4.3 Theoretical Analysis

Based on the n object detection messages from its neighbors, a sensor node needs to find the estimated underlying phenomenon value. Since sensor readings are encoded and transmitted in a digital format, the n neighboring readings form a new distribution. If we assume r is the random number representing the recovered encoded message reading, y_0 is the underlying phenomenon value, $\phi()$ is the pdf and $\Phi()$ is the cdf of the standard normal distribution function, the pdf of r can be formulated as,

$$p(r) = \begin{cases} \Phi\left(\frac{T-\Delta_1-y_0}{\sigma}\right), & r \leq T-\Delta_1; \\ \phi\left(\frac{r-y_0}{\sigma}\right), & r \in (T-\Delta_1, T+\Delta_1); \\ 1-\Phi\left(\frac{T+\Delta_1-y_0}{\sigma}\right), & r \geq T-\Delta_1. \end{cases}$$

Therefore, the expected mean of r is M defined as,

$$\begin{aligned} M &= \int_{-\infty}^{+\infty} xp(x)dx \\ &= (T-\Delta_1)\int_{-\infty}^{T-\Delta_1} \phi\left(\frac{x-y_0}{\sigma}\right)dx + \int_{T-\Delta_1}^{T+\Delta_1} x\phi\left(\frac{x-y_0}{\sigma}\right)dz \\ &\quad + (T+\Delta_1)\int_{T+\Delta_1}^{+\infty} \phi\left(\frac{x-y_0}{\sigma}\right)dx, \end{aligned}$$

in which the middle term can be reformulated as,

$$\begin{aligned}
\int_{T-\Delta_1}^{T+\Delta_1} x\phi\left(\frac{x-y_0}{\sigma}\right)dx &= \int_{T-\Delta_1}^{T} x\phi\left(\frac{x-y_0}{\sigma}\right)dx + \int_{T}^{T+\Delta_1} x\phi\left(\frac{x-y_0}{\sigma}\right)dx \\
&= \lim_{c\to\infty}\sum_{i=1}^{c}\frac{\Delta_1}{c}\left(T-i\frac{\Delta_1}{c}\right)\phi\left(\frac{T-y_0}{\sigma}-i\frac{\Delta_1}{c\sigma}\right) \\
&\quad + \lim_{c\to\infty}\sum_{i=1}^{c}\frac{\Delta_1}{c}\left(T+i\frac{\Delta_1}{c}\right)\phi\left(\frac{T-y_0}{\sigma}+i\frac{\Delta_1}{c\sigma}\right) \\
&= T\lim_{c\to\infty}\sum_{i=1}^{2c}\frac{\Delta_1}{c}\phi\left(\frac{T-\Delta_1-y_0}{\sigma}+i\frac{\Delta_1}{c\sigma}\right) \\
&\quad + \lim_{c\to\infty}\sum_{i=0}^{c} i\left(\frac{\Delta_1}{c}\right)^2\left(e^{-\frac{(T+i(\Delta_1/c)-y)^2}{2\sigma^2}} - e^{-\frac{(T-i(\Delta_1/c)-y)^2}{2\sigma^2}}\right) \\
&= T\int_{T-\Delta_1}^{T+\Delta_1}\phi\left(\frac{x-y_0}{\sigma}\right)dx \\
&\quad + \lim_{c\to\infty}\sum_{i=0}^{c} i\left(\frac{\Delta_1}{c}\right)^2\left[\phi\left(\frac{T-y_0}{\sigma}+i\frac{\Delta_1}{c\sigma}\right)-\phi\left(\frac{T-y_0}{\sigma}-i\frac{\Delta_1}{c\sigma}\right)\right].
\end{aligned}$$

So M can be stated as,

$$\begin{aligned}
M =& T + \Delta_1\left[\int_{T+\Delta_1}^{+\infty}\phi(\frac{x-y_0}{\sigma})dx - \int_{-\infty}^{T-\Delta_1}\phi(\frac{x-y_0}{\sigma})dx\right] \\
&+ \lim_{c\to\infty}\sum_{i=0}^{c} i\left(\frac{\Delta_1}{c}\right)^2\left[\phi\left(\frac{T-y_0}{\sigma}+i\frac{\Delta_1}{c\sigma}\right)-\phi\left(\frac{T-y_0}{\sigma}-i\frac{\Delta_1}{c\sigma}\right)\right].
\end{aligned}$$

We now can state and prove one important property of the encoding schema of NED.

Property 1 *The object estimation result based on the arithmetic average of recovered neighboring sensor readings is unbiased.*

Proof: This property can be proven under three different conditions.

1. $y_0 = T$:

   When $y_0 = T$,$\int_{T+\Delta_1}^{+\infty}\phi\left(\frac{x-T}{\sigma}\right)dx = \int_{-\infty}^{T-\Delta_1}\phi\left(\frac{x-T}{\sigma}\right)dx$ and $\phi\left(i\frac{\Delta_1}{c\sigma}\right) = \phi\left(i\frac{\Delta_1}{c\sigma}\right)$, since $\phi()$ is symmetric around zero. Therefore, $M = T$ in this case.

2. $y_0 > T$:

   If $y_0 > T$, then $\int_{T+\Delta_1}^{+\infty} \phi\left(\frac{x-T}{\sigma}\right) dx > \int_{-\infty}^{T-\Delta_1} \phi\left(\frac{x-T}{\sigma}\right) dx$ and $\phi\left(\frac{T-y_0}{\sigma} + i\frac{\Delta_1}{c\sigma}\right) > \phi\left(\frac{T-y_0}{\sigma} - i\frac{\Delta_1}{c\sigma}\right)$. So $M > T$ when $y_0 > T$.

3. $y_0 < T$:

   Similarly we can find that when $y_0 < T$, $M < T$.

To make further statistical tests on object boundary, we need to know the variance, V^2, of r, when $y_0 = T$.

$$\begin{aligned}
V^2 =& \int_{-\infty}^{+\infty} (x-M)^2 p(x) dx \\
=&(\Delta_1)^2 \int_{-\infty}^{T-\Delta_1} \phi\left(\frac{x-T}{\sigma}\right) dx + \int_{T-\Delta_1}^{T+\Delta_1} (x-T)^2 \phi\left(\frac{x-y_0}{\sigma}\right) dx \\
&+ (\Delta_1)^2 \int_{T+\Delta_1}^{+\infty} \phi\left(\frac{x-T}{\sigma}\right) dx \\
=&2(\Delta_1)^2 \int_{-\infty}^{T-\Delta_1} \phi\left(\frac{x-T}{\sigma}\right) dx + \int_{T-\Delta_1}^{T+\Delta_1} (x-T)^2 \phi\left(\frac{x-T}{\sigma}\right) dx \\
=&2(\Delta_1)^2 \Phi\left(\frac{-\Delta_1}{\sigma}\right) + \left[\Phi\left(\frac{x-T}{\sigma}\right)\sigma^2 + \phi\left(\frac{x-T}{\sigma}\right)(T-x)\sigma^2\right]\Bigg|_{T-\Delta_1}^{T+\Delta_1} \\
=& \left(2(\Delta_1)^2 - \sigma^2\right)\Phi\left(\frac{-\Delta_1}{\sigma}\right) + \sigma^2\Phi\left(\frac{\Delta_1}{\sigma}\right) - 2\Delta_1\phi\left(\frac{\Delta_1}{\sigma}\right)\sigma^2
\end{aligned} \tag{4.3}$$

4.4 Object and Object Boundary Detection

After receiving messages from neighboring nodes, a sensor node needs to recover significant object and non-object readings first. Then the node can use the arithmetic average, $\bar{m}$, based on recovered readings and insignificant object readings, to estimate the local phenomenon value. Based on the property 1, NED estimates the local object status, $\hat{O}(s_i)$,

as,

$$\hat{O}(s_i) = \begin{cases} 1, \text{ if } \bar{m} \geqslant T; \\ \\ 0, \text{ else.} \end{cases} \tag{4.4}$$

Besides the arithmetic average, statisticians often use several different averages to estimate the mean value. For example, the median value can be used [DCXC05]. The median value has several advantages, such as being more robust against outliers. However, it is hard to use a variable length encoding schema to exchange the median value. We use the arithmetic average because the probability of finding an outlier reading among a few neighboring nodes is small. The encoding confidence level Δ_1 also works as a filter to remove outlier readings outsides $[T - \Delta_1, T + \Delta_1]$. Those outlier readings can be recovered as $T - \Delta_1$ or $T + \Delta_1$ before the further analysis. Our simulation results also show that the arithmetic average value performs better than the median value.

The variance of recovered neighboring readings is based on Equation 4.3 for the given sensor device variance, σ^2, the object threshold, T and the encoding significant level Δ_1. According to the central limit theorem, the arithmetic average based on Equation 4.4 is an estimation of local mean with the estimated variance defined by,

$$\hat{\sigma^2} = \frac{V^2}{n}, \tag{4.5}$$

where n is the number of nodes in $\mathbb{N}(s_i)$.

We cannot directly apply Equation 1.4 based on the estimated local phenomenon values, $\hat{Y}(s_i)$. One important reason is that sensor nodes are discretely distributed in space. There may be no node located on the object boundary. The sensing noise can also affect the local

boundary estimation results. NED uses Equation 4.6 to estimate the boundary, $\hat{B}()$, for another given confidence level, Δ_2.

$$\hat{B}(s_i) = \begin{cases} 1, \text{if } T \in [\bar{m} - \Delta_2, \bar{m} + \Delta_2]; \\ \\ 0, \text{else.} \end{cases} \tag{4.6}$$

Equation 4.6 is equivalent to the statistical model in [CG03]. $\hat{B}(s_i)$ in Equation 4.6 indicates whether the sensor node, s_i, has the $\Phi(\frac{\Delta_2}{\hat{\sigma}}) - \Phi(-\frac{\Delta_2}{\hat{\sigma}})$ confidence that the node is located on the boundary $Y(s_i) = T$. NED's encoding mechanism symmetrically trims the sensor readings around the object threshold, T. Since the underlying phenomenon is isotropic, and the sensor nodes are evenly distributed, the arithmetic average of recovered messages, $\bar{m}$, around an object boundary should be close enough to T. Thus, in NED, the nodes around an object boundary can report the boundary detection.

4.5 Algorithms of NED

A sensor specification manual usually gives the variance of sensing noise. As shown by Table 4.1, sensor nodes can calculate the value Δ_1 based on the user-defined confidence level, $Level1$. Based on the value of Δ_1, a sensor node can encode the local sensor readings accordingly. The time space is divided into rounds. In each round, each sensor node broadcasts its local encoded object readings. After receiving neighbors' local encoded object readings, a sensor node restores significant object and non-object readings (binary values)

into real numbers. The node then computes the arithmetic average to reduce the noise effect, and to estimate a better local object and object boundary status based on Equation 4.4 and Equation 4.6. Based on the value of Δ_2, sensor nodes can report significant object boundary detection results as illustrated by Table 4.1.

Table 4.1. Algorithm of NED

Require: Sensor nodes know the sensing noise level of σ, and the two confidence levels $Level1$ and $Level2$.
Ensure: Sensor nodes make local object and boundary estimation.

$\Delta_1 \Leftarrow computeSignificantLevel(Level1)$
$\Delta_2 \Leftarrow computeSignificantLevel(Level2)$
$r \Leftarrow getMySensorReading()$
$msg \Leftarrow encodeObject(r, T, \Delta_1)$
$broadcast(msg)$
$msgList \Leftarrow receiveFromNeighbors()$
$readingList \Leftarrow restore(msgList, T, \Delta_1)$
$avg \Leftarrow estAvg(readingList)$
$sqrtVar \Leftarrow estSqrtVar(readingList)$
if $avg \geqslant T$ **then**
  $estObject \Leftarrow$ *TRUE*
else
  $estObject \Leftarrow$ *FALSE*
end if
if $T \geqslant (avg - \Delta_2)$ *AND* $T \leqslant (avg + \Delta_2)$ **then**
  $estBoundary \Leftarrow true$
else
  $estBoundary \Leftarrow$*FALSE*
end if

NED does not consider the detailed locations of neighboring nodes. If the spatial variation of a phenomenon is large, and sensor nodes are not evenly distributed, we can apply different weights on neighboring readings. For example, a weight based on the Euclidean

distances between node locations has been used in Chapter 3. This type of approach can produce better estimation results but also requires more wireless communication.

4.6 Chapter Summary

In this chapter, we present an efficient approach, NED, to detect 2D object and object boundary in WSNs. The NED approach allows users to specify different confidence levels for encoding local estimation results and estimating object boundary based on neighboring messages. If the noise level is small, a small Δ_1 value can be used to save more communication cost. For more noisy readings, a large Δ_1 value is preferable to achieve better estimation results. When the phenomenon change is more influential than the sensing noise among neighboring nodes, the variance among neighboring readings may be larger than we expected as shown by Equation 4.3. In such a case, users can set a larger confidence level for Δ_2 to get better boundary estimation results. A phenomenon may not be spatially continuous. For example, a phenomenon can be a step function over space, as assumed by [DCXC05], in which phenomenon values are y_1 in object regions and y_2 in non-object regions. For such discontinuous phenomena, the boundary threshold T can be any values in (y_1, y_2) to separate object and non-object regions. The NED approach can use the boundary threshold $T = \frac{y_1+y_2}{2}$ to keep the symmetry of sensor readings around the threshold, T, and to detect the object boundary for this type of phenomenon. For spatially continuous phenomena, the NED approach allows nodes far from the object boundary to communicate

by only 2-bit messages. Those nodes close to the object boundary use 33-bit messages to achieve high quality estimation results. As shown by our experimental results, the NED approach is resource efficient in the constrained environment.

Chapter 5

TRACKING DEFORMABLE 2D OBJECTS IN WSN

Based on efficient boundary detection algorithms, such as NED, distributed nodes report boundary points. Related approaches also help SDMSs to find the geometric representation of the object boundary. The next logical step, which is also the focus of this chapter, is using a WSN to track 2D objects based on their geometric representations.

This chapter presents a SNAKE-based algorithm to track 2D objects in WSNs. The proposed approach uses a deformable curve to represent a 2D object. The representative curve virtually deforms and optimizes its shape and location to track the 2D object. Different from detecting boundary points (or connecting them into closed geometric curves), the proposed approach focuses on tracking the movement of the representative curve by using WSNs.

We use sensor nodes to detect object boundary changes in their neighboring regions. Thus, distributed nodes adapt the representative curve locally to track the overall object. In

the proposed approach, a node does not need to know the global shape of boundary geometry and saves communication cost. Furthermore, the proposed approach locally adapts the representative curve to the topology changes caused by the interactions among multiple 2D objects (i.e., splitting and merging). In this way, the proposed approach tracks 2D objects efficiently in WSNs.

Based on the deformable curve representation, an SDMS derives abstract spatiotemporal properties of underlying 2D objects via the in-network aggregation operations. By detecting the object boundary change and adapting the representative curve, distributed sensor nodes are able to extrapolate the object's location and shape in the near future. In such a way, different types of abstract information and queries are processed fully in network. The communication cost to report the geometric information to a base station, consequently, is further reduced.

First, we explain several preliminary concepts related to the in-network 2D object tracking.

5.1 SNAKE Model

We use $\{V^t, E^t\}$ to indicate a closed curve representing the object boundary at time t.

$$V^t = \left\{v_1^t, v_2^t, \cdots, v_n^t\right\}, \tag{5.1a}$$

$$E^t = \left\{\overrightarrow{v_1^t v_2^t}, \overrightarrow{v_2^t v_3^t}, \cdots, \overrightarrow{v_{n-1}^t v_n^t}, \overrightarrow{v_n^t v_1^t}\right\}. \tag{5.1b}$$

A closed curve consists of n vertices and n edges. The vertices are 2D points (i.e., $v_i^t = (x_i^t, y_i^t)$). The curve is a closed curve, as indicated by Equation 5.1b. $\{V^t, E^t\}$ is assumed to represent a simple curve (i.e., the curve does not cross itself). The edges in E^t are directed, as illustrated by Equation 5.1b. We also assume that a WSN can correctly detect the object boundary. In short, compared to the resolution of spatial distribution of sensor nodes, we assume a 2D object needs to be large enough. Due to the monitoring granularity, the boundary of a small 2D object may not be detected. Since the vertices in V^t are sufficient to describe the edges in E^t, as shown by Equation 5.1b and Equation 5.1a, we will use V^t to represent the closed curve in the following parts of this dissertation. Therefore, we use V^0 to indicate the initial boundary geometry.

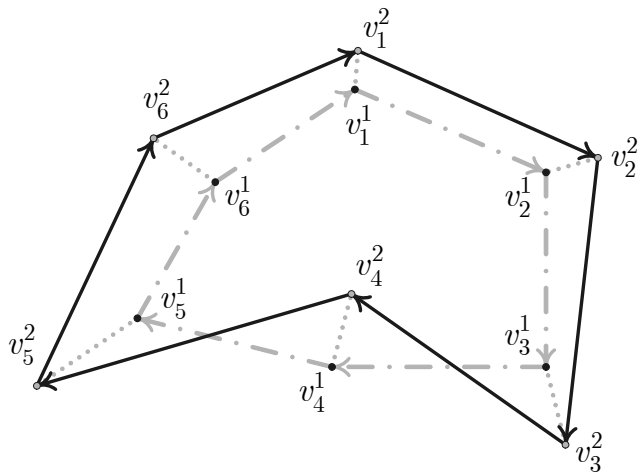


Figure 5.1. Example of deformable 2D object tracking

$$leftVertex(v_i^t) = \begin{cases} v_n^t, \text{ if } i = 1; \\ v_{i-1}^t, \text{ else.} \end{cases} \tag{5.2}$$

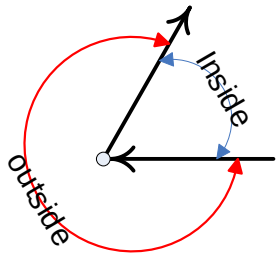


Figure 5.2. Topological relationship based on local angle

$$rightVertex(v_i^t) = \begin{cases} v_1^t, \text{ if } i = n; \\ v_{i+1}^t, \text{ else.} \end{cases} \tag{5.3}$$

For a vertex v_i^t, we name v_{i+1}^t as the immediate right neighboring vertex of v_i^t; v_{i-1}^t is the immediate left neighbor, by facing the interior region at v_i^t. As shown by Equation 5.2 and 5.3, the only exception is that v_1^t is the immediate right vertex of v_n^t; v_n^t is the immediate left vertex of v_1^t. The directed edge connecting v_i^t and $rightVertex(v_i^t)$ is the right edge of v_i^t, while the directed edge from $leftVertex(v_i^t)$ to v_i^t is the left edge of v_i^t. v_i^t and its immediate left and right neighboring vertices form the local angle centered at v_i^t. For $\angle rightVertex(v_i^t)v_i^t leftVertex(v_i^t)$, we define a point p is inside this angle, if p is located on the right of both the right and left edges of v_i^t, as illustrated by Figure 5.2.

In the field of computer vision, the deformable curve model is known as the *SNAKE* (or *Active Contour*) model [KWT88]. Here, a deformable curve is used to approximate an object boundary (e.g., coast lines in remotely sensed images) by using sparse points and computing a coarse curve first. Then an energy model is used to adjust the location and number of vertices. In a way, the deformable curve V^t can be treated as a rubber band

around a "solid" object. We can use the rubber band to represent the object's boundary. The adjustment of the representative curve is based on basic physical rules. When the rubber band is stabilized under different physical forces, the overall elastic energy is minimal. As shown by Figure 5.1, the vertices should be able to "move" over time under the influence of different "forces", and therefore deform the shape of the closed curve. At time t, the placement of V^t needs to minimize the "elastic" energy, E, as,

$$E = \alpha E_{ten} + \beta E_{cur} + \gamma E_{ext}. \tag{5.4}$$

Equation 5.4 describes the requirements for a curve to represent an object boundary. E_{ten} in the first term of Equation 5.4 is the first order continuity constraint. This term can be viewed as the tension along the rubber band. If the rubber band is stabilized, the tension should be equal along the band. In other words, the vertices need to be evenly distributed along the boundary, which is controlled by E_{ten}. E_{cur} in Equation 5.4 is the second order continuity constraint, and indicates V^t's curvature. E_{cur} controls the smoothness of V^t. E_{ten} and E_{cur} are also called internal forces, which model the geometric information about V^t. Given only E_{ten} and E_{cur}, a deformable curve cannot represent a concave shape well. E_{ext}, which is known as the external force or edge strength, provides another force to attach a deformable curve well to a 2D object of arbitrary shape. α, β and γ are relative weights of each force model, and describe the importance of different forces to the final shape and location of V^t. By applying the SNAKE model and using deformable curves to represent 2D objects, distributed sensor nodes adjust nearby vertices without knowing

the global detailed shape of V^t. Since the vertex movement is only influenced by different forces, we need to find appropriate force models, which can be efficiently implemented in the constrained WSN.

5.2 In-network Deformable Curve Tracking

Under the constraints of WSN, sensor nodes should minimize the communication consumption to maintain the deformable curve structure. In this section, we demonstrate that our revised SNAKE model achieves this design goal.

5.2.1 Efficient Force Models

To use the deformable curve model in WSNs, we constrain that a vertex, v_i^t, can only move to the location of sensor nodes. We call a sensor node, s_i, a vertex node at time t, if a vertex v_j^t is at the location of s_i (i.e., $v_j^t = s_i$). To implement the tracking algorithm, appropriate force models must be resource friendly. In our proposed approach, a node locally detects three states, whether the node is located within the object (sensed value above a user threshold), outside an object (value below the user-defined threshold), or on the boundary based on the values of its neighboring nodes.

First, we need to find neighboring boundary nodes, $\mathbb{NB}()$, defined as,

$$\begin{aligned} \mathbb{NB}(s_i, t) : \{s_j | \quad & s_j \text{ can communicate with } s_i \text{ directly } \textit{AND} \\ & s_j \text{ detects the object boundary at } t\}. \end{aligned} \tag{5.5}$$

As indicated by Equation 5.5, $\mathbb{NB}(s_i, t)$ is the set of s_i's neighboring nodes that detect the object boundary at time t. For simplicity, $\mathbb{NB}(s_i, t)$ may contain s_i, if s_i detects the boundary. Based on the discussion in Chapter 4, sensor nodes are able to prepare local object status and local object boundary status. Sensor nodes only exchange local boundary detection results among immediate neighbors to generate $\mathbb{NB}()$.

External force models for image processing are an active research area. For example, based on Fuzzy Set theory, E_{ext} can be represented as a local certainty value about the boundary [ASG01]. Additionally to the local boundary certainty, gradient vectors provide directions towards the 2D object's boundary [XP98]. Generating the gradient vectors, however, would require several iterations of messages exchanged among sensor nodes (not just among vertex nodes), which is expensive.

In our revised SNAKE-based approach, E_{ext} uses the local boundary detection results provided by $\mathbb{NB}()$. Based merely on $\mathbb{NB}()$, however, an external force may not work well when a vertex node cannot find the boundary report among its immediate neighbors. In the balloon model [Coh91], the proposed E_{ext} contains an outward pressure. By applying the outward pressure, a deformable curve behaves like an inflating balloon to expand itself to represent the object boundary. The balloon model works fine only if the deformable curve is contained within the real object boundary. We need the deformable curve to be able to also "deflate". To better track the object boundary and save the communication cost, we

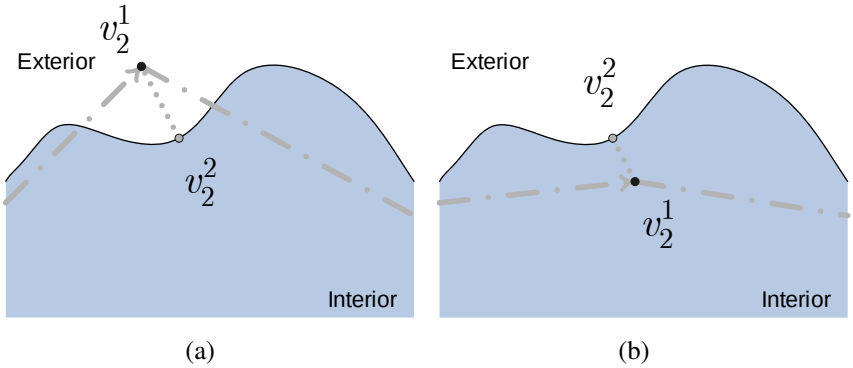


Figure 5.3. External forces when $\mathbb{NB}() = \varnothing$

define our external force model as follows.

$$E_{ext}(s_i, t) = \begin{cases} \mathbb{NB}(s_i, t), & \text{if } \mathbb{NB}(s_i, t) \neq \varnothing; \\ \\ \{s_j | s_j \in \mathbb{N}(s_i) \textit{ AND} & \text{if } \mathbb{NB}(s_i, t) = \varnothing \\ s_j \text{ is inside the curve}\}, & \textit{AND } \mathbb{O}(s_i, t) = 1; \\ \\ \{s_j | s_j \in \mathbb{N}(s_i) \textit{ AND } s_j & \text{if } \mathbb{NB}(s_i, t) = \varnothing \\ \text{is not inside the curve}\}, & \textit{AND } \mathbb{O}(s_i, t) = 0. \end{cases} \tag{5.6}$$

As shown by Equation 5.6, the proposed external force, E_{ext}, only requires message exchange among neighboring nodes. If some neighboring nodes detect the object boundary, E_{ext} allows the vertex to move onto anyone among them. Note here, a vertex may not need to move, if the vertex node at $t-1$ detects the boundary at t.

In some situations, a vertex node may lose track of the object boundary (e.g., when a 2D object moves fast), and none of its immediate neighbors detect the object boundary. Its local object detection result and the curve's topology information, however, provide useful information to adapt the curve shape correctly. If a vertex node detects that it is not located

within the object and cannot find the object boundary in the neighboring region, the node must be located in the exterior region of the object. In this case, the deformable curve needs to "deflate" locally, as shown by Figure 5.3(a). The neighboring nodes located in the interior region of the closed curve are the candidate locations for the vertex. As illustrated by Figure 5.3(b), if a vertex node detects that it is located inside of the object and finds no boundary in its nearby region, the deformable curve "inflates" locally. The neighboring nodes located in the exterior region of the closed curve are the candidate locations for the vertex. Vertices can eventually find the object boundary by using the proposed E_{ext}. Although our E_{ext} is light-weighted, our model flexibly adapts the deformable curve to track the underlying 2D object.

Equation 5.6 provides several candidate locations for a vertex node to move to. A vertex can only move to one location among the candidate locations. To calculate the energy weight among the candidate locations, we revise Equation 5.4 as,

$$E = \alpha E_{ten} + \beta E_{cur}. \tag{5.4$'$}$$

Based on Equation 5.4$'$, a vertex moves to the location with the minimal energy weight among the candidate locations given by Equation 5.6.

The internal forces need to be resource efficient as well. A general model for E_{ten} is defined by,

$$\overline{d^{t-1}} - \left| v_i^t - v_{i+1}^t \right|,$$

where $\overline{d^{t-1}}$ indicates the average length of edges,

$$\overline{d^{t-1}} = \frac{1}{n}\sum_{i=1} n\left|v_i^{t-1} - v_{i+1}^{t-1}\right|.$$

This model requires updating the average edge length, $\overline{d^{t-1}}$, among all vertex nodes if V^t changes. Perrin et al. proposed a new E_{ten} model for detecting object boundaries in digital images [PS01]. Perrin et al. showed that their E_{ten} model constrains the vertices to be evenly dispersed along the curve. So their E_{ten} model is ideal for our tracking quality requirements. We slightly modify their E_{ten} model. Our E_{ten} model is resource-efficient and only requires message exchange among consecutive vertex nodes, as defined by,

$$E_{ten} = Var\left(\left|v_i^{t+1} - v_{i-1}^t\right|, \left|v_i^{t+1} - v_{i+1}^t\right|\right). \tag{5.7}$$

For a candidate location, v_i^{t+1}, of v_i^t, $Var()$ measures the variance of the lengths of two consecutive edges, $\left|v_i^{t+1} - v_{i-1}^t\right|$ and $\left|v_i^{t+1} - v_{i+1}^t\right|$. When the two edges are equal length, E_{ten} is zero. To minimize E_{ten}, the vertices need to be located at equal intervals along the curve.

When the 2D object expands and shrinks, the deformable curve, like the rubber band, should expand and shrink simultaneously. The parameter D_{split} controls the number of vertices when the curve deforms.

$$\begin{cases} \left|v_{i+1}^t - v_i^t\right| \leqslant D_{split}, & \text{No change;} \\ \\ \left|v_{i+1}^t - v_i^t\right| > D_{split}, & \text{Add a vertex between } v_i^t \text{ and } v_{i+1}^t. \end{cases} \tag{5.8}$$

When the distance between v_i^t and v_{i+1}^t is larger than D_{split}, a new vertex is added between v_i^t and v_{i+1}^t. As illustrated by Figure 5.4(a), to ensure the even vertex spacing, the new

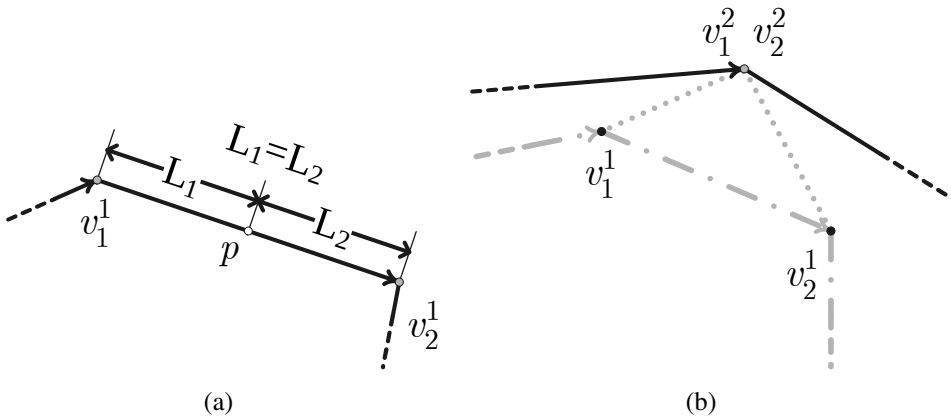


Figure 5.4. Examples of dynamic adding and folding
(a) adding a new vertex, and (b) folding vertices

vertex is placed at

$$\left(\frac{x_i^t + x_{i+1}^t}{2}, \frac{y_i^t + y_{i+1}^t}{2}\right). \tag{5.9}$$

D_{split} ensures the largest disparity in the vertex spacing, and influences the tracking quality of the deformable curve. When the deformable curve shrinks, multiple vertices may move to a single sensor node. Some vertices moving onto a single node are consecutive neighbors, and can be folded into a single vertex, as shown by Figure 5.4(b). A more complex case will be explained in Section 5.2.2.

E_{cur} controls the curve's smoothness. We use the value of the inner angle to represent E_{cur}. The second order curvature can be used to represent the smoothness, which minimizes the angle variation of three consecutive angles [PS01]. In short, the second order curvature model requires that the three consecutive angles are similar. To find the curvature value of the next location, p, for v_3^1 in Figure 5.5(b), the second order curvature model needs

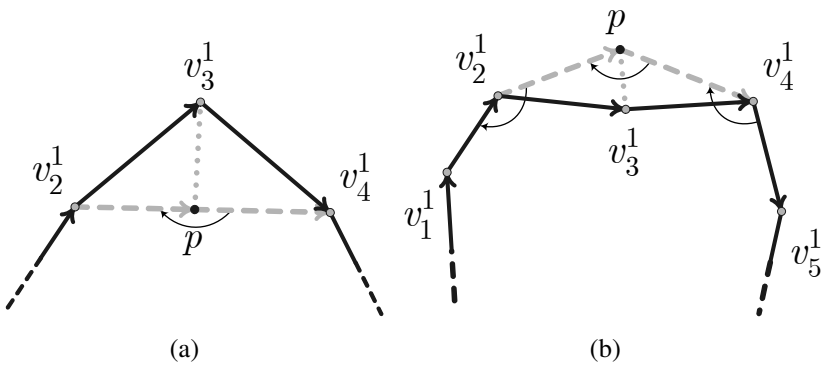


Figure 5.5. Curvature models
(a) the 1st order curvature model, and (b) the 2nd order curvature model

to know the value of three internal angles, $\angle p v_2^1 v_1^1$, $\angle v_4^1 p v_2^1$ and $\angle v_5^1 v_4^1 p$. The second order curvature needs a vertex location to be updated among five consecutive vertices, which is expensive in communication. In Figure 5.5(b), the updated location of v_3^1 should be sent to v_1^1, v_2^1, v_4^1 and v_5^1. To save the energy and communication cost, we choose the first order curvature defined as,

$$E_{cur} = Var\left(\pi, \angle v_{i+1}^t v_i^{t+1} v_{i-1}^t\right). \tag{5.10}$$

As indicated by Equation 5.10 and illustrated by Figure 5.5(a), the first order curvature model is biased towards straight lines. The first order curvature requires a vertex update to be exchanged only among three consecutive vertex nodes. For example, in Figure 5.5(a), the updated location of v_3^1 should only be sent to v_2^1 and v_4^1. Our experiments showed that the first and second order curvature models have almost the same tracking quality. One possible explanation is that E_{cur} dominates the final curve shape only when D_{split} is large. Similarly along a rubber band, a local bend only affects a nearby area.

The revised E_{ext}, E_{ten} and E_{cur} models are light-weighted, and need message exchange among neighboring nodes only. Based on the revised E_{ext}, E_{ten} and E_{cur} models, a WSN efficiently "moves" vertices and therefore tracks the underlying 2D objects. We also need to consider the topology changes when multiple 2D objects interact, which will be explained by the next section.

5.2.2 Tracking Multiple Objects

When multiple 2D objects change their shapes and locations in space, basically two types of topological changes (i.e., splitting and merging) are involved [JW09]. Deformable curves representing 2D objects consequently should adapt their shapes to the topological changes. The original SNAKE model is too rigid to do so, since the connected edges are unbreakable. A flexible model is necessary for deformable curves to adapt to the topological changes. Note here, we do not consider the dimensional changes in this dissertation. We assume that the remaining objects after the splitting or merging changes need to be large enough for a WSN to correctly detect the boundary and treat them as 2D objects.

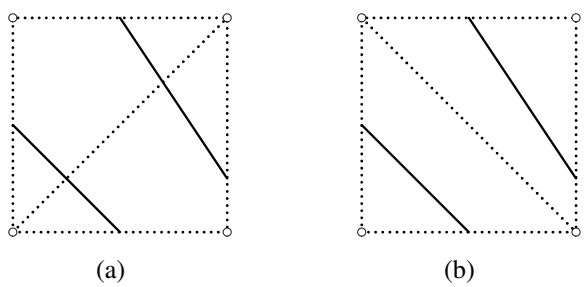


Figure 5.6. Ambiguity caused by different triangulation patterns

Today, several revised SNAKE models have been proposed to use breakable curves to track 2D objects [MT00, LV04]. Most models are based on a centralized infrastructure, which is not suitable for the constrained WSN. In the T-Snake approach [MT00], the space is partitioned into non-overlapping triangles. In a triangle cell, nonconsecutive edges need to be removed and replaced by a single edge. The T-Snake approach, however, faces the ambiguity caused by different triangulation patterns. In Figure 5.6, the solid lines indicate the edge of deformable curves; the dotted lines represent the triangulation partition. The edges in Figure 5.6(a) are identical to the edges in Figure 5.6(b). Due to the different triangulation patterns, the edges in Figure 5.6(a) need to be removed, whereas the same edges in Figure 5.6(a) can be kept. A global uniform triangulation pattern is necessary for the T-Snake approach. Finding the global triangulation pattern in the constrained WSN, however, has to consume additional resources, especially if sensor nodes are unevenly distributed or nodes are mobile [SS04, HH05].

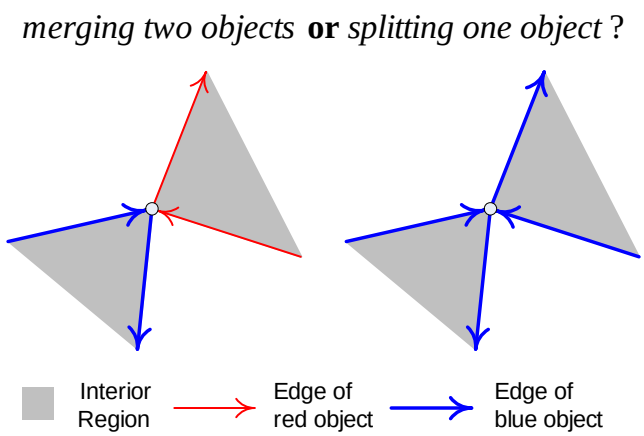


Figure 5.7. Ambiguity when two 2D objects touches at a single point

One observation is that any 2D object can be represented by a set of simple closed curves. A 2D object can contain holes. A hole can also be represented by a simple closed curve. Nonconsecutive edges in a simple closed curve cannot intersect, overlap or touch with each other. Purely based on the geometric shape of deformable curves, our model focuses on converting non-simple curves into simple curves.

Two 2D objects can touch at a single point. If we try to reconnect the edges linked to the same point, we shall face an ambiguity. The reconnected edges simultaneously can indicate a 2D object is splitting, as illustrated by Figure 5.7. To better adapt the deformable curves to the topological changes, our model is based on detecting and removing overlapping and intersected edges.

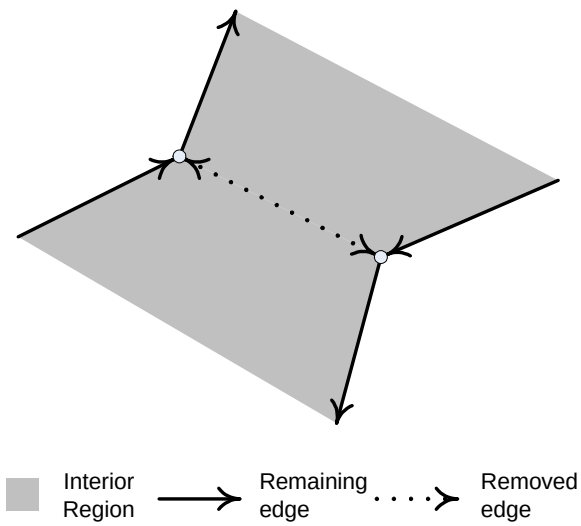


Figure 5.8. Removing and reconnecting overlapping edges

The original SNAKE model is based on physical laws. It is intuitive to explain our model by the example of soap bubbles. When two soap bubbles are merging, some parts of

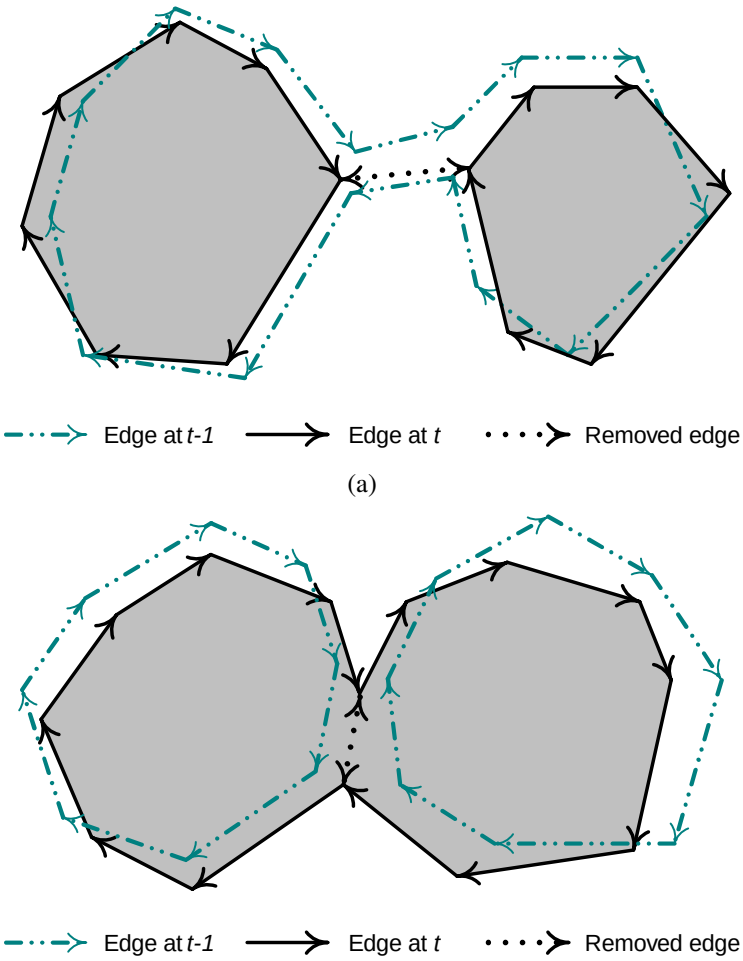


Figure 5.9. Example of splitting and merging
(a) splitting, and (b) merging

the bubble walls from two bubbles overlap first. Then the overlapping bubble wall breaks and two bubbles become a single bubble. Figure 5.8 shows a zoom-in picture of Figure 5.9(b). Let us consider one end-point of the overlapping edges in Figure 5.8. The end-point is actually covered by two different vertices that were previously located at different points but moved onto the same point. By removing the overlapping edges, we get two open curves. One of the vertices on the same point then has the right edge removed; another one has the left edge removed. The two vertices are locally reconnected and merge into a single vertex. The merged vertex now has the left and right edges from remaining edges of the previous two vertices. In this way, two open curves are reconnected into a single closed curve.

When a bubble is splitting, a part of the bubble wall overlaps another part from the same bubble, as illustrated by Figure 5.9(a). This can also be represented by Figure 5.8. The only difference is that the edge direction is reversed, and the interior and exterior regions are reversed. An interesting observation from Figure 5.8 is that detecting overlapping edges can be done locally on distributed nodes.

Due to the discrete distribution of sensor nodes, the vertex movement cannot be continuous. In some cases (e.g., uneven node distribution), edges may intersect with each other. Since the closed curves are simple, the intersected edges need to be removed. We get open curves after removing the intersected edges. As explained by Figure 5.10, those vertices located on the non-boundary region should be removed, when the edge intersection occurs.

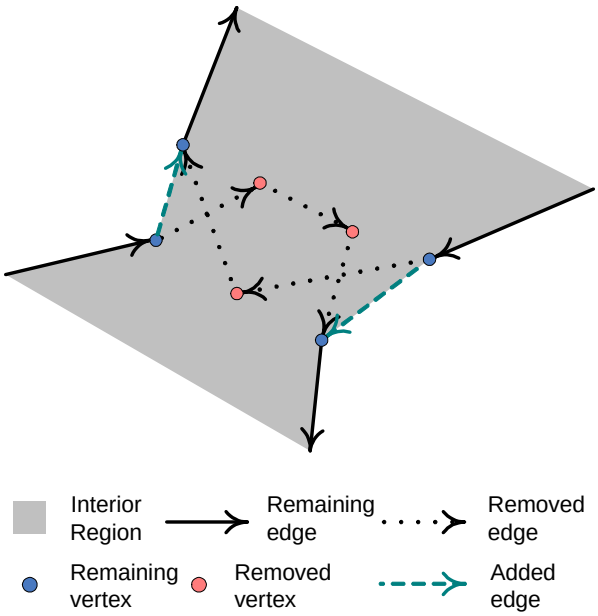


Figure 5.10. Removing and reconnecting intersected edges

For a pair of removed intersected edges, two vertices (one without left neighbor, another one without right neighbor) may remain. A new edge, therefore, should be added here to reconnect the open curves, as shown by Figure 5.10. The two vertices on the open curves consequently are consecutive vertex neighbors. As explained above, D_{split} is the longest edge length. Suppose R_{comm} indicates the communication range of wireless radio. Due to the broadcasting nature of wireless channel, if $D_{split} \leqslant \sqrt{2}R_{comm}$, no additional communication is required to detect the intersected edges based on our in-network deformable curve tracking.

5.3 Algorithms

In the proposed approach, vertex location information is exchanged among neighboring vertex nodes. When a vertex v_i^t is located at a particular sensor node, the sensor node needs to know the locations of v_{i-1}^t and v_{i+1}^t. By assuming that the vertices are facing the exterior region, we use the "LEFT" and "RIGHT" relations to identify the neighboring vertices. For the vertex node of v_i^t, we use two local variables, $leftVertex$ and $rightVertex$, to store its left and right neighboring vertices v_{i-1}^t and v_{i+1}^t. A timer is used in our implementation to control the sensors and the vertex movement. When time elapses from t to $t+1$, sensors collect new local readings. Afterwards, sensor nodes exchange local object and boundary detection results. We use GPSR [KK00] as the communication protocol, and assume a position service running on the background [LJD+00, DF03]. Sensor nodes therefore communicate with each other based on their locations. We also assume that the background services handle node failures and communication failures.

5.3.1 Pseudo-codes and Description

Based on the neighboring boundary detection at time $t+1$, a vertex node uses the location of previous neighboring vertices at time t to calculate the vertex's next location, as shown by Table 5.1.

After exchanging local boundary detection results, a vertex node finds nearby nodes, which detect the object boundary. If a neighboring node in the neighboring area detects

Table 5.1. Algorithm of finding the next location of v_i^t

Require: Sensor nodes exchange local object detection, $objectDetected$, and boundary detection results among immediate neighboring nodes, $NSensors$. The neighboring boundary reports is stored into a point array $NBReports$.

Ensure: v_i^t moves to a node at location v_i^{t+1} with minimal energy.

```
if NBReports.length ≠ 0 then
    CandLocs ⇐ NBReports
else
    for all s ∈ NSensors do
        if objectDetected =FALSE then
            if s INSIDE ∠v_{i+1}^t v_i^t v_{i-1}^t then
                CandLocs.add(s)
            end if
        else
            if s OUTSIDE ∠v_{i+1}^t v_i^t v_{i-1}^t then
                CandLocs.add(s)
            end if
        end if
    end for
end if
MinE ⇐ +∞
r ⇐ v_{i+1}^t
l ⇐ v_{i-1}^t
for all s ∈ CandLocs do
    E_ten ⇐ Var(|s − l|, |s − r|)
    E_cur ⇐ Var(π, ∠rsl)
    E ⇐ αE_ten + βE_ten
    if E < MinE then
        MinE ⇐ E
        nextLoc ⇐ s
    end if
end for
return nextLoc
```

the boundary, the node is a candidate for the vertex's next location as shown by Table 5.1. If the vertex node cannot detect the boundary in the nearby area, the node uses the local angle's topology information and its local object detection result to find the next candidate location. If a vertex node detects neither the object nor the object boundary in the nearby area, the candidate locations are within the interior region defined by the local angle. If a vertex node detects the object but does not find the object boundary in the nearby area, the candidate locations are within the exterior region, as illustrated by Table 5.1. Among the candidate locations, the location with the minimal tension and curvature energy is the next location for the vertex. A designation message is sent to the sensor node located at the next location. When a sensor node receives a vertex movement message, the sensor node caches the vertex movement into a cached vertex movement array, CVM. An element in CVM contains the vertex's previous location, and the vertex's previous left and right neighboring vertices. Since multiple vertices may move onto a single node at the same time, Table 5.2 is used to fold multiple vertices.

Multiple vertices may move onto the same sensor node. After receiving a vertex movement message, a sensor node caches the vertex movement into a vertex movement array, VM. An element in VM contains the vertex's previous location, and the vertex's previous left and right neighboring vertices. Some vertices can be folded into a single vertices (e.g., consecutive left and right neighboring vertices), as illustrated by Table 5.2. By comparing the $LEFT$ and $RIGHT$ relationships among vertices, Table 5.2 folds consecutive vertices

Table 5.2. Algorithm of folding consecutive vertices

Require: A sensor node receives multiple vertex movement messages and caches the messages into cached vertex movement array VM.

Ensure: Folding vertices on the local sensor nodes and prepare the vertices list VL.

```
VL ⇐ VL.init()
repeat
    mostLeft ⇐ VM.getFirst()
    VM ⇐ VM.remove(mostLeft)
    repeat
        toRepeat ⇐FALSE
        for all m ∈ VM do
            if mostLeft.preLeft = m.preLocation then
                VM ⇐ VM.insert(mostLeft)
                mostLeft ⇐ m
                VM ⇐ VM.remove(m)
                toRepeat = TRUE
            end if
        end for
    until toRepeat =FALSE
    mostRight ⇐ mostLeft
    repeat
        toRepeat ⇐FALSE
        for all m ∈ VM do
            if mostRight.preRight = m.preLocation then
                mostRight ⇐ m
                VM ⇐ VM.remove(m)
                toRepeat =TRUE
            end if
        end for
    until toRepeat =FALSE
    v ⇐ newVertex()
    v.preRight ⇐ mostRight.preRight
    v.currentRight ⇐ null
    v.preLeft ⇐ mostLeft.preLeft
    v.currentLeft ⇐ null
    VL ⇐ VL.insert(v)
until VM.length = 0
return VL
```

Table 5.3. Algorithm of adding a vertex

Require: Vertex nodes update their current location to the left and right neighbors.
Ensure: Adding a new vertex operation if the distance between a vertex and its right vertex is larger than D_{split}.

1: **if** $|myLocation - rightVertex| > D_{split}$ **then**
2: $\quad v.x \Leftarrow \frac{myLocation.x + rightVertex.x}{2}$
3: $\quad v.y \Leftarrow \frac{myLocation.y + rightVertex.y}{2}$
4: $\quad newVertex \Leftarrow PositionService.FindNearestNode(v)$
5: $\quad notifyNewVertexTo(rightVertex)$
6: $\quad designateNewVertex(newVertex)$
7: $\quad rightVertex \Leftarrow newVertex$
8: **end if**

into a single vertex and inserts the vertex into the vertex list, VL. If vertices are not consecutive (e.g., the vertices are from two 2D objects), VL may contain multiple vertices. Until now, a vertex movement is finished. The new vertex node then notifies the vertex's current location to its previous left and right vertex nodes. The current left and right vertex nodes may get the message through the previous left and right vertex nodes. The updated location messages are exchanged only among neighboring vertices through necessary relays. After receiving the updated location of its neighboring vertices, a vertex node knows the locations about its current right and left vertices. After the vertex updates are done, a vertex node checks the distance to its right vertex. If the distance is larger than D_{split}, a new vertex is added in between, as illustrated by Table 5.3.

The new vertex is the middle point of the local vertex and its right neighboring vertex. Since sensor nodes are discretely distributed, the nearest sensor node to the middle point is

found through the background position service [DF03]. As shown by Table 5.3, if a new vertex is inserted, the nearby vertex links are updated, and the new vertex node is notified.

After receiving the updates from neighboring vertices, a sensor node needs to update the corresponding vertex entry in the vertex list, VL. An element in VL, therefore, contains the locations of the vertex's current right and left neighboring vertices. Based on the content of VL, a sensor node detects the overlapping edges locally. After removing overlapping edges, the open curves need to be reconnected. Some vertices may also be removed accordingly as illustrated by Table 5.4. If no vertex remains (e.g., a vertex has its current right and left neighboring vertices overlapping), the sensor node becomes a non-vertex node. The edge between the local sensor node and $brokenNeighbor$ indicates the removed overlapping edges. The location of $brokenNeighbor$ is useful to determine whether the topological change is *splitting* or *merging*. After removing overlapping edges and reconnecting open curves, the remaining vertex moves based on the force models afterwards.

In the in-network deformable curve tracking, vertex nodes need to update the current vertex locations to neighbors. Vertex nodes can detect the intersected edges based on the broadcasted vertex location updates. No additional communication is required over the deformable curve tracking, if $D_{split} \leqslant \sqrt{2}R_{comm}$. After the pair of intersected edges are detected, IE, the four vertex nodes need to be notified. Some vertices may need to be removed if the vertices are not located on the object boundary, as illustrated by Table 5.5. Intersected edges need to be removed. We need to close the open curves by reconnecting

Table 5.4. Algorithm of removing overlapping edges and reconnecting open curves

Require: A sensor node folds multiple vertices and has the current neighboring vertices' locations updated into the vertices list VL.

Ensure: Removing overlapping edges and corresponding vertices; reconnecting open curves; reporting the removed edge.

```
for all v ∈ VL do
    for all o ∈ VL do
        if v.currentLeft = o.currentRight then
            brokenNeighbor ⇐ v.currentLeft
            v.currentLeft ⇐ null
            o.currentRight ⇐ null
        end if
    end for
end for
for all v ∈ VL do
    if v.currentLeft = null AND v.currentRight = null then
        VL.remove(v)
    end if
end for
for all v ∈ VL do
    for all o ∈ VL do
        if v.currentLeft = null AND o.currentRight = null then
            v.currentLeft ⇐ o.currentLeft
            VL.remove(o)
        end if
    end for
end for
return brokenNeighbor
```

Table 5.5. Algorithm of removing intersected edges and reconnecting open curves

Require: The intersected edges have been detected; an IE structure contains the four vertices of the intersected edges; consequently, four vertices have been notified about the intersection and run this algorithm.

Ensure: Removing intersected edges and corresponding vertices; reconnecting open curves; reporting the removed edges and vertices.

```
if localBoundaryStatus = false then
    resignVertex(mySelf)
    return reportRemovedVertex(myLocation)
end if
for i = 0 to 1 do
    if i = 0 then
        j ⇐ 1
    else
        j ⇐ 0
    end if
    if myLocation = e[i].leftVertex then
        rightVertex ⇐ e[j].rightVertex
        return reportRemovedEdge(e[i])
    else if myLocation = e[i].rightVertex then
        leftVertex ⇐ e[j].leftVertex
        return reportRemovedEdge(e[i])
    end if
end for
```

remaining vertices, as explained by Table 5.5. Similar to the $brokenNeighbor$ in Table 5.4, the location of intersected edges in Table 5.5 also helps to determine the type of this topological change.

5.3.2 Discussion

We assume the initial curve V^0 is given. The initial curve V^0 can be found by distributed algorithms [SO05, GHS07, LL07], or from the distributed detection result based on the different models [XLCL06]. For example, the emergence of a 2D object matching a user-defined shape can provide the initial boundary V^0. Due to the constrained environment, the V^0 shape given by a distributed object detection is usually coarse, such as a simple rectangle [XLCL06]. Our tracking algorithm changes and optimizes the shape and location of V^0 based on the revised SNAKE model to effectively attach to the 2D object. Similar results can be found in the related studies on SNAKE [Coh91].

The proposed tracking algorithm for deformable curves maintains the curves by localized message exchange. When a vertex moves, the vertex node sends a designation message to one of its immediate neighbors, and resigns. The new vertex node reports the updated vertex location to the previous left and right vertex nodes. The previous left and right vertex nodes may need to relay the update messages to the vertices' current locations. D_{split} roughly bounds the geographical range for a vertex update message to be transmitted. If the curve keeps a constant number of vertices, the maintenance cost of the tracking algorithm

is constant. If the curve expands and requires more vertices, the maintenance cost increases linearly to the number of vertices.

D_{split} also controls the number of vertices along a closed curve. If D_{split} is large, fewer vertices are added when a curve deforms. D_{split} is useful to control the quality of the deformable curve to represent the underlying 2D object. Similar techniques have also been applied to simplify the curve shape [GHS07]. Compared with reporting points along the object boundary, the network requires less communication to send linked vertices, if D_{split} is large. The difference is approximately scaled by D_{split}, since only the two end vertices of a line with $length = D_{split}$ represents the whole set of points along the line.

The location of $brokenNeighbor$ is useful to locally judge the type of the topological change, as explained by Algorithm 5.4. After removing overlapping edges, a sensor node forms a new angle, which has the remaining left and right edges as the new angle's left and right edges. By comparing Figure 5.9(b) and Figure 5.9(a), we shall see that if $brokenNeighbor$ is within the new angle, then the topological change is a merging event. If $brokenNeighbor$ is outside the angle, a splitting event occurs. In some cases, a 2D object may partially merge itself. For example, a band is bent into a ring. Similarly, a ring can be broken into a band. To better solve this issue on how to efficiently and locally determine the type of the topological change, we may need to assign unique identifications to 2D objects [FZWN08]. For example, a sensor node can combine the object ID of the removed edge with the topological test result based on the location of $brokenNeighbor$ to determine if a

ring is newly formed. We do not address this issue in detail, since it is beyond the scope of this dissertation.

As illustrated by Figure 5.8, Algorithm 5.4 requires no additional communication cost over the in-network deformable curve tracking. Detecting intersected edges may need additional communication cost. If D_{split} is small enough, vertex nodes are able to detect intersected edges through the broadcasting vertex location updates. After intersected edges are found, the four vertex nodes need to be notified. Algorithm 5.5 removes the intersected edges and reconnects the representative curves, as shown by Figure 5.10.

Based on the algorithms described in this section, a WSN is able to track 2D objects separately and their interactions in-network. A WSN can update the deformable curves to users and allow users to get the spatiotemporal properties from the geometric information. The deformable curve tracking algorithm provides more than just the snapshot results about representative curves. Based on the deformable curves, a WSN is able to directly extract abstract spatiotemporal properties of 2D objects without returning users the detailed geometric information about the representative curves.

5.4 Abstract Information

As explained by A. Galton, several abstract spatiotemporal properties are useful for cognition, linguistics and reasoning [Gal00]. In daily life, people can describe and exchange information about 2D objects by abstract spatial and spatiotemporal information without

any graphical aid. For example, a radio broadcast can report news about wild fires without giving any images or videos. In Section 5.2.2, we have explained how to adapt the representative curves to the topological changes involved by the interaction between multiple objects. In this section, we show how to efficiently compute other abstract spatial and spatiotemporal properties based on the in-network deformable curve tracking.

5.4.1 Aggregated Information

We use the aggregated information of deformable curves to extract the overall spatial and spatiotemporal properties about 2D objects.

$$MBR^t = \left(MIN(X^t), MIN(Y^t), MAX(X^t), MAX(Y^t)\right), \tag{5.11}$$

where

$$X^t = \left\{x_1^t, x_2^t, \cdots x_n^t\right\},$$

$$Y^t = \left\{y_1^t, y_2^t, \cdots y_n^t\right\}.$$

The *Minimal Bounding Rectangle* (MBR) of a 2D object is a simple geometry to approximate the 2D object. As shown by Equation 5.11, we use the aggregation operation to find the MBRs of 2D objects.

$$P^t = \sum_{i=1}^{n} \left|v_i^t - v_{i+1}^t\right|. \tag{5.12}$$

As illustrated by Equation 5.12, the perimeter value of a closed curve is found by aggregated information.

$$P^{t+1} - P^t = \sum_{i=1}^{n} \left(\left| v_i^{t+1} - v_{i+1}^{t+1} \right| - \left| v_i^t - v_{i+1}^t \right| \right). \tag{5.13}$$

The perimeter change is illustrated by Equation 5.13. If a pair of neighboring vertex nodes remain relatively unchanged, Equation 5.13 is useful to suppress unnecessary local reports about the perimeter calculation.

$$A^t = \frac{1}{2} \sum_{i=1}^{n} \left(x_i^t y_{i+1}^t - x_{i+1}^t y_i^t \right). \tag{5.14}$$

As indicated by Equation 5.14, the area value of a closed curve is expressed by an aggregated result. A vertex node prepares its local partial results based on its location and its right neighboring vertex. The area about the region covered by current curve, therefore, is aggregated through the partial results.

$$A^{t+1} - A^t = \sum_{i=1}^{n} \left(DA1_i^{t+1} + DA2_i^{t+1} \right), \tag{5.15}$$

where

$$DA1_i^{t+1} = \frac{1}{2} \left| v_i^{t+1} - v_i^t \right| \left| v_{i+1}^t - v_i^t \right| \sin \angle v_i^{t+1} v_i^t v_{i+1}^t, \tag{5.16a}$$

$$DA2_i^{t+1} = \frac{1}{2} \left| v_{i+1}^{t+1} - v_{i+1}^t \right| \left| v_{i+1}^{t+1} - v_i^{t+1} \right| \sin \angle v_{i+1}^t v_{i+1}^{t+1} v_i^{t+1}. \tag{5.16b}$$

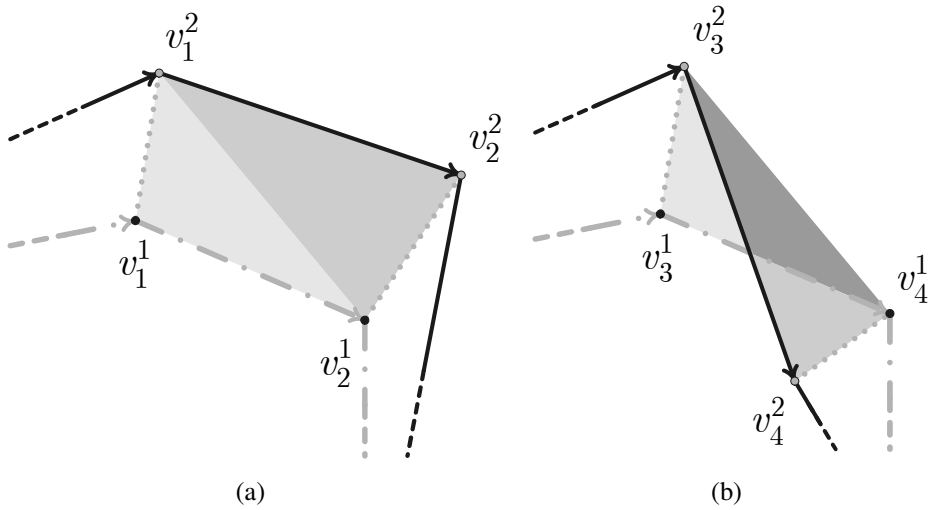


Figure 5.11. Examples of area changes

A variation of Equation 5.14 is the area change as indicated by Equation 5.15. As shown by Figure 5.11, a local vertex node prepares the local area change based on the nearby vertices' locations. The local area change values, $DA1_i^{t+1}$ and $DA2_i^{t+1}$, are signed scalars, as indicated by Equation 5.16a and 5.16b. For example, Figure 5.11(a) shows that there is a local enlarging defined by the two triangles $\triangle v_1^1 v_1^2 v_2^1$ with $area = DA1_1^2$ and $\triangle v_2^1 v_2^2 v_1^2$ with $area = DA2_1^2$. The local area change may also be negative values. For example, in Figure 5.11(a), the area of $\triangle v_4^1 v_4^2 v_3^2$, $DA2_3^2 < 0$, indicates a local shrinking. The area change operation, as indicated by Equation 5.15, can be used to suppress local partial aggregation reports. For example, if the local area change is zero, a vertex node can suppress the local report to its parent node.

$$C^t = (x_C^t, y_C^t), \tag{5.17a}$$

$$x_C^t = \frac{1}{6A^t} \sum_{i=1}^{n} \left[(x_i^t + x_{i+1}^t)(x_i^t y_{i+1}^t - x_{i+1}^t y_i^t) \right], \tag{5.17b}$$

$$y_C^t = \frac{1}{6A^t} \sum_{i=1}^{n} \left[(y_i^t + y_{i+1}^t)(x_i^t y_{i+1}^t - x_{i+1}^t y_i^t) \right]. \tag{5.17c}$$

The *centroid* of a 2D object is also called the center of mass or the center of gravity. We treat the centroid of a 2D object as a 2D point, as indicated by Equation 5.17a. Since the area of a 2D object is represented by aggregated information, the location of centroid is aggregated as indicated by Equation 5.17b and 5.17c.

$$\begin{aligned} A^{t+1}x_C^{t+1} - A^t x_C^t =& \frac{1}{3} \sum_{i=1}^{n} \left[DA1_i^{t+1} \left(x_i^t + x_i^{t+1} + x_{i+1}^t \right) \right. \\ & \left. + DA2_i^{t+1} \left(x_i^t + x_{i+1}^t + x_{i+1}^{t+1} \right) \right], \end{aligned} \tag{5.18a}$$

$$\begin{aligned} A^{t+1}y_C^{t+1} - A^t y_C^t =& \frac{1}{3} \sum_{i=1}^{n} \left[DA1_i^{t+1} \left(y_i^t + y_i^{t+1} + y_{i+1}^t \right) \right. \\ & \left. + DA2_i^{t+1} \left(y_i^t + y_{i+1}^t + y_{i+1}^{t+1} \right) \right]. \end{aligned} \tag{5.18b}$$

The centroid change is represented as a weighted sum, as indicated by Equation 5.18a and Equation 5.18b. A vertex node prepares the local partial result based on the nearby vertices' location changes and the area change. Similar to the area change, Equation 5.18a and Equation 5.18b can also be used to suppress the local partial aggregated results.

$$\overrightarrow{C^t C^{t+1}}. \tag{5.19}$$

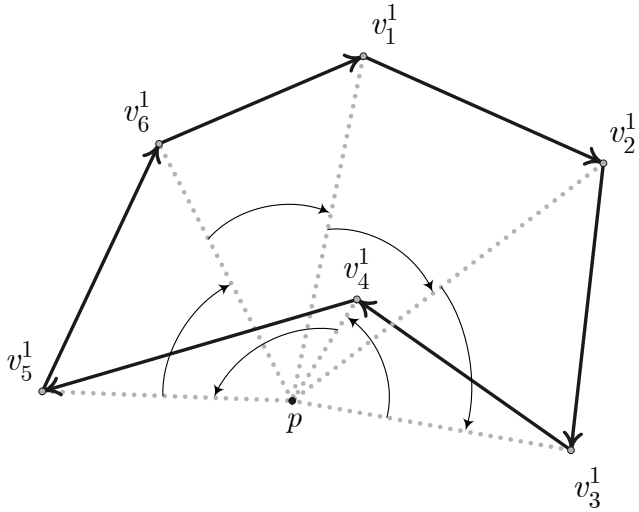
Based on the updates about the centroid's location, users can understand an object's overall location changes in the spatiotemporal space. Equation 5.19 describes the trajectory of the 2D object between time t and $t+1$. Based on Equation 5.19, we support spatiotemporal queries about the object's movement and moving direction. For example, "how fast is the 2D object moving?" and "is the 2D object moving north?"

$$\angle C^t p C^{t+1}. \tag{5.20}$$

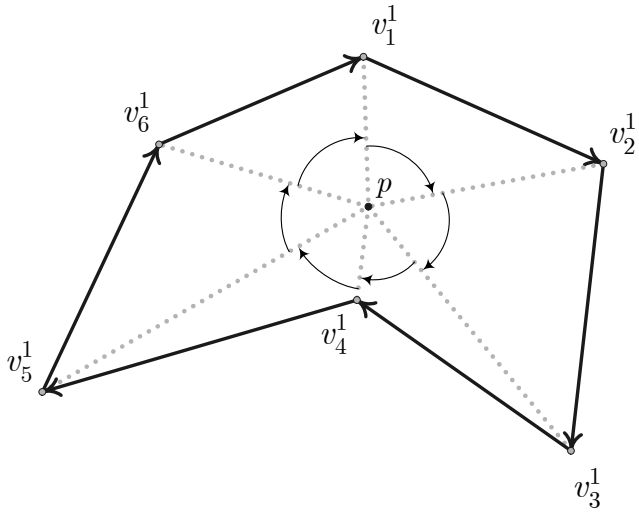
We also use the centroid to define the rotation information. The rotation information is defined by the centroid's change relative to a given point, p. Based on Equation 5.20, we can answer spatiotemporal queries about the object's rotation for the given point p, such as "is the 2D object moving anticlockwise for the point p?"

$$\sum_{i=1}^{n} \angle v_i^t p v_{i+1}^t = \begin{cases} 2\pi, & \text{inside;} \quad (5.21a) \\ 0, & \text{outside.} \quad (5.21b) \end{cases}$$

Many point-set topological relationships are based on the test of the $INSIDE$ relation. Users are also interested in queries like "is the 2D object covering a point p?" As indicated by Equation 5.21a and 5.21b, we do the $INSIDE$ test through an aggregated angle sum. A vertex node computes the local angle value defined by its location, its right vertex and the given point p. If a point p is not located inside the closed curve, the angle sum is zero,



(a)



(b)

Figure 5.12. Examples of $INSIDE$ relation test
(a) not inside, and (b) inside

as shown by Figure 5.12(a). If a point p is located inside the closed curve, the aggregated angle sum is 2π, as illustrated by Figure 5.12(b).

5.4.2 Predictive Information

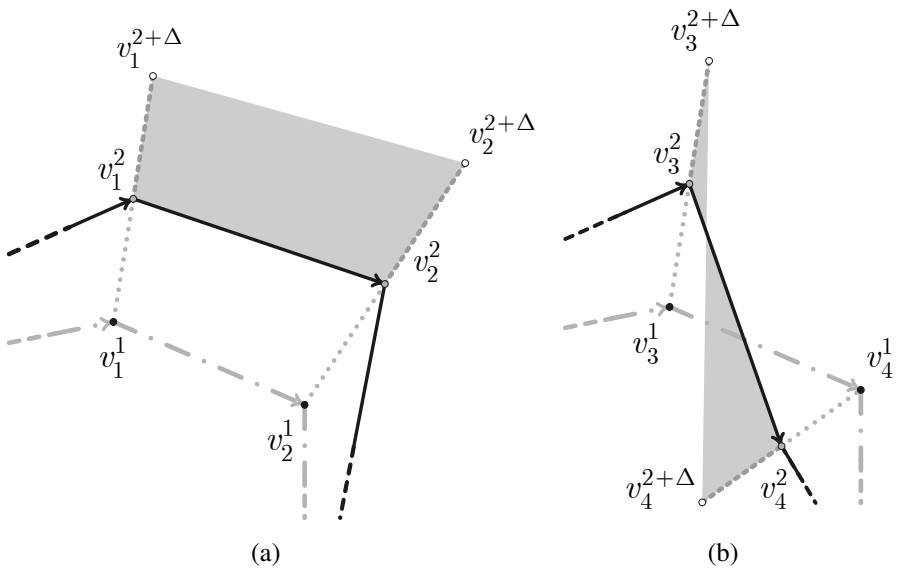


Figure 5.13. Examples of edge projection

$$x_i^{t+\Delta} = x_i^t + \Delta\left(x_i^t - x_i^{t-1}\right), \tag{5.22a}$$

$$y_i^{t+\Delta} = y_i^t + \Delta\left(y_i^t - y_i^{t-1}\right). \tag{5.22b}$$

The tracking of deformable curves also supports extrapolating the curves' future location and shape. This type of estimation is done based on the edge projection over time. We estimate a vertex's location in the near future based on Equation 5.22a and Equation 5.22b. A vertex node projects its right edge to find the region that may be affected by the

object in the near future. For example, in Figure 5.13(a), the sensor node at v_1^2 is able to extrapolate the location of v_1 at time $2 + \Delta$ based on the vertex movement from v_1^1 to v_1^2. Similarly, the sensor node is able to compute $v_2^{2+\Delta}$. The locations of v_1^2, $v_1^{2+\Delta}$, v_2^2 and $v_2^{2+\Delta}$ define a quadrangle. We use the localized edge projection, and test if a point p is inside this quadrangle to support spatiotemporal queries for the future. For example, "is the 2D object going to cover (or uncover) the point p in the next Δ time?" We use this localized operation to set real-time alerts for the near future. The quadrangle may be non-simple, as illustrated by Figure 5.13(b). A vertex sensor node may be folded from multiple vertices. In this case, the previous location is defined as the centroid of the multiple previous locations.

5.4.3 Discussion

Based on the in-network tracking of deformable curves, many types of spatial and spatiotemporal properties of 2D objects can be extracted by the aggregation operations. Compared with reporting boundary points or linked vertices, processing the aggregated information greatly reduces the communication consumption. The abstract spatial and spatiotemporal properties are useful for people to describe and exchange information about the underlying phenomena. By processing the aggregated information, an SDMS provides real-time reports about the spatial and spatiotemporal properties of 2D objects. For example, "how is the wildfire changing its area?", or "how is the wildfire moving?" In this way, an SDMS is able to provide useful spatial and spatiotemporal properties of 2D objects

for users and high-level reasoning mechanisms, while saving expensive communication to report the detailed geometric information about the objects.

Although an individual sensor node does not know the global and detailed geometric information about the 2D objects, the node is still able to detect local representative curve changes and extrapolate the local object boundary in the near future. Through the localized edge projection, distributed sensor nodes are able to provide real-time alerts for many applications, such as an emergency evacuation.

5.5 Chapter Summary

This chapter presents an efficient approach to track 2D objects by using WSNs. Our approach uses a deformable curve to represent and track a 2D object. The representative curve is also breakable. Therefore, the shape of deformable curves are adjusted according to the interactions between multiple 2D objects. In our approach, sensor nodes track individual vertices on the representative curve without knowing the global detailed geometric information about the curve. Sensor nodes only need to exchange messages among neighbors to maintain deformable curves. Consequently, our approach is resource-efficient to the constrained environment. Furthermore, based on the tracking algorithm, an SDMS is able to provide many abstract spatiotemporal properties of 2D objects through the in-network aggregation operations and localized edge projection. In this way, an SDMS can

directly answer qualitative spatiotemporal queries while the communication cost to return the detailed geometric representations to a base station is saved.

Chapter 6

EXPERIMENTAL EVALUATION

This chapter presents the experimental results of our approaches.

6.1 Analysis of SWOP

We assume that SWOP is implemented in a challenging environment (i.e., mobile WSNs). Here, we used ($128bits$) x-y coordinates to identify sensor nodes, chose the HEED-based clustering procedure, and assumed the node communication range is larger than the required cluster radius (i.e., a direct communication link between a non-head member and its cluster head). We implemented SWOP in Java and ran it over different data sets. In our simulations, the behavior of WSN was simulated by treating each sensor node as a thread running independently and communicating with each other by exchanging messages. The data sets consist of two real data sets from the CalCOFI survey off the coast of Southern California [BWS$^+$02] and from an experiment in the Intel Lab [Int04], and two synthetic

data sets. Without losing any generalization, we normalized the sensor readings to $[0, 1]$. Finding an optimal bandwidth has been researched well for Kernel estimation [LS97], and the fast optimization algorithm [RD06] for the Gaussian Kernel bandwidth is also available. Therefore, we only tested SWOP under pre-chosen bandwidths. The fixed bandwidth is also useful to test two synthetic data sets, since we compared the SWOP estimation results with alternative estimation techniques, including spatial regression and Voronoi-diagram, with regard to their processing costs based on the estimation quality. Most related solutions only compare their results with the results from the centralized solution [GBT$^+$04, SS04]. It is difficult to cross-evaluate different approaches since the code of other solution is not available or not compatible. In our experiments, we compared the estimation results of SWOP on "real" underlying phenomena (i.e., two synthetic data sets) to test the estimation quality. Our experiments demonstrate the high estimation quality of SWOP.

6.1.1 Coefficient Ordering Strategy and Error Evaluation

SWOP returns the distribution of an underlying phenomenon for a given region. An efficient in-network query processing should target to minimize the difference between results of the traditional centralized techniques and itself. The following tests are based on the average MSE from multiple runs. We first compared the MSEs between the results of SWOP and centralized Kernel estimation as shown by Table 6.1 based on different truncation strategies. Table 6.1 confirms that by ordering the polynomial order of Hermite

coefficients, SWOP achieves high quality results while relaxing the data requirement compared with taking the p^2 terms by the original FGT. Aggregating more Hermite coefficients with higher polynomial-order decreases the difference between results of SWOP and the centralized Kernel estimation. Since the largest MSE values between results of SWOP based on zero polynomial order Hermite coefficients and the centralized Kernel estimation results are around 10^{-3}, we performed other quality tests based on the zero-order Hermite coefficients.

The first two real data sets only provide us with point samples of a realistic underlying phenomenon. There is no reading available between the point samples. From the two synthetic data sets, we pick a part of the readings as input point samples, and use other readings as the "real" phenomenon values. The two synthetic data sets allow us to compare the estimated results with "real" values as shown by Table 6.2. Based on our parameter choice for the first synthetic data set, the mean squared errors between the SWOP result and the "real" phenomenon are around 10^{-3}. The SWOP result of #1 set is reliable for many practical purposes. We fixed the bandwidth for the second synthetic data to test SWOP against alternative approaches, although the MSE on the second synthetic data indicates an over-smoothed result. In practice, the method introduced by [RD06] can help users to find the optimal bandwidth according to different phenomenon distributions.

Table 6.1. Cost and quality based on different truncating strategies

p	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Number Of Total Terms In 2D Space based on different truncating strategies										
p-1 order	1	3	6	10	15	21	28	36	45	55
p terms	1	4	9	16	25	36	49	64	81	100
MSE on the salinity data based on different truncating strategies										
p-1 order	6.93E-04	5.20E-04	7.06E-05	2.01E-05	3.98E-06	8.67E-07	1.66E-07	2.78E-08	5.22E-09	7.18E-10
p terms	6.93E-04	4.53E-04	2.48E-05	1.06E-05	7.88E-07	2.54E-07	1.86E-08	4.18E-09	2.83E-10	4.55E-11
MSE on the Intel Lab data based on different truncating strategies										
p-1 order	1.03E-03	3.19E-04	1.01E-04	1.39E-05	4.76E-06	5.30E-07	1.27E-07	1.55E-08	2.18E-09	3.01E-10
p terms	1.03E-04	3.19E-04	4.55E-05	5.46E-06	1.18E-06	1.21E-07	1.62E-08	1.79E-09	1.33E-10	1.76E-11
MSE on the synthetic data #1 based on different truncating strategies										
p-1 order	7.51E-04	5.24E-04	5.39E-05	9.08E-06	2.19E-06	3.42E-07	5.27E-08	6.47E-09	8.35E-10	7.54E-11
p terms	7.51E-04	4.96E-04	1.01E-04	1.67E-05	5.96E-06	8.55E-07	2.33E-07	2.74E-08	6.36E-09	6.17E-10
MSE on the synthetic data #2 based on different truncating strategies										
p-1 order	2.58E-03	3.64E-04	3.03E-04	2.42E-05	1.29E-05	1.13E-06	4.12E-07	3.70E-08	1.04E-08	8.76E-10
p terms	2.58E-03	3.98E-04	1.78E-04	1.27E-05	4.72E-06	3.47E-07	8.44E-08	6.19E-09	1.23E-09	7.32E-11

Table 6.2. Mean squared errors relative to "real" values

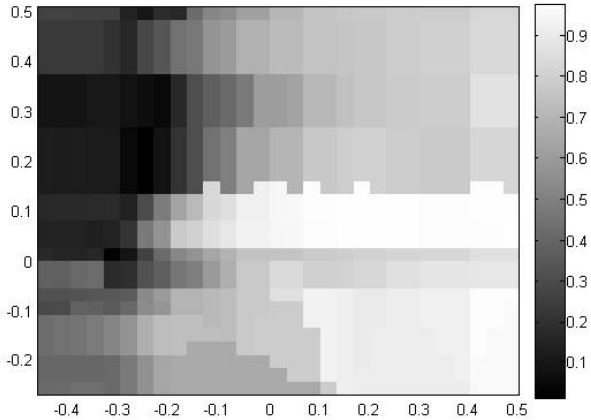
Data set	Kernel	SWOP
Synthetic #1	4.6E-03	7.16E-03
Synthetic #2	3.23E-02	4.40E-02

6.1.2 Estimation Results

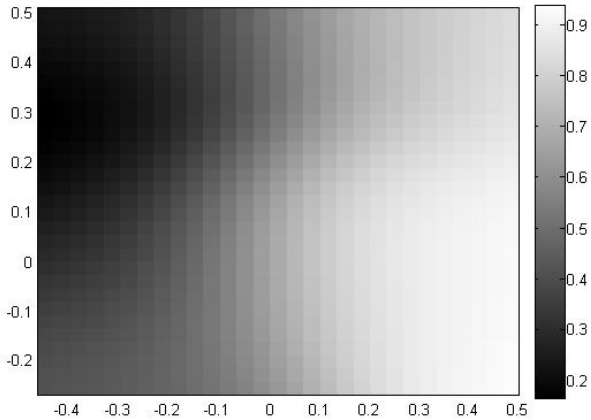
To demonstrate the estimation result using SWOP, we ran SWOP a multitude of times for each data set. The estimation results with the highest compression rates were chosen for display.

The first data set has 372 measurements of salinity density off the coast of Southern California in the CalCOFI survey [BWS$^+$02], based on which a 30×30 unit estimation map with $\tau = 0.2$ is generated. Figure 6.1(b) shows the estimation result based on the traditional centralized Kernel estimation while the result using SWOP with 0 order coefficients and the result based on Voronoi-diagram are shown in Figure 6.1(c) and Figure 6.1(a) respectively. In this example, the x-coordinate is the distance from the coast, y indicates the depth of the sample from the mean sea surface. Negative values in x indicate in-land river water readings. A lighter point in Figure 6.1 indicates the saltier water.

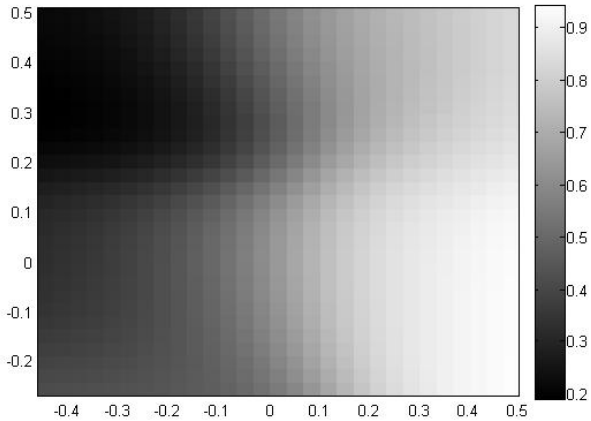
Figure 6.2 illustrates the second test based on a smaller data set from the Intel Lab. In this data set, 48 point temperature samples of sensor nodes distributed over the ceiling of a Intel lab were taken from a snapshot during an experiment in the Intel Lab [Int04]. A 30×30 unit map is estimated. The results based on Voronoi-diagram, the centralized



(a)

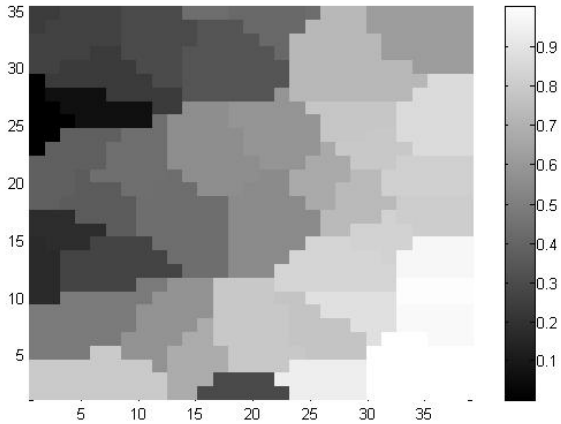


(b)

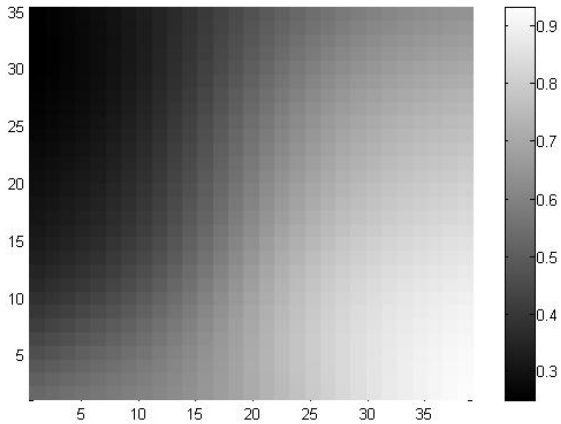


(c)

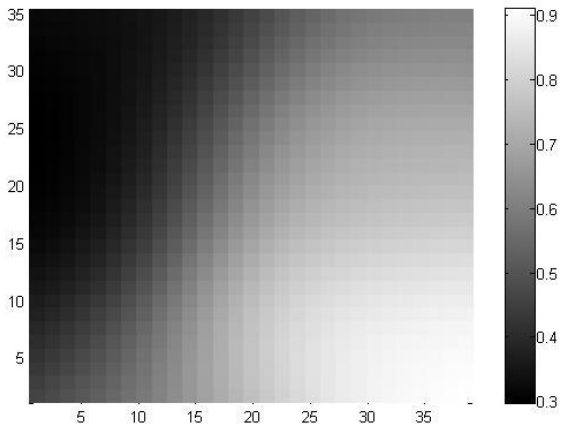
Figure 6.1. Query results on the salinity data
(a) the result based on Voronoi diagram, (b) the result of centralized Kernel, and (c) the result of SWOP



(a)



(b)

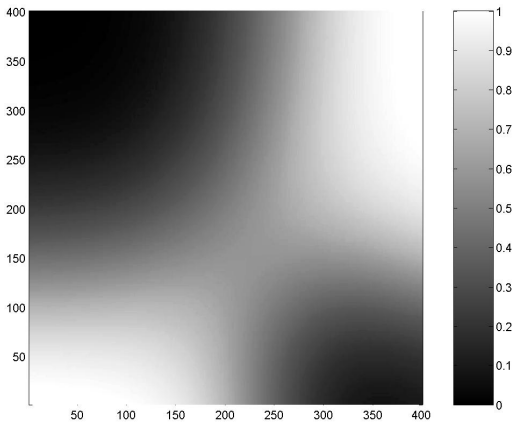


(c)

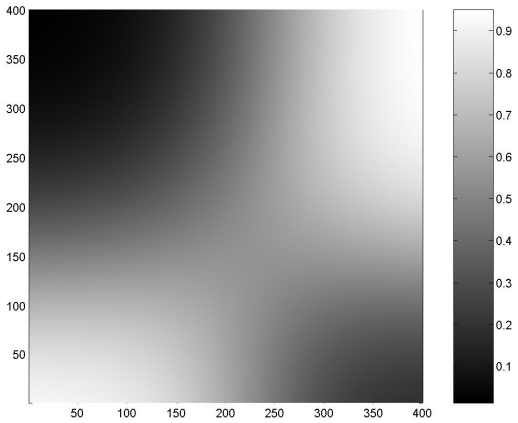
Figure 6.2. Query results on the Intel lab data
(a) the result based on Voronoi diagram, (b) the result of centralized Kernel, and (c) the result of SWOP

Kernel estimation with $\tau = 11$ and SWOP with 0-order coefficients are shown by Figure 6.2(a), Figure 6.2(b) and Figure 6.2(c) respectively, where a darker point indicates the colder temperature.

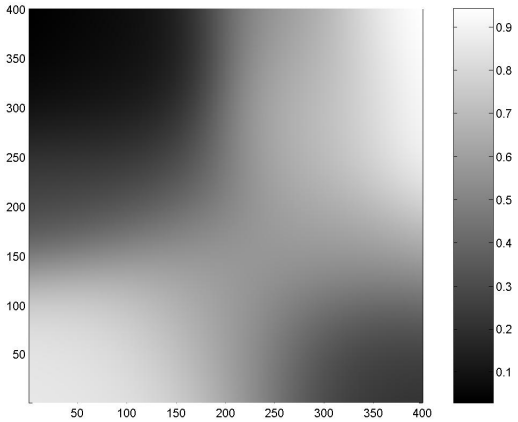
Both measured data sets (i.e., salinity and Intel-Lab data sets) only provide point samples, but the validation of the estimation quality compared to the real underlying phenomenon values is not possible. However, we can compare SWOP's estimation quality with regard to other estimation methods, performed in a central setting. Furthermore, we use two synthetic data sets to test the effectiveness of SWOP. Two 401×401 unit continuous gray scale pictures were synthetically generated as shown in Figure 6.3(a) and Figure 6.4(a). These two data sets can be interpreted as two different distributions of a "real phenomenon". For example, we can assume two gas leaks in the upper-left and lower-right corner of Figure 6.3(a). We set $\tau = 80$ to test the performance of SWOP based on 21×21 point samples taken from the underlying "phenomenon" at the interval of 20 pixels. Figure 6.3(b) and Figure 6.4(b) illustrate the results of centralized Kernel estimation. Figure 6.3(c) and Figure 6.4(c) show the SWOP estimation results with 0-order coefficients for the two synthetic data sets. For the fixed bandwidth, both the centralized Kernel estimation and SWOP return a truthful estimation result on the synthetic data #1. For the second data set, the two small "gas leaks" are obscured, which indicates an over-smoothed result. The result of SWOP based on the 0-order Hermite coefficients is a little distorted compared with



(a)

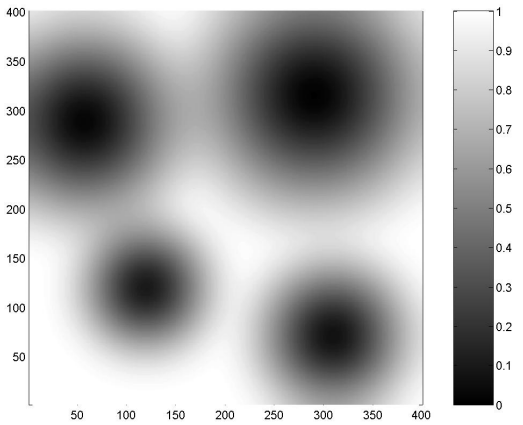


(b)

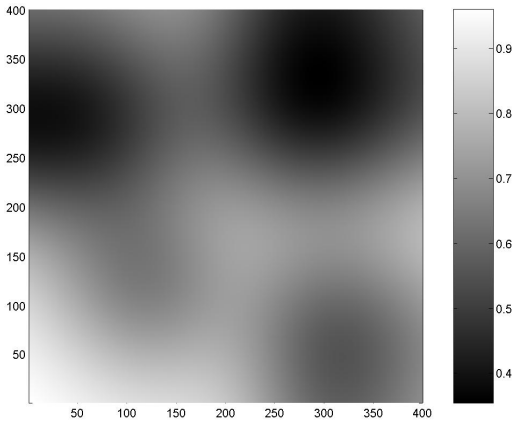


(c)

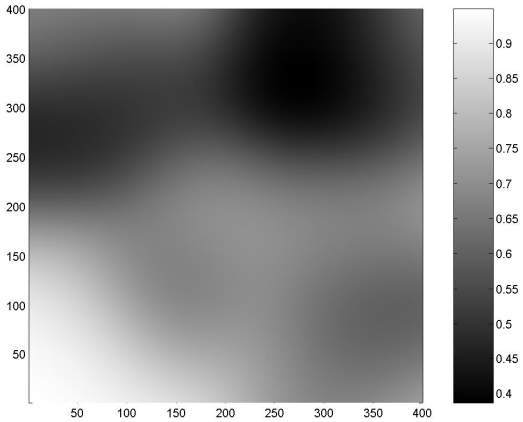
Figure 6.3. Query results on the synthetic Data #1
(a) the original data, (b) the result of centralized Kernel, and (c) the result of SWOP



(a)



(b)



(c)

Figure 6.4. Query results on the synthetic Data #2
(a) the original data, (b) the result of centralized Kernel, and (c) the result of SWOP

the centralized Kernel estimation result. Here, we set the bandwidth fixed on purpose to compare the quality of SWOP estimation results with other alternative estimation results.

The estimation results based on Voronoi-diagram depict the layout and readings of sensor nodes directly, but the results are coarse compared to the results based on Kernel estimation. Furthermore, the cost of processing a Voronoi-diagram-based approach limits its application in the constrained WSNs. Whereas, even compared with the "real" phenomena, the results of SWOP still directly illustrate the phenomenon's distributions.

6.1.3 Cost Evaluation

In our tests, we use one double value ($64bits$) to represent a sensor reading and two double values ($128bits$) to represent a sensor node identity (i.e., its location). We recorded the average number of clusters and the average size of raw data and SWOP data for each cluster based on 0-order Hermite coefficients from multiple independent tests on each data set, as shown by Table 6.3. The clustering algorithm plays an important role in SWOP for the compression gain. After being clustered, a non-head node requires 192 bits to send its reading and ID to the cluster head. The message size for each cluster head to represent its cluster members depends on the chosen order of Hermite coefficient. For the zero polynomial order, each cluster head needs 256 bits to represent its member nodes for both the numerator and denominator in Equation 3.9. Since small clusters just send their raw readings, the average message size is a little smaller than 256 bits as shown by

Table 6.3. Compared with transmitting raw data for each cluster, SWOP saves 94% in the communication cost. The total communication cost of a network depends on different communication protocols and network layouts. It is difficult to simulate SWOP for all cases, and thus, we only consider the size of data collected from cluster heads.

Table 6.3. Required data size for each cluster(in bit)

Data set	**# of clusters**	**Raw data**	**SWOP**
Salinity	21	3475.39	253.85
Intel-lab	8.8	1093.12	249.7
Synthetic #1	23.1	3572.66	251.4
Synthetic #2	22.7	3730.04	252.3

6.1.4 Comparison with Alternative Approaches

Wavelet and Delta Compression

A compression technique can be applied in clustering protocols to compress raw sensor readings for each cluster. We implemented the Haar wavelets and let cluster heads transform raw readings and node IDs into wavelets. Table 6.4 illustrates the experiment results on the two real data sets for different wavelet coefficient settings. An advantage of wavelets is that they can represent data in different scales and compress data losslessly based on which we can apply any analytical models. As shown in Table 6.4, the Haar wavelets can compress lossless data in about 60% size of the raw data for each cluster. In our experiments, we only compared the centralized Kernel estimation results based on the wavelet data with the Kernel estimation results on original data. By eliminating small

wavelet coefficients, we can achieve higher compression rates, but degrade the estimated results. However, to achieve a similar quality of SWOP, wavelet-based methods require a larger data size than SWOP does. More tests on the synthetic data sets and the Delta compression show similar results to wavelets, therefore we exclude the detailed comparison about them. By evaluating the Haar wavelets, the Delta-compression and SWOP, we conclude that SWOP requires less communication cost but still returns high quality estimation results.

Table 6.4. Evaluation on wavelets

Coefficient threshold		**Data size**	**MSE**
readings	node ID		
Intel-Lab data			
0	0	698.98	0
0.2	0	449.71	9.53E-04
0.4	0	391.79	1.71E-02
0	6	661.98	1.23E-03
0	10	649.04	2.63E-03
0.3	6	310.82	8.53E-03
0.4	6	317.28	2.49E-02
0.3	8	323.61	8.6E-03
0.4	8	329.81	2.92E-02
Salinity data			
0	0	2896.26	0
0.2	0	1967.57	9.63E-04
0.4	0	1919.22	1.11E-02
0	0.10	1529.13	1.93E-03
0	0.2	1534.45	7.43E-03
0.2	0.1	651.84	1.97E-03
0.3	0.1	687.52	7.43E-03
0.2	0.15	619.79	6.43E-03
0.3	0.15	594.22	1.33E-02

Spatial Regression

Since we fixed the bandwidth for both synthetic data sets, we compare SWOP with different 2D spatial regression methods on the synthetic data sets based on different estimation qualities. We did our tests to evaluate the estimation results against the "real" phenomenon values and the cost of processing alternative approaches in the network.

Table 6.5. Evaluation on 2D polynomial regression

Polynomial Order	**MSE**	**# of** $f()s$
Synthetic data #1		
1	7.5E-02	3
2	1.2E-02	6
3	4.8E-03	10
4	1.5E-03	15
Synthetic data #2		
1	6.2E-02	3
2	6.0E-02	6
3	4.9E-02	10
4	2.6E-02	15

We ran different 2D spatial regression methods in a traditional centralized setting on raw data. Table 6.5 shows the results based on different orders of polynomial regressions. With higher orders of polynomial equations, the estimation results reach higher quality. To achieve a similar estimation quality of SWOP with the current bandwidth setting, a 2D spatial polynomial regression requires 10 or more basis functions for both synthetic data sets. For Kernel regression, we tested different numbers of kernels based on different kernel functions separated at fixed intervals with different bandwidths. Table 6.6 illustrates the

minimal MSE based on different numbers of kernels and different kernel functions. Table 6.6 also shows the chosen bandwidth and kernel-center interval for the different kernel functions to return the best estimation results based on different numbers of kernels. To achieve a similar quality of SWOP, the Kernel regression requires 9 or more kernels. Figure 6.5 and Figure 6.6 show the estimation results based on the cubic polynomial, and the best estimation results based on 9 cone kernels and 9 Gaussian kernels for synthetic data #1 and #2 respectively.

Generally, both regression estimation methods require 9 or more basis functions to achieve a similar or better quality of SWOP. To return the final estimation results, we need at least $(81 + 9) \cdot k_i$ data from the network. Applying several types of kernel functions decreases the size of data exchanged among neighboring nodes, but the estimation results are not smooth due to the discontinuity of the kernel functions (e.g., the estimation results based on cone kernels Figure 6.5(b)). On average, for both synthetic data sets, SWOP returns around 23 clusters, and requires a similar size of data, about $23 \cdot 4 \cdot k_i$, from the network for a similar quality compared to the 2D spatial regression methods. However, almost all nodes involved in regression methods need to receive and send the same large size of data. In SWOP, only the cluster heads near to the central base or a micro-server need to communicate with the large-sized messages. The nodes within a cluster and the nodes at the bottom on a routing tree in SWOP reduces their communication costs. Furthermore, for the current cluster radius setting $0.9(80) = 72$ and the current spatial window size $401 \times$

Table 6.6. Evaluation on Kernel regression

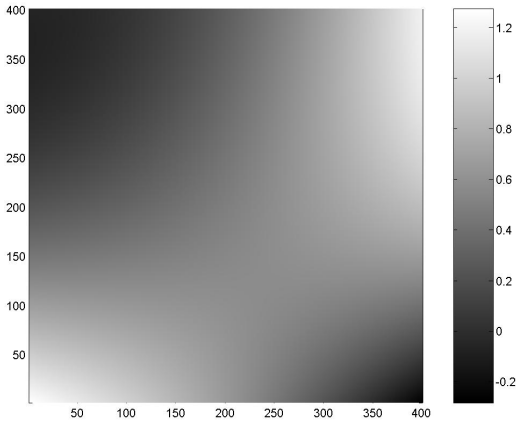
# of kernels	**Block kernel**			**Cone kernel**			**Gaussian kernel**		
	Min MSE	Kernel Interval	Bandwidth	Min MSE	Kernel Interval	Bandwidth	Min MSE	Kernel Interval	Bandwidth
Synthetic data #1									
16	8.37E-03	110	165	4.06E-04	130	195	4.60E-04	110	95
9	2.09E-02	140	210	1.72E-03	140	210	1.15E-03	160	130
4	6.84E-02	200	300	1.37E-02	210	315	1.19E-02	210	265
Synthetic data #2									
16	2.96E-02	120	260	2.83E-02	110	325	3.85E-02	130	65
9	3.78E-02	170	255	4.13E-02	160	310	4.19E-02	150	75
4	6.26E-02	200	300	5.82E-02	200	300	5.89E-02	170	165

401 unit, a compact clustering pattern should contain less than 9 clusters. The distributed clustering algorithm does not return a good clustering pattern. SWOP can achieve a higher compression gain by applying more sophisticated clustering methods.

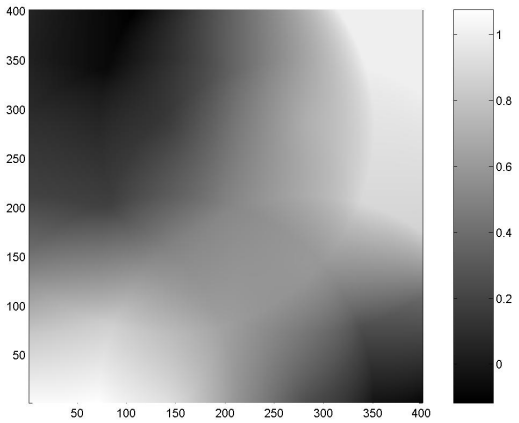
Regression estimation methods focus on minimizing global errors, while SWOP and non-parametric estimation methods focus on revealing local variations. If we compare the estimation results of SWOP and regression estimation methods with the "real" underlying phenomenon values, the local change is better preserved by SWOP than by regression estimation methods for the similar global quality, MSE. For example, in Figure 6.6(a), one of the small peaks totally disappeared.

6.2 Analysis of NED

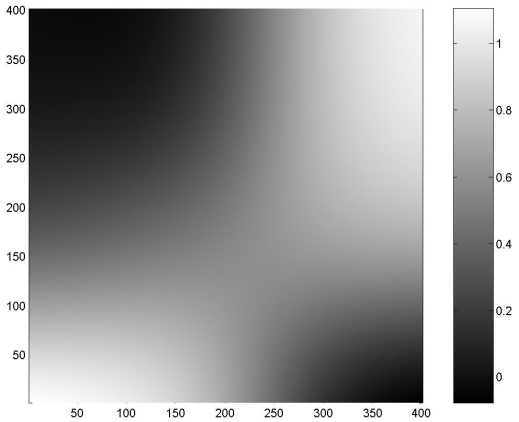
We simulated NED using MatLab. To test the performance of NED, we used a graphic tool to generate several gray-level pictures in which the gray-level values represent the underlying phenomena. The gray-level value is represented as from 0 (pure white) to 1 (pure black) without loss of generality. The unit distance is 1 pixel distance in our experiments. A 101×101 picture, as shown by Figure 6.7, is used by most of our experiments on NED. The phenomenon illustrated by Figure 6.7(a) continuously changes over the space. The cross section at $Y = 50$ clearly indicates the continuity of the phenomenon as shown by Figure 6.7(b). Sensor nodes were assumed to be located distributed over the graphic area and took the local pixel gray values as the sensor readings. We applied normal white noise



(a)

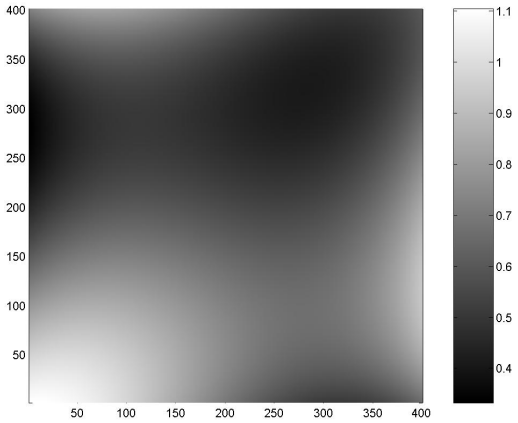


(b)

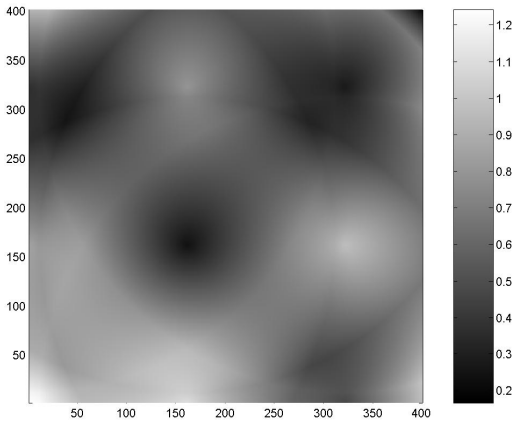


(c) Gaussian Kernel

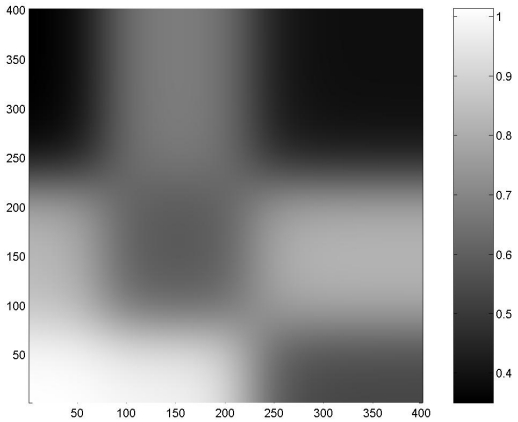
Figure 6.5. Alternative estimations on the synthetic data #1
(a) the result of Polynomial regression, (b) the result of Cone Kernel regression, and (c) the result of Gaussian Kernel regression



(a)



(b)



(c)

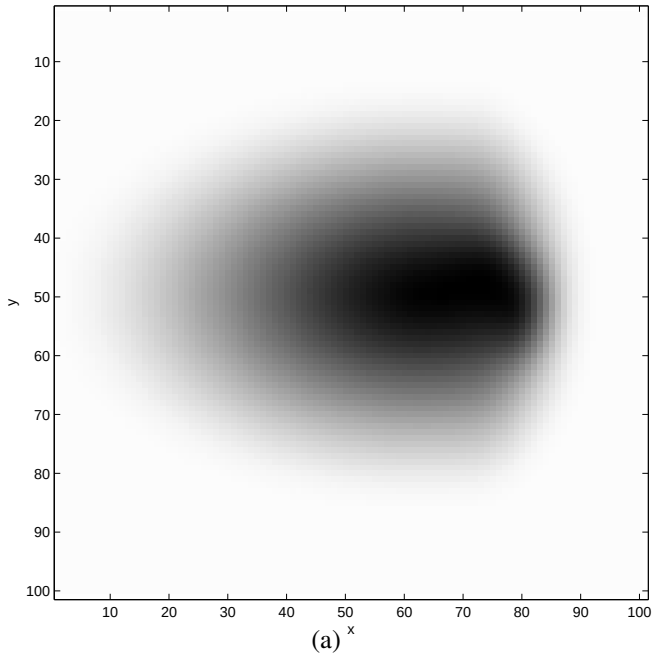
Figure 6.6. Alternative estimations on the synthetic data #2
(a) the result of Polynomial regression, (b) the result of Cone Kernel regression, and (c) the result of Gaussian Kernel regression

to each sensor reading. We also assume that sensor nodes can communicate with each other within the distance 5, and run different estimation methods 500 times for different parameter settings. In the following tests, Δ_1 and Δ_2 are set to the 95% confidence level if there is no more specification.

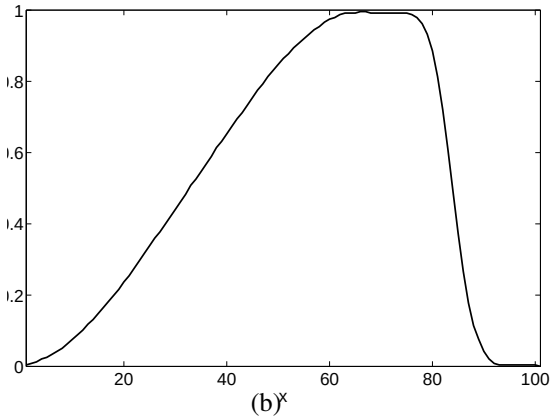
6.2.1 Object and Boundary Detection

For the first experimental test set, the sensor nodes were located in a grid layout. The distance between two neighboring nodes is 3 pixel distance. Figure 6.8 shows an object detection result based on $T = 0.5$ with different noise levels. In Figure 6.8, the object is located inside the solid line. The dots indicate the sensor nodes that detect the object, whereas the circles indicate the nodes detecting the non-object. Figure 6.8(a)-6.8(d) illustrate the results with noise variance settings $\sigma = 0.1$ to $\sigma = 0.4$ respectively. As we can see, NED effectively estimates the object distribution with the noise setting, $N(0, 0.1^2)$. For more noisy readings, the estimation result of NED is degraded, but is still acceptable.

Figure 6.9 shows the boundary detection results of NED, in which the solid line indicates the exact boundary and the circles are the sensor nodes reporting the boundary. Figure 6.9(a)-6.9(d) show the boundary detection results with noise variance settings $\sigma = 0.1$ to $\sigma = 0.4$ respectively. The boundary detection result based on $N(0, 0.1^2)$ is still the best detection quality. The estimation quality decreases as the noise variance increases. The result based on $\sigma = 0.4$ is not as clear as the result based on smaller σ values. But the



(a)



(b)

Figure 6.7. A synthetic phenomenon
(a) the field distribution, and (b) the cross section at $y = 50$

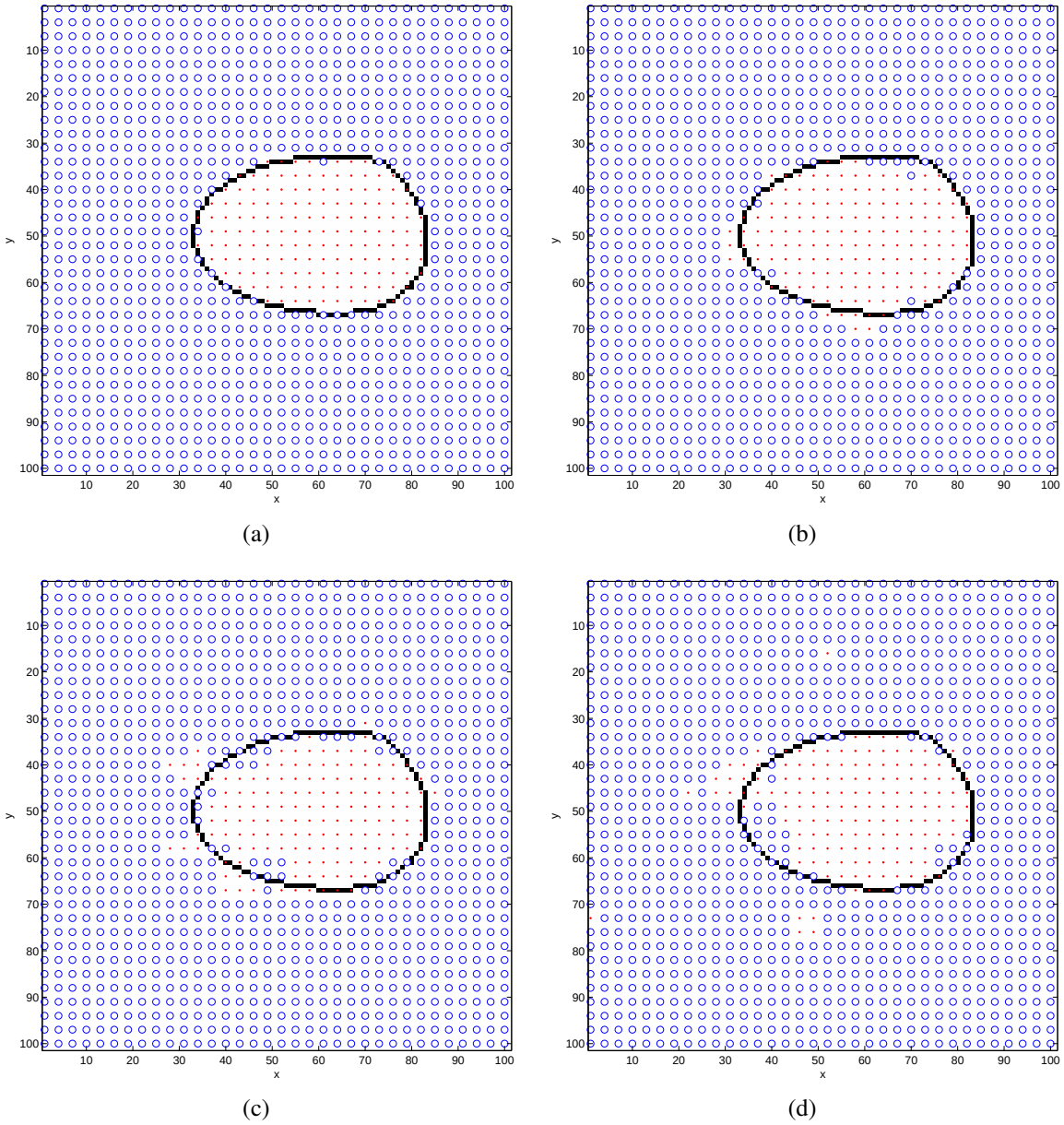
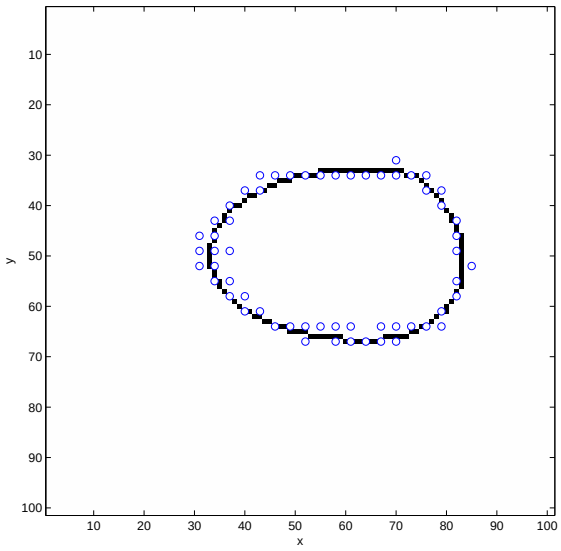
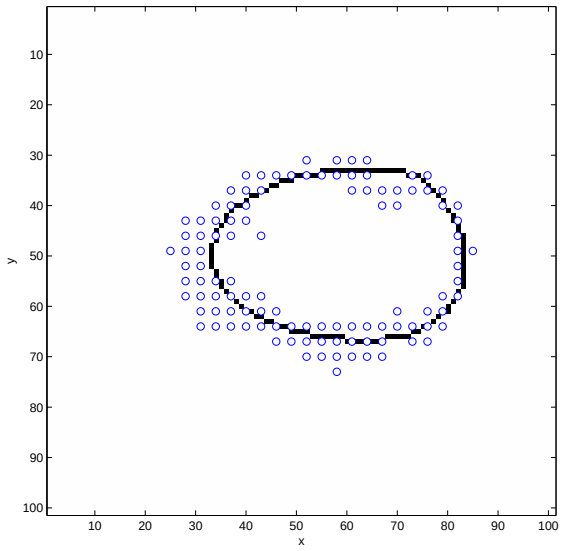


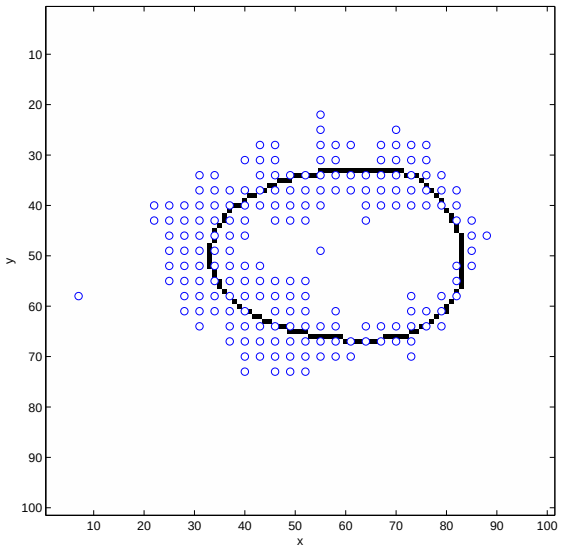
Figure 6.8. Object detection results with $T = 0.5$ under different noise levels (a) $\sigma = 0.1$, (b) $\sigma = 0.2$, (c) $\sigma = 0.3$, and (d) $\sigma = 0.4$



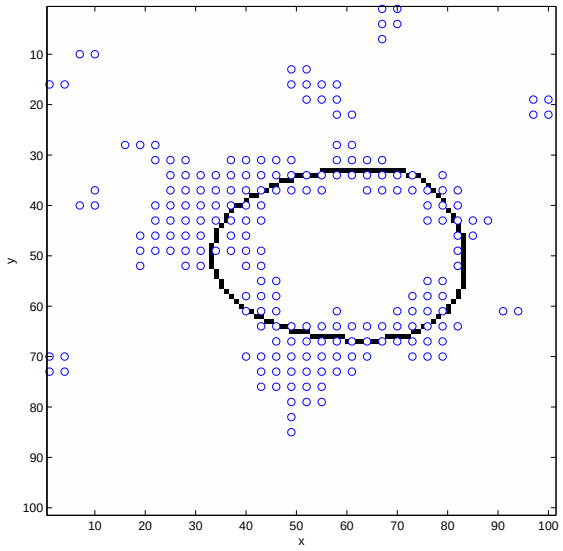
(a) $\sigma = 0.1$



(b) $\sigma = 0.2$



(c) $\sigma = 0.3$



(d) $\sigma = 0.4$

Figure 6.9. Boundary detection results with $T = 0.5$ under different noise levels (a) $\sigma = 0.1$, (b) $\sigma = 0.2$, (c) $\sigma = 0.3$, and (d) $\sigma = 0.4$

result based on $\sigma = 0.4$ delivers still discernable results, although the noise level, $\sigma = 0.4$ is large compared with the boundary setting $T = 0.5$.

NED supports arbitrary settings on the object boundary thresholds. Figure 6.10 shows the boundary detection results based on different threshold settings with the noise setting $\sigma = 0.1$. For the moderate noise level setting, NED returns precise boundary detection results. The sensor nodes around the object boundaries successfully report the boundary locations. Figure 6.10 indicates that the size of object $Y(p) \geqslant 0.8$ is smaller than the size of object $Y(p) \geqslant 0.6$, as we can observe from Figure 6.7.

We conducted other tests to simulate mobile sensor nodes. As illustrated by Figure 6.11, we randomly selected 1500 pixels from the simulated phenomenon to provide the locations and readings of sensor nodes. A white normal noise with variance $\sigma = 0.1$ was also applied to each reading. As shown by Figure 6.11(a), NED returns a clear object boundary detection result based on the threshold $T = 0.5$. Figure 6.11(b) illustrates the nodes almost perfectly report the object status.

We used a binary phenomenon as shown by Figure 6.12(a) to test the performance of NED on discontinuous phenomena. The cross section at $y = 30$, in Figure 6.12(b), shows that the phenomenon is a step function across the space. We set the boundary threshold $T = 0.5$ and the noise variance as $\sigma = 0.1$ to test the performance of NED. Since the phenomenon is not spatially continuous around the object boundary, we set the significance level of Δ_2 to 99% to increase the boundary estimation quality. Figure 6.12(c) shows the

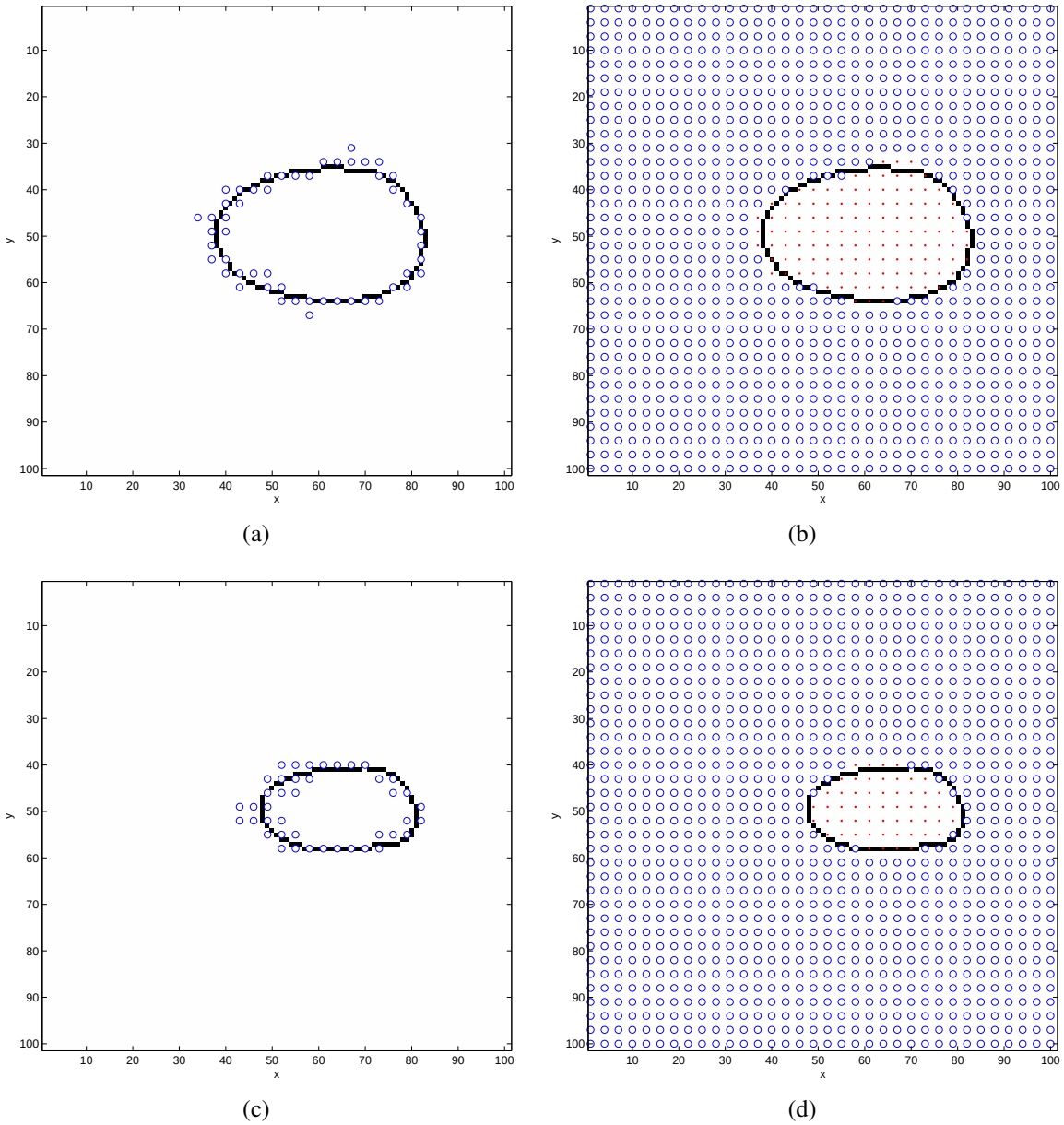
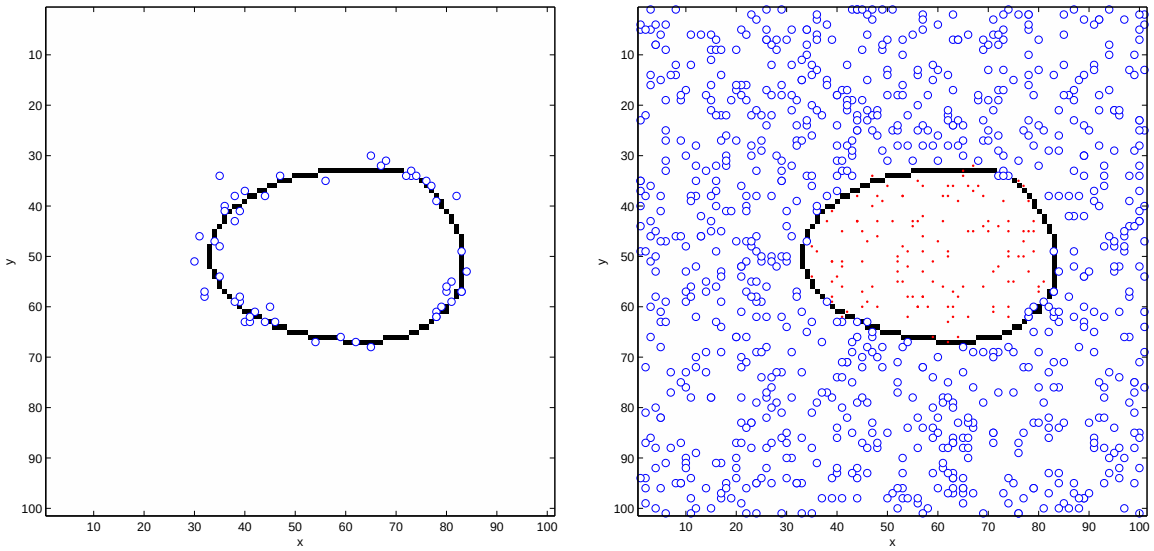


Figure 6.10. Detection on arbitrary thresholds
(a) the boundary detection on $T = 0.6$, (b) the object detection on $T = 0.6$, (c) the boundary detection on $T = 0.8$, and (d) the object detection on $T = 0.8$



(a) (b)

Figure 6.11. Detection results based on random layouts
(a) the boundary detection, and (b) the object detection

sensor nodes successfully report the object boundary. The nodes almost perfectly detect the object as illustrated by Figure 6.12(d). Overall, Figure 6.12 exemplifies the effectiveness of NED on discontinuous phenomena.

6.2.2 Estimation Quality of NED

We ran NED and alternative approaches a multitude of times to test the object and object boundary estimation quality. Here, we use triple values in the format of ($minValue$, $meanValue$, $maxValue$) to indicate the object detection quality. The $minValue$ ($maxValue$) indicates the lower (upper) bound of the number nodes which successfully report the local object status in a test from 500 tests. The $meanValue$ indicates the average number of

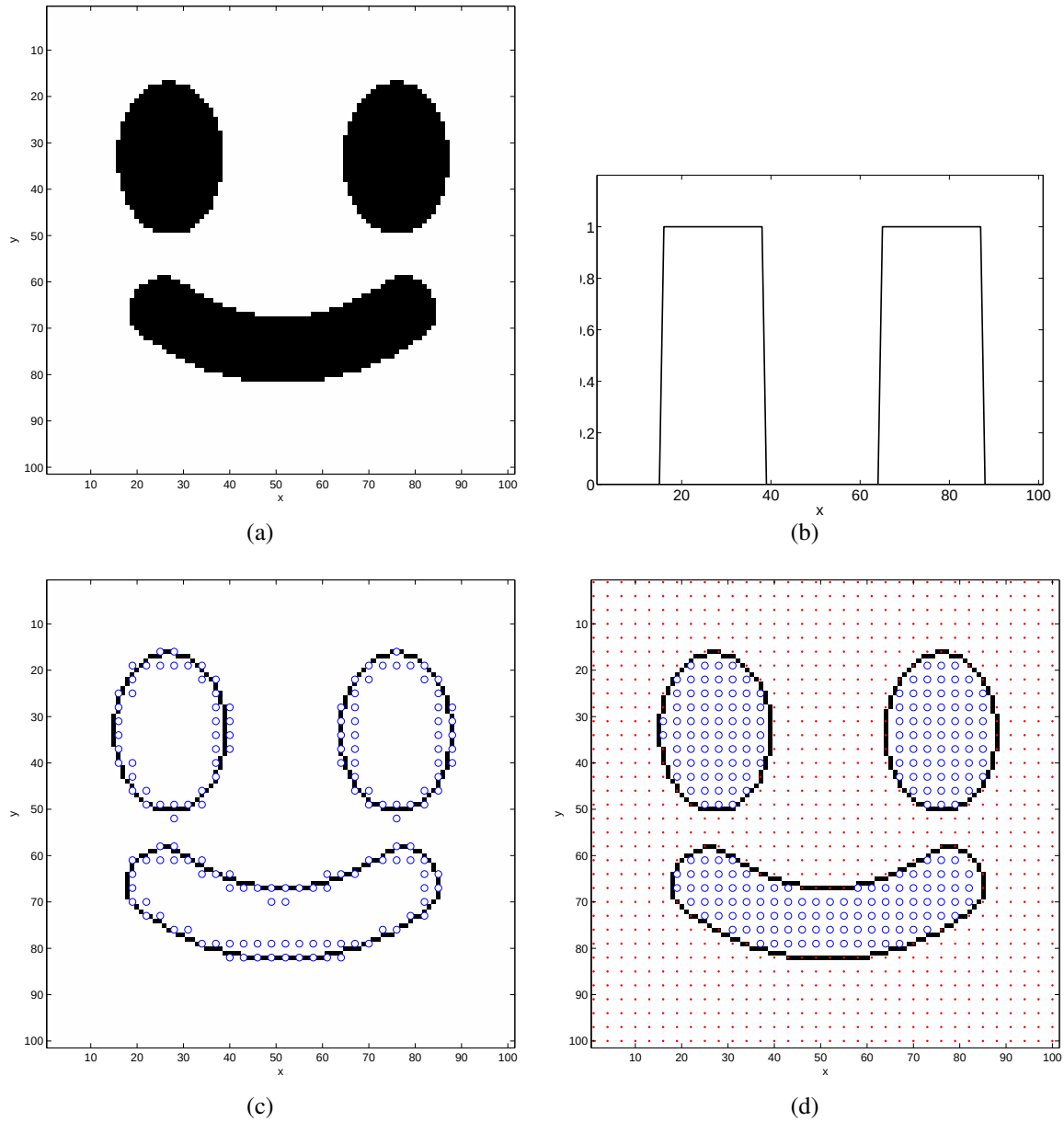


Figure 6.12. NED results on a binary phenomenon
(a) the field distribution of the phenomenon, (b) the cross section at $y = 30$, (c) the NED boundary detection result, and (d) the NED object detection result

nodes which successfully reports from the multiple tests. Table 6.7 illustrates the quality of estimation results for $T = 0.5$ with different noise variance settings based on the grid-like network layout from 500 tests. We used different methods to estimate the local phenomenon value. The nodes in large spatial distance from the object boundary rarely make erroneous estimation results. All tested methods achieved a detection success rate over 90%. We set the corresponding significance level to 95% for Δ_1 in NED, and tested the performance of the uncompressed moving arithmetic average and median methods without transforming significant float readings into binary values. While the variance of noise increases, the estimation quality of different methods decreases. As shown by Table 6.7, the estimation quality of NED and uncompressed arithmetic average method are almost the same and the best among these methods. The moving median and majority voting methods report more erroneous results than NED does. One possible explanation is the limited number of neighboring nodes, which restricts the performance of methods based on moving median and majority voting.

6.2.3 Effectiveness of Δ_1

NED allows users to choose different Δ_1 settings to encode corresponding local significant object estimation results into binary messages. Different Δ_1 settings can affect both the estimation quality and the communication cost.

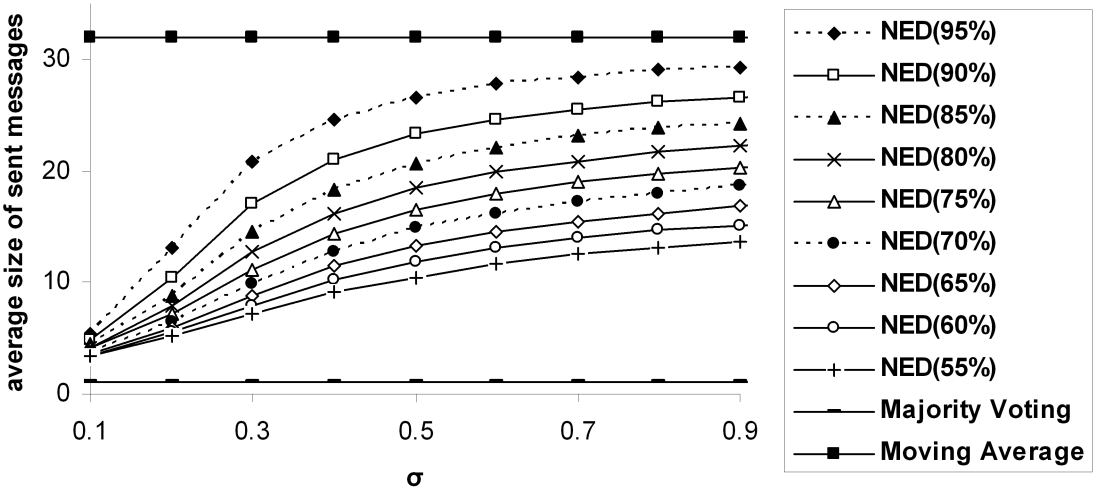
Table 6.7. Estimation quality

σ	**# of successful estimations(min,mean,max)**			
	NED	Moving Arithmetic Average	Moving Median	Majority Voting
0.1	(1132,1146.5,1154)	(1135,1146.8,1153)	(1130,1142.7,1150)	(1130,1142.7,1150)
0.2	(1115,1135.9,1147)	(1118,1136.2,1146)	(1105,1130.1,1145)	(1105,1130.1,1145)
0.3	(1101,1124.4,1140)	(1101,1124.8,1138)	(1083,1115.8,1134)	(1083,1115.8,1134)
0.4	(1071,1113.3,1133)	(1071,1113.6,1132)	(1056,1101.5,1127)	(1056,1101.8,1128)
0.5	(1056,1097.3,1124)	(1056,1097.7,1122)	(1028,1079.5,1118)	(1029,1080.6,1120)
0.6	(1036,1080.6,1111)	(1044,1081.4,1113)	(989,1054.5,1097)	(990,1056.9,1099)
0.7	(1007,1060.9,1098)	(1010,1061.7,1098)	(958,1027.7,1078)	(963,1031.8,1080)
0.8	(992,1037.8,1082)	(991,1038.9,1083)	(919,999.92,1057)	(925,1005.8,1063)
0.9	(944,1013.1,1062)	(943,1014.4,1062)	(885,970.62,1033)	(892,977.68,1045)
1.0	(914,989.49,1052)	(914,990.78,1057)	(865,945.36,1011)	(871,953.48,1019)

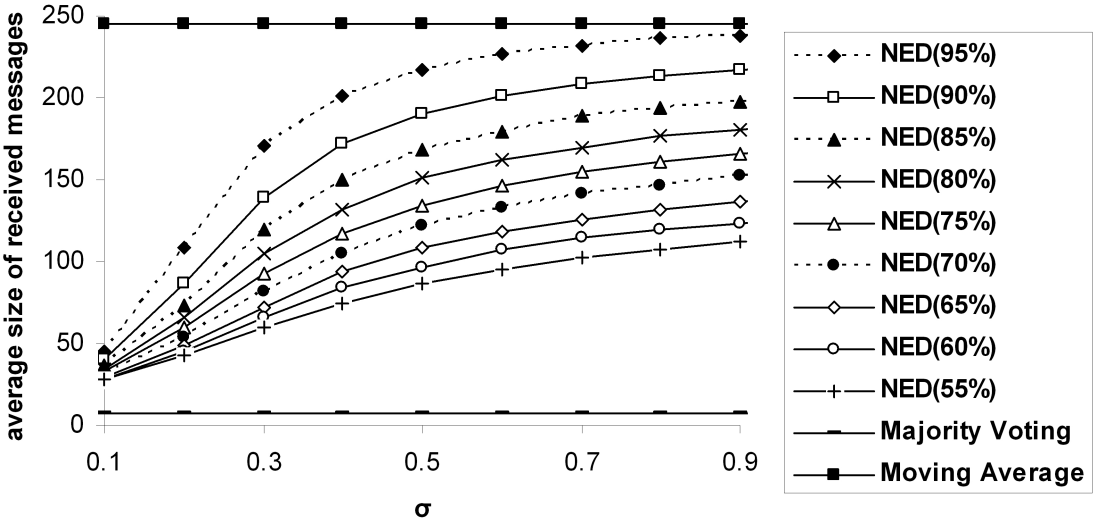
Table 6.8. Estimation quality for different Δ_1 significant levels

σ	**Significant Level of Δ_1(min,mean,max)**			
	90%	80%	70%	60%
0.1	(1133,1146,1151)	(1133,1145.7,1152)	(1130,1144.9,1151)	(1131,1143.7,1149)
0.2	(1116,1135.2,1147)	(1118,1134.8,1144)	(1111,1134.2,1145)	(1114,1132.5,1148)
0.3	(1087,1123.6,1140)	(1085,1123.1,1138)	(1099,1122.2,1137)	(1080,1120.6,1138)
0.4	(1084,1111.9,1129)	(1074,1110.8,1134)	(1076,1110,1135)	(1079,1106.5,1129)
0.5	(1059,1099.2,1125)	(1057,1097.5,1124)	(1059,1095.1,1124)	(1047,1090.9,1121)
0.6	(1035,1081.3,1115)	(1035,1079.9,1115)	(1023,1076.2,1107)	(1025,1071.1,1106)
0.7	(990,1060.6,1106)	(1014,1058.7,1109)	(998,1053.9,1097)	(988,1047.8,1101)
0.8	(983,1038.6,1082)	(978,1036.3,1088)	(957,1031.2,1072)	(951,1023.9,1070)
0.9	(938,1014.5,1077)	(945,1012.8,1065)	(929,1007.5,1067)	(920,999.08,1054)
1.0	(893,993.85,1063)	(883,991.55,1048)	(921,985.16,1060)	(900,976.24,1037)

Table 6.8 depicts the estimation quality of different Δ_1 settings under different noise variances from 500 tests. As the Δ_1 value decreases, the estimation quality of NED degrades and becomes closer to the majority voting. For a small noise effect, the difference between different Δ_1 settings is small as illustrated by Table 6.8.



(a)



(b)

Figure 6.13. Data requirement of NED
(a) the average size of sent data, and (b) the average size of received data

Wireless radio communication is one of the most resource consuming components of processing in WSNs. Figure 6.13 shows the average data requirement for the object detection results based on $T = 0.5$ with different noise levels from the grid network layout. As explained by Figure 6.13(a), the majority voting method only needs 1 bit to encode local object estimation whereas the uncompressed moving arithmetic average or median methods require 32 bits to encode a sensor reading. The average size of received data, however, depends not only on particular algorithms, but also on the wireless radio communication range. Nodes can hear from each other within the distance of 5. A node receives more data than it sends, as shown by Figure Figure 6.13(b). Overall, Figure 6.13(a) and Figure 6.13(b) show similar results. The communication requirement of NED is between the two methods. The Δ_1 setting also affects the communication cost. A small Δ_1 can reduce the communication cost of NED. When the noise effect is small, users can set a small Δ_1 to achieve a low-cost communication and still maintain a good estimation quality.

6.2.4 Effectiveness of Δ_2

NED uses Δ_2 to control the object boundary estimation. Δ_2 represents the confidence level of local boundary estimation results. Based on Δ_2, distributed sensor nodes are able to report significant boundary estimation results. Figure 6.14 shows boundary estimation results for different Δ_2 settings. As the Δ_2 value increases, the width of the estimated

boundary gets "thinner" as illustrated by Figure 6.14(a) to Figure 6.14(d). Some nodes around the boundary may fail to report the boundary if Δ_2 is too small.

6.3 Evaluation on Tracking Deformable Curves

We implemented and tested the proposed distributed deformable curve tracking algorithm in TinyOS [LLWC03], and used CLDP [KGKS05], which is an enhanced TinyOS implementation of GPSR as the communication protocol. We run our codes in TOSSIM [LLWC03], and set the simulated environment as follows: the network was set to a grid layout and in the network, 169 sensor nodes were distributed evenly in a 100×100 unit 2D space at the interval of 8 unit. The root node was located at $(2, 2)$, and connected to a base station. The wireless radio range was set to 10 units, which allowed a sensor node to directly communicate with up to four neighbors within the range. The weights α and β were equal to 1. Sensor nodes collected sensor readings based on video clips to simulate a dynamic continuous phenomenon. Each sensor node collected sensor readings from the corresponding pixel values in the video clips based on the node's location. Table 6.9 summarizes the parameter settings in our experiments.

Table 6.9. Parameter settings

Parameter	**Value**	**Parameter**	**Value**
Network Layout	Grid	Network size	169
Node Interval	8 unit	R_{comm}	10 unit
α	1	β	1
Root location	$(2, 2)$		

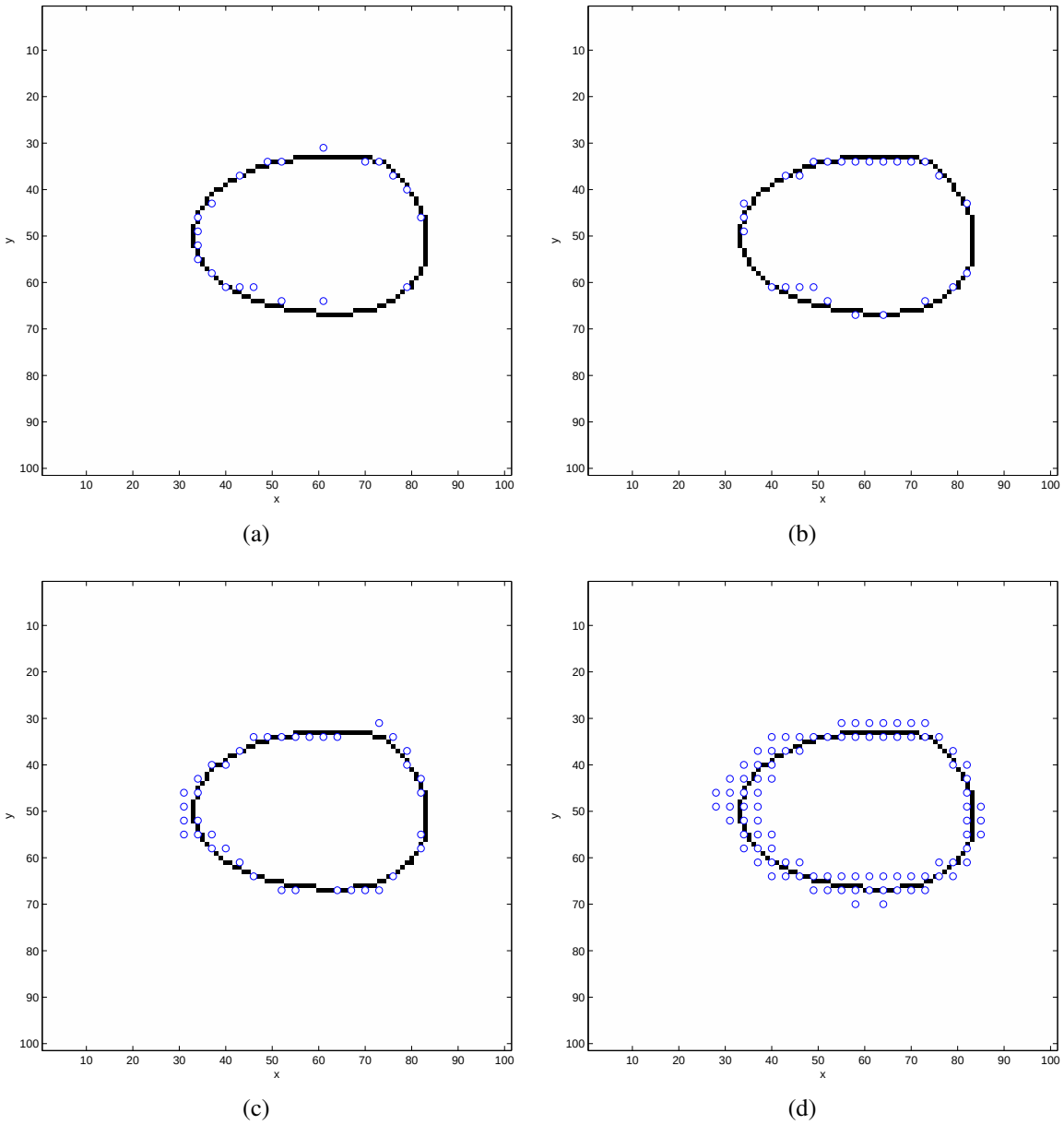


Figure 6.14. Boundary detection results of different Δ_2 significance levels (a) 60%, (b) 70%, (c) 80%, and (d) 99%

6.3.1 Tracking Cost

In the first test sets, we focused on tracking a single 2D object. To control the curve tracking quality, D_{split} was set to 18. Two video clips containing two different objects were used. The initial shapes of both objects were a solid circle with $radius = 25$. The initial curves were both an inscribed regular octagon of the circle. The object 1 started with $center = (35, 35)$, and moved $(x + 4, y + 4)$ in each frame while keeping the size constant. The object 2 started with $center = (50, 50)$, and enlarged $radius + 4$ in each frame while the center was unchanged. A video frame was updated to TOSSIM in every 700 seconds. Sensor nodes were awakened in every 350 seconds to collect updated sensor readings, detect objects and boundaries, and deform the tracking curves. A sensor node obtained a sensor reading as the corresponding pixel value in the concurrent video frame based on the node's location.

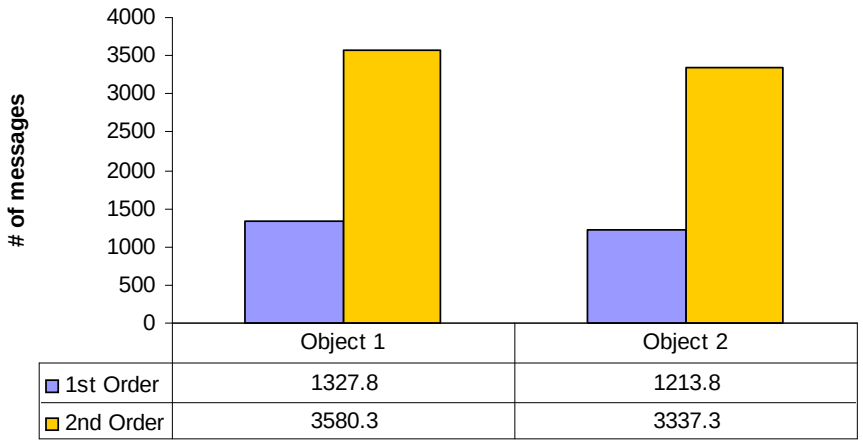


Figure 6.15. Maintenance cost

We implemented the first order and second order curvature models in our experiments. The two curvature models preformed almost the same in the tracking quality, since we set D_{split} to a small value. The first order curvature model prefers the local angle to be π, while the second order curvature model constrains local three consecutive angles to be similar. The first order curvature model requires the location update of a vertex to be exchanged among three neighboring vertex nodes. The second order model needs to exchange a vertex location update message among five consecutive vertex nodes to update the three angles' values. The second order model also needs more communication resources to send the folding vertices and adding vertex notifications. Figure 6.15 shows the average maintenance costs of the first and second curvature models from our tests. The first order curvature model consumes around 36% maintenance communication cost as required by the second order curvature model. Since the first order curvature model requires less maintenance cost and shows no difference in the tracking quality, we did the following tests based only on the first order curvature model.

We compared the communication cost of tracking deformable curves with the cost of reporting inner boundary points. We did not implement any 2D image or video extraction, since reporting inner boundary points usually consumes less than the approaches based on centralized 2D image or video results require [CG03, KI04, DCXC05, JN06]. Figure 6.16 illustrates the average communication costs from our tests. To track both objects, reporting the inner boundary points requires the most expensive communication cost. Reporting

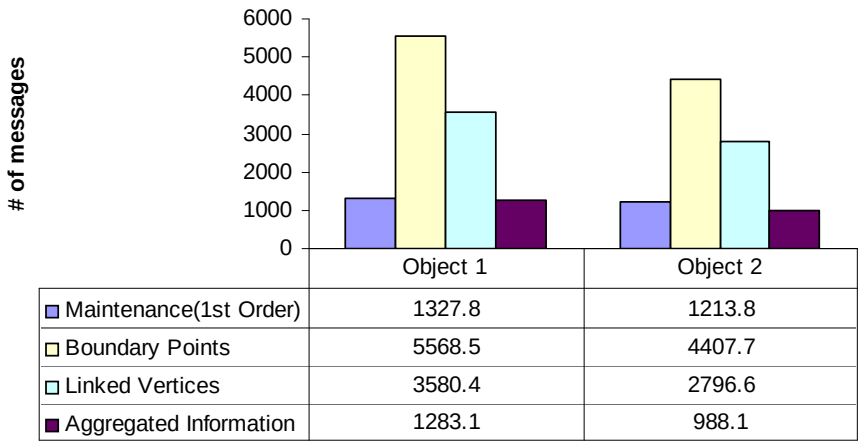
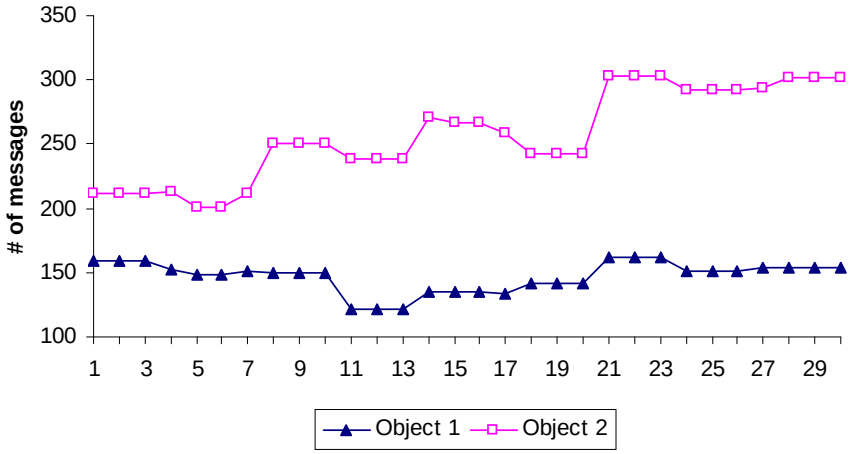


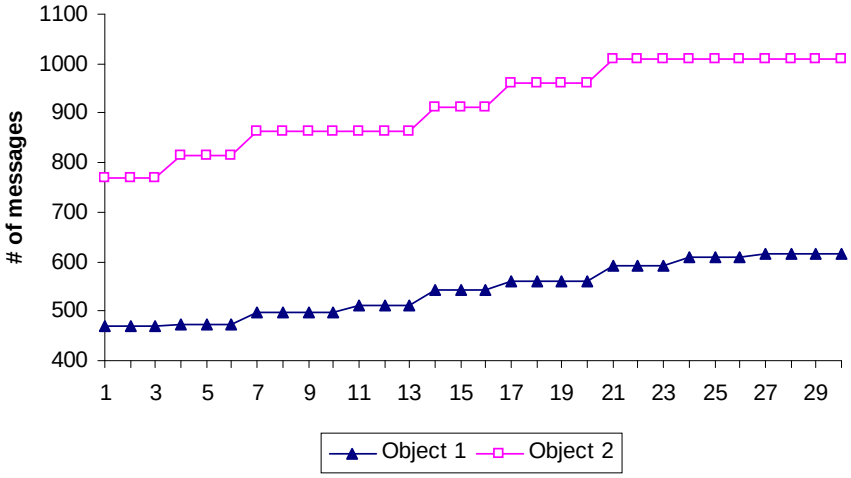
Figure 6.16. Communication cost

linked vertices required less WSN resources than reporting the inner boundary point did. As we expected, the difference between the two types of communication costs was approximately scaled by D_{split}. The ratios of reporting linked vertices against inner boundary points to track the two objects were both around 0.6. If we consider the communication to maintain the deformable curves, the total communication cost of tracking deformable curves was still a slightly less expensive than the cost of reporting inner boundary points. Tracking deformable curves supports extracting complex spatiotemporal properties about objects. We did the tests based on the aggregation and localized computation. As shown in Figure 6.16, processing the aggregated information consumed much less communication resources than reporting inner boundary points or linked vertices did.

Another interesting performance test is the comparison of communication rates over time. Figure 6.17 shows the results from two tests. The object 1 moved and kept its area



(a)



(b)

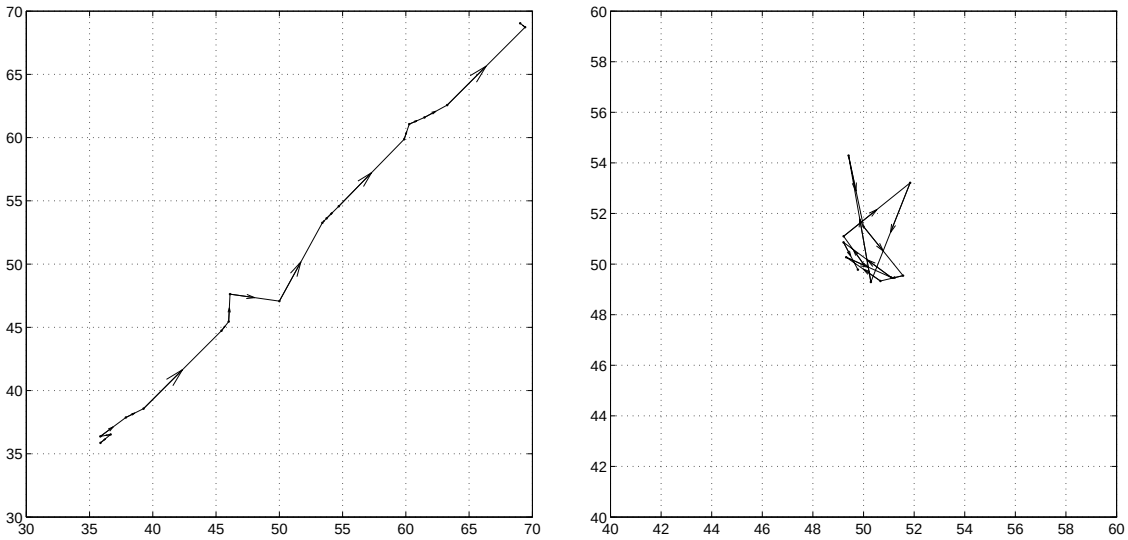
Figure 6.17. Communication rates
(a) the rate of maintenance cost of the 1st order curvature model), and (b) the rate of reporting boundary point

constant in size while the object 2 increased its area and kept the center stationary. As illustrated by Figure 6.17(a), to track the object 1, the maintenance cost rate remained constant. The communication cost rate to maintain the deformable curve for the object 2 increased since more vertices were added to track the enlarged region, as shown by Figure 6.17(a). Figure 6.17(b) explains that more communication resources are required to report inner boundary points of both objects while time elapsed. The object 1 required the same number of points to represent the boundary. While the object 1 moved further away from the root node's location, the inner boundary points required more hops to be relayed back to the root node. The object 2 needed more points to represent the boundary while the area was enlarged. More communication resources are needed to report the increasing number of points along the boundary of object 2. Other types of communication messages for linked vertices and the aggregated information showed similar results as presented in Figure 6.17(b).

6.3.2 Extracting Abstract Spatiotemporal Property

Without loss of generality, we only present our experimental results with regard to the area and centroid of 2D objects here. Figure 6.18 draws the centroid's moving paths based on the aggregated information from the two tests. In the experimental data set, the object 1 moved from the southwest to the northeast, while the object 2 kept its centroid at the center of the 100×100 unit space. Based on the aggregated information, people can understand

the moving patterns of two objects. The object 1 moved while the object 2 roughly kept still as explained by Figure 6.18(a) and Figure 6.18(b). The paths in Figure 6.18 are not smooth because of the relatively low monitoring resolution.



(a) (b)

Figure 6.18. Centroid moving paths
(a) the path of object, and (b) the path of object 2

Figure 6.19 shows the area changes based on the aggregated information from the two tests. Users can draw the conclusion that the area of object 1 remained constant while the area of object 2 kept enlarging. Due to the relatively low monitoring resolution, the area changes were not smooth, as shown in Figure 6.19. By combining Figure 6.18 and Figure 6.19, we can see that the object 1 moved with a constant area, and the object 2 did not move but increased its area.

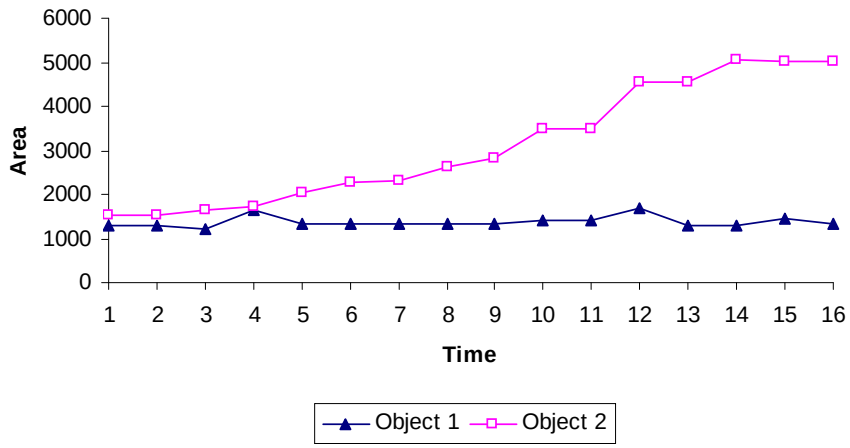


Figure 6.19. Area change

We implemented the area change and centroid change operations to apply lossless suppression. In short, if the local partial aggregated area change or centroid change is zero, a sensor node suppresses the local partial result to be further transmitted to its parent node. Figure 6.20 shows the comparison of the processing costs of unsuppressed and suppressed aggregation. The suppressed aggregation consumed around 40% of communication cost as required by the unsuppressed one. This result proves the effectiveness of suppression techniques, and shows the future direction to combine other suppression technologies [SBY06] with our approaches.

Table 6.10. Quality of predictive information

Predict Range(in $350s$)	5	4	3	2	1
Time Difference(in $350s$)	8	8	4	4	2

We tested the predictive information for real-time alerts. To generate the alerts, a vertex only needs to know the previous locations of the local consecutive vertices. A vertex can

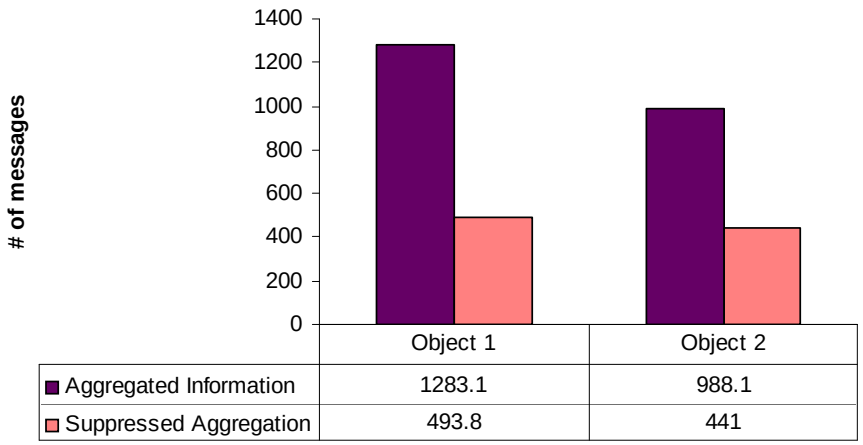


Figure 6.20. Suppressed aggregation

compute the alert based on the local information. Table 6.10 shows the test results on the object 1. In this test, the point for forecast is $(82, 82)$. The prediction range was chosen from different discrete ranges (in $n \times 350$ seconds). The equivalent query is "Will the object 1 move onto the point $(82, 82)$ in next $n \times 350$ seconds?" Table 6.10 also illustrates the time difference between the time when the first alert was given and the time when the object 1 really affected the point $(82, 82)$. The results shown by Table 6.10 are useful, although the prediction quality is not perfect. The main reason is that the movement of object 1 in our tests is not continuous. The edge projection introduced by Section 5.4.2 can be extended to improve the prediction quality. For example, instead of only based on the vertex locations at two consecutive time slots, the velocity of vertex and edge movement can be better estimated based on more historical vertex locations. The quality of predictive information can be improved through methods that are more sophisticated.

6.3.3 Tracking Multiple 2D Objects

In the second set of tests, we used additional video clips as the simulated dynamic continuous phenomenon to test the interactions among multiple objects. These video clips contained multiple 2D objects. Each frame was a snapshot of these objects. A video frame was updated to TOSSIM every 600 seconds. Sensor nodes were awakened up in every 200 seconds to collect updated sensor readings, detect objects and boundaries, deform the tracking curves, and adapt curves to the topological changes. D_{split} was set to 15 for these tests.

The first video contained a single 2D object located at the network center initially. Afterwards, the 2D object split into two objects. The two 2D objects started moving towards two opposite corners of the 100×100 unit space. Later on, the two 2D objects moved back to the center and merged into a single object. Figure 6.21 illustrates a series of snapshots of the tracking curves and underlying 2D objects. In Figure 6.21, the gray region indicates the region covered by underlying 2D objects. The small blue circles represent the location of sensor nodes. The red circles indicate the sensor nodes that detected the inner object boundary. The small squares represent the vertices on deformable tracking curves. The black lines indicate the edges of deformable curves at the current time slot, while the gray lines are the edges of deformable curves at the previous time slot. The dotted gray lines represent the vertex movement. Figure 6.21(a), 6.21(b) and 6.21(c) show the sequence of

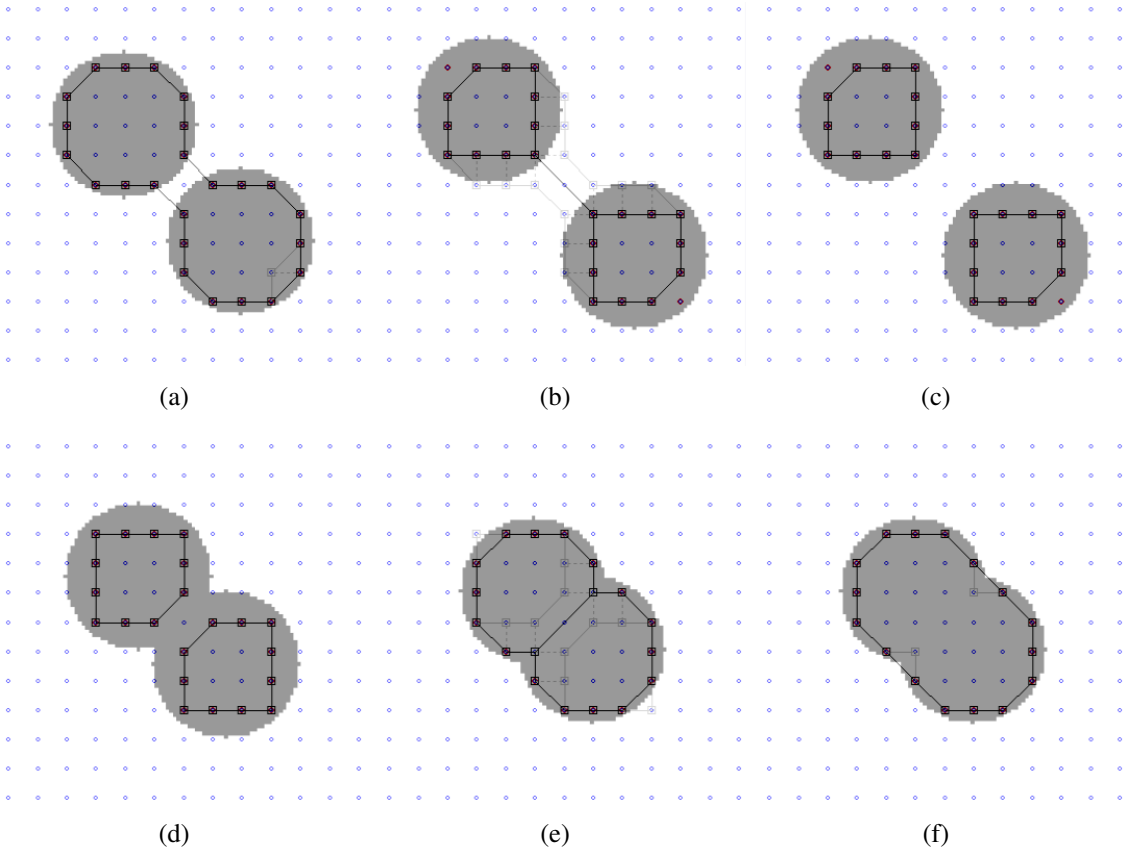


Figure 6.21. Test results on splitting and merging

the splitting event. As shown in Figure 6.21(b), the sensor nodes can detect the overlapping edges. By removing the overlapping edges and reconnecting the open curves, sensor nodes can locally adapt the deformable curves into two closed curves as illustrated by Figure 6.21(c). Figure 6.21(d), 6.21(e) and 6.21(f) explain the sequence of the merging event. Similar to Figure 6.21(b), when the two 2D objects were merging, some edges in the two closed curves overlapped together as shown by Figure 6.21(e). Removing the overlapping edges can allow sensor nodes to locally adapt the deformable curves into a single closed curve, as explained by Figure 6.21(f).

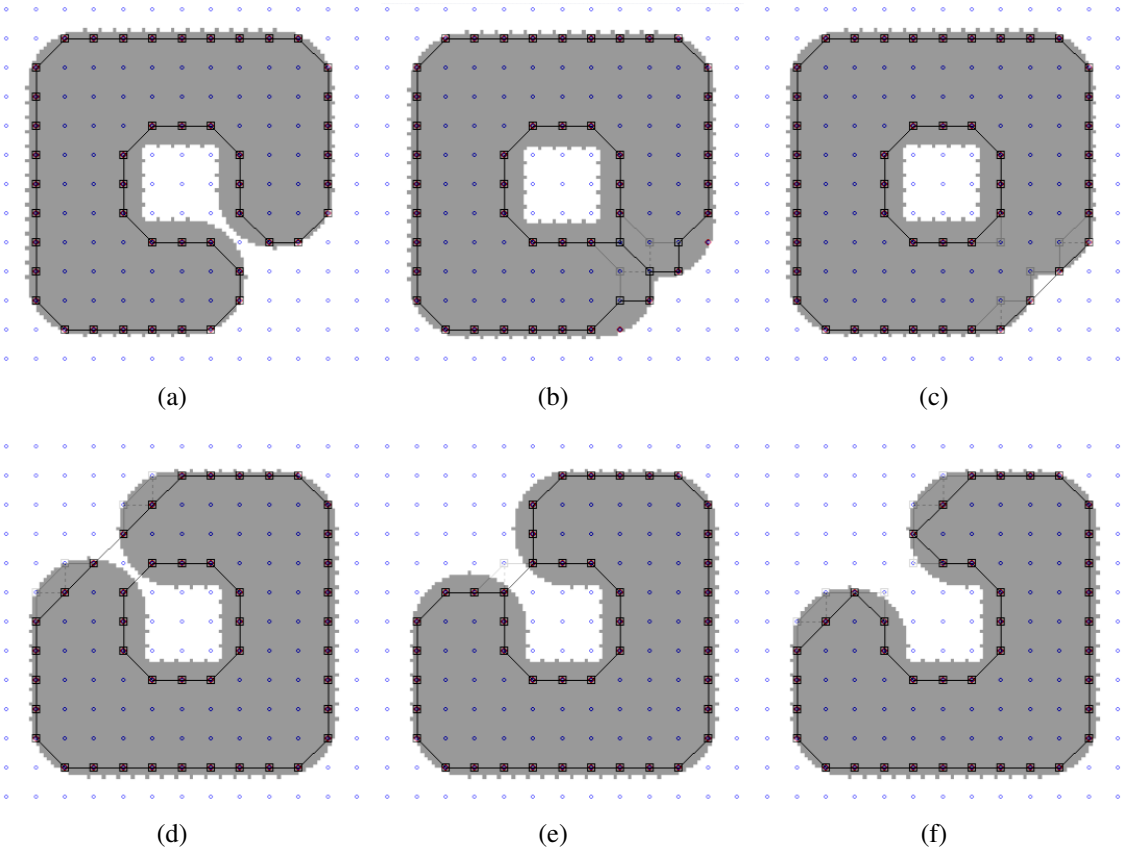


Figure 6.22. Test results on a hole development

We used the second video to illustrate the development of a hole. The video used for the second test began with a single 2D object located at a corner in the network. Afterwards, the 2D object grew two "arms" both horizontally and vertically. The two arms merged at the opposite corner, which resulted in a hole inside the 2D object. Figure 6.22(a), 6.22(b) and 6.22(c) show the sequence of how the two arms merged. Similar to Figure 6.21(e), the edges of two arms overlapped partially as illustrated by 6.22(b). By removing the overlapping edges and reconnecting open curves, sensor nodes can adapt the deformable curves locally to represent the ring shape of the 2D object as shown by Figure 6.22(c).

After the establishment of the inner hole, the ring started breaking at one corner. The breaking sequence is illustrated by Figure 6.22(d), 6.22(e) and 6.22(f). Again, sensor nodes can locally detect the overlapping edges as shown by Figure 6.22(e). Through removing the overlapping edges, the deformable curve can adapt its shape locally to the shape of underlying 2D object as illustrated by Figure 6.22(f).

6.4 Chapter Summary

In this chapter, we present the experimental results of our approaches. The experimental results show the effectiveness of our approaches. Compared to alternative approaches, the experimental results prove that our approaches are resource efficient with respect to the constrained WSNs.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this dissertation, we have presented energy-efficient in-network algorithms to collect and process sensor readings to detect and monitor continuous phenomena by using WSNs. The approaches support both field-based and object-based representations of continuous phenomena. This chapter summarizes our findings and contributions, and discusses potential future research topics.

7.1 Major Results

As the first major result, we have introduced an in-network estimation technique, SWOP, which returns the estimated fine-grained value distribution of a continuous phenomenon based on the Gaussian Kernel estimation. The SWOP approach breaks the entangled links between estimation points and sensor nodes by utilizing the Hermite expansion. The SWOP approach clusters sensor nodes based on their locations into non-overlapped groups. In

each cluster, a small number of Hermite coefficients represent the information about the sensor nodes inside including their locations and readings. In such a way, the wireless communication consumption is reduced. After receiving the Hermite coefficients from the network, a base station or a micro server with more computation power generates the final estimated spatial window result at a user specified resolution. Our simulation results have shown that SWOP reduces the communication cost by 90% compared with transmitting raw sensor readings. The computation complexity is constant with regard to the distributed sensor nodes. Since SWOP utilizes a dynamic node clustering algorithm, SWOP is independent on the node distribution. SWOP can be extended to support arbitrary shapes of query region, by revising the final query result generation part based on Hermite coefficients. The main drawback of SWOP is lacking a ?exible choice with regard to the estimation bandwidth. The current version of SWOP uses the same bandwidth for the entire estimation window region. An improvement of SWOP could use different bandwidths for different locations based on the local phenomenon properties. For example, if a phenomenon is homogeneous in a region, a large bandwidth for this region can be chosen. Consequently, the clustering algorithm in SWOP would need to be revised, if a flexible bandwidth is used.

Secondly, we have proposed the NED approach, an object-based approach of continuous phenomena, to identify objects and object boundaries within a WSN. In NED, the

object and object boundary is identified by user-specified thresholds with regard to sensor readings. The NED approach uses a variable length encoding mechanism for nodes to exchange their local object detection results. If a node detects either a significant object or non-object result by comparing its local sensor reading with the threshold value, the node has more confidence about its local object detection status. In NED, a node with more confidence about its local object detection result uses a 2-bit message to encode and broadcast the result. If a node is nearby the object boundary, the node tends to make faulty object detection results and has less confidence. The nodes with less confidence about their local detection results use 33-bit messages to broadcast their local detection results. In this way, the nodes nearby the object boundary communicate more for better boundary detection results, whereas the nodes in other regions communicate less to save energy. Although the NED approach was mainly designed for spatially continuous phenomena, our experimental results have illustrated the effectiveness of NED for discontinuous phenomena. The efficiency of NED mainly depends on the choice of two user parameters, Δ_1 and Δ_2, for different detection confidence levels and different sensing noise levels. NED can achieve similar detection quality as moving-average-based approaches, while only using the inexpensive communication cost as exhibited by majority-voting-based approaches. Δ_2 is globally defined by users to control the object boundary detection results in NED. A real phenomenon, however, may have different spatial variations at different locations. By

using a global fixed Δ_2 value, NED may fail to report the object boundary in some subregions. To improve NED, we could take the local phenomenon variation into account. An improvement of NED could choose different Δ_2 values at different locations based on the local phenomenon variation with respect to the boundary threshold setting. In this way, only the object boundary detection would be revised, while other parts in NED can be kept.

Thirdly, we have presented an approach for in-network tracking of 2D objects. Our approach is built on the SNAKE model. We have proposed a revision of the original SNAKE model, and have implemented resource-efficient force models for the SNAKE model. In our approach, sensor nodes track individual vertices on the deformable SNAKE curve incrementally without knowing the detailed global curve shape. To update the shape of the representative curve, sensor nodes only need to exchange messages among neighbors. In this way, the maintenance cost of our approach is resource efficient in the constrained WSNs. The original SNAKE model is a rigid model, which cannot be used to track multiple 2D objects. Our approach allows the representative curves to be breakable to track multiple 2D objects. It adapts the representative curves locally to the topological changes caused by the interactions between multiple objects. Based on the in-network tracking of deformable curves, we have shown that different types of abstract spatiotemporal properties can be extracted via the proposed in-network aggregation operations. By tracking the object boundary change and adapting the representative curve, our approach provides the

real-time prediction about the near-future shape of underlying 2D objects. Our experimental results have proven that our approach tracks 2D objects in a WSN and provides abstract properties about the objects more efficiently than, transmitting boundary points or boundary geometry to a central base station and computing spatiotemporal changes "outside" of the WSN. In this dissertation, we have not considered dealing with inside hole detection of 2D objects. The inside hole detection can be solved by running additional boundary detection and geometry formation algorithms on the outer boundary of the hole, without significantly changing our algorithms. In the current implementation of the SNAKE-based tracking, we have not considered node and communication failures. However, the node and communication failures are common in real WSN deployments. To apply our algorithms in the real world, we would need to improve the robustness of our algorithms against the failure-prone nature of WSN. An improvement of the SNAKE-based tracking can use the redundancy to overcome the node and communication failures. For example, vertex nodes can broadcast their local and neighboring vertices information to their neighbors. If a vertex node fails, its neighbors can take over the vertex node's processing and recover the in-network tracking.

Using traditional sensing platforms such as remote sensing instruments, large scale sensor platforms or airborne instruments, interpolation is generally used to compile the sample points for a very large region due to the low spatial resolution of those instruments (with regard to the large area, which can encompass entire states or countries). In current WSNs,

the spatial resolution is usually only a few meters, and thus, WSNs are best used to monitor phenomena, which cannot be observed by using traditional sensing platforms. For example, the microclimate of the plants in orchards, vineyards or other precision agriculture areas cannot be observed by remote satellites, and does need the novel platform and resolution of WSN. For example, the redwood tree project has shown that the microclimate can be visualized using interpolated discrete sensor readings as a smooth spatially continuous phenomenon [TPS+05]. Although WSNs can provide higher resolution than traditional sensing platforms, the sensed samples are still discrete points. Due to the discrete node distribution, the interpolation is necessary to understand the phenomenon properties in-between sensor nodes. Our approaches have application to a wide range of problems such as precision agriculture [BBB04]. Our approaches can help to understand the spatial properties of microclimates in vineyards or tomato greenhouses, and help users to make appropriate responses. Our approaches for object-based queries can be applied to detect the boundary of 2D objects and track these regions in environmental observations. For example, sensor nodes have been used to detect contaminated regions in wide-area environments [JMGRP09]. We can treat the contaminated regions as objects and use our approaches to generate real-time reports about the spatial and spatiotemporal properties about the objects.

7.2 Future Work

So far, this dissertation has shown several approaches for SDMSs to support spatial and spatiotemporal queries for continuous phenomena. Our approaches have proven that WSNs are more than simple data collectors with regard to such phenomena detection and monitoring. By using intelligent in-network data aggregation, estimation and processing techniques, distributed sensor nodes process queries and information collaboratively, while saving significant amounts of valuable energy and extending the lifetime of WSN applications. WSNs can be distributed information processors and respond to user queries in real time.

Undoubtedly, sensor nodes will be smaller, more powerful and more economical in the future. Future WSNs will be able to observe the physical world in a remarkable detail. A general bottleneck in current SDMSs, which will remain in the near future, is that a centralized data analysis (in most cases, a human expert) is required to interpret the quantitative real-time query results. Our approaches have shown that WSNs are able to track 2D objects in real time. More importantly, abstract spatial and spatiotemporal properties about underlying phenomena can be extracted in real time. These abstract properties are useful in the human linguistic communication and reasoning. Currently, we only focus on the efficient extraction of spatiotemporal properties. Our future work will answer how to use the real-time updated spatiotemporal properties of 2D objects. In such a way, more complex spatiotemporal queries can be defined and executed efficiently in future SDMSs.

Our future work will bring more spatial intelligence to WSNs and enhance the automation level of SDMSs.

Tracking multiple point objects is an interesting research direction for us. In this topic, usually a hardware ID is attached to an individual point object. A WSN is deployed to monitor the movement of multiple point objects. Users are interested in the common movement patterns, such as the movement of a flock of birds. Although the proposed approaches in this dissertation focus on spatially continuous phenomena, these approaches can still be applied on the flock tracking by appropriate revisions. We can convert the point locations into a density function. In this way, the discrete phenomenon can be converted into a continuous field representation. For example, the approach presented in Chapter 5 can be easily applied over the object density function to track the spatiotemporal changes of flocks, and to provide the abstract spatiotemporal properties of flocks.

Most today's WSNs are static. With the development of robots, sensor nodes will be more mobile in the future. A node will be able to collect sensor readings at any spatial point at any time, controlled by user specified programs. If a node is able to move fast enough with respect to the temporal variation of underlying phenomena, a mobile node can provide the same sensor readings as provided by current static WSNs. The monitoring resolution can be greatly enhanced by using the mobile WSNs. The proposed approaches in this dissertation, however, are mostly based on static WSNs. For example, the discrete vertex movement assumption in Chapter 5 must be revised for the mobile WSNs. Our future work

needs to efficiently extend the proposed approaches to process spatial information in such a mobile environment.

All the future directions will bring new challenges as well as new discoveries to us. Future WSNs will be more powerful and bring us what we cannot see today.

BIBLIOGRAPHY

[AK00] F. Aurenhammer and R. Klein. *Handbook Of Computational Geometry*, chapter 5, pages 201–290. Elsevier Science, 1st edition, 2000.

[AML05] D.J. Abadi, S. Madden, and W. Lindner. REED: Robust, efficient filtering and event detection in sensor networks. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, pages 769–780, 2005.

[And95] J.G.T. Anderson. Pilot survey of mid-coast maine seabird colonies: An evaluation of techniques. Technical report, the State of Maine Department of Inland Fisheries and Wildlife, 1995.

[ASG01] P. Agouris, A. Stefanidis, and S. Gyftakis. Differential snakes for change detection in road segments. *Photogrammetric Engineering & Remote Sensing*, 67:1391–1399, December, 2001.

[ASSC02] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, 40(8):102–114, 2002.

[AWSBL99] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proceedings of the seventeenth ACM Symposium on Operating Systems Principles (SOSP '99)*, pages 186–201, 1999.

[BBB04] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, 2004.

[BE97] Marshall Bern and David Eppstein. *Approximation algorithms for NP-hard problems*, chapter 8, pages 296–345. PWS Publishing, 1997.

[BG96] Trevor Bailey and Tony Gatrell. *Interactive Spatial Data Analysis*. Prentice Hall, 1996.

[BGS00] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, 7:10–15, 2000.

[BGS01] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proceedings of the 2nd International Conference on Mobile Data Management (MDM '01)*, pages 3–14, 2001.

[BKOS00] M.D. Berg, M.V. Krefeld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms And Applications*. Springer, 2nd edition, 2000.

[BR02] B.J.C. Baxter and G. Roussos. A new error estimate of the fast gauss transform. *SIAM Journal on Scientific Computing*, 24(1):257–259, 2002.

[BWS+02] A. Bucklina, P.H. Wiebeb, S.B. Smolenacka, N.J. Copleyc, and M.E. Clarke. Integrated biochemical, molecular genetic, and bioacoustical analysis of mesoscale variability of the euphausiid nematoscelis difficilis in the california current. *Deep-Sea Research*, 49:437–462, 2002.

[BY03] M. Broadie and Y. Yamamoto. Application of the fast gauss transform to option pricing. *Management Science*, 49(8):1071–1088, 2003.

[CES04] D.E. Culler, D. Estrin, and M. Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–49, 2004.

[CG03] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad Hoc Networks*, 1(2-3):273–291, 2003.

[Coh91] L.D. Cohen. On active contour models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, 1991.

[DCXC05] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, volume 2, pages 902–913, 2005.

[DF03] M. Demirbas and H. Ferhatosmanoglu. Peer-to-peer spatial queries in sensor networks. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing (P2P '03)*, page 32, Washington, DC, USA, 2003. IEEE Computer Society.

[DGM+05] A. Deshpande, C. Guestrin, S. Madden, J.M. Hellerstein, and W. Hong. Model-based approximate querying in sensor networks. *VLDB Journal*, 14(4):417–443, 2005.

[DGR+03] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao. The cougar project: a work-in-progress report. *SIGMOD Record*, 32(4):53–59, 2003.

[DKR04] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management Of Data (SIGMOD '04)*, pages 527–538, 2004.

[DNB04] V. Delouille, R. Neelamani, and R.G. Baraniuk. Robust distributed estimation in sensor networks using the embedded polygons algorithm. In *Proceedings of the 3rd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '04)*, pages 405–413, 2004.

[DNW05] M. Duckham, S. Nittel, and M.F. Worboys. Monitoring dynamic spatial fields using responsive geosensor networks. In *Proceedings of the 13th annual*

ACM International Workshop on Geographic Information Systems (ACMGIS '05), pages 51–60, New York, NY, USA, 2005. ACM Press.

[DP73] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[Ett98] M. Ettus. System capacity, latency, and power consumption in multihop - routed ss-cdma wireless networks. In *Proceedings of the 1998 IEEE Radio and Wireless Conference (RAWCON '98)*, pages 55–58, 1998.

[FZWN08] C. Farah, C. Zhong, M.F. Worboys, and S. Nittel. Detecting toplogoical change using wireless sensor networks. In *Proceedings of the 5th International Conference on Geographic Information Science (GIScience 2008)*, pages 55–69, 2008.

[Gal00] A. Galton. *Qualitative Spatial Change*. Oxford University Press, 2000.

[Gal01] A. Galton. A formal theory of objects and fields. In *Proceedings of the 2001 International Conference On Spatial Information Theory (COSIT 2001)*, pages 458–473, 2001.

[Gal03] A. Galton. Desiderata for a spatio-temporal geo-ontology. In *Proceedings of the 2003 International Conference On Spatial Information Theory (COSIT 2003)*, pages 1–12, 2003.

[GBT+04] C. Guestrin, P. Bodi, R. Thibau, M. Paski, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of the 3rd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '04)*, pages 1–10, 2004.

[GD06] A. Galton and M. Duckham. What is the region occupied by a set of points? In *Proceedings of the 4th International Conference on Geographic Information Science (GIScience 2006)*, pages 81–98, 2006.

[GHS03] S. Ganeriwal, C.C. Han, and M.B. Srivastava. Poster abstract: spatial average of a continuous physical process in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 298–299, New York, NY, USA, 2003. ACM Press.

[GHS07] S. Gandhi, J. Hershberger, and S. Suri. Approximate isocontours and spatial summaries for sensor networks. In *Proceedings of the 6th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '07)*, pages 400–409, New York, NY, USA, 2007. ACM.

[GLB+03] D. Gay, P. Levis, R.V. Behren, M. Welsh, E.A. Brewer, and D.E. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the 2003 ACM International Conference on Programming Language Design and Implementation (SIGPLAN 2003)*, pages 1–11, 2003.

[GNC+01] J.A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile. IEEE 802.15.4: a developing standard for low-power low-cost wireless personal area networks. *IEEE Network*, 15(5):12–19, 2001.

[GS91] L. Greengard and J. Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[HCB02] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670, 2002.

[HH05] B. Harrington and Y. Huang. In-network surface simplification for sensor fields. In *Proceedings of the 13th annual ACM International Workshop on Geographic Information Systems (ACMGIS '05)*, pages 41–50, 2005.

[HHMS03] J.M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *Proceedings of the 2nd ACM/IEEE International Workshop on Information Processing in Sensor Networks (IPSN '03)*, pages 63–79, 2003.

[HK01] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 1st edition, 2001.

[HKS+05] C.C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd International*

Conference on Mobile Systems, Applications, and Services (MobiSys '05), pages 163–176, 2005.

[Hol00] S. Hollar. Cots dust. Master's thesis, University of California, Berkeley, 2000.

[IEE06] IEEE Standards Association. *IEEE 802 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, 2006.

[Int04] Intel lab data. http://berkeley.intel-research.net/labdata/, April 2004.

[JCW04] A. Jain, E.Y. Chang, and Y.F. Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management Of Data (SIGMOD '04)*, pages 11–22, 2004.

[JMGRP09] K.J. Johnson, C.P. Minor, V.N. Guthrie, and S.L. Rose-Pehrsson. Intelligent data fusion for wide-area assessment of uxo contamination. *Stochastic Environmental Research and Risk Assessment*, 23(2):237–252, 2009.

[JN05] G. Jin and S. Nittel. UDC: A self-adaptive uneven clustering protocol for dynamic sensor networks. In *Proceedings of the 1st International Conference on Mobile Sensor Networks (MSN '05)*, pages 897–906, 2005.

[JN06] G. Jin and S. Nittel. One adaptive event and event boundary detection algorithm for noisy wireless sensor networks. In *Proceedings of the International Workshop on Mobile Location-Aware Sensor Networks (MLASN '06)*, 2006.

[JW08] J. Jiang and M.F. Worboys. Detecting basic topological changes in sensor networks by local aggregation. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances In Geographic Information Systems (ACMGIS '08)*, pages 1–10, 2008.

[JW09] J. Jiang and M.F. Worboys. Event-based topology for dynamic planar areal objects. *International Journal of Geographical Information Science (To appear)*, 2009.

[KGKS05] Y.J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI '05)*, pages 217–230, 2005.

[KI04] B. Krishnamachari and S. Iyengar. Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241–250, 2004.

[Kin04] P. Kinney. Zigbee technology: Wireless control that simply works. http://www.zigbee.org, 2004.

[KK00] B. Karp and H.T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 243–254, 2000.

[KK05] A.J. Karaki and A.E. Kamal. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, chapter 6. CRC Press, 1st edition, 2005.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[LJD+00] J. Li, J. Jannotti, D.S.J. DeCouto, D.R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, 2000.

[LL07] Y. Liu and M. Li. Iso-map: Energy-efficient contour mapping in wireless sensor networks. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*, page 36, 2007.

[LLWC03] P. Levis, N. Lee, M. Welsh, and D.E. Culler. TOSSIM: accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 126–137, 2003.

[LS97] O.V. Lepskii and V. Spokoiny. Optimal pointwise adaptive methods in non-parametric estimation. *The Annals of Statistics*, 25(6):2512–2546, 1997.

[LV04] S. Lefèvre and N. Vincent. Real time multiple object tracking based on active contours. In *Proceedings of the International Conference on Image Analysis and Recognition (ICIAR 2004)*, pages 606–613, 2004.

[MCP+02] A. Mainwaring, D.E. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pages 88–97, 2002.

[MFHH02] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Operating Systems Review*, 36(SI):131–146, 2002.

[MFHH05] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[MLNL04] X. Meng, L. Li, T. Nandagopal, and S. Lu. Event contour: An efficient and robust mechanism for tasks in sensor networks. Technical report, University of California, Los Angeles, 2004.

[MT00] T. McInerney and D. Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4:73–91, 2000.

[NM03] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *Proceedings of the 2nd ACM/IEEE International Workshop on Information Processing in Sensor Networks (IPSN '03)*, pages 80–95, 2003.

[Oxf08] Online oxford english dictionary. http://dictionary.oed.com/, March 2008.

[PS01] D.P. Perrin and C.E. Smith. Rethinking classical internal forces for active contour models. In *Proceedings of the 2001 IEEE Conference on Computer Vision Pattern Recognition (CVPR '01)*, volume 2, pages 615–620, 2001.

[PSC05] J. Polastre, R. Szewczyk, and D.E. Culler. Telos: enabling ultra-low power wireless research. In *Proceedings of the 4th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '05)*, pages 364–369, 2005.

[Rac02] J. Racine. Parallel distributed kernel estimation. *Computational Statistics and Data Analysis*, 40(2):293–302, 2002.

[RD06] V.C. Raykar and R. Duraiswami. Fast optimal bandwidth selection for kernel density estimation. In *Proceedings of the 6th SIAM International Conference on Data Mining*, pages 524–528, 2006.

[SBY06] A. Silberstein, R. Braynard, and J. Yang. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management Of Data (SIGMOD '06)*, pages 157–168, 2006.

[Sim87] P. Simons. *Parts: A Study in Ontology*. Routledge, 1987.

[SO05] I. Solis and K. Obraczka. Efficient continuous mapping in sensor networks using isolines. In *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, pages 325–332, 2005.

[SS04] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *Proceedings of the 12th annual ACM International Workshop on Geographic Information Systems (ACMGIS '04)*, pages 166–175, 2004.

[Sza51] O. Szasz. On the relative extrema of the hermite orthogonal functions. *The Journal of the Indian Mathematical Society*, 25:129–134, 1951.

[TPS+05] G. Tolle, J. Polastre, R. Szewczyk, D.E. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)*, pages 51–63, 2005.

[TYD+05a] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Multi-query optimization for sensor networks. In *Proceedings of the 1st IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2005)*, pages 307–321, 2005.

[TYD+05b] N. Trigoni, Y. Yao, A. J. Demers, J. Gehrke, and R. Rajaraman. Multi-query optimization for sensor networks. Technical report, Cornell University, 2005.

[UKT08] M. Umer, L. Kulik, and E. Tanin. Kriging for localized spatial interpolation in sensor networks. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM '08)*, pages 525–532, 2008.

[WD04] M.F. Worboys and M. Duckham. *GIS: A Computing Perspective*. CRC Press, 2nd edition, 2004.

[WLLP01] B. Warneke, M. Last, B. Liebowitz, and K.S.J. Pister. Smart dust: Communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51, 2001.

[WM04] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proceedings of the 1st International Conference on Networked Systems Design and Implementation (NSDI '04)*, page 3, 2004.

[WTC03] A. Woo, T. Tong, and D.E. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 14–27, 2003.

[WTW+01] M.C. Wheeler, J.E. Tiffany, R.M. Walton, R.E. Cavicchi, and S. Semancik. Chemical crosstalk between heated gas microsensor elements operating in close proximity. *Sensors and Actuators B: Chemical*, 77(1-2):167–176, 2001.

[XLCL06] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management Of Data (SIGMOD '06)*, pages 145–156, 2006.

[XP98] C. Xu and J.L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

[YDGD03] C. Yang, R. Duraiswami, N.A. Gumerov, and L. Davis. Improved fast gauss transform and efficient kernel density estimation. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, pages 664–671, 2003.

[YF04] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of the 23rd Annual Joint*

Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), pages 629–640, 2004.

[Zho08] C. Zhong. Generating contour maps for dynamic fields monitored by sensor networks. Master's thesis, University of Maine, 2008.

[Zig06] ZigBee Alliance. *ZigBee Specification 2006*, 2006.

[Zim88] H. Zimmermann. OSI reference model - the ISO model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1988.

[ZSGM08] X. Zhu, R. Sarkar, J. Gao, and J.S.B. Mitchell. Light-weight contour tracking in wireless sensor networks. In *Proceedings of the 27th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2008)*, pages 1175–1183, 2008.

[ZW08] C. Zhong and M.F. Worboys. Continuous contour mapping in sensor networks. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC 2008)*, pages 152–156, 2008.

BIOGRAPHY OF THE AUTHOR

Mr. Guang Jin was born in Wuhan, Hubei, P.R. China on November 1st, 1977. Guang was raised in Wuhan, and graduated from the Wuhan Experimental High School in 1996. He was enrolled by the Huazhong University of Science and Technology, and graduated in 2000 with a Bachelor degree of Science in Physics majoring in Computer Science and Engineering. Guang continued his graduate study in the Huazhong University of Science and Technology, and graduated in 2003 with a Master degree of Engineering majoring in Computer Software and Theory.

After receiving his master degree, Guang entered the graduate program of the department of Spatial information Science and Engineering at the University of Maine in 2003. During his graduate study in Maine, Guang was interested in the spatial information and query processing in wireless sensor networks, and published papers related to his study. The selected publications are listed as follows.

- *G. Jin* and S. Nittel, "Tracking deformable 2D objects in wireless sensor networks", 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS 2008), Irvine CA, November 2008.

- *G. Jin* and S. Nittel, "Towards Spatial Window Queries Over Continuous Phenomena in Sensor Networks", IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol 19(4), pp. 559-571, April 2008.

- *G. Jin*, "GeoSensor Networks, Estimating Continuous Phenomena", In book of "Encyclopedia of GIS GeoSensor Networks", S. Shekhar & H. Xiong(ed.), Springer, pp. 371-372, 2008.

- *G. Jin* and S. Nittel, "UDC: A self-adaptive uneven clustering protocol for dynamic sensor network" (Journal Version), International Journal of Sensor Networks(IJSNet) 2(1/2): 25-33 (2007).

- *G. Jin* and S. Nittel, "NED: Efficient Event Detection in Sensor Network", Workshop "Mobile Location-Aware Sensor Networks", in conjunction with MDM, Nara, Japan, May 13 2006.

- S. Nittel, *G. Jin* and Y. Shiraishi, "In-Network Spatial Query Estimation in Sensor Networks", IEICE Transactions (A), Vol.J88-A, No.12, pp.1413-1421, December 2005.

Since April 2007, Guang has become a member of the honor society of Phi Kappa Phi. Guang is a candidate for the Doctor of Philosophy degree in Spatial Information Science and Engineering from The University of Maine in May, 2009.