Stony Brook University

LOCALIZATION AND LOCATION VERIFICATION IN SENSOR NETWORKS

A DISSERTATION PRESENTED

BY

SOLOMON LEDERER

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

December 2009

Stony Brook University
The Graduate School

Solomon Lederer

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation

Dr. Jie Gao - Dissertation Advisor
Assistant Professor, Department of Computer Science

Dr. Himanshu Gupta - Chairperson of Defense
Associate Professor, Department of Computer Science

Dr. Samir Das
Associate Professor, Department of Computer Science

Dr. Sangjin Hong
Associate Professor, Department of Electrical and Computer Engineering

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Localization and Location Verification in Sensor Networks

by

Solomon Lederer

Doctor of Philosophy

in

Computer Science

Stony Brook University

2009

Sensor networks are still in need of efficient algorithms to bootstrap and maintain network operation in order for them to become a practical technology. Localization continues to be a challenging problem for such networks. Typically sensor networks consist of thousands of nodes disseminated haphazardly over some area without location awareness. A localization algorithm is thus necessary to locate the nodes for data integrity and network operation. We study what can be achieved using connectivity information alone for localization, in particular, in situations where the network shape is complex. We present our algorithm that exploits the global rigidity property of combinatorial Delaunay complex on selected landmark nodes. The key insight is that the combinatorial Delaunay complex has a unique realization in the plane. Thus an embedding of the landmarks by simply gluing the Delaunay triangles properly recovers the faithful network layout. We also demonstrate how this algorithm can be performed without any knowledge of the network boundary by selecting landmarks according to an incremental Delaunay refinement method. In addition, we illustrate how the algorithm works in 3D where nodes are equipped with altimeters to get altitude information. We then tackle the related problem of location verification where the objective is to correctly assess location claims of un-trusted (potentially compromised) nodes. The mechanisms we introduce prevent a compromised node from generating illicit

event reports for locations other than its own. To achieve this goal, in a process we call location certification, data routed through the network is tagged by participating nodes with belief ratings, collaboratively assessing the probability that the claimed source location is indeed correct.

Contents

List of Figures

Acknowledgements

I would like to express my heartfelt thanks and appreciation to my advisor, Professor Jie Gao. It is with Jie's guidance, support, and collaboration that brought me to where I am today. It is though her that I came upon the fascinating area of sensor network research, and through her learned how to explore and investigate the area professionally. I am deeply indebted to her for always being available to help in every way.

I'd like to thank my coauthors, especially those who have made contributions to this dissertation, Yue Wang and Radu Sion.

Special thanks to the faculty members who served as referees during the various benchmarks of my Ph.D. studies (Research Progress Exam, Preliminary Exam and Thesis Defense): Prof. Himanshu Gupta, Prof. Samir Das, and Prof. Sangjin Hong. I also owe thanks to Prof. Joe Mitchell whose course on computational geometry planted the seed for my chosen research focus.

I want to thank my friends and colleagues Rik Sarkar, Dengpan Zhou, Xiaomeng Ban, and Sandra Tinta with whom discussions and friendship were a cherished part of my time here. Special thanks to Jeremy Watts for helping with a coding challenge.

Most importantly I want to thank my wife, Belle, whose love and encouragement led me to the finish line. I also want to thank my parents, David and Malka, for their overarching positive influence and my in-laws, Abraham and Haia Guttmann, for their love.

This dissertation is dedicated to them.

Publications

1. Sol Lederer, Jie Gao, Radu Sion. On Certifying Location claims in Sensor Networks, *Applied Cryptography and Network Security ACNS*, 2007.

2. Sol Lederer, Jie Gao, Radu Sion. Collaborative Location Certification for Sensor Networks, *IEEE Sarnoff Symposium* (invited paper), 2008.

3. Sol Lederer, Yue Wang, Jie Gao. Connectivity-based Localization of Large Scale Sensor Networks with Complex Shape, *IEEE Conference on Computer Communications INFOCOM*, 2008.

4. Yue Wang, Sol Lederer, Jie Gao. Connectivity-based Sensor Network Localization with Incremental Delaunay Refinement Methods, *IEEE Conference on Computer Communications INFOCOM*, 2009.

5. Sol Lederer, Yue Wang, Jie Gao. Connectivity-based Localization of Large Scale Sensor Networks with Complex Shape, *ACM Transactions on Sensor Networks*, 5(4), November, 2009.

6. Sol Lederer, Jie Gao, Radu Sion. Collaborative Location Certification for Sensor Networks, *Transactions on Sensor Networks TOSN* (to appear).

7. Yue Wang, Sol Lederer, Jie Gao. Connectivity-based Sensor Network Localization with Incremental Delaunay Refinement Methods, *Transactions on Sensor Networks TOSN* (under review).

Chapter 1

Introduction

1.1 Overview

Sensor networks are networks of hundreds or thousands of wireless sensing devices used to monitor and classify our physical world. Various stimuli of our environment can be captured, recorded, processed and disseminated by such networks. Forests may have sensors to alert of fires or unlawful hunting; animal habitats may be studied using these devices; urban warfare environments may be littered with sensors to watch enemy movements. Other applications may include detecting trespassers crossing country borders, observing road traffic, capturing structural faults in bridges or skyscrapers, and monitoring water and nutrients in arable soil.

We envision sensors being used in large quantities, collaborating in networks of thousands or tens of thousands of nodes. In order to deploy such large networks, they will typically be dispersed haphazardly over a particular region, perhaps by air. Sensors will be cheap enough that they can be used in such high numbers.

New algorithms for the operation of such networks must be developed from scratch as algorithms used in smaller ad-hoc networks are not portable to sensor network because of their unique constants and objectives. Sensor networks suffer from a limited power supply, weaker radios, and low computing power and storage. And more importantly, certain problems such as localization—determining the location of sensor nodes—are unique to these networks, since they are not

equipped with GPS hardware.

Sensor networks also have an interesting property that as individual nodes they are simple, weak devices in terms of computing power and memory. However, collectively, as a network of thousands working in unison they can be very powerful. Certain problems that these networks face, such as localization, are too difficult for individual nodes to solve on their own, therefore nearly all existing algorithms employ special beacon or anchor nodes, that have more capabilities and knowledge then ordinary nodes, to tackling these problems. However if we recognize that collectively the network is quite intelligent and powerful, perhaps solutions to these problems can be devised by involving the whole network. Indeed, we present algorithms for localization and location verification that does not use anchors nodes or special ranging hardware. For localization, through a structured hierarchical embedding process, nodes can elect certain opportune nodes to be landmark nodes and embed them first. Then all nodes can embed themselves with the use of nearby landmark nodes.

Moreover, since these networks are composed of thousands of immobile nodes, we can view a sensor network as a collection of points in the plane placed about arbitrarily, and its communication network can be viewed as a mathematical graph. Therefore we can borrow ideas and algorithms from graph theory and computational geometry when approaching localization problems and others. Algorithms from surface reconstruction, convex hull algorithms, can be applied to analyzing and deconstructing the sensor network graph. This is immensely useful for designing routing, data collection, or localization protocols. In addition, data structures used in computational geometry such as quad trees and kd-trees are naturally applicable to sensor networks where you may want to structure data according to spacial correlations.

In this dissertation we tackle two overlapping problem areas in sensor networks, localization and location certification. Localization is the challenge of determining where the nodes are located, and location certification involves verifying a node's claimed location in an environment where the nodes may be falsifying its location reports.

1.2 Related Work

There is a strong research focus on algorithms for sensor networks. All software and protocol challenges must be addressed practically for sensor networks to become a realizable, ubiquitous technology. Many such areas of research include ways to minimize the strain on the limited hardware of such devices. This includes designing databases with small memory footprints to account for the limited storage of these nodes, to battery-saving protocols involving wake and sleep time regulation to maximize battery lifetime, to data aggregation techniques to minimize the quantity of data routed through the network. For each of these areas there are myriad approaches of how to deal with the stated problem. For instance, with regard to data aggregation, one protocol might incorporate ideas of network coding to transmit more information without increasing the size of the data, or it may involve smartly dropping superfluous information as it is routed through the network. Since there are so many solutions to the various problems, each with there own advantages and handicaps, research must also be done into comparing the different approaches to determine which are "best", or which are best for a particular scenario.

With regard to localization there are a number of approaches on how to deal with this problem. The most obvious choice is to equip each node with a GPS device. However this is a poor choice do to cost and other limitations such as inability to work where access to satellites are not available. The next natural approach is to equip some nodes with a GPS device or whose location is a priori none somehow. This may mean having a percentage of anchor nodes or beacon nodes that aid all other nodes in localizing their position. It may mean having a mobile device such as an airplane which contacts each node and aids in its localization. With regard to how a node measures its distance to some known locations there are perhaps a dozen options on how this is done. This generally involves the use of sensitive radios or clocks that can measure one of the following: received signal strength (RSSI), the angle of arrival (AoA), or the time of arrival (TOA), time difference of arrival (TDoA) [53,69]. An alternative to this approach

is to only use the binary information of connectively to give some sort of distance measurement. That it, since the communication radius is known, then if 2 nodes can communicate we know they must lie within a particular range, and conversely if they cannot communicate they most lie outside that range. If we take the shortest path between 2 nodes, the path's hop count gives a good estimate of the relative distances between nodes.

Most localization algorithms are for sensors deployed on a flat plane. They can be categorized by anchor-based or anchor-free algorithms, depending on whether there are nodes with known locations (through GPS, for example). Or, they can be grouped by range-based or range-free algorithms, depending on whether one has distance measurements.

Anchor-based range-based algorithms are mostly variations of the basic trilateration framework [69, 70]. When range information is not available, hop count information is used as a substitute [61]. One can also use angle measurements for anchor-based localization [62]. Real systems have also been developed for in-door localization, see for example the Cricket system [64].

When there are no anchors in the network, one may use global optimization techniques such as MDS or semi-definite programming [11] to solve for the network relative locations. This now brings to attention the issue of graph rigidity [41] which has been explored in the graph theory community. That is, given the specified edge lengths of a graph, find an embedding of that graph in a particular dimension, if one exists.

The pioneer work of using rigidity theory in network localization [6, 12, 26, 38, 39, 60, 75] focuses on identifying special graphs that do admit efficient localization algorithms. The first idea is to use *trilateration graphs* [26, 38, 39, 60]. A trilateration graph is defined recursively. It is either a triangle or a trilateration graph with a trilateration extension, defined as adding an additional vertex with three edges to existing vertices. If the network contains a trilateration graph, one can exhaustively search for the 'seed' triangle in the graph and greedily find the trilateration extensions. Thus an incremental algorithm can be adopted to find the realization of the network. A trilateration graph is globally rigid but it is strictly

stronger (i.e., there are globally rigid graphs that are not trilateration graphs), and thus may require more edges than necessary to uniquely embed the graph.

The second idea is to examine *d-uniquely localizable graphs*. A graph with known edge lengths is called uniquely d-localizable if there is a unique realization of the graph in $\mathbb{R}^d$ and there is no non-trivial realization in $\mathbb{R}^k$ with $k > d$. For example, a generic simplex of $d + 1$ vertices is uniquely d-localizable, as its embedding in any higher dimensional Euclidean space will not get any different. For uniquely d-localizable graphs, So and Ye [12, 75] have shown that a semidefinite program is able to find the realization. It is not known, however, whether d-localizability is a generic graph property[1] and it is not clear whether there is a combinatorial characterization of graphs that are d-localizable. The only known method to test uniquely d-localizability is to run the semidefinite program to see whether it succeeds or not. Both approaches taken in prior work require that the network has sufficiently many edges to be globally rigid. In addition, when we do not have edge length values, as in the setting of this thesis, it is not clear how to use these rigidity results.

Our work is different from previous work. We focus on the rigidity of the combinatorial Delaunay complex and apply the algorithm in the more challenging anchor-free, range-free setting. The only prior literature except our work in this setting is to apply multi-dimensional scaling on the hop count values [73].

Notwithstanding the theoretical interest of localization from pure network connectivity, the considered setting also reflects a number of practical application scenarios, in particular, for large scale networks of inexpensive sensor nodes. As the state of the art, a few deployments of sensor networks are in the size of hundreds or thousands of sensor nodes [1,59]. The target size is to achieve hundreds of thousands of nodes in the next few years. As sensor networks scale in size, it is unlikely that sensors are deployed in a uniform homogeneous environment. Terrain variations and obstacles may well prevent the deployment of sensor nodes, resulting in networks with complex shapes respecting the underlying deployment environment. In our model we do not assume distance measurements for the reasons above. We

[1] irrespective of the edge length values, intuitively.

also do not assume anchor nodes (with fixed locations), as networks deployed in remote, unaccessible environments may not have any fixed reference points. We work on the localization problem with the minimalist approach with only connectivity information. Partial distance estimations or anchor nodes, if available, can be easily incorporated to our algorithm to improve localization performance.

1.3 Our Contribution

In our model we do not assume distance measurements for the reasons above. We also do not assume anchor nodes (with fixed locations), as networks deployed in remote, unaccessible environments may not have any fixed reference points. We work on the localization problem with the minimalist approach with only connectivity information. Partial distance estimations or anchor nodes, if available, can be easily incorporated to our algorithm to improve localization performance.

In this work we approach the localization problem and location certification from a theoretical perspective first in order to determine what is the best we can hope for using only very weak assumptions. That is, instead of tackling the problem by adding more hardware or restrictions to the problem setting, we seek to capitalize on the geometric properties intrinsic to any network to design algorithms that offer a solution. In stead of, for instance, assuming there are anchor nodes or beacon nodes to help with localization, we first ask, "Is it strictly necessary to have anchor nodes?" For each assumption found in the literature, we ask, "What would happen if we do away with such assumption?" Once we remove the assumptions we are forced to think along the lines of whether the problem becomes impossible and, if not, what algorithm can we then design that utilizes these properties. Thinking along these lines we can demonstrate the efficacy of our algorithms not just through simulations but also prove theoretically that they meet certain benchmarks and will not fail under various conditions.

While our foray into localization leads to the design of efficient algorithms to solve practical problems, it also leads to interesting results for theoretical ideas in computational geometry, rigidity theory, and topology.

Once localization is achieved the next challenge is to ensure that the location claims made by nodes are indeed correct. Since localization takes place in a distributed manner, there is no base station that is privy to the location of all nodes at the conclusion of the embedding process. Rather each node knows its individual location and appends it to sensory reports passed on to the base station. The concern here is that a node may be making a false location claim, either accidentally or maliciously. It is not enough to simply corroborate a nodes location based on the location of its one-hop neighbors as they too may be corrupted. Rather we take into account the location of all nodes along the path from sensor to base station (sink). Making this idea formal and efficient is the focus of Chapter 4.

With respect to location certification, existing work investigates secure localization [55], i.e., how nodes determine their own location in a hostile environment, and secure location verification [68, 79], determining the location of a node in the face of liars. Typically these protocols involve special anchor nodes, or nodes whose location is not corruptible. Based on the distance to these nodes, the location of the remaining nodes is determined with certain assurances by deploying distance-measuring RF or ultrasound-based mechanisms and performing multi-way handshakes under synchronized clocks assumptions. These methods are designed to be used when the network is first deployed, to establish the location of all nodes during its initial setup. However, when operating in hostile environments, it is essential to secure location information claims *at runtime*, in the presence of compromised nodes, that could falsify location claims and inject incorrect event reports into the information stream. False location information may lead the data sink to take action in a location where none is warranted, and vice versa, not take action in the area where a response is necessary.

This is the only work that I am aware of that does not make use of anchor nodes or beacons for location verification, but instead uses a collaborative effort to certify the location of nodes.

1.4 Dissertation Organization

The dissertation is divided into two parts. Part I is a thorough investigation into localization for sensor networks and how using a collaborative approach, localization can be achieved using only connectivity information. We explore both 2D and 3D versions of this problem. Part II is an investigation into location verification and offers a collaborative approach to tackle this problem.

In the first part of this work we present an anchor-free, range-free localization algorithm that uses the involvement of all nodes in a collective scheme to embed the network. We incorporate ideas from rigidity theory, and simplicial complexes to give theoretical guarantees and bounds on the effectiveness of our algorithm. The basic idea is to select landmark nodes along the network boundaries (inner and outer) according to a certain density and embed these landmarks first. we prove that our landmark selection process will select a near optimal number of landmarks, not too few to make the process impossible, and not much that will negatively effect the final result. Once the basic layout is determined by the embedded landmarks, it becomes straightforward to embed all nodes in the network.

We then expand the above algorithm to select landmarks incrementally. While the above algorithm relies on first determining the network boundary, we here show how we can actually achieve better results through a different approach. While a number of algorithms exist to determine which nodes lie on the network boundary, they perform poorly in networks of low average degree, so we replace boundary detection with an alternative algorithm. Not knowing the network boundary introduces a number of new challenges which we address theoretically, algorithmically, and through simulations.

We then venture into performing localization in 3D, something which has barely been previously explored. We expand our algorithm to work on a non-planar surface. In real world environments the nodes will not normally land on a flat surface but will cover hills and uneven terrain. We present the new challenges in such environments and demonstrate solutions to these problems with theoretical guarantees.

In Part II, we apply the philosophy of community intelligence to location verification. Location verification is determining that a node is indeed at its claimed location. This is needed in hostile environments where you suspect that nodes may be compromised and are issuing false location claims for destructive ends, or in scenarios where the nodes are faulty and are mistaken about its location. When a node sends a packet it must pass through a chain of other nodes along the way before reaching its destination. This collection of nodes can be a powerful judge in determining the honesty of a packet's source. By collectively attaching belief ratings to a packet, the network community can give assurances as to the integrity of a sensor report.

Part I

Beacon-free, Range-free, Localization

Chapter 2

Connectivity-based Localization

2.1 Introduction

The physical location of sensor nodes is critical for both network operation and data interpretation. In this work we focus on anchor-free localization in which none of the nodes know their location and the goal is to recover a relative coordinate system up to global rotation and translation. This is motivated by sensor network applications in remote areas or indoor/underwater environments in which GPS or explicitly placed anchor nodes are not available or too costly. Philosophically, anchor-free localization addresses a very fundamental problem: can we recover the network geometry, simply from the network connectivity information? That is, with local knowledge (knowing which nodes are nearby), can we reconstruct the global picture?

As sensor networks scale in size, retrieving the locations of the nodes becomes even more challenging. The difficulty comes from the network scale, error accumulation, and the increase to both the communication and computation load. Moreover, large deployments of sensor nodes are more likely to have irregular shapes as obstacles and terrain variations inevitably come in to the picture. Our emphasis here is to localize a large sensor network with a complex shape, by using only the network connectivity.

Incorrect flips vs. graph rigidity. A major challenge in network localization

is to figure out the correct global layout and resolve flip ambiguities. To give some intuition, Figure 2.1 illustrates that with only network connectivity information (or even with measurements of the edge lengths), one is unable to tell the "flip" of triangle $\triangle bcd$ relative to a neighboring triangle $\triangle abc$ locally. Both are valid embeddings.

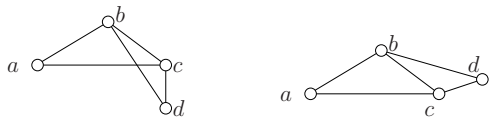


Figure 2.1. A connectivity graph with two distinct embedding having the same set of edge lengths.

Figure 2.2 shows a more severe error, a global flip, that may result from some local flips. The right figure has almost all the nodes correctly localized but has one corner folded over on itself. This is particularly devastating because a node communicating with only its neighbors cannot realize this global error. Indeed, it has been observed that localization algorithms by local optimization may get stuck at one configuration far from the ground truth (see Figure 2 in [60]).

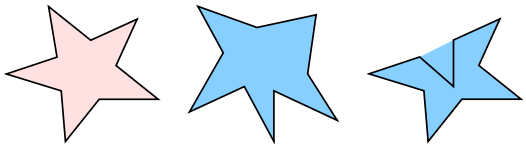
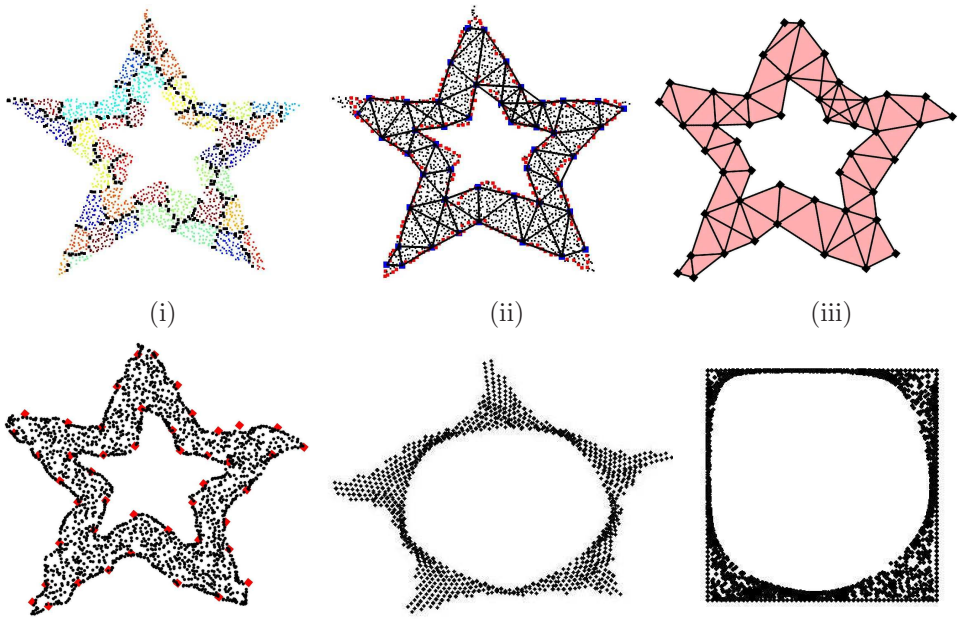


Figure 2.2. Left to right: the ground truth; one possible embedding; a more devastating embedding with a global flip.

It thus represents a major difficulty to resolve flip ambiguities in anchor-free localization. When we know the edge lengths, localization is closely related with graph rigidity [41] in 2D. A graph is *rigid* if one cannot continuously deform the graph embedding in the plane without changing the edge lengths. A graph is *globally rigid* if there is a unique realization in the plane. Rigidity without global rigidity may yield flip ambiguities. For example, Figure 2.1 is rigid but not globally rigid. Thus in anchor-free localization, global rigidity is the desirable property.

A number of localization algorithms deal with the problem of rigidity by exploring the graph structure [26, 38, 39, 60]. These algorithms either require that



(i) (ii) (iii)

(iv) (v) (vi)

Figure 2.3. Anchor-free localization from network connectivity, on a double star shape. The number of nodes is 2171. The connectivity follows a unit disk graph model with average node degree 10. (i) The Voronoi cells of the landmarks (black nodes are on the Voronoi edges); (ii) The Delaunay edges extracted from the Voronoi cells of the landmarks; (iii) Our embedding result of the extracted Delaunay complex; (iv) Our localization result of the entire network. (v) Embedding result by multi-dimensional scaling. (vi) Embedding result by the rubberband representation with the outer boundary fixed along a square.

the network is dense enough to guarantee the network is a tri-lateration graph[1] (such that iterative trilateration method resolves the ambiguity of flips—an even stronger notion than global rigidity) [26, 39, 60]; or, when the network is sparse, record all possible configurations and prune incompatible ones whenever possible, which, in the worst case, can result in an exponential space requirement [38]. All these algorithms require that neighbors are able to estimate their inter-distances, and they do not work with network connectivity alone. Estimating the inter-distances from received signal strength can be quite noisy in a complex environment, and accurate distance estimation requires special ranging hardware.

An approach on anchor-free localization with only network connectivity is to use global optimization such as multi-dimensional scaling (MDS) [74]. MDS takes an inter-distance matrix on n nodes and extracts the node location in $\mathbb{R}^n$. For 2D embedding, the locations are taken as the largest 2D linear projection. Figure 2.3 (v) shows the result of (MDS) on figure 2.3 (vi). Intuitively, MDS tries to stretch the network out in every direction. For a well-connected dense network it gives an effective localization result. But it does not have any notion of rigidity and may produce results with global flips. See more examples in Figure 2.8.

Discovery of global topology. Aside from localization algorithms, recently there is a growing interest in the study of global topology of a sensor field, and its applications in point-to-point routing and information discovery. The focus is to identify high-order topological features (such as holes) from network connectivity [29, 30, 32, 33, 52, 78] and construct virtual coordinate systems with which one can route around holes [14, 27, 28, 34, 35]. These virtual coordinates are by no means close to the real node coordinates — they are not meant to be close. But one may ask the following question: can the identification of the network geometric features (network boundaries, holes, etc.) help in recovering the true node locations? In other words, with the understanding of the network global topology such as where the holes are, does it allow us to infer some information

[1] A tri-lateration graph G in dimension d is one with an ordering of the vertices $1, \ldots, d+1, d+2, \ldots, n$ such that the complete graph on the initial $d+1$ vertices is in G and from every vertex $j > d+1$, there are at least $d+1$ edges to vertices earlier in the sequence. Tri-lateration graph is globally rigid.

on graph rigidity that can be used to prevent global flips?

One piece of work that uses network boundaries to generate topologically faithful (i.e., no global folding) embeddings is to use the rubberband embedding, by Rao *et al.* in [66] and by Funke and Milosavljevic in [35]. The idea is to fix the network outer boundary on a rectangle and then each internal node iteratively takes the center of gravity of its neighbors' locations as its own location. The rubberband relaxation converges to what is called the rubberband representation [77]. With the identification of the network outer boundary, this method does give a layout without incorrect folds, but unfortunately induces large distortion as holes are typically embedded much larger than they are. An example is shown in Figure 2.3 (vi). In the literature [35,66] the rubberband representation is mainly used in assigning virtual coordinates to the nodes for geographical routing purposes and is not used to recover the true node location.

Our contribution. The key idea presented her is to derive a globally rigid substructure from the extraction of high-order topological features of a sensor field, that recovers the global network layout and provide a basis for a localization algorithm.

We assume the sensor nodes are embedded in a geometric region or on a terrain, possibly with holes. The nodes nearby can directly communicate with each other but far away nodes cannot[2]. We do not use anything beyond the network connectivity information and do not assume neighbors can measure their inter-distances, although such information can be easily incorporated to further improve the localization accuracy.

Briefly, the algorithm can be explained as follows (see Figure 2.3): Suppose the network boundaries (both the outer boundary and inner hole boundaries) have been discovered (say with any of the algorithms in [29, 30, 32, 33, 52, 78]). We take samples on the network boundaries and call them *landmarks*. Each node in the network records the closest landmark in terms of network hop distance. The network is then partitioned into *Voronoi cells*, each of which consists of one

[2]Specifically, in our simulations we have adopted unit disk graph model, quasi-unit disk graph model and probabilistic connectivity model.

landmark and all the nodes closest to it (Figure 2.3(i)). The Delaunay graph (Figure 2.3(ii)) as the dual of the Voronoi diagram, has two landmarks connected by a Delaunay edge if their corresponding Voronoi cells are adjacent (or share some common nodes).

Now, here is the key insight: given two *Delaunay triangles* sharing a common edge, there is only *one* way to embed them. Thus there is no flip ambiguity! This is because the Delaunay triangles are induced from the underlying Voronoi partitioning so intuitively we can think them as 'solid' triangles, which, when embedded, must keep their interiors disjoint (the case in Figure 2.1 left cannot happen). In this chapter we make this intuition rigorous. We prove in the case of a continuous geometric domain that when the landmarks are sufficiently dense (with respect to the local geometric complexity), the induced Delaunay graph is rigid. Moreover, the Delaunay complex (with high-order simplices such as Delaunay triangles) is *globally rigid*, i.e., there is a unique way to embed these 'solid' Delaunay triangles in the plane.

The identification of the Delaunay triangles and, more importantly, how to embed them relative to each other overcomes a major hurdle toward anchor-free localization. We use an incremental algorithm to glue the triangles one by one. Each Delaunay edge is given a length equal to the minimum hop count between the two landmarks. Since the hop count is only a poor approximation of the Euclidean distance, we use mass-spring relaxation to improve the quality of the embedding and evenly distribute the error (Figure 2.3 (iii)).

Now with the landmarks localized and the network layout successfully recovered, the landmarks serve as 'anchor' nodes such that each additional node localizes itself by using trilateration with its hop count distances to 3 or more nearby landmarks (in Figure 2.3 (iv)).

In our algorithm the discovery of the sensor layout, i.e., landmark selection and discovery of the Delaunay edges is done in a distributed way. The discovered Delaunay complex is delivered to the base station where the embedding of the landmarks is produced. This network layout is then disseminated to the remaining nodes to localize themselves.

The outline of the chapter is as follows. In Section 3.2 we prove the rigidity of the Delaunay complex and describe the criterion for landmark selection, in the case of a continuous domain. Readers can also choose to read Section 2.3 first, in which we explain the algorithm for the discrete network. Simulation results are presented in Section 2.4.

2.2 Theoretical Foundations

In this section we introduce notations and the theoretical foundation of our algorithm ideas, in particular, the density requirement for landmarks to guarantee the global rigidity of the combinatorial Delaunay complex. Some proofs are put in the Appendix.

2.2.1 Medial axis, local feature size and r-sample

We consider a geometric region $\mathcal{R}$ with obstacles inside. The boundary $\partial\mathcal{R}$ consists of the outer boundary and boundaries of inner holes. For any two points $p, q \in \mathcal{R}$, we denote by $|pq|$ their Euclidean distance and $d(p, q)$ the *geodesic* distance between them inside $\mathcal{R}$, i.e., the length of the shortest path avoiding obstacles. In a discrete network we can use the minimum hop length between two nodes as their distance, whose analog in the continuous case is the geodesic distance. In this thesis paper all the distances are by default measured by the geodesic distances unless specified otherwise. A ball centered at a point p of radius r, denoted by $B_r(p)$, contains all the points within geodesic distance r from p.

Definition 2.2.1. *The* medial axis *of $\mathcal{R}$ is the closure of the collection of points, with at least two closest points on the boundary $\partial\mathcal{R}$.*

The medial axis of ∂R consists of two components, one part inside $\mathcal{R}$, called the *inner medial axis*, and the other part outside $\mathcal{R}$, called the *outer medial axis*. See Figure 2.4. In this chapter paper we only care about the inner medial axis.

Now we are ready to explain how to measure the local geometric complexity of $\mathcal{R}$, which determines the sampling density. An example is shown in Figure 2.4.

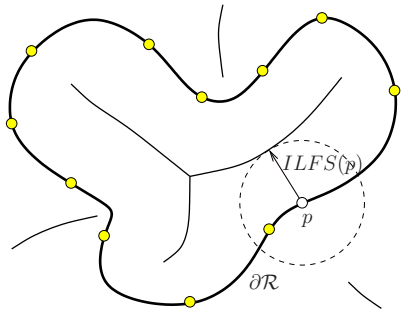


Figure 2.4. The region $\mathcal{R}$'s boundary is shown in dark curves. The medial axis and landmarks selected on the boundaries. Point $p \in \partial\mathcal{R}$ has a landmark within distance $ILFS(p)$.

Definition 2.2.2. *The* inner local feature size *of a point $p \in \partial\mathcal{R}$, denoted as $ILFS(p)$, is the distance from p to the closest point on the inner medial axis.*

Definition 2.2.3. *An r-sample of the boundary $\partial\mathcal{R}$ is a subset of points S on $\partial\mathcal{R}$ such that for any point $p \in \partial\mathcal{R}$, the ball centered at p with radius $r \cdot ILFS(p)$ has at least one sample point inside.*

Landmark density criterion. Our algorithm selects the set of landmarks as an r-sample, with $r < 1$ and selects at least 3 landmarks on each boundary cycle. We will show that these landmarks capture important topological information about the network layout and can be used to reconstruct the network layout.

2.2.2 Landmark Voronoi diagram and combinatorial Delaunay graph

We take some points in $\mathcal{R}$ and denote them as *landmarks* S. Construct the *landmark Voronoi diagram* $V(S)$ as in [27]. Essentially each point in $\mathcal{R}$ identifies

the closest landmark in terms of geodesic distance. The *Voronoi cell* of a landmark u, denoted as $V(u)$, includes all the points that have u as a closest landmark:

$$V(u) = \{p \in \mathcal{R} \,|\, d(p,u) \leq d(p,v), \forall v \in S\}.$$

Each Voronoi cell is a connected region in $\mathcal{R}$. The union of Voronoi cells covers the entire region $\mathcal{R}$. A point is said to be on the *Voronoi edge* if it has equal distance to its two closest landmarks. A point is called a *Voronoi vertex* if its distances to three (or more) closest landmarks are the same. A Voronoi edge ends at either a Voronoi vertex or a point on the region boundary $\partial\mathcal{R}$. The *Voronoi graph* is the collection of points on Voronoi edges. The *combinatorial Delaunay graph* $D(S)$ is defined as a graph on S such that two landmarks are connected by an edge if and only if the corresponding Voronoi cells of these two landmarks share some common points. See Figure 4.3 for some examples. We state some immediate observations

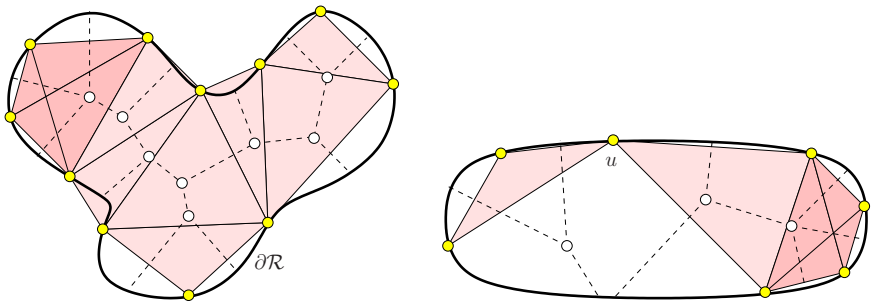


Figure 2.5. (i) The Voronoi graph (shown in dashed lines) and the Delaunay graph/complex for a set of landmarks that form an r-sample with $r < 1$. (iii) When the set of landmarks is not an r-sample (with $r < 1$), the combinatorial Delaunay graph may be non-rigid.

about the Voronoi diagram and the corresponding combinatorial Delaunay graph below.

Observation 2.2.4. *A point on the Voronoi edge of two landmarks u, v* certifies *that there is a Delaunay edge between u, v in $D(S)$. A Voronoi vertex of three landmarks u, v, w* certifies *that there is a triangle between u, v, w in $D(S)$.*

In the case of a degeneracy, four landmarks or more may become cocircular and thus share one Voronoi vertex. See the left top corner in Figure 4.3 (i). We will

capture these high-order features by defining the Delaunay complex in the notion of abstract simplicial complex [24]. The notion of abstract simplicial complex is defined in a completely combinatorial manner and is described in terms of sets. Formally, a set α is an (abstract) *simplex* with dimension $\dim \alpha = \operatorname{card} \alpha - 1$, i.e., the number of elements in α minus 1. A finite system A of finite sets is an *abstract simplicial complex* if $\alpha \in A$ and $\beta \subseteq \alpha$ implies $\beta \in A$. That is, each set α in A has all its subsets in A as well. In our setting, we construct an abstract simplicial complex from the Voronoi diagram, named the *abstract Delaunay complex*, by taking the *Čech complex* of the Voronoi cells, defined below.

Definition 2.2.5. *The* (abstract) Delaunay complex *is the collection of sets*

$$DC(S) = \{\alpha \subseteq S \mid \bigcap_{u \in \alpha} V(u) \neq \emptyset\}.$$

In other words, a set $\alpha \subseteq S$ is a Delaunay simplex if the intersection of the Voronoi cells of landmarks of α is non-empty. The dimension of the Delaunay simplex α is the cardinality of α minus 1.

Thus a landmark vertex is a Delaunay simplex of dimension 0. A Delaunay edge is a simplex of dimension 1. A Delaunay triangle is a simplex of dimension 2 (intuitively, think of the triangle as a 'solid' triangle with its interior filled up). In case of a degeneracy, k landmarks are co-circular and their Voronoi cells have non-empty intersection. This corresponds to a simplex of dimension $k - 1$. The rightmost 4 landmarks in Figure 4.3 (iii) form a dimension-3 simplex (again, intuitively think the simplex as a solid object). We drew the Delaunay complex as shaded regions.

The definition of an abstract simplicial complex is purely combinatorial, i.e., no geometry involved, thus the name of 'abstract' complex. We can talk about an embedding or realization of an abstract simplicial complex (without geometry) in a geometric space as a *simplicial complex* (with geometry). A simplicial complex is geometric and is embedded in a Euclidean space. We give the definitions below. In this chapter, we take the abstracted Delaunay complex from the network

connectivity graph, and find the geometric realization of the abstract Delaunay complex as a simplicial complex in the plane, thus recovering the global shape of the sensor network.

A finite set of points is *affinely independent* if no affine space of dimension i contains more than $i+1$ of the points, for any i. A *k-simplex* is the convex hull of a collection of $k+1$ affinely independent points S, denoted as $\sigma = \operatorname{conv} S$. The dimension of σ is $\dim \sigma = k$. Figure 2.6 shows 0, 1, 2, 3-simplex in $\mathbb{R}^3$. The convex hull of any subset $T \subseteq S$ is also a simplex. It is a subset of $\operatorname{conv} S$

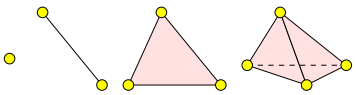


Figure 2.6. 0, 1, 2, 3-simplex in $\mathbb{R}^3$.

and called a *face* of σ. For example, take the convex hull of three points in a 3-simplex, it is a 2-simplex (a triangle). A *simplicial complex* is the collection of faces of a finite number of simplices such that any two of them are either disjoint or meet in a common face. A *geometric realization* of an abstract simplicial complex A is a simplicial complex K together with a bijection φ of the vertex set of A to the vertex set of K, such that $\alpha \in A$ if and only if $\operatorname{conv} \varphi(\alpha) \in K$ [24]. Of course the ambient space in which the simplicial complex is embedded has to have dimension at least equivalent to the highest dimension of the simplex in A. In our case, when there is degeneracy theoretically we will have to embed in a space with dimension higher than 2. We will discuss how to get around this problem in the next section after the discussion of rigidity. In the rest of the chapter, when we say the Delaunay *graph*, we refer to the Delaunay edges and vertices. When we say the Delaunay *complex*, we also include the higher order simplices such as Delaunay triangles and tetrahedrons.

2.2.3 Global rigidity of combinatorial Delaunay complex

The property of the combinatorial Delaunay graph clearly depends on the selection of landmarks. The goal of this section is to show that the Delaunay graph is rigid

when there are at least 3 landmarks on each boundary cycle and they form an r-sample of $\partial\mathcal{R}$ with $r < 1$. In addition, and the Delaunay complex is globally rigid (i.e., it admits a unique 2D realization). An example when the combinatorial Delaunay graph is not rigid due to insufficient sampling is shown in Figure 4.3 (ii). Now we prepare to prove the rigidity results by first showing that the Voronoi graph (collection of points on Voronoi edges) is connected within $\mathcal{R}$. In this subsection we assume that the landmarks are selected according to the landmark selection criterion mentioned above. The proofs of some of the following Lemmas can be found in the Appendix.

Observation 2.2.6. *Two Voronoi vertices connected by a Voronoi edge correspond to two Delaunay triangles sharing an edge.*

Lemma 2.2.7. *For any two adjacent landmarks u, v on the same boundary cycle, there must be a Voronoi vertex inside $\mathcal{R}$ whose closest landmarks include u, v.*

Lemma 2.2.7 implies that the Delaunay graph has no node with degree 1 – since every node is involved in 2 triangles with its adjacent 2 nodes on the same boundary.

Lemma 2.2.8. *If there is a continuous curve C that connects two points on the boundary $\partial\mathcal{R}$ such that C does not contain any point on Voronoi edges, then C cuts off a topological 1-disk[3] of $\partial\mathcal{R}$ with at most one landmark inside.*

Corollary 2.2.9. *The Voronoi graph $V(S)$ is connected.*

Now we are able to show that the combinatorial Delaunay graph is rigid. In other words, given a realization of $D(S)$ in the plane, one cannot deform its shape in the plane without changing the lengths of the edges. To prove this, we use a seminal result about graph rigidity [by G. Laman in 1970], known as the *Laman condition*. It states that generically rigid graphs in 2D can be classified by a purely combinatorial condition. A graph is called a *Laman graph* if it has n vertices, $2n - 3$ edges and any subset of k vertices spans at most $2k - 3$ edges.

[3]Intuitively, a topological 1-disk can be continuously deformed into a straight unit length line segment, without any cutting or gluing operations.

Theorem 2.2.10 (Laman condition [54]). *A graph G with n vertices is generically rigid* [4] *in 2 dimensions if and only if it contains a Laman graph on n vertices.*

Theorem 2.2.11. *The combinatorial Delaunay graph $D(S)$ is rigid, under our sampling condition.*

Proof: In this proof we assume without loss of generality that there is no degeneracy, i.e., four or more landmarks are not co-circular. Indeed degeneracy will only put more edges to the combinatorial Delaunay graph, which only helps with graph rigidity.

From the Voronoi graph $V(S)$, we extract a subgraph V' that contains all Voronoi vertices and the Voronoi edges that connect these Voronoi vertices. Some Voronoi edges end at points on the boundary $\partial\mathcal{R}$ and we ignore those. By Corollary 2.2.9 this graph V' is connected. Now we find a spanning tree T in V' that connects all Voronoi vertices. Take the corresponding subgraph D' of the combinatorial Delaunay graph $D(S)$ such that an edge exists between two landmarks in D' if and only if there is a point in T that certifies it. D' is a subgraph of $D(S)$. Now we argue that D' is a Laman graph.

First the number of landmarks is n. We argue that the number of edges in D' is $2n-3$. Assuming the number of Voronoi vertices is m, T has $m-1$ Voronoi edges. We start from a leaf node on T and sweep along the edges on T. Each time we add one new vertex that is connected to the piece that we have explored through an edge. During the sweep we count the number of landmarks and the number of Delaunay edges that we introduce. To start, we have T' initialized with one Voronoi vertex, thus we have three landmarks and three Delaunay edges. The new Voronoi vertex x we introduce is adjacent to one and only one vertex in T'—if x is adjacent to two vertices in T', then there is a cycle since T' is connected. This will contradict with the fact that T is a tree. Thus in each additional step we will introduce one Voronoi vertex that is connected to T' through one Voronoi edge.

[4] Intuitively, generic rigidity means that almost all (except some degenerate cases) realizations of the graph in the plane are rigid. Generic rigidity is a graph property. However, a generically rigid graph may have some degenerate assignment of edge lengths such that the realization is not rigid.

This will introduce one new landmark and two new Delaunay edges. When we finish exploring all Voronoi vertices we have a total of $3 + (m-1) = m + 2 = n$ landmarks, and $3 + 2(m-1) = 2n - 3$ Delaunay edges between them. Thus D' has n landmarks and $2n - 3$ edges.

With the same argument we can show that any subgraph of D' with k landmarks, denoted by S', has at most $2k - 3$ edges. This is because a Delaunay edge is certified by a Voronoi edge. Thus we take the Voronoi edges of T whose corresponding landmarks all fall inside S'. These Voronoi edges span at most a tree between Voronoi vertices involving only landmarks in S', because they are a subset of a tree T. By the same argument there are at most $2k-3$ edges between landmarks in S'. Thus the graph D' is a Laman graph. By the Laman condition the combinatorial Delaunay graph $D(S)$ is rigid. □

The above theorem shows the rigidity of the combinatorial Delaunay graph, but not the global rigidity yet—there might be several different realizations of the graph in the plane. Indeed for an arbitrary triangulation one may flip one triangle against another adjacent triangle one way or the other to create different embedding. However, this is no longer possible if we embed the *combinatorial Delaunay complex*, induced from the Voronoi diagram $V(S)$. The intuition is that when the triangles are 'solid' and two triangles cannot share interior points there is only one way to embed the Delaunay complex. Thus the recovered Delaunay complex does reflect the true layout of the sensor field $\mathcal{R}$.

Recall that we want to find an embedding of the abstract Delaunay complex in 2D. That is, we want to find a mapping φ of the vertices in the plane such that any abstract simplex $\sigma \in DC(S)$ is mapped as a simplex $\operatorname{conv} \varphi(\sigma) \in \mathbb{R}^2$. Notice that in the case of degeneracy there are high-order k-simplices, $k \geq 3$, for which a geometric realization requires embedding into a space of dimension k or higher. However, this is not really a problem if we force the dimension to be 2. Indeed, look at all the edges of a k-simplex, $k \geq 3$, they form a complete graph of $k + 1 \geq 4$ vertices. Thus it is a 3-connected graph and redundantly rigid (a graph remains rigid upon removal of any single edge). Existing results in rigidity theory [10,42] show that a graph is globally rigid (uniquely realizable) in 2D under

edge lengths constraints if and only if it is tri-connected and is redundantly rigid. Thus all high-order simplices have unique embedding in the plane (up to global translation and rotation). In this work, we find a geometric realization of the abstract Delaunay complex in the plane. For all the simplices with dimension 2 or smaller, they are mapped to simplices in the plane. For simplices of dimension 3 or higher, the induced *graph* is globally rigid and subject to a unique embedding, as explained above.

Now the Delaunay complex is composed of a set of Delaunay triangles (2-simplices) and high-order simplices (and their sub-simplices, of course). We already know that the high-order simplices are embedded in the plane as globally rigid components. The Delaunay 2-simplices/triangles are embedded as a geometric complex, i.e., the geometric realization of the abstract Delaunay complex. What is left is to show that given two Delaunay triangles $\triangle uvw$ and $\triangle uvp$ sharing an edge, there is only one way to embed them in the plane as required by the definition of simplicial complex—that is w and p are on opposite sides of the shared edge uv, as in Figure 4.2(i). Otherwise, w and p are embedded on the same side of

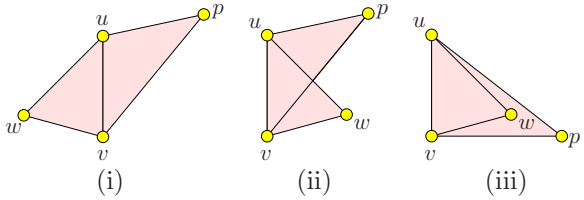


Figure 2.7. Two Delaunay triangles $\triangle uvw$ and $\triangle uvp$ sharing an edge. (i) is the only valid embedding with the two triangles not sharing any interior points.

uv. Then either w is inside $\triangle uvp$ (as in Figure 4.2 (iii)), or p is inside $\triangle uvw$, or two edges intersect at a non-vertex point (as in Figure 4.2 (ii)). This will violate the properties of a simplicial complex that any two simplices are either disjoint or meet at a common face. If w is inside $\triangle uvp$, then the two simplices, a 0-simplex w and a 2-simplex $\triangle uvp$ intersect at a vertex w which is not a face of $\triangle uvp$. In the other case, if two edges intersect at a non-vertex point, this intersection is not a face of either edge.

Now we can conclude with the main theoretical result:

Theorem 2.2.12. *Under our landmark selection criterion, the combinatorial Delaunay complex $DC(S)$ has a unique embedding in the plane up to a global translation and rotation.*

2.3 Algorithm Description

We assume a large number of sensor nodes scattered in a geometric region. In general nearby nodes can directly talk to each other and far away nodes can not but the algorithm does not strictly enforce a unit disk graph model. The algorithm basically realizes the landmark selection and embedding described in the previous section. Thus we will not re-iterate many things said already and instead focus on the implementation and robustness issues, for the geodesic distance is only poorly approximated by the minimum hop count between two nodes.

We first outline the algorithm and explain each step in detail.

2.3.1 Select landmarks

We use a distributed boundary detection algorithm that identifies nodes on both outer and inner boundaries and connects them into boundary cycles [78]. With the boundary detected we can identify the medial axis of the sensor field, defined as the set of nodes with at least two closest boundary nodes [14]. The boundary nodes flood inward at roughly the same time [25, 36]. The flooding messages are suppressed by the hop count to the boundary nodes to reduce message complexity. Each node learns its closest boundary node. The nodes at which the flooding frontiers collide are nodes on the inner medial axis.

In a discrete network, the medial axis may contain a lot of noises due to the discrete hop count values. For example, a node that is a neighbor of adjacent two boundary nodes is identified to be on the medial axis according to the definition, and is clearly not what we want. There are a number of heuristic algorithms in the past literature to 'clean up' the medial axis of a discrete network [14, 84]. The idea is to take the nodes with two or more closest intervals on the network

boundary [84]. A node having its closest points on the boundary in a consecutive interval is not identified as the medial axis node.

With the boundary and medial axis identified, we select landmarks from boundary nodes such that for any node p on the boundary, there is a landmark within distance $ILFS(p)$, where $ILFS(p)$ is the inner local feature size of p defined as the hop count distance from p to its closest node on the inner medial axis. In order to find the local feature size of each node on the boundary, nodes on the medial axis flood the network at roughly the same time with proper message suppression. Each boundary node learns its local feature size as the hop count to its closest node on the medial axis.

Now, landmark selection can be performed by a message traversing along the boundary cycles and select landmarks along the way in a greedy fashion to guarantee the sampling criterion. Alternatively, we can let each boundary node p wait for a random period of time and select itself as a landmark. Then p sends a suppression message with TTL as $ILFS(p)$ to adjacent boundary nodes. A boundary node receiving this suppression message will not further select itself as landmarks. Thus landmarks are selected with the required density.

2.3.2 Compute Voronoi diagram and combinatorial Delaunay complex

The landmark Voronoi diagram is computed in a distributed way as in [27]. Essentially all the landmarks flood the network simultaneously and each node records the closest landmark(s). Again a node p will not forward the message if it carries a hop count larger than the closest hop count p has seen. Thus the propagation of messages from a landmark ℓ is confined within ℓ's Voronoi cell. All the nodes with the same closest landmark are naturally classified to be in the same cell of the Voronoi diagram. Nodes with more than one closest landmarks stay on Voronoi edges or vertices.

Unlike the Euclidean case that there is always a point with equal distance to any two or three landmarks, when we adopt the integer hop count measurement

as the distance metric, there may not be a point with equal distance to two or three landmarks. Thus we re-define Voronoi vertices in the discrete setting.

Definition 2.3.1. *An* interior node *is a node p with distance to its closest landmark strictly smaller than its distances to all the other landmarks. A* border node *is a node that is not an interior node.*

Figure 2.3 (i) is an example of the landmark Voronoi diagram with different Voronoi cells colored differently. Border nodes are colored black. We group these border nodes into Voronoi edges and vertices, i.e., the k-witnesses of $(k-1)$-simplices.

Definition 2.3.2. *A* k-witness *is a border node which is within 1-hop from interior nodes of k different Voronoi cells. The border nodes that witness the same set of Voronoi cells are grouped into connected* clusters.

One subtle robustness issue, due to the discreteness of sensor nodes, is that there might not be a node that qualifies for the witness defined above (especially for high-order simplices). Thus we propose a *merge* operation: For two clusters A and B that are both k-witnesses, if there exists a node p in cluster A, or there exists a node q in cluster B, and all nodes in cluster B are neighbors of p or all nodes in cluster A are neighbors of q, then we merge cluster A and B into one cluster that certifies the union of their corresponding landmarks. The benefit of doing so is to generate high order Delaunay simplices even when there are no corresponding witnesses due to the discrete resolution. The above algorithm to identify the abstract Delaunay complex is a heuristic algorithm that uses the intuition from the continuous case. Alternatively we can use the notion of the witness complex [19, 22]. This is explored in a paper by Gao and Guibas [37].

The witnesses certify the existence of Delaunay simplices and by definition can be identified locally. A k-witness node w, after it identifies itself, reports to the corresponding landmarks. Such a report contains the IDs of the landmarks involved in this dimension $k-1$ Delaunay simplex, together with the distance vector from the witness node w to each of the k landmarks. Remember that

nodes in a Voronoi cell store their minimum hop count distances to their home landmark. Thus, the report just follows the natural shortest path pointer to the landmarks involved (so routing is simple). It can happen that multiple witnesses certify the same Delaunay simplex (say, in the case of a Delaunay edge) and they individually report to the same landmark. These report messages are again suppressed during routing. If a node sees a report about a previously received Delaunay simplex, it will not forward it. Naturally the report from the witness with the smallest hop count to its landmarks will arrive the earliest. With these reports, a landmark learns the combinatorial Delaunay simplices it is involved in, and in addition, an approximate hop count to the other landmarks in those simplices through the distance vectors carried in the reports. In particular, a landmark p estimates the hop count distance to landmark q as the minimum of the sum of distances from the witness node to p and q, over all reports received with q involved. This distance estimation can be directly used to embed the Delaunay simplices. Alternatively, if the minimum hop count distances between neighboring landmarks are desired, one can let the messages initiated by the landmarks travel to the adjacent Voronoi cells. Thus each landmark learns the minimum hop count to all neighboring landmarks.

We remark that in the protocol we aggressively use message suppression to reduce the communication cost. With reasonable synchronization most of the flood messages are pruned and the average number of messages transmitted by each node is within a small constant. We also remark that local synchronization (with possible global clock drifts) is sufficient as message suppression occurs mostly among neighboring landmarks.

2.3.3 Embed Delaunay complex

Now we are ready to glue the simplices together to embed the landmarks and generate the network layout. Since there is only one way to glue two adjacent simplices (to keep their interiors disjoint, as shown by Theorem 3.2.2), the embedding is unique. We first embed one simplex S_1 arbitrarily. Then we can embed

its neighbor S_2 as follows: Let ℓ_1 and ℓ_2 be the landmarks they share in common. For each landmark ℓ_i in S_2 not yet embedded, we compute the 2 points that are with distance $d(\ell_1, \ell_i)$ from ℓ_1 and $d(\ell_2, \ell_i)$ from ℓ_2, where $d(\cdot, \cdot)$ is the hop-count distance between landmarks, estimated in the previous section. Among the two possible locations we take the one such that the orientation of points $\{\ell_1, \ell_2, \ell_i\}$ is different from the orientation of $\{\ell_1, \ell_2, \ell_r\}$, where ℓ_r is any landmark of S_1, other than ℓ_1 and ℓ_2. Thus ℓ_i and ℓ_r lie on opposite sides of edge $\ell_1\ell_2$.

In some cases one landmark may have two or more neighboring simplices that are already embedded and is thus given multiple coordinate assignments. A natural solution is to take ℓ at the centroid of the different positions. After we have a rough embedding of the entire Delaunay complex, we apply a mass-spring algorithm [31,43,48,51,65] to "smooth out" the disfigurements caused by the conflicting node assignments. It is important to recognize however, that mass-spring plays a minor role in our algorithm and its utility is only apparent here because we initially start with topologically correct landmarks positions, i.e., no global flips. Without this initial configuration with good layout a naive mass-spring algorithm can easily gets stuck at local minima, as observed by many [51, 65].

Briefly, the idea of mass-spring embedding is to think of the landmarks as masses and each edge as a spring, whose length is equal to the estimated hop count distance between two landmark nodes. The springs apply forces on the nodes and make them move until the system stabilizes. The objective is to have the measured distances (based on their current locations) between landmarks match as closely as possible the expected distances (indicated by hop count values).

We remark that this heuristic embedding algorithm only guarantees that adjacent Delaunay triangles are embedded 'side-by-side'. It does not prevent two chains of triangles from wrapping around and overlapping each other. In fact, given a planar graph with specified edge lengths, it is a NP-hard problem to find a planar embedding [9, 15]. Our problem is more difficult as we only have approximate edge lengths. It remains as future work to develop efficient approximation algorithms to embed a planar graph with approximate edge lengths.

In a distributed environment the embedding of the Delaunay simplices can

be done incrementally with message passing. Alternatively, the combinatorial Delaunay complex can be collected at a central station where the embedding is performed and disseminated to the remaining nodes. As the number of landmarks is only dependent on the geometric complexity of the sensor field, it is much smaller than the total number of nodes. Thus a centralized collection and dissemination of the landmark positions is manageable.

2.3.4 Network localization

Since the locations of the landmarks are known, each non-landmark node just runs a tri-lateration algorithm to find its location (e.g., the atomic trilateration in [69]) by using the hop count estimation to 3 or more landmarks.

2.4 Simulations

We conducted simulations on various network topologies and node densities to evaluate our algorithm and compare with existing solutions.

2.4.1 Simulation setup and models

In the simulations we use three different models for the network connectivity.

1. *Unit disk graph model:* two nodes are connected by an edge if and only if the Euclidean distance between them is no greater than 1.
2. *Quasi-unit disk graph model:* two nodes are connected by an edge if the Euclidean distance between them is no greater than a parameter α, $\alpha < 1$, and are not connected by an edge if the Euclidean distance is larger than 1. If the Euclidean distance d is in the range $(\alpha, 1]$, there may or may not be an edge between them. We include this edge with probability $(1 - d)/(1 - \alpha)$.
3. *Probabilistic connectivity model:* with unit disk graph model, we additionally remove each edge with probability q.

The nodes are distributed according to a perturbed grid distribution. Each node is perturbed from the grid point with a uniform distribution. That is, for any node $p(x, y)$ on the grid, we created two random numbers r_x and r_y between 0 and the grid width. Then we use $(x + r_x, y + r_y)$ as the node position. We then control the communication range to vary the average node degree.

To vary the network "shape", We tried different network topologies by including single or multiple holes, convex or concave holes, and some difficult cases such as a U-shape or a Spiral-shape. The network setup and parameters are shown in the caption for each topology.

2.4.2 Algorithms in comparison

Since most localization algorithms assume node inter-distance measurements and/or anchor nodes, to make a fair comparison we only compare with two algorithms that also use network *connectivity information only*:

Multi-dimensional scaling (MDS). Multidimensional scaling has been used by Shang et al. [74] for sensor network localization with connectivity information only. It is also the only anchor-free localization algorithm so far using connectivity information. For n nodes, the input to MDS is the pairwise distance estimation of size $O(n^2)$. In this case, since only rough hop-count distances are known, MDS has trouble capturing a twist within the graph, making a long narrow graph not differentiable from a spiral-shaped graph. In addition, MDS is a centralized algorithm and can not be executed in sensor nodes with limited resources. At the heart of MDS is singular value decomposition (SVD) which has a time complexity of $O(n^3)$. In our simulation we tested MDS in two cases, once on all the nodes and once on the landmarks only. They produce similar layout results. MDS on all nodes is very slow. For some experiments with 5000 nodes the matrix operation involved in MDS requires more than 1GB memory. This computation is only feasible on powerful nodes such as the base station.

.

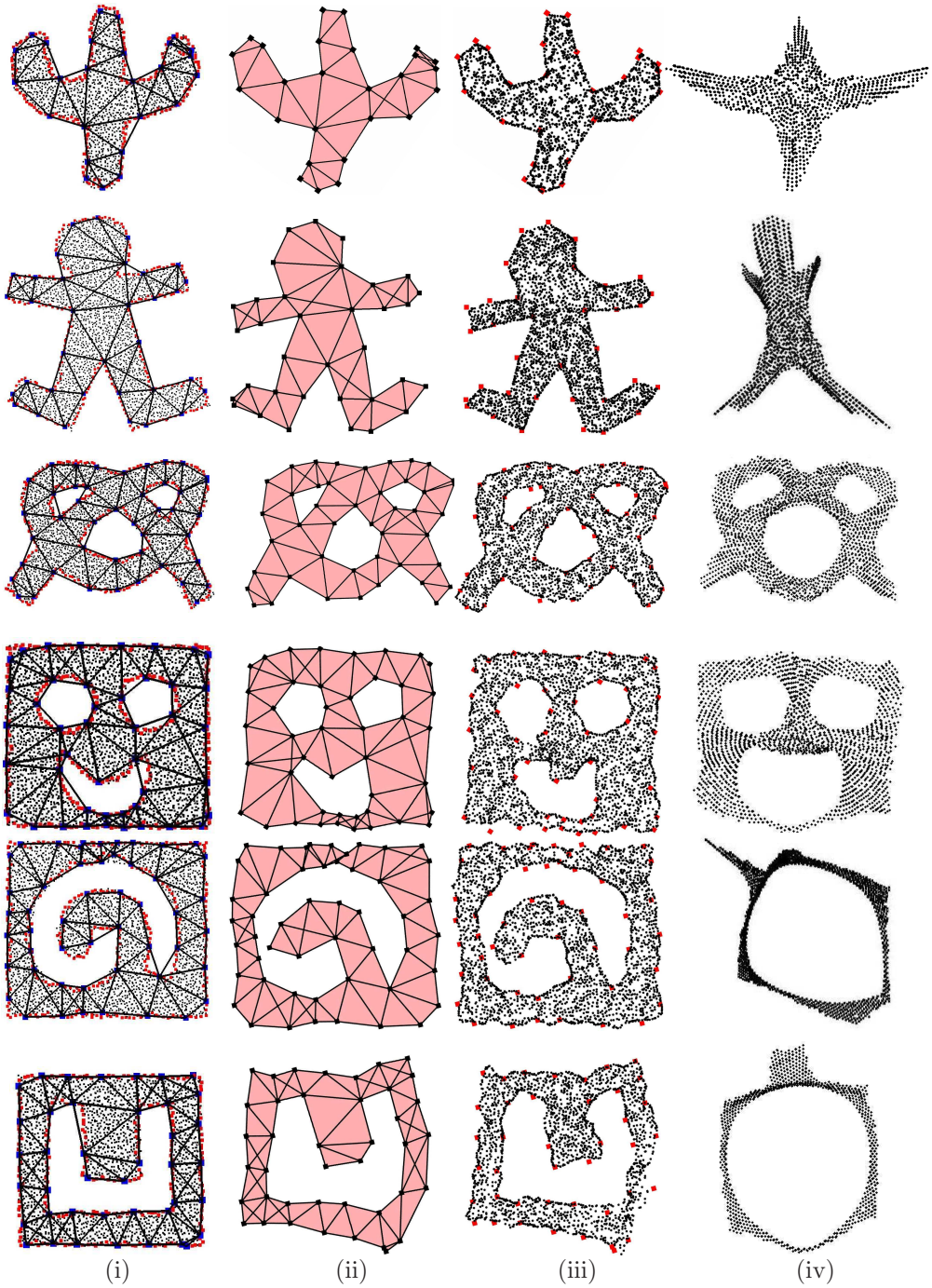


Figure 2.8. From left to right, we have: (i) the true sensor locations and extracted combinatorial Delaunay complex; (ii) embedding of the combinatorial Delaunay complex; (iii) localization of all nodes by our algorithm; (iv) the results produced by MDS on all nodes in the network. The connectivity network is generated with unit disk graph model on nodes placed at perturbed grid points. First row: Cactus, 1692 nodes with average degree of 6.9. Second row: Ginger man, 2807 nodes with average degree of 10. Third row: Pretzel, 2993 nodes with average degree of 9.1. Fourth row: Smiley face, 2782 nodes with average degree of 9.5. Fifth row: Spiral in a box, 2910 nodes with average degree of 9.5. Sixth row: Square with a concave hole, 2161 nodes with average degree of 10.4.

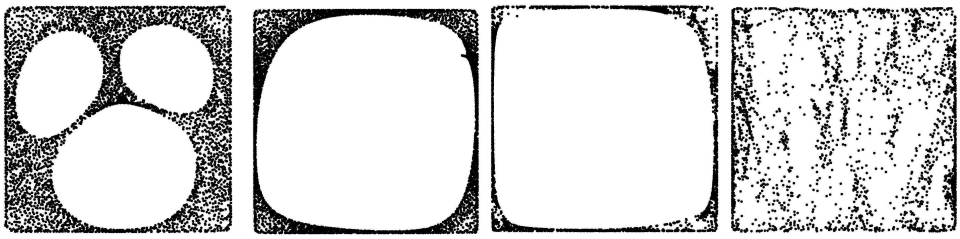


Figure 2.9. Rubberband algorithm results for (i) face (ii) spiral in a box (iii) square with a concave hole (iv) U shape.

2.4.3 Simulation results

The objective of the following simulations is to evaluate our algorithm and compare with MDS or rubberband representations. In particular, we would like to investigate how does the algorithm performance depend on different factors such as the network shape, the node density, landmark density, and communication models.

Influence of network shapes

We applied our algorithm to a number of networks with different layouts, or "shapes". We observed that the performance of our algorithm is fairly stable for all kinds of shapes, but the performance of MDS depends a lot on the shape of the sensor field. We thus include here a few representative pictures in Figure 2.8.

MDS gives reasonable results for some cases (the 1st and 2nd example) but performs quite poorly when the real network has curved pieces (like spirals), and may even introduce an incorrect global flip, as in the 5nd and 6th examples. For a qualitative measure, We have computed the average distance error between the true location and our localization result and that of MDS, scaled by the communication range[5]. In all cases we are consistently better. In some cases when MDS does not produce the correct network layout, we are $4 \sim 7$ times better as shown in Table 2.1.

[5]For alignment, we take three arbitrary landmarks and compute a rotation matrix for both results.

Topology	concave	face	man	pretzel	spiral	cactus	star
Our Alg	1.88	0.91	1.94	0.95	1.11	2.39	2.16
MDS	4.42	2.78	3.24	1.45	7.10	2.82	3.24

Table 2.1. Average location error, scaled by communication range.

Influence of network communication models

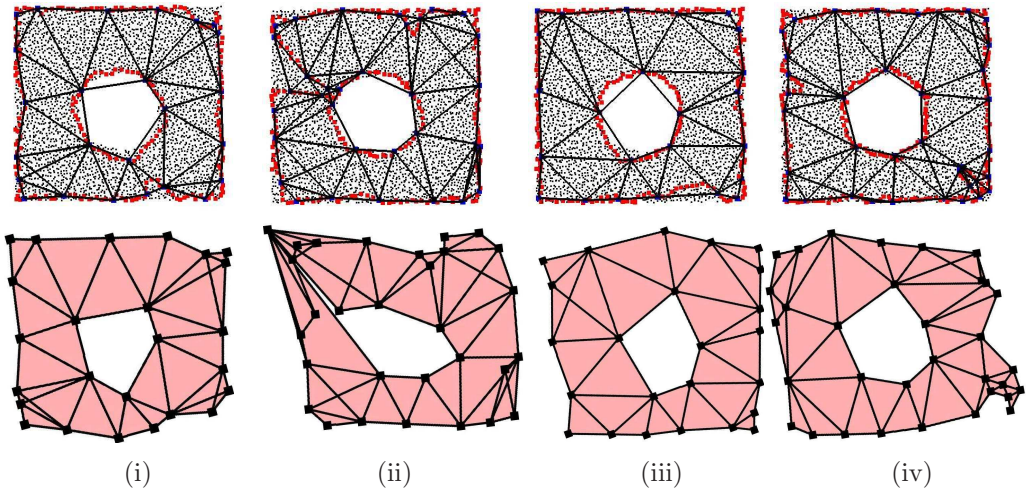


Figure 2.10. Embedding the landmarks under challenging network conditions. The first row shows the ground truth; the second row our embedding of the landmark nodes. From left to right the models depicted are (i) 3443 nodes, avg. degree 10.66. only keep α edges and delete (1-α) edges randomly. α= 0.9. (ii) 3443 nodes, avg. degree 11.95. $\alpha = 0.8$ (iii) 3443 nodes, avg. degree 9.58. quasi-UDG model: We assume that for two nodes whose distance d is between α and 1, there is an edge with probability (1-d)/(1-α). If $d < \alpha$, there must be an edge between them. $\alpha = 0.8$ (iv) 3443 nodes, avg. degree 7.57. $\alpha = 0.6$.

We tested our algorithm on different communication models. The observation is that the embedding result heavily depends on the performance of the boundary detection algorithm. If the boundary detection algorithm faithfully detected the network boundary, the embedding result is satisfactory as well. If the boundaries detected have local deficiencies, then the embedding may have local errors or flips. We show some representative cases in Figure 2.10. Figure 2.10 (i) and (ii) show what happens when a percentage of the links are broken. In (i) a fraction q of the edges in the unit disk graph, randomly selected, are deleted, for $q = 0.1$ and

(ii) $q = 0.2$. In (iii) a quasi-UDG model is used: for two nodes whose distance d is between α and 1, there is an edge with probability (1-d)/(1-α). If $d < \alpha$, there must be an edge between them. $\alpha = 0.8$ in this case. In (iv), we use a quasi-UDG model with $\alpha = 0.6$. As you can see (ii) and (iv) give poor results. The problem in these cases is that the network boundary was not detected accurately. Whenever the boundary deviates from the real network boundary, we discovered that the embedding of the Delaunay triangles may incur local flips (such as the left top corner in (ii) and the right bottom corner in (iv)), as the information carried by the landmarks and the Delaunay triangles on these landmarks is now misleading.

Influence of node density

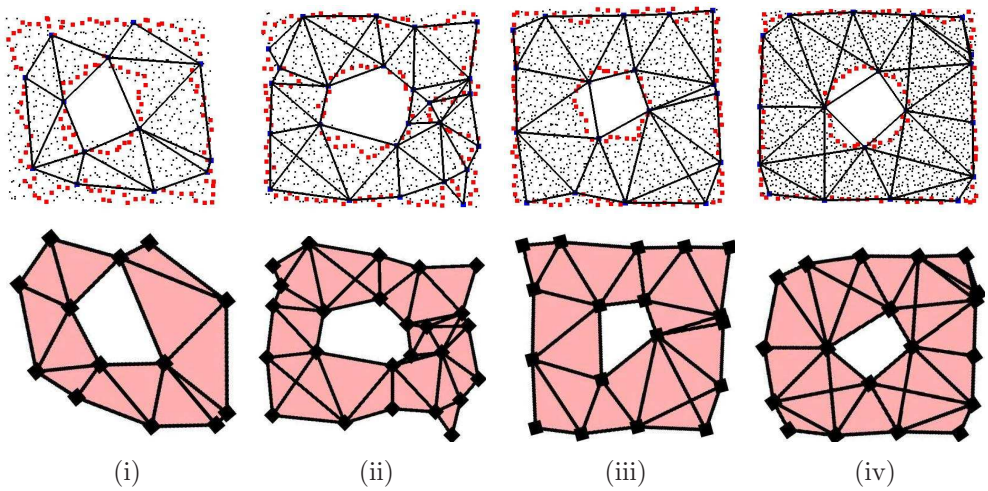


Figure 2.11. Effect of node density/average degree on the embedding, the node densities increase from left to right and the communication ranges are the same for all networks. (i) 677 nodes, avg. degree 5.59 (ii) 840 nodes, avg. degree 6.56 (iii) 1162 nodes, avg. degree 9.2 (iv) 1740 nodes, avg. degree 14.57.

As node density goes higher, the performance of our algorithm improves. There are two reasons for this. One is that the boundary detection algorithm works better with higher node density. The second is that the hop-count distance between nodes is a better approximation of the geodesic distance between them.

The simulations in Figure 2.11 show the results of networks having increasingly

denser nodes from left to right with the same communication range. Networks with higher density normally perform better than lower density networks. Specially, if the average degree is below 7, the boundary detection step fails to faithfully recover the boundary causing the rest of the algorithm performs not good as well.

Influence of landmark density

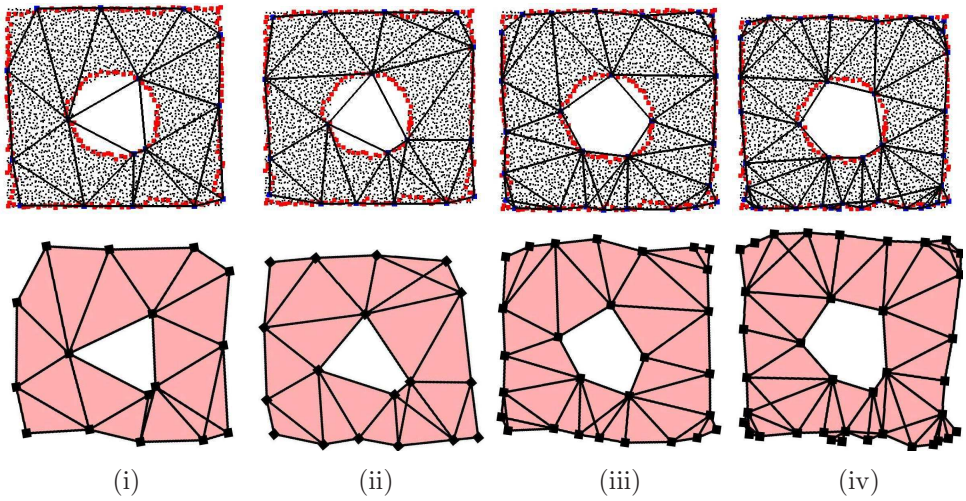
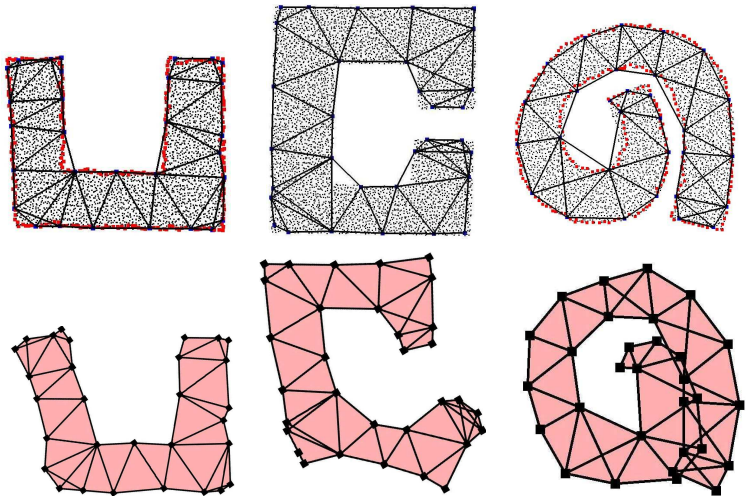


Figure 2.12. Effect of landmark density. All figures with 3443 nodes and avg. degree 11.95. (i) decrease the number of landmarks (ii) standard number of landmarks as we described in algorithm section (iii) increase the number of landmarks (iv) increase the number of landmarks more

The theoretical results in the previous section gives a lower bound on the landmark density to ensure the rigidity of the Delaunay complex. One can certainly select much more landmarks than that. In general, a higher density of landmarks may allow for a slightly better embedding of the network since bends and corners of the network can be captured more accurately. With a very sparse set of landmarks the distance between 2 neighboring landmarks can be grossly exaggerated because the multi-hop path may need to get around a corner. But a denser set of landmarks means that the mass spring embedding of the Delaunay complex runs on a larger set, increasing the computation and communication cost of the algorithm. As shown in Figure 2.12, the result of the algorithm is fairly stable with

different landmark density. Thus the benefit of using a denser set of landmarks may not outweigh the increased cost of doing so.

Error accumulation



(i) (ii) (iii)

Figure 2.13. Possible error accumulation in networks with an elongated shape. In column (i) 3297 nodes, avg. degree 3297. We show a U-shaped graph properly embedded with minor distortion due to the use of hop-count distances. In (ii), 5028 nodes, avg. degree 14.9. The embedded network with a 'C' shape endures higher distortion. In (iii), 3910 nodes, avg. degree 15. Error accumulation causes the spiral to overlap on itself.

Recall that the algorithm uses the hop count distance between landmarks to approximate their geodesic distance. Thus we may observe error accumulation in the embedding when the network has an elongated shape as shown in Figure 2.13. In these examples, the embedded shape is distorted and may have self-overlap (as in example (iii)), due to error accumulation.

2.4.4 Further discussion

MDS. Multidimensional scaling is a standard statistical approach that takes the all pairs proximity and recovers a 2D embedding of the vertices with linear projection methods such as principle component analysis (PCA). To better understand why MDS introduces incorrect flips, the intuition behind it is that the network hole causes the hop count distances to be not necessarily a good estimation of the Euclidean distance of the nodes. For example, the node at the tip of the spiral has a fairly long network distance to the opposite node 'across the lake'. MDS has no way of distinguishing this imprecise and misleading measurements from other good distance estimates. In fact, the misleading measurements seem to 'outweigh' the good measurements and MDS eventually chooses to flip the spiral over. Our other examples also show that the MDS tends to enlarge the hole in the middle. Another limitation is that MDS behaves more or less like a blackbox and it is not easy to interpret the results and not to mention improving it.

On a different note, we remark that using multi-dimensional scaling on the shortest path distance matrix in a unit-disk graph setting is essentially the same algorithm as in Isomap [76], proposed by Tenenbaum, de Silva and Langford, for non-linear dimension reduction for high-dimensional data embedded in a low dimensional manifold. The famous result tested in Isomap is a 2D swiss roll shape manifold in 3D. With shortest path distance metric instead of the Euclidean metric in the ambient space, Isomap is able to 'flatten up' the swiss roll and recover the non-linear manifold. If the points are embedded on a 2D manifold but with possibly holes, i.e., a slice of Swiss cheese rolled up in 3D, our algorithm will recover a much more faithful representation of the unfolded 2D manifold. The fundamental idea here of using carefully selected short distances and patching up the local simplices suggests a generic rule of recovering the inherent topology and geometry of data points in an ambient space. This is one direction we will explore further. In a general setting, it requires both the understanding of topological features inherent in the geodesic distances and rigidity results in higher dimensions, both of which are not trivial.

Graph rigidity. The theory of graph rigidity in 2D has been relatively well understood. For example, there is a combinatorial condition, the Laman condition, to characterize graphs that are generically rigid. There is also an efficient algorithm, the pebble game [47], to test whether a graph is generically rigid in time $O(n^2)$. Similarly, both a combinatorial characterization of globally rigid graphs and polynomial algorithms for testing such graphs are known [10,42]. It is however not trivial to apply these rigidity results in the development of efficient localization algorithms. Given a graph with the edge lengths specified, finding a valid graph realization in $\mathbb{R}^d$ for a fixed dimension d is a NP-complete problem [7,8,72]. Even if we know a graph is globally rigid in 2D, there is no known efficient algorithm to find the realization of the graph in 2D with given edge lengths.

See section 4.6 for further discussion of previous work in rigidity theory.

Comparatively, we focus on the global rigidity of the combinatorial Delaunay complex, that has high-order topological structures than graphs. The combinatorial Delaunay *complex* is globally rigid but the combinatorial Delaunay *graph* is not necessarily globally rigid. Different from the graph rigidity approach, this algorithm does not require explicitly that the network to be embedded is globally rigid. This sheds some light on solving the network localization problem when the network is (uniformly) sparse but not rigid, such as a grid-like network with punched holes. Our current algorithm does not work well in the case of extremely low density networks because the boundary detection algorithm fails to find the network boundary effectively. In future work we plan to remove the dependency of boundary detection step in the algorithm and hope to apply it in localizing low-density non-rigid networks.

2.5 Conclusion

The novelty of our localization scheme is to extract high-order topological information to solve the notoriously difficult problem of resolving flip ambiguities. Geometric information of sensor nodes (e.g. node locations) has been recognized as an important character in sensor networks. The global topology of the sensor

field is shown here to be helpful in recovering the network geometry.

2.6 Appendix

2.6.1 Proofs in section 2.2.1

Observation 2.6.1. *The inner medial axis of $\mathcal{R}$ measured in terms of Euclidean distance is the same as that measured in terms of geodesic distance.*

Proof: Take the maximum size ball centered at a point p on the medial axis under Euclidean distance measure. This ball touches two or more points on the boundary and has no boundary points in its interior. Thus the geodesic distances from p to the tangent points are the same as the Euclidean distances. In other words, a point p is on the medial axis under the Euclidean distance is also on the medial axis under the geodesic measure.

On the other hand, take a maximum size ball centered at a point p on the medial axis under the geodesic distance measure and its tangent points on $\partial\mathcal{R}$. We argue that the geodesic shortest path from p to its tangent point must be a straight line. If otherwise it can only bend at a point q on the boundary $\partial\mathcal{R}$. This means q is a closer boundary point than the tangent point, which contradicts with the assumption. Thus the point p is also on the medial axis under the geodesic distance measure. □

Next we prove an important Lemma about the inner local feature size. This Lemma and its proof are motivated by [4] and will be useful for the proofs in Subsection 2.2.3.

Lemma 2.6.2. *Given a disk B containing at least two points on $\partial\mathcal{R}$, for each connected component of $B \cap \mathcal{R}$, either it contains a point on the inner medial axis, or its intersection with $\partial\mathcal{R}$ is connected.*

Proof: We take one connected component C of $B \cap \mathcal{R}$ and assume that it does not contain a point on the inner medial axis and intersects $\partial\mathcal{R}$ in two or more

connected pieces. Now we take a point u in C but u is not on $\partial\mathcal{R}$. Now take u's closest point on $C \cap \partial\mathcal{R}$. If the closest point is not unique, then u is on the inner medial axis and we have a contradiction. Now the closet point p stays on one connected piece of $C \cap \partial\mathcal{R}$. We take u's closest point on a different piece of $C \cap \partial\mathcal{R}$, denoted as q. See Figure 2.14. Now as we move a point x from u to q

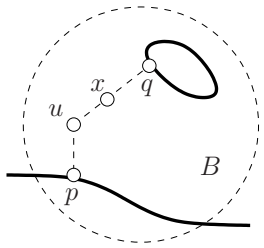


Figure 2.14. Each connected component of $B \cap \mathcal{R}$ either contains a point on the inner medial axis or its intersection with $\partial\mathcal{R}$ is connected.

along the geodesic path between u and q, x's closest point on $C \cap \partial\mathcal{R}$ starts with p and eventually becomes q. So at some point x the closest point changes. That point x is on the inner medial axis. This leads to a contradiction, and hence the claim is true. □

2.6.2 Proofs in section 2.2.3

Observation 2.2.6. *Two Voronoi vertices connected by a Voronoi edge correspond to two Delaunay triangles sharing an edge.*

Proof: Recall that each Voronoi vertex x certifies a Delaunay triangle of three landmarks u, v, w. First we argue that the points on the Voronoi edge connecting Voronoi vertices x and y must have their two closest landmarks among u, v, w. Certainly if one point on the Voronoi edge has one of its closest landmark to be p and p is not any of u, v, w, then this point is a Voronoi vertex. Without loss of generality, we assume that y has three closest landmarks u, v, z. Thus the corresponding Delaunay triangles of x, y are $\triangle uvw$ and $\triangle uvz$ sharing an edge uv. □

Observation 2.2.7. *For any two adjacent landmarks u, v on the same boundary cycle, there must be a Voronoi vertex inside $\mathcal{R}$ whose closest landmarks include u, v.*

Proof: We take two adjacent landmarks u, v and consider the set of points in $\mathcal{R}$ with equal distance from u, v. The mid-point on the geodesic path connecting u, v, denoted by x, is at an equal distance from u, v. We take a disk through u, v centered at x and move the disk while keeping it through u, v. Its center will trace a curve called $C(u, v)$ with all the points on $C(u, v)$ having equal distances from u, v. $C(u, v)$ has two endpoints p, q with q on the boundary segment in between u, v and p also on the boundary. Take $r = d(p, u) = d(p, v)$. See Figure 2.15.

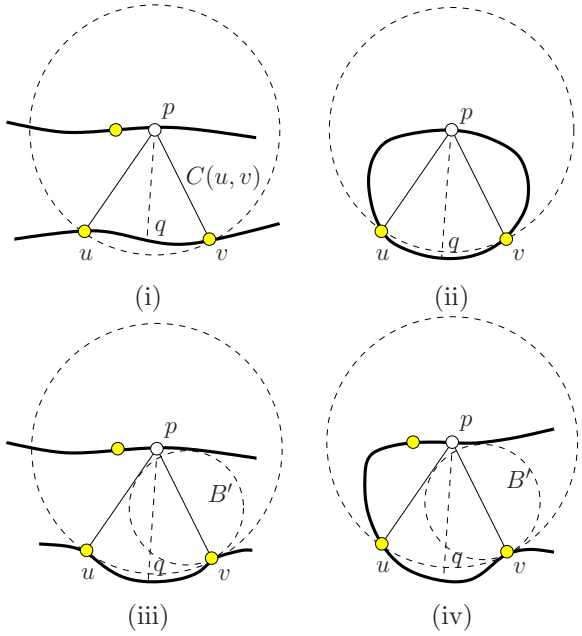


Figure 2.15. u, v are two adjacent landmarks. The point p on the boundary has its closest landmarks as u, v. (i)-(iv) four possible cases.

We claim that there must be a Voronoi vertex on $C(u, v)$ that involves u, v and we prove this claim by contradiction. Otherwise, p's two closest landmarks are u, v — the ball $B_r(p)$ centered at p with radius r contains no other landmark inside. We take $r^- = r - \varepsilon$ with $\varepsilon \to 0$. Thus $B_{r^-}(p)$ contains no landmark. Now we see that this will violate the sampling condition if we can show that there is a point on the inner medial axis inside $B_{r^-}(p)$ (meaning that $r^- \geq ILFS(p)$).

We take the connected component of $B_{r^-}(p) \cap \mathcal{R}$ that contains the curve $C(u, v)$, denoted by F. By Lemma 4.8.4, if F does not contain a point on the inner medial axis, then its intersection with the boundary $\partial\mathcal{R}$ is connected. Now we do a case analysis depending on how the boundary curve goes through u and v. In Figure 2.15 (i) & (ii), the ε-neighborhood of the boundary at u, v also intersects $B_{r^-}(p) \cap \mathcal{R}$. In (i), $F \cap \partial\mathcal{R}$ has two connected pieces, thus leading to a contradiction. In (ii), the boundary between u, v through p is completely inside $B_{r^-}(p)$, which has no other landmark inside. In this case there are only 2 landmarks, namely u, v, on the boundary cycle containing p. This contradicts our sampling condition.

If the boundary at v (or u, or both) is only tangent to $B_{r^-}(p) \cap \mathcal{R}$ (meaning that $B_{r^-}(p)$ does not contain any ε-neighborhood of v, see Figure 2.15 (iii) & (iv)), we argue that F contains a point on the inner medial axis. To see that, we take the ball $B_r(p)$ tangent at v with v's ε-neighborhood outside the ball. Now we shrink it while keeping it tangent to v until it is tangent to two points on the boundary of F. Now the center of the small ball B' is on the inner medial axis, which is inside $B_{r^-}(p)$. Thus we have the contradiction. The claim is true. □

Lemma 2.2.8. *If there is a continuous curve C that connects two points on the boundary $\partial\mathcal{R}$ such that C does not contain any point on Voronoi edges, then C cuts off a topological 1-disk of $\partial\mathcal{R}$ with at most one landmark inside.*

Proof: Without loss of generality we assume that C has no other boundary points in its interior. Assume C connects two points p, q on the boundary. Since C does not cut any Voronoi edges, C must stay completely inside the Voronoi cell of one landmark say u. Without loss of generality assume that u is to the right of boundary point q. See Figure 2.16(i). Now the boundary of Voronoi cell of u is partitioned by the curve C, with one part completely to the left of C. Consider one of the intersections between the Voronoi cell boundary of u with the region boundary $\partial\mathcal{R}$, say p'. We consider the ball $B_r(p')$ with $r = d(p', u)$. The point p' has two closest landmark, with one of them as u and the other to the left of C, denoted as w. Now, this ball cannot contain any other landmark besides u, w. We

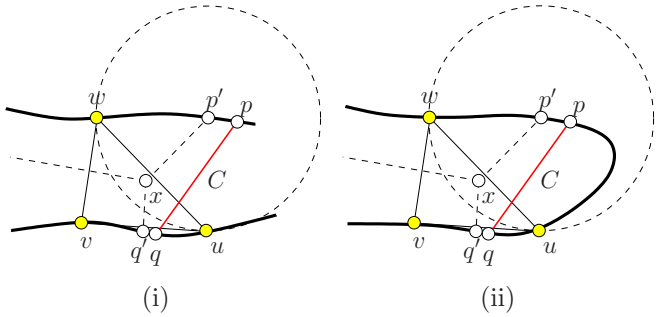


Figure 2.16. (i) C is inside the Voronoi cell of landmark u to the right of C. (ii) the curve C cuts off a segment of $\partial\mathcal{R}$ with no other landmark inside.

argue by Lemma 4.8.4 that the component of $B_r(p') \cap \mathcal{R}$ containing p' intersects $\partial\mathcal{R}$ in a connected piece. Otherwise $B_r(p')$ contains a point on the inner medial axis, which means $r > ILFS(p')$. Thus by the sampling condition there must be a landmark inside $B_r(p')$.

Now, since the component of $B_r(p') \cap \mathcal{R}$ containing p' intersects $\partial\mathcal{R}$ in a connected piece, this intersection is a continuous segment between u and w on $\partial\mathcal{R}$, completely inside $B_r(p')$, by using the same argument as in the previous lemma; see Figure 2.16 (ii). In this case, the curve C cuts off a segment of $\partial\mathcal{R}$ with at most one landmark inside. The claim is true. □

Chapter 3

Localization using Incremental Delaunay Refinement Methods

3.1 Introduction

In this chapter we continue our investigation into the network localization problem for a large-scale sensor network with a complex shape but we. Here we design an algorithm that does not rely on boundary detection as a precondition to localization. Again, we do not assume any anchor nodes with known locations and use only network connectivity information to recover the relative positioning of all the nodes. Thus we require no extra hardware supplements (e.g., for angle or distance measurements) and investigate a same fundamental problem: can the network geometry be reconstructed using network connectivity alone?

Challenge: graph rigidity. As mentioned above in section 2.4.4, a major challenge in anchor-free localization is to handle possible flip ambiguities. For a simple example, two triangles sharing an edge can be embedded in two possible ways, with the two triangles on the same side, or on opposite sides of the common edge. In general, whether a graph has a unique embedding or not is investigated in graph rigidity theory [41]. Refer to section 2.4.4 for more details.

Our Approach. The work in this chapter is a follow-up to chapter 2 in which

we proposed a framework for connectivity-based anchor-free localization with a different approach. We select some sensor nodes as *landmarks* and compute the Voronoi diagram with all the nodes closest to the same landmark grouped into the same Voronoi cell. We extract the *combinatorial Delaunay complex*[1] as the dual complex of the landmark Voronoi diagram – there is a Delaunay edge (or in general a k-simplex) between two (or k) landmarks if their Voronoi cells share some common nodes. In contrast with the previous rigidity work on *graphs*, we focus on the global rigidity property of the *combinatorial Delaunay complex*, that has high-order topological structures (such as Delaunay triangles) compared with graphs that do not (having only vertices and edges). The combinatorial Delaunay *complex* may be globally rigid when the combinatorial Delaunay *graph* is not. See Figure 4.2 for an illustration. As opposed to the graph rigidity approaches

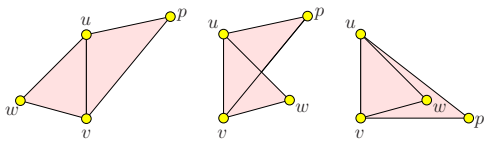


Figure 3.1. Two Delaunay triangles $\triangle uvw$ and $\triangle uvp$ sharing an edge. The first figure is the only valid embedding, because in a simplicial complex two simplices can only intersect at a common face. The graph is not globally rigid.

discussed earlier, this approach does not explicitly require that the network to be embedded is globally rigid. This sheds some light on providing reasonable localization results when the network has low node density (even uniformly sparse but non-rigid graphs such as a grid-like network with punched holes).

In chapter 2 we proposed an algorithm for landmark selection to guarantee that the generated combinatorial Delaunay complex is globally rigid and admits a unique realization in the plane. This leads to an algorithm to put together the Delaunay triangles in an incremental manner to find the unique embedding of the Delaunay complex. The Delaunay complex, once embedded, can be used to localize the rest of the nodes in the network. The algorithm for selecting landmarks

[1]The Delaunay complex is defined in the notion of abstract simplicial complex [24]. A set α is an (abstract) *simplex* with dimension $\dim \alpha = \operatorname{card} \alpha - 1$, i.e., the number of elements in α minus 1. A finite system A of finite sets is an *abstract simplicial complex* if $\alpha \in A$ and $\beta \subseteq \alpha$ implies $\beta \in A$.

first uses a boundary detection algorithm (see [29, 30, 32, 33, 52, 78]) to identify the network boundary nodes and then selects the landmarks to be a γ-*sample* with $\gamma < 1$. Specifically, every boundary node has a landmark within its *inner local feature size*, defined as the distance to the closest node on the *medial axis* (which is the collection of nodes with two or more closest nodes on the boundary). More details on this algorithm and compared to the one above will be further explained in the next section.

The dependency on the boundary detection algorithm puts limitations on the applicability of the localization algorithm, in particular, in the case of extremely low density networks, for which boundary detection algorithms do not work well. Examples of some of these cases were shown above.

Our Contribution. The main contribution in this chapter is an incremental landmark selection algorithm that does not assume knowledge of the network boundary. In particular, we start with no knowledge of the network topology (whether there are holes or how many there are, etc.) and develop local conditions to test whether a node should be included as a new landmark. Landmarks are included in a distributed manner until the global rigidity property and the coverage property are satisfied. The global rigidity property is to guarantee that the Delaunay complex is globally rigid and thus has a unique embedding. The coverage property ensures that every node is not far from the embedded Delaunay complex. Thus the embedded Delaunay complex approximates the shape of the original network. The landmarks selected naturally adapt to the local geometry of the network, with a higher density of landmark nodes selected in regions with more detailed and complex features. This new landmark selection algorithm greatly enhances the robustness of our algorithm in cases of extremely sparse or even non-rigid networks, or networks with very complicated shape that are challenging for boundary detection algorithms.

Once the landmarks are selected, we build the combinatorial Delaunay complex in the same way as in chapter 2. The combinatorial Delaunay complex is embedded by gluing adjacent triangles side by side. The resulted embedding of the landmarks are then used to embed all other nodes by using simple trilateration.

We demonstrate the improved performance of our algorithm in various network settings in the simulation section and compared with our previous algorithm. So far, we are not aware of any other localization algorithms using only connectivity information with comparable performance.

3.2 Incremental Delaunay Refinement: Theory

In this section we use a continuous setting to go though the framework of network localization by the combinatorial Delaunay complex and provide the theoretical foundation of the incremental Delaunay refinement algorithm. The algorithm implementation in the network setting is elaborated on in the next section. The sensor field is assumed to be a continuous domain $\mathcal{R} \in \mathbb{R}^2$ with perhaps some interior holes. For any two points $p, q \in \mathcal{R}$, we denote by $|pq|$ their Euclidean distance and $d(p, q)$ the *geodesic* distance (shortest path distance) between them inside $\mathcal{R}$. The geodesic distance is an analog of the minimum hop count distance in the discrete setting. A ball centered at a point p of radius r, denoted by $B_r(p)$, contains all the points within geodesic distance r from p.

The boundary of $\mathcal{R}$ is denoted as $\partial\mathcal{R}$ and may have multiple cycles. The *inner medial axis* of $\mathcal{R}$ is the collection of points in $\mathcal{R}$ that have two or more closest points on the boundary $\partial\mathcal{R}$. The *inner local feature size* of a point $p \in \partial\mathcal{R}$, denoted by $ILFS(p)$, is the distance from p to the inner medial axis of $\mathcal{R}$. A set of landmarks L on the boundary $\partial\mathcal{R}$ is called a γ-sample[2] if for any point $p \in \partial\mathcal{R}$, there is at least one landmark within distance $\gamma \cdot ILFS(p)$ from p.

Suppose L is a set of landmarks on the domain boundary $\partial\mathcal{R}$, the *Voronoi cell* of a landmark u, denoted as $V(u)$, includes all the points that have u as a closest landmark:

$$V(u) = \{p \in \mathcal{R} \,|\, d(p, u) \leq d(p, v), \forall v \in L, v \neq u\}.$$

[2]Notice that the definition of γ-sample is different from the typical definition in geometric processing [4, 5] where the local feature size is used. The medial axis for a domain $\mathcal{R}$ has two parts, one inside $\mathcal{R}$ and one outside $\mathcal{R}$. For our setting we do not have access to the part of the medial axis outside of $\mathcal{R}$ and we can only use the inner local feature size.

The collection of Voronoi cells is denoted as the landmark Voronoi diagram $V(L)$ for the set L of landmarks. A point is called a *Voronoi vertex* if it has equal distance to at least three landmarks. The Voronoi vertices inside $\mathcal{R}$ are called the *inner Voronoi vertices*. A ball $B_r(p)$ centered at an inner Voronoi vertex p with radius r equivalent to the distance from p to the closest landmarks is called a *Voronoi ball*.

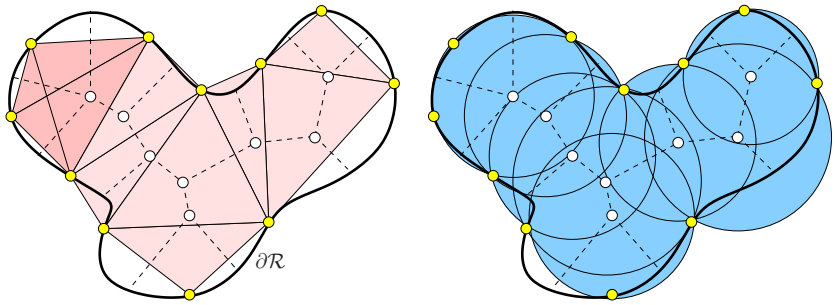


Figure 3.2. Left: The Voronoi graph (shown in dashed lines) and the Delaunay complex for a set of landmarks on the boundary $\partial\mathcal{R}$. The Delaunay simplices (vertices, edges, triangles, tetrahedrons) are shaded. Right: The union of Voronoi balls approximately covers the domain $\mathcal{R}$.

The *combinatorial Delaunay complex* of the landmarks L, denoted by $DC(L)$, is the collection of sets

$$DC(L) = \{\alpha \subseteq L \mid \cap_{u\in\alpha} V(u) \neq \emptyset\}.$$

In other words, a set $\alpha \subseteq S$ is a Delaunay simplex if the intersection of the Voronoi cells of landmarks of α is non-empty. The Delaunay complex has naturally 0-dimensional simplices such as the landmarks, 1-dimensional simplices such as Delaunay edges, and 2-dimensional simplices such as Delaunay triangles, and possibly higher order simplices such as tetrahedrons. See Figure 4.3 for an example.

3.2.1 γ-sample, rigidity and coverage

In chapter 2, we showed a framework for network localization by embedding the Delaunay complex $DC(L)$ extracted from the network connectivity. We proved

that when the landmarks are selected as a γ-sample of the domain $\mathcal{R}$ with $\gamma < 1$, the Delaunay complex $DC(L)$ is globally rigid and thus admits a unique realization in the plane. This establishes the foundation of the localization algorithm as we can now embed the Delaunay complex incrementally and then localize the entire network with the Delaunay complex as a structural skeleton. See Figure 3.3 (i) for an example when the rigidity condition is violated. In this case one does not know how to embed the Delaunay triangles as the left part can rotate freely around the right one.

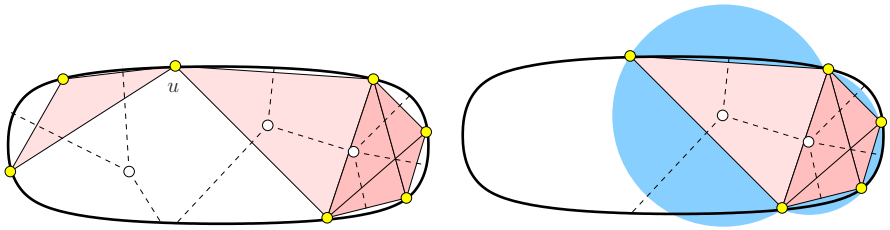


Figure 3.3. Bad cases for localization. Left: the Voronoi edges of landmark u form two connected components. The combinatorial Delaunay complex is not rigid. Right: The union of Voronoi balls do not well cover the domain $\mathcal{R}$. The problem is that the current combinatorial Delaunay complex does not capture the shape of the sensor field that are not yet covered.

For localization, we also want that the Delaunay complex provides good 'coverage' of the sensor field in the sense that every node is not very far from the Delaunay complex, so that the Delaunay complex faithfully represents the network shape. In particular, we take $\mathcal{B}$ to denote the union of all the Voronoi balls, and $\mathcal{U}$ the shape of the union of these balls. As we will prove later, the γ-sample condition guarantees that the union of Voronoi balls is a good approximation of $\mathcal{R}$ and the approximation is improved as the density of landmarks increases. See Figure 4.3 (ii) for an example of a good case and Figure 3.3 (ii) for an bad example. Rigorously, we define that the Delaunay complex $DC(L)$ *δ-covers* $\mathcal{R}$ if every point $x \in \mathcal{R}$ is within distance $(1+\delta) \cdot r$ from the center p of a Voronoi ball $B_r(p)$, where r is the radius of this Voronoi ball.

Using the union of the Voronoi balls to approximate the shape $\mathcal{R}$ was initially proposed in geometric processing and computer graphics [5]. It has been shown that the errors in the position and normal of the surface of $\mathcal{U}$ with $\mathcal{R}$ is bounded everywhere, given sufficiently dense samples on the $\partial\mathcal{R}$. However, we cannot

directly apply the results in [5] as there are a couple of differences with our setting. First, the metric we are working with is the geodesic shortest path metric, instead of the Euclidean metric used in [5]. In addition, as we only have sensors in the interior of $\mathcal{R}$, we do not have access to the part of medial axis that is outside $\mathcal{R}$ and we are only able to use the inner local feature size to define the γ-sample.

Before we prove the coverage theorem, let us first understand the boundary of the union of balls $\mathcal{U}$. The boundary of $\mathcal{U}$ contains some circular arcs from the balls in $\mathcal{B}$. We first characterize what arc can possibly stay on the boundary of $\mathcal{U}$. Each Voronoi edge in $V(L)$ has two endpoints, being either a Voronoi vertex or a point on the boundary $\partial\mathcal{R}$. A Voronoi edge with two Voronoi vertex endpoints is called an *inner edge*. A Voronoi edge with two endpoints on the boundary is called an *outer edge*. A Voronoi edge with both a Voronoi vertex and a point on the boundary is called a *mixed edge*. For each Voronoi ball B, the three landmarks partition its boundary ∂B into three circular arcs. We label the arc between two landmarks u, v with the label of the Voronoi edge of u, v as either inner, outer, or mixed. We now claim that only mixed arcs can possibly appear on $\partial\mathcal{U}$. First realize that the interior points of an inner arc cannot stay on the boundary of $\mathcal{U}$, since the arc is enclosed inside the union of the two Voronoi balls that go through u, v. Second if we choose $\gamma < 1$, then the Voronoi diagram inside $\mathcal{R}$ is connected as proved in Corollary 2.10 in chapter 2. Thus there cannot be an outer edge in $V(L)$, since this edge will be disconnected from the rest of the Voronoi diagram. Now for a Voronoi ball $B_r(p)$ with a mixed edge between landmarks u, v we define a *pie* as the set of points in $\mathcal{R}$ bounded by the boundary segment between u, v and the shortest paths from p to u, v. Only the points inside a pie with a mixed arc can possibly stay outside $\mathcal{U}$. See Figure 3.13. Notice that in the case of degeneracy, a Voronoi ball can have four or more landmarks. The classification of edges and the proof later are the same in that case.

In the following theorem we show that the landmarks being a γ-sample not only guarantees the global rigidity of the Delaunay complex (as shown in chapter 2) but also the good coverage property (proof in the Appendix). In the next section we describe a new algorithm to achieve both global rigidity and coverage, without

using network boundary explicitly.

Theorem 3.2.1. *For a connected region $\mathcal{R} \subseteq \mathbb{R}^2$, we select landmarks L as a γ-sample on the region boundary $\partial\mathcal{R}$ with $\gamma < 1$. Then the Delaunay complex δ-covers $\mathcal{R}$, with $\delta = 2\gamma/(1-\gamma)$.*

3.2.2 Landmark selection for both rigidity and coverage

Based on the previous discussion, there are two desirable criteria, namely, global rigidity and coverage, for the final Delaunay complex. Choosing the landmarks as a γ-sample will lead to both properties. Nevertheless our previous algorithm would require the knowledge of the network boundary. In this subsection we investigate *local* conditions for landmark selection to guarantee both rigidity and good coverage of the induced Delaunay complex:

1. **Local Voronoi edge connectivity:** The Voronoi edges for each landmark u form a connected set.
2. **Local Voronoi ball coverage:** Each node x inside a Voronoi cell $V(u)$ is δ-covered by a Voronoi ball $B_r(p)$, where p is a Voronoi vertex with landmark u.

We first show that if both conditions are satisfied for a set of landmarks L, then the Delaunay complex $DC(L)$ satisfies both the global rigidity and coverage property. This is relatively straightforward. After this, we examine how to design a landmark selection algorithm to meet these conditions.

Rigidity of the Delaunay complex

When the local Voronoi edge connectivity condition is met, we argue that the Delaunay complex is globally rigid. To do that, we will make use of a theorem proved in chapter 2:

Theorem 3.2.2. *[Global Rigidity] If $V(L)$ is connected inside $\mathcal{R}$, the Delaunay complex $DC(L)$ is globally rigid.*

The local Voronoi edge connectivity immediately implies the global Voronoi edge connectivity. If otherwise, there must be one landmark whose Voronoi edges have two or more connected components, since the union of all the Voronoi cells is $\mathcal{R}$. Thus the local Voronoi edge connectivity condition implies the global rigidity of $DC(L)$.

Coverage of the Delaunay complex

If the local Voronoi ball coverage condition is met for every Voronoi cell, then the coverage property of the Delaunay complex follows directly.

Incremental Delaunay refinement algorithm

Since both conditions can be tested locally, we naturally have the following incremental landmark selection algorithm: for each Voronoi cell $V(u)$,

1. If the first condition is not met, the Voronoi edges with u have two or more connected components. Since each Voronoi edge has either a Voronoi vertex or a point on $\partial\mathcal{R}$ as endpoints, we select, among all the endpoints of Voronoi edges of u on $\partial\mathcal{R}$, the one that is *furthest* from u as a new landmark.
2. If the first condition is met, we check the second condition. Among all the points that violate the local Voronoi ball coverage condition, we select the one that is least covered as a new landmark: $\arg\max_x \min_{B_r(p)}\{\delta' \,|d(x,p) = (1+\delta')r\}$. That is, for each such point x, we choose the Voronoi ball $B_r(p)$ with p such that $d(x,p) = (1+\delta')r$ with smallest possible δ'. And we select the point x with the largest such δ'.

This landmark selection algorithm always selects landmarks on the network boundary[3] but it does not require the detection of the network boundary, nor does it require the knowledge of the medial axis and local feature size, whose computation is sensitive to noise. New landmarks in different Voronoi cells can

[3]Landmarks added by condition 1 will be on $\partial\mathcal{R}$ for sure. For the landmarks added by the 2nd condition, by the same argument as in Theorem 3.2.1 the points outside the Voronoi balls must be inside the pies. And the least uncovered point stays on the region boundary $\partial\mathcal{R}$.

be inserted in parallel as the algorithm executes locally inside each Voronoi cell. Thus the new landmark selection method is more robust and practical compared with the γ-sampling used in our previous algorithm. Next, we show the algorithm terminates and the landmarks generated have bounded density.

Landmark density by incremental refinement

Here we show that every landmark q added by the incremental algorithm is not sufficiently covered by existing landmarks, i.e., the distance to its closest landmark is at least $\gamma \cdot ILFS(q)$ for an appropriate parameter $\gamma < \min(\frac{1}{3}, \frac{\delta}{2+\delta})$. If a point $x \in \partial\mathcal{R}$ is within $\gamma \cdot ILFS(x)$ from a landmark, we say x is γ-*covered*. The proofs of the following results are shown in the Appendix.

Lemma 3.2.3. *If a Voronoi cell $V(u)$ violates the local Voronoi edge connectivity condition, the new landmark q selected is not covered by any landmark within $\gamma \cdot ILFS(q)$, for any $\gamma < 1/3$.*

Lemma 3.2.4. *If a Voronoi cell $V(u)$ for a landmark u violates the local Voronoi ball coverage condition, the new landmark q selected is not covered by any landmark within $\gamma \cdot ILFS(q)$, $\gamma = \delta/(2+\delta)$.*

The above results show that our local conditions do identify points on the boundary that need to be γ-covered for $\gamma < \min(\frac{1}{3}, \frac{\delta}{2+\delta})$. If the inner local feature size for any point $x \in \partial\mathcal{R}$ is at least ε for some fixed ε, then the incremental Delaunay refinement algorithm will eventually terminate, as every new landmark included covers at least an interval of length $2\varepsilon\gamma$ centered at itself on $\partial\mathcal{R}$. This procedure cannot go on indefinitely.

We remark that the algorithm will certainly terminate when the landmark set is a γ-sample for any $\gamma < 1$, but it may also terminate before that if both the rigidity and coverage conditions are met, as shown in Figure 3.4. This can be understood in terms of our algorithm picking up the major geometric features and ignoring the noisy features of $\mathcal{R}$. The rigidity and coverage properties guarantee that the reconstructed Delaunay complex will approximate $\mathcal{R}$ and are what we

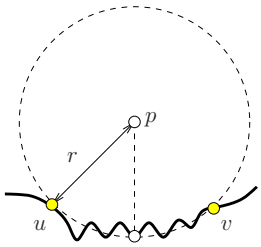


Figure 3.4. The landmark set may not be a γ-sample of $\partial\mathcal{R}$. The local feature size for points on the segments between u, v is however much smaller than the distance to u or v. Unless we really want to capture the wiggling features between u, v, the γ-sample is an overkill.

really care about in our localization algorithm. The γ-sample for $\mathcal{R}$ can be much denser than what is needed in practice.

Last we show that the landmark set generated by the incremental algorithm has bounded density.

Theorem 3.2.5. *Suppose L is the generated landmark set by the incremental algorithm. If any landmark from L is removed, then it is not a γ'-sample of $\partial\mathcal{R}$, with $\gamma' = \gamma/(1+\gamma)$, $\gamma < \max(1/3, \delta/(2+\delta))$.*

3.3 Incremental Delaunay Refinement: Distributed Implementation

3.3.1 Algorithm description

Suppose a large number of sensor nodes are scattered in a geometric region, where nearby nodes can directly communicate with each other. Similar to the setting in chapter 2, we do not enforce that the communication graph follows the unit disk graph model (in our simulations we use both a quasi-UDG model and a probabilistic radio model), nor do we assume any knowledge of the node locations or inter-distances. We select landmarks incrementally in the network until both the global rigidity and the coverage property are satisfied as described in Section 3.2. Next we will explain the distributed implementation of each algorithm step in detail. Unless specified otherwise, all the distances, by default, refer to the geodesic distance, which is measured by the minimum hop count between two nodes in a

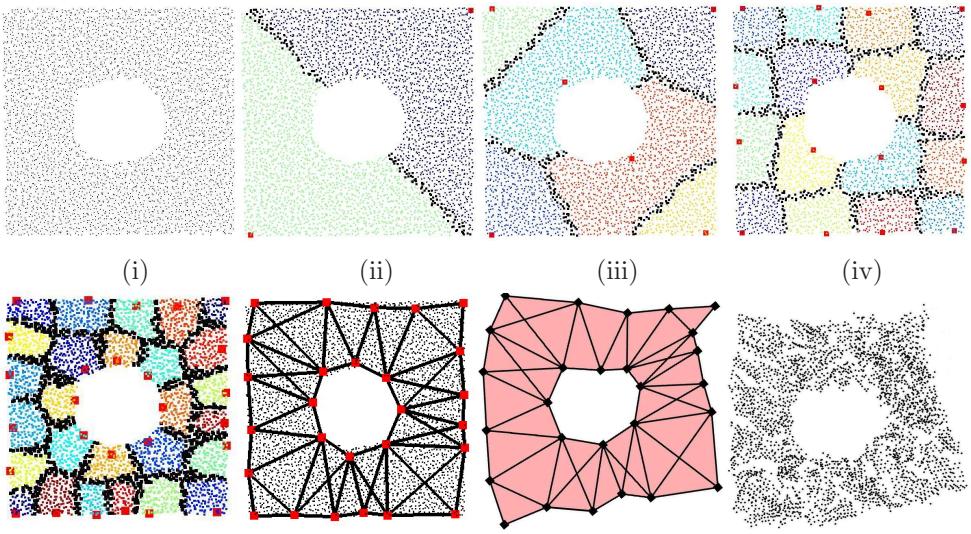


Figure 3.5. A step by step illustration of the incremental Delaunay refinement method. The total number of sensor nodes is 3887. The communication graph follows a unit disk graph model with average node degree 7.5. (i-iv) Start with two landmarks on the boundary and incrementally add more landmarks. (v) The final Voronoi diagram when the algorithm stops. (vi) The Delaunay edges extracted from the Voronoi cells of the landmarks. (vii) The embedding result of combinatorial Delaunay complex. (viii) The embedding result of all nodes.

discrete network.

Select initial landmarks

We start with two landmarks arbitrarily selected on the boundary. In order to guarantee these two starting landmarks are definitely on the boundary, we flood the network from a random node r and find the farthest node p from r, p must be on the network boundary. Then we flood from p and find the farthest node q from p. q will be on the boundary as well. We use p and q as our two initial landmarks. See Figure 3.5(i).

Compute Voronoi diagram

Once we have some landmarks, we calculate the landmark Voronoi diagram in a distributed way. Each landmark learns of its closest landmark(s) and all the nodes with the same closest landmark are naturally classified to be in the same Voronoi

cell. Recall that the landmarks are included incrementally. A new landmark initiates a flood message which is propagated only inside its Voronoi cell—when a node receiving this message sees that its hop count to some preexisting landmark is equal or smaller, the message is dropped. As more landmarks are included, the size of Voronoi cells decreases, and so does the communication cost of each restricted flooding operation.

Nodes with more than one closest landmarks lie on a Voronoi edge or vertex. Although the straightforward definition of *Voronoi vertex* is a node with equal distance to at least three landmarks, one robustness concern is that there may not be a node that qualifies for this definition by the discrete network hop count measure. In chapter 2 we proposed a merging heuristic to get Voronoi vertices. Here we refine this process with rigor and propose the following witness definition to guarantee the existence of Voronoi vertices.

Definition 3.3.1. *A node p is called a* 2-witness *for a pair of landmark $\{u, v\}$, if $d(p, u), d(p, v)$ are among the top m smallest hop count distances from landmarks to p and these hop count distances differ at most by β_2. β_2 is called the relaxation parameter for 2-witnesses.*

In other words, we denote by $\ell_i(p)$ the set of landmarks with the i-th smallest distance to p and $d_i(p)$ the i-th smallest distance from landmarks to p. Then a node p is the 2-witness for all pairs of landmarks in $L_2 = \cup_{i=1}^{m} \ell_i(p)$ such that $d_m(p) - d_1(p) \leq \beta_2$ and $d_{m+1}(p) - d_1(p) > \beta_2$. We call L_2 the *2-witness landmark set* for p. p witnesses every pair in L_2.

The boundary of a Voronoi cell of a landmark u is the collection of 2-witnesses with u in their landmark set. With the 2-witnesses we will detect 3-witnesses for triples of landmarks, a.k.a. the Voronoi vertices, by properly merging neighboring 2-witnesses with different landmark sets. In general we define a k-witness as follows, for $k \geq 3$.

Definition 3.3.2. *A node p is called a* k-witness *for a tuple of k landmarks, if p is a $k-1$-witness and the k landmarks are among the top m closest landmark set*

$L_k = \cup_{i=1}^{m} \ell_i(p)$ with $d_m(p) - d_1(p) \leq \beta_k$, $d_{m+1}(p) - d_1(p) > \beta_k$. β_k is called the relaxation parameter for k-witnesses. L_k is called the k-witness landmark set for p.

The parameters β_k are appropriately chosen as explained below. By the analog of the continuous case, the 2-witnesses correspond to the 1-dimensional Voronoi edges. The k-witnesses for $k \geq 3$ correspond to 0-dimensional Voronoi vertices. Thus we hope that the collection of 2-witnesses for each landmark (i.e., its Voronoi edges) is connected, and that the k-witnesses with $k \geq 3$ for different k-tuples form isolated connected components that separate 2-witness groups with different landmark pairs.

To show this we first give a number of observations.

Lemma 3.3.3. *If $\beta_k \geq 1$, there cannot be two neighboring k-witnesses p, q such that the landmark sets they witness do not share any common landmark.*

Now we examine what nodes among the 2-witnesses are selected to be 3-witnesses. The Voronoi boundary of each landmark is required to be connected, therefore, we group the 2-witnesses for each landmark u by the set of landmarks they witness. Adjacent 2-witnesses that witness different landmark sets will be selected as 3-witnesses with a properly selected relaxation parameter β_3. We choose $\beta_2 = 1$.

Lemma 3.3.4. *If there are two neighboring 2-witness nodes p, q that witness different landmark set, i.e., $L_2(p) \neq L_2(q)$, and $\beta_3 = 2\beta_2 + 2$, p, q are both 3-witnesses of the landmarks in $L_2(p) \cup L_2(q)$.*

Therefore, the Voronoi boundary of a landmark will have connected components of 2-witnesses (with the same witness landmark set) connected by 3-witnesses. Intuitively, this corresponds to Voronoi edges connected by Voronoi vertices. We will perform this witness selection operation further so that among the 3-witnesses, neighboring nodes with different witness landmark sets will be identified as 4-witnesses, if $\beta_4 = 2\beta_3 + 2$. Each connected component of k-witnesses

with the same landmark set will generate the corresponding Delaunay simplices. The witness identification procedure continues until the groups of k-witnesses with the same witness landmark set are isolated components.

The witness identification algorithm only uses local information. With the witnesses identified, we can output the combinatorial Delaunay complex as we will explain later.

Select more landmarks incrementally

With the Voronoi diagram from the initial 2 landmarks, we then select more landmarks incrementally. Corresponding to Section 3.2, for each landmark u and its Voronoi cell $V(u)$, we check:

- If the 2-witnesses (a.k.a. Voronoi edges) of u are not connected (this can be checked by having each connected component of the union of u's Voronoi edges send a message to u), we choose among all nodes that are endpoints of Voronoi edges lying on the network boundary[4] and select the one furthest from u as a new landmark.
- If the 2-witnesses of u are connected, we check each point p in Voronoi cell $V(u)$ and any Voronoi vertex v associated with u. We select point p as the new landmark if p is furthest away from any relaxed Voronoi ball $B_{(1+\delta)r}(v)$ among all points that are not yet δ-covered by Voronoi balls of u. Here r is the hop-count distance between u and v.

As the conditions are local, new landmarks can be selected in different Voronoi cells in parallel. The Voronoi diagram is then updated until no more landmarks are selected.

Figure 3.5 (v) is the final Voronoi diagram when the landmark selection stops. The Delaunay edges extracted from the final Voronoi diagram are shown in Figure 3.5 (vi). When the algorithm stops, both the global rigidity and good coverage are guaranteed.

[4]Notice that we can discover such nodes as each Voronoi edge is a connected set of 2-witnesses with the same landmark set, whose endpoints are either 3-witnesses or nodes on the network boundary.

Extract Delaunay complex

When all the landmarks are in place and the final Voronoi diagram is computed, using the witnesses we identified earlier, each connected component of k-witnesses with the same landmark set will generate a corresponding Delaunay simplex. In particular, for each k-witness p, $k \geq 3$, we output for each k-tuple in the witness landmark set $L_k(p)$ a $k-1$-dimensional simplex that implicitly includes all its faces. These simplices are collected to be embedded in the next step. The embedding of the Delaunay complex is the only centralized operation in the algorithm. Once the Delaunay complex is embedded, its realization is disseminated to the entire network to localize the rest of the nodes. Notice that since the Delaunay complex is a compact structure whose size depends on the network geometric complexity, and since only Voronoi nodes are involved in embedding it, the cost of collection and dissemination is substantially smaller than the cost of collecting the entire connectivity graph for any centralized localization algorithm.

Embed Delaunay complex

In brief, we choose one simplex, embed it as a starting point, and then embed each neighboring simplex side-by-side to the one already embedded. As mentioned earlier, two k-witnesses ($k \geq 3$) with different landmark sets are connected through m-witnesses with $2 \leq m < k$. Thus each simplex we extract must share an edge with a neighboring simplex and the 2 simplices cannot overlap, so the embedding is unambiguous. For example, suppose a simplex S is already embedded, and we want to embed a neighboring simplex (triangle) S' that shares a common edge with S. We use bilateration to find the 2 possible positions for the third landmark of S' that has not yet been embedded and choose the one that does not cause S and S' to overlap.

Since we ran our new algorithm on more complicated topologies than what our original algorithm was capable of, we encountered many high dimensional simplices (see for example the sun shape in Figure 3.9). In this case we embed each high-dimensional simplex using multi-lateration to the other landmarks of

the simplex that are already embedded, in order to take advantage of all known distance measurements. Since we only have estimated distances, we solve the optimization problem of minimizing the mean square error among the distances as described in [70]. And as another optimization, we run a mass-spring relaxation on the simplex in order to smooth out the distance errors.

We remark that our embedding algorithm only makes sure that adjacent Delaunay triangles are embedded 'side-by-side', thereby allowing us to get a very good embedding of the network. However, it does not guarantee a planar embedding—one part of the network can still curve around and intersect with another part of the network. It is an NP-hard problem to find a planar embedding given a planar graph with specified edge lengths. A particularly challenging scenario is when embedding a network with a hole and we want to connect the loop of simplices cycling back to itself. One approach we use to prevent one simplex from landing atop another is by setting some boundary lines defined by the first embedded simplex that no other simplex may cross. If a landmark goes over this line, it is embedded to the line. If a landmark should receive more than one coordinate assignment (arising from two simplices coming around the hole), we simply embed it at the centroid of its different assignments. These steps work well in many cases, and are what we used to get the result in Figure 3.8. Unfortunately they do not ensure planarity, and can introduce flipped simplices, as can be seen in the flower and music images of Figure 3.9, and elsewhere. We emphasize that our algorithm guarantees correct orientation of the simplices, but once other heuristics are applied, the guarantees no longer hold. This problem is hard to get around by the NP hardness result. It still remains for future work to develop efficient approximation algorithms with theoretical guarantees for planar graph embedding.

Network localization

When the Delaunay complex is embedded and disseminated to all the nodes, each non-landmark node uses its hop count estimation to 3 (or more) landmarks to trilaterate its own location (as in the atomic trilateration in [69]).

3.3.2 Discussion

We remark that when there are anchor nodes with known locations, we can select them as initial landmarks. More landmarks can be included to ensure sufficient density. Later when we embed the landmarks, the anchor nodes are fixed at their known locations. When there are enough anchor nodes placed with proper density, our algorithm deteriorates to simple trilateration algorithm (in [69]). Our refinement algorithm can also be used to identify candidates for additional anchor nodes.

Li and Liu [56] have observed that when the network has holes or complex shape, the shortest path may bend on hole boundaries. Thus the network hop count distance is a poor approximation to the straight line Euclidean distance. They have also proposed ways to correct the biases of the distances and shown improvement to the localization accuracy of simple trilateration algorithm. In our algorithm, by partitioning the network into Voronoi cells, in some sense we avoid the shortest paths that bend on holes. This also partially explains why our algorithm performs better than multi-dimensional scaling[5] [74] (as shown above), when all shortest path lengths are dumped to a global optimization routine to generate the embedding as the best fit. The biased distance measurements necessarily distort the embedding.

[5]MDS takes an inter-distance matrix on n nodes and extracts the node location in $\mathbb{R}^n$. Consider the matrix B, where each entry $b_{ij} = \sum_{k=1}^{p} x_{i_k} x_{j_k} = x_i^\top x_j$, where p is the dimension, each $x_i = (x_{i_1}, \ldots, x_{i_p})^\top$ is the coordinates for point i. It is possible to derive B from the known squared distances d_{ij} alone, where d_{ij} is the shortest path between nodes i and j. Each entry $b_{ij} = -\frac{1}{2} d_{ij}^2 - \frac{1}{n} \sum_{j=1}^{n} a_{ij} - \frac{1}{n} \sum_{i=1}^{n} a_{ij} + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$, where $a_{ij} = -\frac{1}{2} d_{ij}^2$. From B we derive the unknown coordinates. Since B is symmetric, positive semi-definite we can write $B = VAV^\top$, where A is the diagonal matrix of the eigenvalues of B, and V is the matrix of corresponding eigenvectors. The coordinate matrix $X = VA^{\frac{1}{2}}$, that is $X = (x_1, \ldots, x_n)^\top$, an $(n \times p)$ matrix of coordinates. If the inter-node Euclidean distances are known exactly, then MDS would precisely determine the coordinates of the points (up to global transformations). Taking the first 2 eigenvalues and eigenvectors yields the coordinates in 2D.

3.4 Simulation

We conducted extensive simulations under various scenarios to evaluate how well our algorithm extracts the network topology and how performance is affected by different factors such as node density, different communication model (quasi-UDG, probabilistic model, etc.) and node random failures. Typically our examples have an average node degree of around 10, and we also tested on examples with average degree as low as 6. We also demonstrate results for a special case where nodes are aligned on a perfect grid having an average degree of 4. We evaluate the communication cost of our algorithm at the end.

Influence of node density. Theoretically, our algorithm performs better under higher node density since the hop-count distance between nodes is a better approximation of the geodesic distance between them.

Figure 3.6 shows the results of networks with different densities but with the same communication range. Notice that when the average degree is below 7, not all selected landmarks are on the boundary, as Voronoi edges may be broken at small holes in the network. The performance deteriorates when the average degree drops below 6, when error accumulation by using hop-count distance becomes too large for an accurate embedding.

Influence of network communication models. Unit disk graph model is normally too ideal in practical scenario. In real scenario, there could be noise, nodes power off, network disconnection for any unknown reason, we also evaluate our algorithm on connectivity models other than unit disk graph model, in particular, *quasi-unit disk graph model (quasi-UDG)* and *probabilistic connectivity model*. In *quasi-UDG*, two nodes are connected by an edge if the Euclidean distance between them is no greater than a parameter α, $\alpha \leq 1$, and are not connected by an edge if the Euclidean distance is larger than 1. If the Euclidean distance d is in the range $(\alpha, 1]$, we include this edge with probability $(1 - d)/(1 - \alpha)$. In the *probabilistic connectivity model*, we start with the unit disk graph model and remove each edge with probability $1 - \beta$. We remark that the probabilistic model also take into account the random node/link failures.

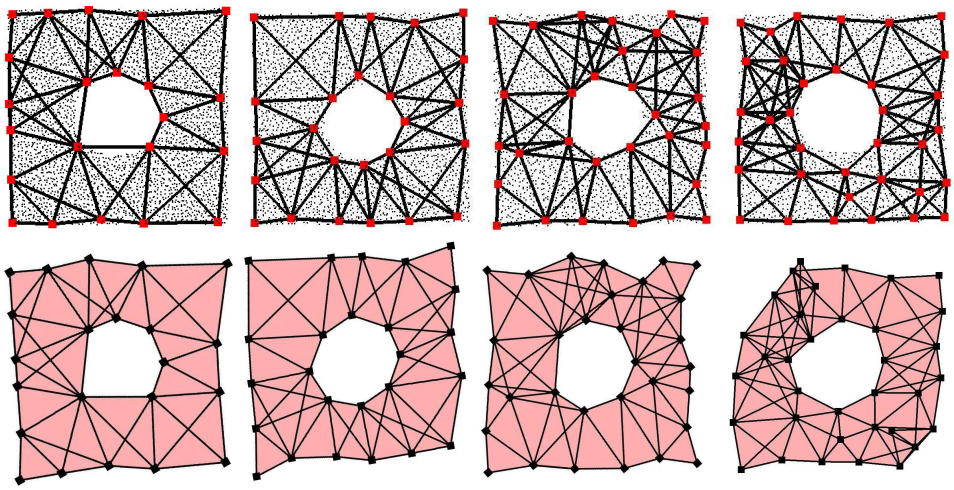


Figure 3.6. The embedding results for networks of different node densities. The communication ranges are the same for all 4 networks. The first row shows the ground truth; the second row shows our embedding of the landmark nodes. From left to right the models depicted have (i) 3887 nodes, avg. degree 10.28. (ii) 3044 nodes, avg. degree 7.6. (iii) 2680 nodes, avg. degree 6.3. (iv) 2320 nodes, avg. degree 5.7.

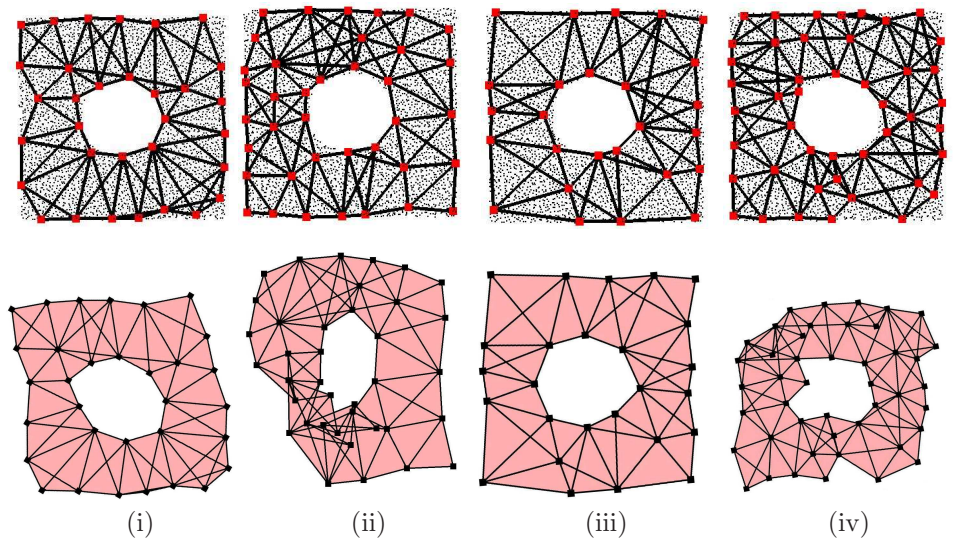
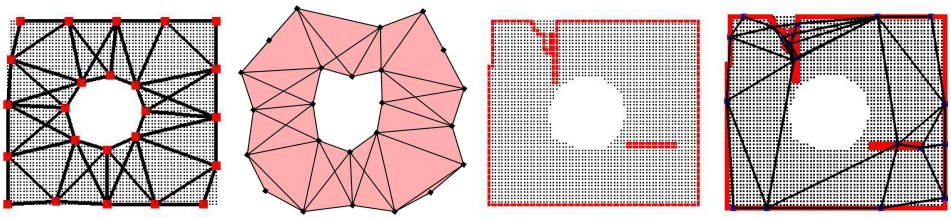


Figure 3.7. Effect of network communication models on the embedding. The first row shows the ground truth; the second row is our embeddings of the landmark nodes. All the networks have 3887 nodes and the same communication range. From left to right the models depicted are (i) quasi-UDG model, avg. degree 6.4, $\alpha = 0.6$ (ii) quasi-UDG model, avg. degree 5.6, $\alpha = 0.5$ (iii) delete each edge with probability $1 - \beta$. $\beta = 0.6$, avg. degree 6.2 (iv) Same model as (iii), $\beta = 0.5$, avg. degree 5.0.

We show some representative cases in Figure 3.7. (i) and (ii) use the quasi-UDG model. (iii) and (iv) use the probabilistic model. We have good embedding results even when α or β is at least 0.6 with an average degree of around 6. When α or β goes below 0.5, the algorithm performance starts to deteriorate.

Comparison with Chapter 2 Since the new algorithm does not depend on boundary detection, it not only avoids the computationally expensive operation of detecting the network boundary, but can work under conditions where the boundary detection would give poor results, causing an unsatisfactory outcome. Figure 3.8 is a network with nodes laid out on a perfect grid with an average degree of only around 4. This is an example that will not work using our previous algorithm as boundary detection will fail. As far as we know, no known boundary detection algorithm can work on networks with such low average degree. Figure 3.8(i)(ii) shows the ground truth and embedding result of the new algorithm. Note the low degree does cause some locally inaccurately embedded pieces. At two top corners, the triangles are degenerate as the hypotenuse has exactly the same length as the other 2 sides measured by hop-count in the grid network. Nevertheless we still capture the topology and the global geometry rather faithfully. Figure 3.8(iii) is the boundary detection result using the method in [78], which generates a Delaunay Complex that does not capture the network geometry (Figure 3.8(iv)).



(i) (ii) (iii) (iv)

Figure 3.8. A perfect grid network. 3388 nodes, avg. degree 3.87. (i) the Delaunay complex extracted from the Voronoi cells of the landmarks using the new algorithm. (ii) the embedding result. (iii) the boundary detection result. (iv) the Delaunay complex result using the previous algorithm.

Different network topology. We show more results using our algorithm for a number of networks with convoluted shapes in Figure 3.9. The reason we show

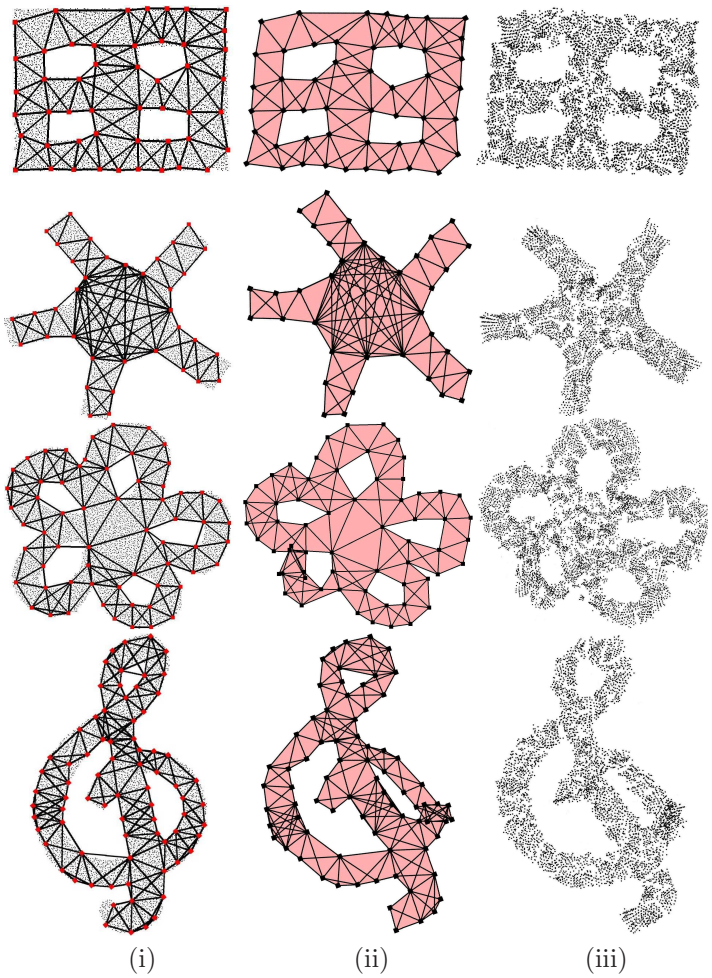


Figure 3.9. Running our algorithm on different topologies. The first row shows a network of windows shape, with 6495 nodes and avg. degree 9.97. The second row shows a network of sun shape, with 5217 nodes and avg. degree 10.3. The third row shows a network of flower shape with 8350 nodes and avg. degree 9.14. The fourth row shows a network of music shape, with 6176 nodes and avg. degree 10.2. Columns: (1) the ground truth. (2) the embedded landmark nodes. (3) all the nodes embedded using multi-lateration to the closest landmark nodes.

these interesting results is to prove the ability of our algorithms on complicated shapes. Some shapes here can also find possible real scenarios corresponding to practical sensor deployment. For example, remote unknown areas or underwater environments could be any shapes, a window-like shape, sun-like shape and etc..

Communication cost of the algorithm. In the execution of the incremental landmark selection algorithm, the new landmarks in different Voronoi cells are selected in parallel and each new landmark only floods locally in its Voronoi cell. In one iteration, many Voronoi cells can be refined and new landmarks are selected. In Figure 3.10, we run our algorithm on a group of networks with the same shape (similar to Figure 3.5), the same communication range and different node densities. We calculate the average number of nodes in each Voronoi cell in each iteration. It is shown that the refinement is very effective. The average size of Voronoi cells drops dramatically and the algorithm typically stops after a small constant number of iterations.

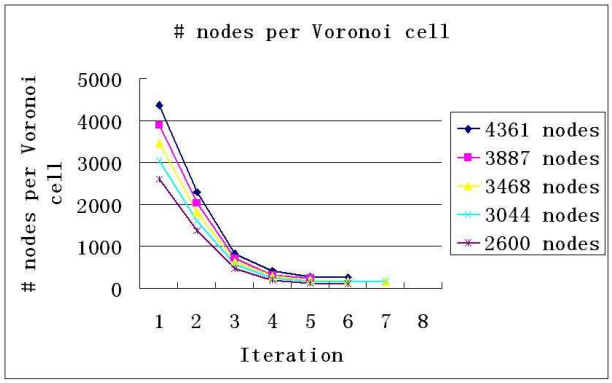


Figure 3.10. The average size of the Voronoi cell in each iteration until the algorithm stops. The size of the network varies from 2600 to 4361. With the same communication range, the average degree of the network varies from 6.29 to 10.68.

We also show the total communication cost for each iteration, in particular, the number of messages involved in the establishment of the Voronoi cells of all newly added landmarks in each iteration. When a new landmark is selected, it only floods locally in its Voronoi cell, claiming these nodes from old landmarks and properly setting up the Voronoi boundary. Recall that the landmark selection rules are local and can be executed by each node on the Voronoi boundary locally,

whose communication cost is negligible. The combinatorial Delaunay complex is only extracted at the final step by a network wide flooding. Thus we evaluate the communication cost for the landmark refinement step. In Figure 3.11, we run our algorithm on the same setting as Figure 3.10. We use the broadcast model, that is, one transmission is received by all neighbors. For each iteration, we take the total communication cost as the number of transmissions during that iteration. In Figure 3.12, we show the average number of transmissions per node, for different network size. As the energy consumption of each node is mainly on the communication cost. This figure shows how the algorithm scales in terms of energy usage.

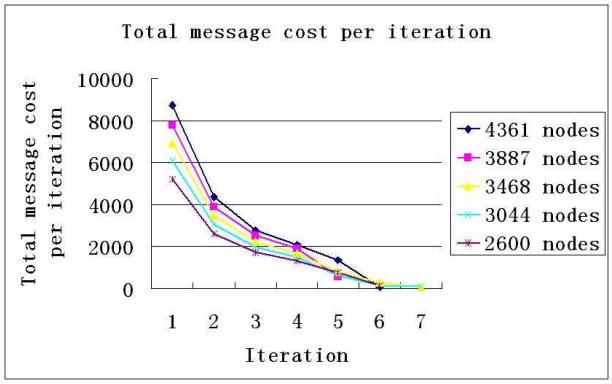


Figure 3.11. The total message cost in each iteration until the algorithm stops. The size of the network varies from 2600 to 4361. With the same communication range, the average degree of the network varies from 6.29 to 10.68.

3.5 Conclusion

This chapter is a follow-ups up to chapter 2 solving the localization problem using connectivity information only. We develop a new landmark selection algorithm using incremental Delaunay refinement method in a distributed manner. The new algorithm keeps the good properties (global rigidity and coverage) needed for localization, and yet is not dependent on network boundary detection. This allows for a more robust algorithm, less sensitive to the noisy results of boundary detection and avoids its high computation cost. Thus our new algorithm is more

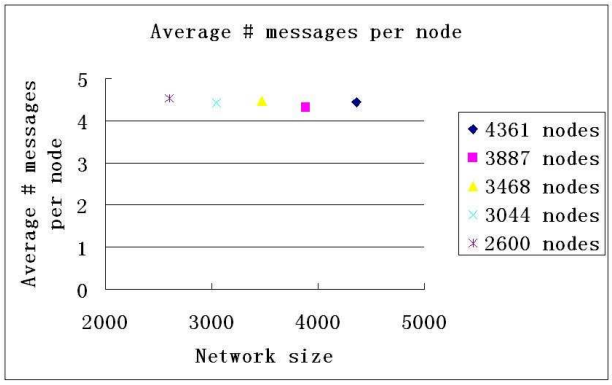


Figure 3.12. The average number of messages per node for different network size. The size of the network varies from 2600 to 4361. With the same communication range, the average degree of the network varies from 6.29 to 10.68.

applicable in practice, performing well in networks with low average degree and complex shapes.

So far the framework only applies for the embedding in 2D. In our on-going work, we are applying the algorithm for embedding of sensors deployed on a non-flat terrain or in three dimensional space (e.g., for underwater deployment). Boundary detection algorithms in dimensions higher than two are unavailable thus the ideas presented in chapter 2 would have little hope on its own to be extended to a more general setting. The incremental landmark refinement is a necessary and critical improvement to make the technique more generic.

3.6 Appendix

We show that the inner local feature size function is 1-Lipschitz.

Lemma 3.6.1 (Lipschitz continuity). *The inner local feature size of any shape* $\mathcal{R} \subseteq \mathbb{R}^2$ *is 1-Lipschitz:* $ILFS(x) \leq ILFS(y) + d(x, y)$ *for any* $x, y \in \mathbb{R}^2$.

Proof: The proof follows from triangle inequality. Suppose that point p is the closest point of y on the inner medial axis of $\mathcal{R}$. Then $d(p, y) = ILFS(x)$. Thus, $ILFS(x) \leq d(p, x) \leq d(p, y) + d(x, y) = ILFS(y) + d(x, y)$. □

Theorem 3.2.1. *For a connected region $\mathcal{R} \subseteq \mathbb{R}^2$, we select landmarks L as a γ-sample on the region boundary $\partial\mathcal{R}$ with $\gamma < 1$. Then the Delaunay complex δ-covers $\mathcal{R}$, with $\delta = 2\gamma/(1-\gamma)$.*

Proof: We first prove the claim for points on $\partial\mathcal{R}$. Consider a point x on $\partial\mathcal{R}$ in

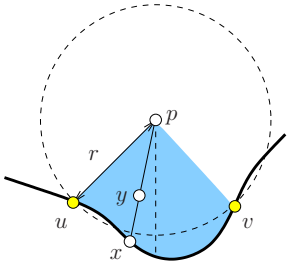


Figure 3.13. Any x is within distance $\delta \cdot r$ from a Voronoi ball. A pie between a mixed arc $\hat{uv}$ is shown in shade.

between two landmarks u, v, as shown in Figure 3.13. Lemma 2.8 in chapter 2 says that there is a Voronoi vertex p with u, v as two closest landmarks and the Voronoi edge with respect to u, v is a mixed edge. Without loss of generality we assume that x's closest landmark is u. By the γ-sample property, $d(u,x) \leq \gamma \cdot ILFS(x)$.

Now we assume by contradiction that $d(p,x) > (1+\delta)r$. Thus $\gamma \cdot ILFS(x) \geq d(u,x) \geq d(p,x) - d(p,u) > (1+\delta)r - r = \delta r$ by the triangle inequality. Thus $ILFS(x) > \delta r/\gamma$.

We also know that the inner local feature size is a 1-Lipschitz function (by triangle inequality, proof in the Appendix). That is, $ILFS(x) \leq ILFS(u) + d(u,x)$. As we know that the Voronoi ball $B_r(p)$ touches three landmarks and contains at least one point on the medial axis in $\mathcal{R}$, $ILFS(u) \leq 2r$. Thus we have, $ILFS(x) \leq 2r + \gamma \cdot ILFS(x)$. Combining the inequalities, we have $\delta r/\gamma < ILFS(x) \leq 2r/(1-\gamma)$. That gives us $\delta < 2\gamma/(1-\gamma)$, a contradiction.

If the claim is true for all points on $\partial\mathcal{R}$, it is true for all points in $\mathcal{R}$. Suppose otherwise, then there is a point y in the interior of $\mathcal{R}$ that is not δ-covered. y can only possibly stay inside a pie, as shown in Figure 3.13. Then there must be another point $x \in \partial\mathcal{R}$ such that y stays on the geodesic shortest path from p to x. Thus y is covered by $B_r(p)$, the same Voronoi ball that covers x. □

We first restate a useful Lemma from chapter 2.

Lemma 3.6.2. *Given a disk B containing at least two points on $\partial\mathcal{R}$, for each connected component of $B \cap \mathcal{R}$, either it contains a point on the inner medial axis, or its intersection with $\partial\mathcal{R}$ is connected.*

Lemma 4.3.2. *If a Voronoi cell $V(u)$ violates the local Voronoi edge connectivity condition, the new landmark q selected is not covered by any landmark within $\gamma \cdot ILFS(q)$, for any $\gamma < 1/3$.*

Proof: Since the boundary of the Voronoi cell $V(u)$ is composed of segments on $\partial\mathcal{R}$ and the Voronoi edges of u, $V(u)$ must have two or more connected components on the domain boundary $\partial\mathcal{R}$ as well. Take a Voronoi edge endpoint p that stays on a different boundary segment with u. $d(u,p) \leq d(u,q)$ since q is the furthest such endpoint. See Figure 3.14. We take a ball $B_r(p)$ with $r = d(u,p) - \varepsilon$, for $\varepsilon \to 0$. We argue that $B_r(p)$ intersects the boundary $\partial\mathcal{R}$ in two or more connected pieces. If otherwise, the ball $B_r(p)$ intersects $\partial\mathcal{R}$ in one component C that goes from p to u (excluding u). Since p is on the Voronoi edge, p is equal distance from u and another landmark w on C. That is, $d(w,p) = d(u,p) > r$. Thus the segment C must leave the ball $B_r(p)$ at some point and come back in. This shows that $C \cup B_r(p)$ must have 2 connected components. By Lemma 4.8.4 there

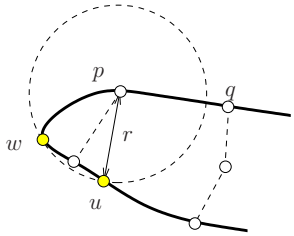


Figure 3.14. The new landmark q is not γ-covered for $\gamma < 1/3$.

is a point on the inner medial axis inside $B_r(p)$. That means $ILFS(p) < d(u,p)$. Since $ILFS$ is 1-Lipschitz, $ILFS(u) \leq ILFS(p) + d(u,p) < 2d(u,p) \leq 2d(u,q)$. Apply this again we get $ILFS(q) \leq ILFS(u) + d(u,q) < 3d(u,q)$. Thus the claim is proved. □

Lemma 4.3.3. *If a Voronoi cell $V(u)$ for a landmark u violates the local Voronoi ball coverage condition, the new landmark q selected is not covered by any landmark within $\gamma \cdot ILFS(q)$, $\gamma = \delta/(2+\delta)$.*

Proof: If q is selected as the new landmark, $d(q,p) > (1+\delta)r$ for any Voronoi vertex p of the landmark u, where $r = d(p,u)$ is the radius of the Voronoi ball at p. Now we have by triangle inequality $d(q,u) \geq d(p,q) - d(p,u) > \delta r$. That is, $r < d(q,u)/\delta$. Similar to the argument in Theorem 3.2.1, $ILFS(q) \leq d(q,u) + ILFS(u) \leq d(q,u) + 2r < (1+2/\delta)d(q,u)$. Thus, $d(q,u) > \gamma \cdot ILFS(q)$ with $\gamma = \delta/(2+\delta)$. □

Theorem 3.2.5. *Suppose L is the generated landmark set by the incremental algorithm. If any landmark from L is removed, then it is not a γ'-sample of $\partial\mathcal{R}$, with $\gamma' = \gamma/(1+\gamma)$, $\gamma < \max(1/3, \delta/(2+\delta))$.*

Proof: For the last landmark q inserted, by Lemma 4.3.2 and Lemma 4.3.3, it is not within distance $\gamma \cdot ILFS(q)$ of any existing landmark. Since $\gamma > \gamma'$ the claim is true for q.

For any landmark q' added before q, we know $d(q,q') > \gamma \cdot ILFS(q)$, since q is added with q' already present in the current landmark set. Since $ILFS$ is 1-Lipschitz, we have $ILFS(q') \leq ILFS(q) + d(q,q') < (1+1/\gamma) \cdot d(q,q')$. Therefore, $d(q,q') > \gamma' \cdot ILFS(q)$. Notice that this argument is true for any pair of landmarks q, q' with q added after q'. Thus for q', the distance to any landmark in L is at least greater than $\gamma' \cdot ILFS(q')$. The claim is true. □

Lemma 3.3.3. *If $\beta_k \geq 1$, there cannot be two neighboring k-witnesses p, q such that the landmark sets they witness do not share any common landmark.*

Proof: We will just prove this for β_2 as the proof is the same for other k. Assume by contradiction that the set of landmarks p witnesses $L_2(p)$ and the set $L_2(q)$ that q witnesses do not share any common landmark. We take $u_1 \in \ell_1(p)$ and $u_2 \in \ell_1(q)$. We have $d(q,u_2)+1 \geq d(p,u_2)$, since p, q are neighboring nodes. Also, as u_2 is not among $L_2(p)$, we have $d(p,u_2) > d(p,u_1) + \beta_2$. Similarly, as u_1 is not among $L_2(q)$, we have $d(q,u_1) > d(q,u_2) + \beta_2$. Now we have

$$\begin{aligned} d(q,u_2) + 1 &\geq d(p,u_2) &&> d(p,u_1) + \beta_2 \\ &\geq d(q,u_1) - 1 + \beta_2 &&> d(q,u_2) + 2\beta_2 - 1. \end{aligned}$$

Thus $\beta_2 < 1$. This shows a contradiction. □

Lemma 3.3.4. *If there are two neighboring 2-witness nodes p, q that witness different landmark set, i.e., $L_2(p) \neq L_2(q)$, and $\beta_3 = 2\beta_2 + 2$, p, q are both 3-witnesses of the landmarks in $L_2(p) \cup L_2(q)$.*

Proof: By Lemma 3.3.3, there is a landmark u_1 such that $u_1 \in L_2(p) \cap L_2(q)$. Choose $u_2 \in L_2(p) \setminus L_2(q)$ and $u_3 \in L_2(q) \setminus L_2(p)$. Now we have,

$$\begin{aligned} d(p, u_3) &\leq d(q, u_3) + 1 &&\leq d(q, u_1) + \beta_2 + 1 \\ &\leq d(p, u_1) + \beta_2 + 2 &&\leq d_1(p) + 2\beta_2 + 2 \quad = d_1(p) + \beta_3. \end{aligned}$$

Thus $u_3 \in L_3(p)$. With a symmetric argument $u_2 \in L_3(q)$. Therefore both p and q are 3-witnesses of the landmarks in $L_2(p) \cup L_2(q)$. □

Chapter 4

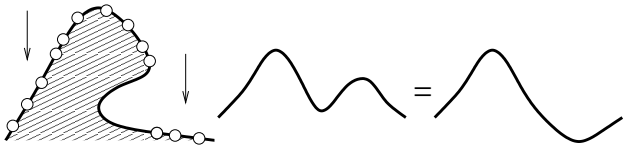
Localization On A Manifold

4.1 Introduction

Up till now our discussion on localization focused on sensors deployed on a flat surface, and indeed this the setting that has been explored in the related research [11, 51, 60–62, 64, 65, 70, 71, 74]. However it is most likely that this will not be the case in real world scenarios. Sensors may be dispersed over a terrain that has hills and valleys. For example, a massive number of inexpensive sensors are dropped from an airplane over an uneven terrain, ocean floor, buildings or other structures. The previous research in localization that does address the 3D scenario is very limited. Zhang *et al.* [82] extend their 2D Landscape algorithm to 3D; it uses a location aware mobile device, such as an airplane, to repeatedly broadcast beacons to sensors on the ground. Similarly Luo *et al.* [58] works with underwater acoustic sensor networks and uses an autonomous underwater vehicle to act as a beacon; the sensors determine there height information (i.e. depth) through the use of a pressure sensor. AbdelSalam and Olariu [2] make use of anchor nodes that have the ability to vary their height and by transmitting at different heights which gives nodes the ability to first determine their own elevation. In this chapter we extend our approach to the 3D setting and still do not resort to any anchors or beacons.

We would model our sensors as samples of a two dimensional manifold surface

$\mathcal{S}$ in 3D. In addition, we require that the surface $\mathcal{S}$ to be *monotonic*, i.e, any vertical line intersects the surface in at most one point. The monotonicity of the sensor field captures a natural constraint when sensors are dropped from above—nodes will not lie directly above or below others, as, for instance, lie on the roof of a building and also inside the building. See a picture on the vertical cross-section of S in Figure 4.1 (i). The collection of sensors are not necessarily uniformly placed on S. The shape of the network can be complex and may have holes.



(i) (ii)

Figure 4.1. (i) Sensors dropped from an airplane stay on a monotonic surface; (ii) Without height information, we can not differentiate a peak from a valley.

In our setting the sensors are inexpensive and tiny so as to be spread by an airplane. No sensors have GPS, which is bulky and costly. We also do not use range information. Obtaining accurate distance estimation between two nodes would require special hardware such as ultrasound ranging hardware, which adds extra cost. The sensors communicate with each other if they are within the transmission range of each other, with the distance measured on the terrain (i.e., the geodesic distance on S). In our model each sensor has a height sensor (such as a barometer) to measure its height. Our localization algorithm uses the network connectivity to find the x, y coordinates of the sensors. We remark that the height of the sensors is necessary to guarantee a unique solution of the localization problem – without the height information we can not differentiate a peak from a symmetric valley as sensors deployed on them may have the same connectivity. For an example, see Figure 4.1 (ii).

Our problem is to develop a range-free, anchor-free localization algorithm for sensors on a terrain with only network connectivity information. This is a very challenging problem as we will need to discover sensor network geometry from graph connectivity, and effectively stretch out the network correctly. Above we have developed a framework for connectivity-based localization in a 2D plane. In

this chapter we extend it to the setting of a terrain. Next we will quickly review the framework from the previous chapters. Then we explain the issues we have for the case of a terrain and how we tackle these problems.

4.2 Overview of embedding combinatorial Delaunay complex

For a sensor network in the plane, the major challenge for anchor-free localization is to avoid incorrect flips – a part of the network folds on top of another. We'll briefly review our algorithm that avoids such incorrect flips.

We select some sensor nodes as *landmarks* L and compute the Voronoi diagram with all the nodes closest to the same landmark grouped into the same Voronoi cell. The cell for landmark u is denoted as

$$V(u) = \{p \in \mathcal{R} \,|\, d(p,u) \leq d(p,v), \forall v \in L, v \neq u\}.$$

A point is called a *Voronoi vertex* if it has equal distance to at least three landmarks. A collection of points with equal distance to two landmarks u, v is called the *Voronoi edge* for u, v. We extract the *combinatorial Delaunay complex*[1] $DC(L)$ as the dual complex of the landmark Voronoi diagram – there is a Delaunay edge (or in general a k-simplex) between two (or k) landmarks if their Voronoi cells share some common nodes.

$$DC(L) = \{\alpha \subseteq L \,|\, \cap_{u \in \alpha} V(u) \neq \emptyset\}.$$

Now, two *triangles* sharing a common edge may have two valid embedding as one can flip one triangle relative to the other. But two *Delaunay triangles* sharing a common edge have only one valid embedding – as these triangles are certified

[1]The Delaunay complex is defined in the notion of abstract simplicial complex [24]. A set α is an (abstract) *simplex* with dimension $\dim \alpha = \operatorname{card} \alpha - 1$, i.e., the number of elements in α minus 1. A finite system A of finite sets is an *abstract simplicial complex* if $\alpha \in A$ and $\beta \subseteq \alpha$ implies $\beta \in A$.

by the underlying Voronoi cells and only the side-by-side embedding maintains the property of a geometric simplicial complex. See Figure 4.2 for an illustration. This critical observation implies that we can glue adjacent Delaunay triangles incrementally to recover the network geometry.

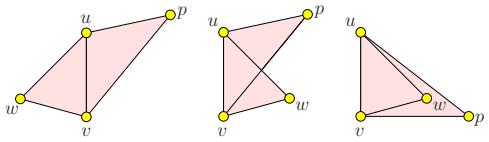


Figure 4.2. Two Delaunay triangles $\triangle uvw$ and $\triangle uvp$ sharing an edge. The first figure is the only valid embedding, because in a simplicial complex two simplices can only intersect at a common face. The graph is not globally rigid.

In chapter 2 and chapter 3, we complete the localization algorithm by properly selecting landmarks L to guarantee two important properties of the combinatorial Delaunay complex $DC(L)$: (i) the rigidity of $DC(L)$: one can not deform the shape of the $DC(L)$ without changing the edge length; (ii) the coverage of the $DC(L)$: every sensor node is close to $DC(L)$ under some measure, i.e., $DC(L)$ well represents the network shape. Altogether the two properties imply that patching the Delaunay triangles will recover the network shape. See Figure 4.3 for an example. The length of a Delaunay edge is taken as the minimum hop count between the two landmarks. With the Delaunay complex embedded in the plane the rest of sensor nodes find their location through trilateration to three closest landmarks.

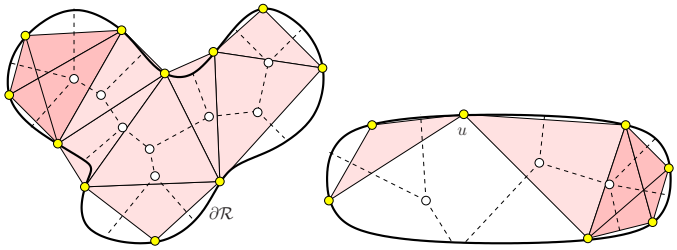


Figure 4.3. Left: The Voronoi graph (shown in dashed lines) and the Delaunay complex for a set of landmarks on the boundary $\partial\mathcal{R}$. The Delaunay simplices (vertices, edges, triangles, tetrahedrons) are shaded. Right: the Delaunay complex is not rigid.

In chapter 2, we first detect the network hole boundary and then select landmarks as a γ-*sample* with $\gamma < 1$. In this case, every boundary node has a landmark within its *inner local feature size* $ILFS$, defined as the distance to the closest node on the *medial axis* (which is the collection of nodes with two or more closest nodes on the boundary). Since the network boundary detection algorithms (as in [29, 30, 32, 33, 52, 78]) are not robust in a sparse network, our approach in chapter 3 selects landmarks such that both rigidity and coverage are guaranteed.

New challenges in 3D. In this chapter we extend the framework from chapter 2 and chapter 3 to sensors placed on a monotonic surface S in 3D. The connectivity graph is based on the distance of sensors on the terrain. If a Delaunay triangle $\triangle abc$ is embedded in 3D, an adjacent triangle $\triangle bcd$ can possibly have two positions with a given height of node d. But only one of them is valid given the monotonicity requirement of S. See Figure 4.4. Thus the main idea carries over. What is left is to develop landmark selection scheme.

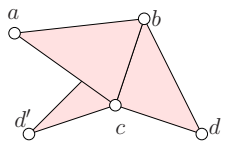


Figure 4.4. A Delaunay triangle has only one valid embedding if an adjacent Delaunay triangle has been embedded, to guarantee the monotonicity of $\mathcal{S}$. In the figure the position d' is not valid.

Contrary to the intuition, going to 3D brings a lot of new issues. It is well known in computational geometry that curve reconstruction (in 2D) is much easier than surface reconstruction (in 3D). The rigidity and coverage properties for a combinatorial Delaunay complex are not sufficient for good localization results. Take a look at Figure 4.5. Imagine a sharp peak with a landmark u_3 at the top, and two landmarks u_1, u_2 at the bottom of the peak, on opposite side. There can be two Voronoi vertices for landmark u_1, u_2, u_3. Thus the peak is actually represented by a single triangle $\triangle u_1u_2u_3$ sticking out of the plane. A similar setting can happen in 2D as well — imagine the Voronoi cell of u_3 is a hole. But in 2D the triangle $\triangle u_1u_2u_3$ sticking out of the plane must be embedded in the plane and we can decide on its correct embedding by the adjacency of the

Voronoi cells. See chapter 2 for more discussions on this. In 3D this becomes a real problem as important terrain features are missed due to insufficient sensor density. You may also notice that there is a hole completely inside the Voronoi cell of u_4. Thus the hole is missed in the representation by the combinatorial Delaunay complex. This is also bothering as trilateration of networks with holes can given distorted localization results [56].

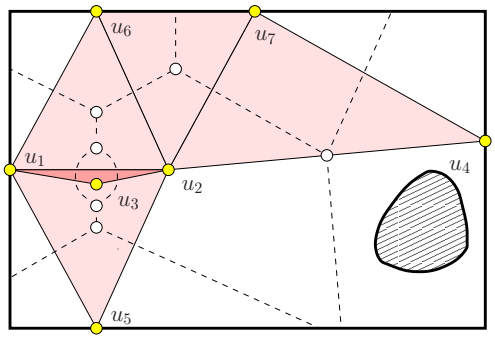


Figure 4.5. An example when the combinatorial Delaunay complex $DC(L)$ is not homotopy equivalent to the surface S. The Voronoi cell of u_4 has a hole inside. The Voronoi edge for landmark u_1, u_2 has two components. There are two Voronoi vertices for u_1, u_2, u_3.

The main issue is that we need to select landmarks to make sure that the combinatorial Delaunay complex has the same topology as the sensor field S, capturing all the spatial features (peaks and network holes). For this, we develop new landmark selection rules. The landmark selection is achieved again by an incremental refinement scheme. Tests are ran locally in each Voronoi cell and its adjacent cells to check whether new landmarks need to be included. Multiple Voronoi cells can be refined at the same time, which leads to inherent parallelism. The same as above we do not assume the knowledge of network holes. Thus the algorithm can be applied to sparse networks.

In the next section we focus on the new landmark selection algorithm, explained in a continuous setting for presentation simplicity and provable properties. The implementation and simulation of the new algorithm are presented afterwards.

4.3 Localization Algorithm and Analysis

4.3.1 Delaunay refinement algorithm

We assume the sensors $\mathcal{P}$ are samples of a monotonic surface $\mathcal{S}$ in 3D. S may have boundaries denoted by $\partial\mathcal{S}$. As discussed before, we will select landmarks such that the combinatorial Delaunay complex is *homotopy equivalent*[2] to the sensor field $\mathcal{S}$. Homotopy equivalence intuitively says that the way the simplices are connected in the Delaunay complex are the same as the connectivity in $\mathcal{S}$. Fortunately the homotopy equivalence of the combinatorial Delaunay complex admits a local condition: as long as the Voronoi edge/vertex set of any k landmarks is either empty or contractible[3], the homotopy equivalence is established. Thus we can check locally whether the conditions are satisfied. The following theorem follows immediately from the Cěch Theorem [13].

Theorem 4.3.1. *If the intersection of the Voronoi cells of any k landmarks is either empty or contractible, the Delaunay complex has the same homotopy type as the region $\mathcal{R}$.*

In the case of landmark sampling a surface S, the homotopy equivalence condition means each Voronoi cell, edge, or vertex, if not empty, should be contractible. Examples when this condition is violated are shown in Figure 4.5.

In the continuous setting, a *Voronoi vertex* is a node with equal distance to at least three landmarks. When the distance is measured by network hop counts, there may not be a node that has exactly the same hop counts to three landmarks. In chapter 3 we proposed a relaxed definition. A node p is on the Voronoi edge of two landmarks u, v if the distance to u, v differ by a relaxation parameter

[2]Two maps f and g from X to Y are homotopic if there exists a continuous map $H : X \times [0,1] \mapsto Y$ with $H(x,0) = f(x)$ and $H(x,1) = g(x)$. Two spaces X and Y have the same homotopy type if there are continuous maps $f : X \mapsto Y$ and $g : Y \mapsto X$ such that $g \circ f$ is homotopic to the identity map of X and $f \circ g$ is homotopic to the identity map of Y. In other words, the maps f and g define a one-to-one correspondence of the topological features such as connected components, cycles, holes, tunnels, etc., and how these features are related.

[3]A set in $\mathbb{R}^d$ which can be reduced to one of its points by a continuous deformation is contractible.

δ_2. p is also called a 2-witness. Similarly, a node p is on the Voronoi vertex of three landmarks u, v, w if the distances differ by another relaxation parameter δ_3. p is called a 3-witness. The collection of 2-witnesses for the same pair of landmarks u, v is called a Voronoi edge $VE(u, v)$. The collection of 3-witnesses for the same triple of landmarks u, v, w is called a Voronoi vertex $VV(u, v, w)$. By choosing the relaxation parameters carefully, we can ensure that (i) Only 2-witnesses can possibly be 3-witnesses; (ii) Two neighboring 2-witnesses p, q must witness a common landmark; (iii) If there is an edge between the nodes in $VE(u, v)$ and nodes in $VE(u, w)$, both $VE(u, v)$ and $VE(u, w)$ are connected to a Voronoi vertex $VV(u, v, w)$. Notice that these correspond to the continuous case that two Voronoi edges intersect at a Voronoi vertex, etc.

With these conditions in mind, we will check whether a Voronoi cell $V(u)$ violates one of the above conditions, and if so refine this cell by including more landmarks.

Detect a hole in a Voronoi cell. We check whether $V(u)$ has one or multiple holes in its interior. This is done by checking whether there is a *cut locus* inside $V(u)$ in the shortest path tree rooted at u. A pair of neighboring nodes $p, q \in V(u)$ is on the cut locus, or denoted a *cut pair*, if the shortest paths from p, q to u have different homotopy types. In other words, they enclose some hole in between. If $V(u)$ has some hole, there must be cut locus which can be detected by a local algorithm. That is, each pair of neighboring nodes p, q following the shortest paths to u check whether they are diverging apart. The details of detecting cut locus can be found in our previous work [78].

We remark that as every node in $V(u)$ records the minimum hop count to u, the shortest path tree at u is implicitly stored in $V(u)$. Thus the cut locus detection is simply a local test at each pair of neighboring nodes and does not require any flooding inside the Voronoi cell.

If a cell $V(u)$ has one or more holes, we will detect a non-empty cut locus. The point on the cut locus furthest from u is selected as a new landmark. Specifically, a node on the cut locus can first check locally whether it is locally the furthest node from u. The locally furthest candidates can report themselves to landmark

u which is going select the furthest one.

Detect connectivity of a Voronoi edge. For two landmarks u, v that share a Voronoi edge, we check whether the Voronoi edge $VE(u, v)$ is disconnected. If so, we will include the point on $VE(u, v)$ furthest from u, v as a new landmark. The procedure for finding this node is the same as before.

Detect connectivity of a Voronoi vertex For any three landmarks u, v, w, their Voronoi vertex has one connected components. We check whether the 3-witnesses for u, v, w form a connected set. If not, we take the node furthest from u as a new landmark.

We remark that the homotopy equivalence condition is in addition to the rigidity and coverage condition of $DC(L)$, as shown below.

1. **Local Voronoi edge connectivity:** The Voronoi edges for each landmark u form a connected set.
2. **Local Voronoi ball coverage:** Each node x inside a Voronoi cell $V(u)$ is δ-covered[4] by a Voronoi ball $B_r(p)$, where p is a Voronoi vertex with landmark u.

For a Voronoi cell $V(u)$, if one of the above conditions is not satisfied, we refine the cell as follows.

1. If the first condition is not met, the Voronoi edges with u have two or more connected components. Since each Voronoi edge has either a Voronoi vertex or a point on $\partial\mathcal{R}$ as endpoints, we select, among all the endpoints of Voronoi edges of u on $\partial\mathcal{R}$, the one that is *furthest* from u as a new landmark.
2. If the first condition is met, we check the second condition. Among all the points that violate the local Voronoi ball coverage condition, we select the one that is least covered as a new landmark: $\arg\max_x \min_{B_r(p)}\{\delta' \,|d(x,p) = (1+\delta')r\}$. That is, for each such point x, we choose the Voronoi ball $B_r(p)$

[4]A ball $B_r(p)$ centered at an inner Voronoi vertex p with radius r equivalent to the distance from p to the closest landmarks is called a *Voronoi ball*. We say x is δ-covered by a Voronoi ball $B_r(p)$ if x is within distance $(1+\delta)\cdot r$ from the center p of $B_r(p)$.

with p such that $d(x,p) = (1+\delta')r$ with smallest possible δ'. And we select the point x with the largest such δ'.

Initially we flood the network from a random node r and find the farthest node p from r. Then we flood from p and find the farthest node q from p. q will be on the boundary as well. We use p and q as our two initial landmarks. We test the Voronoi cell with the above conditions and refine the cells that violate these. The refinement algorithm will stop when no new landmark is added.

4.3.2 Analysis of the Delaunay refinement algorithm

The main objective of this subsection is to show that the Delaunay refinement algorithm will stop with a proper set of landmarks. For simplicity, we will prove this for the continuous setting. Suppose $\mathcal{S}$ is a 2 dimensional smooth manifold with boundary in $\mathbb{R}^3$. The boundary is denoted as $\partial\mathcal{S}$. The Euclidean distance between two points p, q in $\mathbb{R}^3$ is denoted by $|pq|$. The geodesic distance between two points $p, q \in \mathcal{S}$ is denoted by $d(p,q)$. In the following we will assume the distance mentioned is measured as the geodesic distance, unless specified otherwise. $B_r(p)$ contains the points of $\mathcal{S}$ with geodesic distance no greater than r from p. For a path P or a cycle C, we note by $|P|$ or $|C|$ its length.

For a surface $\mathcal{S}$, we denote the *Euclidean medial axis* as the collection of points in $\mathbb{R}^3$ that has two or more closest points in $\mathcal{S}$, with the distance measured by the Euclidean distance. We also denote the *geodesic medial axis* as the collection of points in $\mathcal{S}$ that has two or more closest points in ∂S, with the distance measured by the geodesic distance on $\mathcal{S}$. For example, a cylinder's Euclidean medial axis is its axis and its geodesic medial axis is the circle along the midpoint of two boundary circles. See Figure 4.14. At any point $p \in \mathcal{S}$, we define a number of feature sizes, capturing the local geometric complexity:

- The *Euclidean local feature size* $ELFS(p)$ as the Euclidean distance to the Euclidean medial axis of $\mathcal{S}$.
- The *geodesic local feature size* $GLFS(p)$ as the geodesic distance to the geodesic medial axis of $\mathcal{S}$.

- The *homotopy feature size* $HFS(p)$ at a point p is defined as the maximum radius r such that $B_t(p) \cap \mathcal{S}$ is contractible for all $t \leq r$.

We define the *geometric feature size* $GFS(p)$ at a point p as the minimum of the above three feature sizes $\min\{ELFS(p), GLFS(p), HFS(p)\}$. We assume that

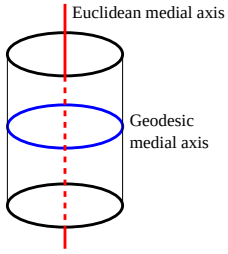


Figure 4.6. A cylinder's Euclidean medial axis is its axis (in red) and its geodesic medial axis is the circle along the midpoint of two boundary circles (in blue).

$\mathcal{S}$ is smooth with the holes far apart such that the geometric feature size is at least ε for some constant ε. We say a point $p \in S$ *γ-covered*, if p has at least one landmark within its geometric feature size. We call a set of landmarks L as a *γ-sample* if every point of S is γ-covered.

In the following we will prove that if a new landmark is inserted, then there must be a point p that has not yet been γ-covered, for some constant γ.

When a Voronoi cell $V(u)$ violates the local connectivity and coverage condition, then some point is not γ covered. The proof of the following two lemmas is the same as in 3.2.2.

Lemma 4.3.2. *If a Voronoi cell $V(u)$ violates the local Voronoi edge connectivity condition, the new landmark q selected is not covered by any landmark within $\gamma \cdot GLFS(q)$, for any $\gamma < 1/3$.*

Lemma 4.3.3. *If a Voronoi cell $V(u)$ for a landmark u violates the local Voronoi ball coverage condition, the new landmark q selected is not covered by any landmark within $\gamma \cdot GLFS(q)$, $\gamma = \delta/(2+\delta)$.*

We only show that when a Voronoi cell violates the homotopy equivalence condition, the new landmark selected is not yet γ-covered for a proper constant γ.

Lemma 4.3.4. *If a Voronoi cell $V(u)$ for a landmark u violates the homotopy equivalence condition, the new landmark q selected is not covered by any landmark within $GFS(q)$.*

Proof: We consider each case separately.

Case I. In the first case, $V(u)$ contains a network hole H. The new landmark q is on the cut locus of the shortest path map from u. Take $r = d(q, u)$. There are two shortest paths $P_1(q, u)$ and $P_2(q, u)$. Both have length r and they altogether surround H. Thus, the cycle C formed by $P_1(q, u)$ and $P_2(u, q)$ is not contractible. Clearly C is inside $B_r(q)$. Thus $B_r(q) \cap \mathcal{S}$ is not contractible. $HFS(q) \leq r$. This means that q is not yet 1-covered. See Figure 4.7 (i) for an example.

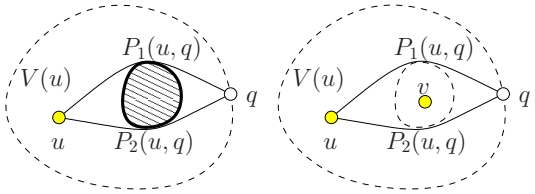


Figure 4.7. Left: The Voronoi cell $V(u)$ contains a hole. Right: The Voronoi cell $V(u)$ has another Voronoi cell $V(v)$ inside.

Case II. In the second case, $V(u)$ contains a Voronoi cell $V(v)$ in its interior. See Figure 4.7 (ii) for an example. We take the new landmark q as the furthest point on the cut locus of the shortest path map from u. Take the ball $B_r(q)$ with $r = d(u, q)$. Now the landmark v is outside of $B_r(q)$ since $d(q, v) > d(q, u)$. Therefore $B_r(q) \cap \mathcal{S}$ is not contractible. This shows that $HFS(q) \leq r$. q is not 1-covered.

Case III. In the third case, the Voronoi edge $VE(u, v)$ for two landmarks u, v has two or more connected components. Take two points p, q on different connected components, with q as the new landmark. See Figure 4.8. Take the point $x_1 \in \mathbb{R}^3$ on the midpoint of the Euclidean line segment pq. $r = |pq|/2$. Take the Euclidean ball $B_r(p)$. Note that x_1 in within $B_r(p)$. If p and q are the 2 closest boundary points to x_1, then x_1 is a point on the medial axis and we are done. Otherwise, there must be some other point a_1 on the boundary that is closer to x_1. Take x_2 to be on the midpoint of the Euclidean line segment pa_1. Notice that

$d(p, x_2) < d(p, x_1)$, since $d(p, a) = d(p, x_1) + d(x_1, a) < d(p, x_1) + d(x_1, q) = d(p, q)$. Therefore $d(p, x_2) \leq r$ so it is within $B_r(p)$. Now repeat the same process for x_2. Either x_2 has p and a_1 as its closest boundary nodes and therefore is a medial axis point, or x_2 has some other boundary point a_2 which is closest. Keep repeating the same process until the midpoint of line segment pa_n is a medial axis point. Note that since $d(pa_n) < d(pa_{n-1})$ this process must terminate, and $d(px_n)$ must necessarily lie within $B_r(p)$. This means $ELFS(p) \leq r$, $ELFS(q) \leq r$. Clearly $|pq| \leq d(p, q) \leq d(p, u) + d(u, q)$. Therefore $ELFS(p) + ELFS(q) \leq |pq| \leq d(p, u) + d(q, u)$. This means either $ELFS(p) \leq d(p, u)$ or $ELFS(q) \leq d(q, u)$. In either case one of the points p, q is not 1-covered.

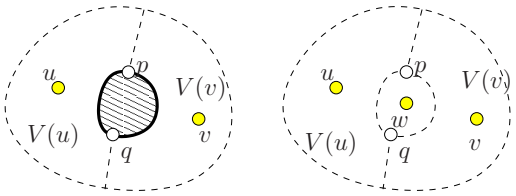


Figure 4.8. Left: There are two segments of the Voronoi edge for u, v. Right: There are two Voronoi vertices p, q for landmark u, v, w.

Case IV. In the last case, the Voronoi vertex $VV(u, v, w)$ for three landmarks u, v, w has two or more connected components. In this case the Voronoi edge $VE(u, v)$ has at least two or more connected components. This has been handled in the previous paragraph. □

Theorem 4.3.5. *The Delaunay refinement algorithm generates a finite set L of landmarks. If any landmark from L is removed, then it is not a γ'-sample of $\partial\mathcal{R}$, with $\gamma' = \gamma/(1+\gamma)$, $\gamma < \max(1/3, \delta/(2+\delta))$.*

Proof: As the minimum geometric feature size for any point p in $\mathcal{S}$ is ε. Then every landmark will cover the points in a ball $B_\varepsilon(p)$. Thus the Delaunay refinement algorithm can not continue infinitely.

For the last landmark q inserted, by Lemma 4.3.2, Lemma 4.3.3 and Lemma 4.3.4, it is not within distance $\gamma \cdot GFS(q)$ of any existing landmark. Since $\gamma > \gamma'$ the claim is true for q.

For any landmark q' added before q, we know $d(q, q') > \gamma \cdot GFS(q)$, since q is added with q' already present in the current landmark set. Since GFS is 1-Lipschitz, we have $GFS(q') \leq GFS(q) + d(q, q') < (1 + 1/\gamma) \cdot d(q, q')$. Therefore, $d(q, q') > \gamma' \cdot GFS(q)$. Notice that this argument is true for any pair of landmarks q, q' with q added after q'. Thus for q', the distance to any landmark in L is at least greater than $\gamma' \cdot GFS(q')$. The claim is true. □

4.4 Implementation

The step-by-step algorithm is presented in chapter 2 and 3. Here we will present the additions made to account for the new cases that emerge in 3D. To review there were previously 2 conditions checked for in the landmark selection phase:

1) If the voronoi cell is not simply connected then we select a new landmark to be the voronoi node that is furthest from the landmark defining the current cell.

2) If a node lies too far outside a voronoi ball as define by the target coverage criteria, then select the furthest such node to be a new landmark.

We add here another 2 conditions to check.

3) As remarked above in Figure 4.5 and as detailed in Figure 4.10 we have to deal with a new scenario of having 2 voronoi vertices for the same set of landmarks. Therefore during the landmark selection phase we add the condition that if there is a disconnected voronoi edge or disconnected voronoi vertex then we select node furthest from all the landmarks it witnesses to be a new landmark.

4) As discussed above in order to capture all holes in the network we must perform the cut locus algorithm within each landmark cell. Otherwise, we may have the situation illustrated in Figure 4.9 where without further refinement the hill will be represented as a single delaunay edge sticking up from the plane. If a hole is discovered, then the node on the cut furthest from the existing

landmark is made into a new landmark. This condition is only checked once all the 3 previous conditions have been satisfied.

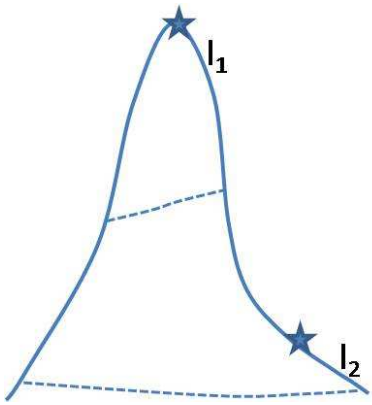


Figure 4.9. l_2 has a disconnected boundary

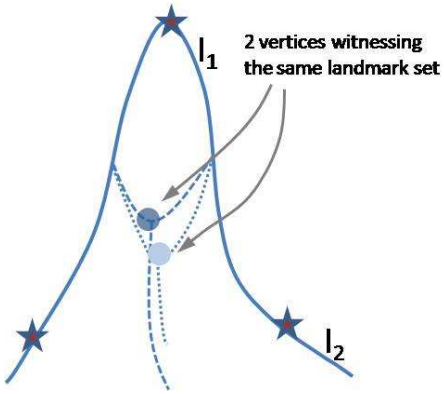


Figure 4.10. Two voronoi vertices witnessing the same landmark set

With regard to the embedding we also had to make changes to account for the added information of knowing the z-coordinate of each node in addition to the Delaunay complex and the hop count distance between each node and its nearby landmarks. An extra step is needed to translate the units of hops into real location units which the z-coordinate is represented in. In the real-world you may assume you know the length of a hop as simply a function of the transmission range of your radios. Alternatively when mapping the virtual coordinates given

by our algorithm to 3 anchors with real coordinates we can adjust the embedding accordingly.

In detail the embedding proceeds as follows: 1. We start with 3 pairwise connected landmarks and embed them relative to each other assign virtual coordinate consistent with the pairwise distance measurements. 2. A new landmark that neighbors 2 already embedded landmarks is selected (Some such node must exist based on the rigidity guarantee of the Delaunay complex). Letting l_1 and l_2 refer to the 2 embedded landmarks, we know (x_1, y_1, z_1) and (x_2, y_2, z_2) and z, the z-coordinate of the new landmark to embed. Given r_1 and r_2 and goal is to recover x and y. Since $((x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2) = {r_1}^2$ and $((x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2) = {r_2}^2$ we have 2 equations with 2 unknowns allowing us to solve for x and y.

Also as mentioned above we use a rubberbanding relaxation on all the nodes to distribute the embedding evenly across the network. Here too we use rubberbanding relaxation but we only allow freedom along the x,y plane while keeping the z-coordinate fixed.

4.5 Simulations

In Part I and II we thoroughly demonstrate the robustness of our algorithm under various conditions. We present results there for various network topologies, and vary parameters such as node density and the communication model. While we show that the algorithm works under low density conditions (with average node degree as low as 4 in some cases) there is a direct correlation between the average node degree (or density) and the accuracy of the final embedding. The reason for this is obvious—the higher the density, the more closely hop count distance matches the Euclidean distance, since routes can follow a straight line, and the distance of a hop is more uniform across the network. Similarly as the communication model moves further away from a unit disk graph model, the less accurate hop count distances become. Less accurate measurements leads to distortion in the embedding. The reader is referred back to chapter 2 and chapter 3 for results

based on these variables.

Here our aim is to show that the algorithm works in 3D by showing different topologies where we the algorithm was used to recover the ground-truth positions of the nodes. We put less emphasis on achieving low node degree and operating under poor communication models, and more of demonstrating the actual results possible using our method.

Figures 4.11,4.12, and 4.13 show the results of our algorithm. The figures show results with 0, 1 and 2 holes and variations in the number of hills and valleys at different heights and depths. The first 2 images of each figure give 2 perspectives on what the ground truth looks like. Image (iii) shows the embedding of the delaunay complex flattened to 2D. Images (iv) and (v) give 2 perspectives on the resulting embedding of all nodes. Finally image (vi) gives another view of the result by draping a surface over the nodes to accentuate the contours (note: holes don't appear in this view).

4.6 Related Work

4.7 Conclusion

This part is concerned with how we can adapt localization algorithms to work in real-world settings where nodes will not lie on a flat surface, but rather occupy regions with hills and valleys. While the basic approach of the algorithm for 2D is used here, there are a number of complex issues that must be addressed that only arise or become significant in 3D. By handling these special cases we prove that the algorithm keeps the good properties (global rigidity and coverage) needed for localization. This presents one of the first works in offering a localization algorithm for a 3D setting.

4.8 Appendix

We show that the geometric feature size function is 1-Lipschitz.

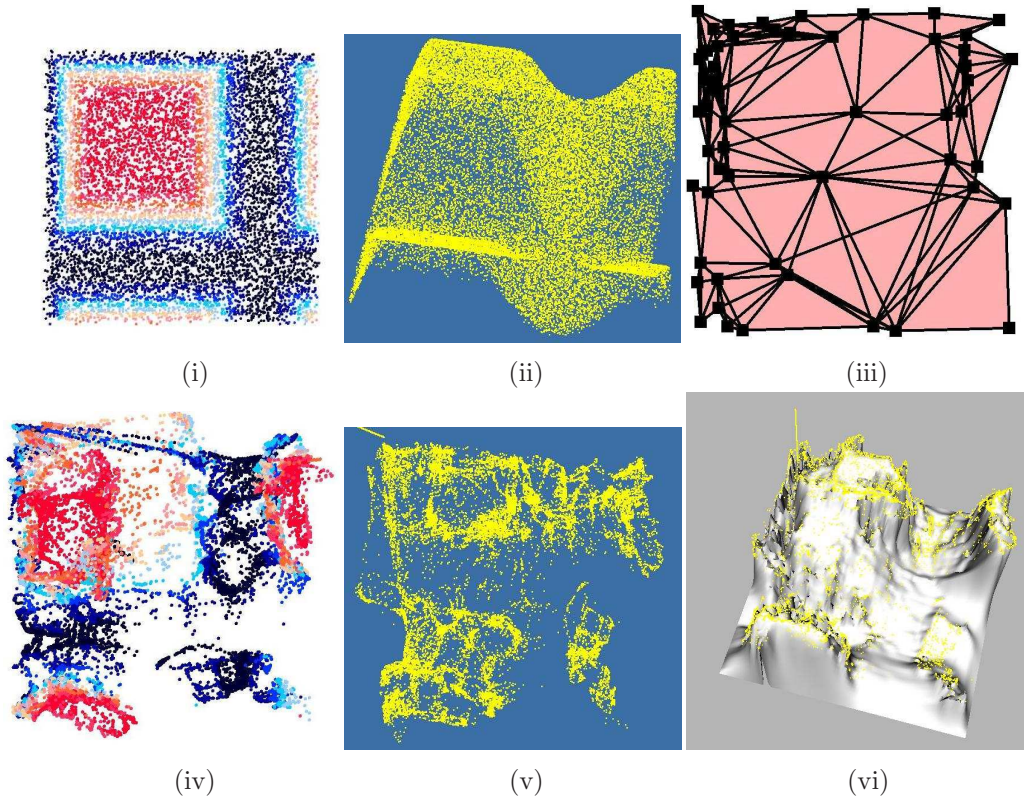


Figure 4.11. Mountain range scenario. (i) shows the ground truth flatted to the plane where red denotes 'high' points and blue 'low' points (ii) shows the ground truth in 3D, (iii) shows the Delaunay complex embedded (iv) shows all the nodes embedded projected onto the 2D plane (v) shows all the nodes embedded in 3D (vi) shows a surface draped over the embedded nodes giving a clear view of the result.

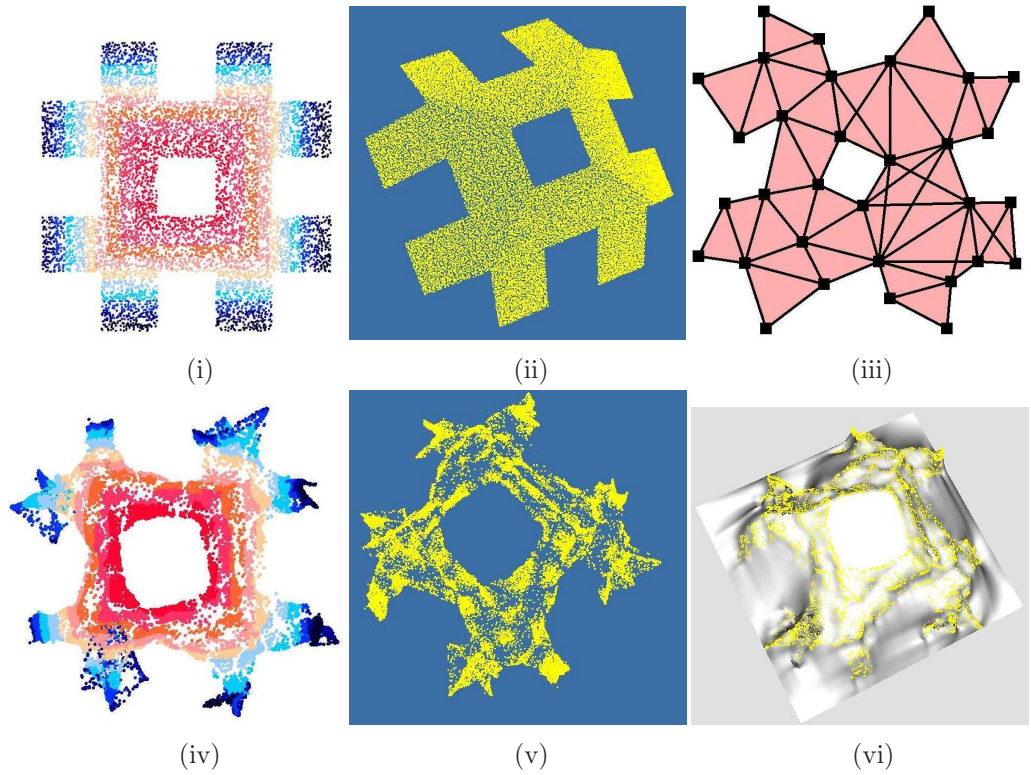


Figure 4.12. Road structure on hill. (i) shows the ground truth flatted to the plane where red denotes 'high' points and blue 'low' points (ii) shows the ground truth in 3D, (iii) shows the Delaunay complex embedded (iv) shows all the nodes embedded projected onto the 2D plane (v) shows all the nodes embedded in 3D (vi) shows a surface draped over the embedded nodes giving a clear view of the result.

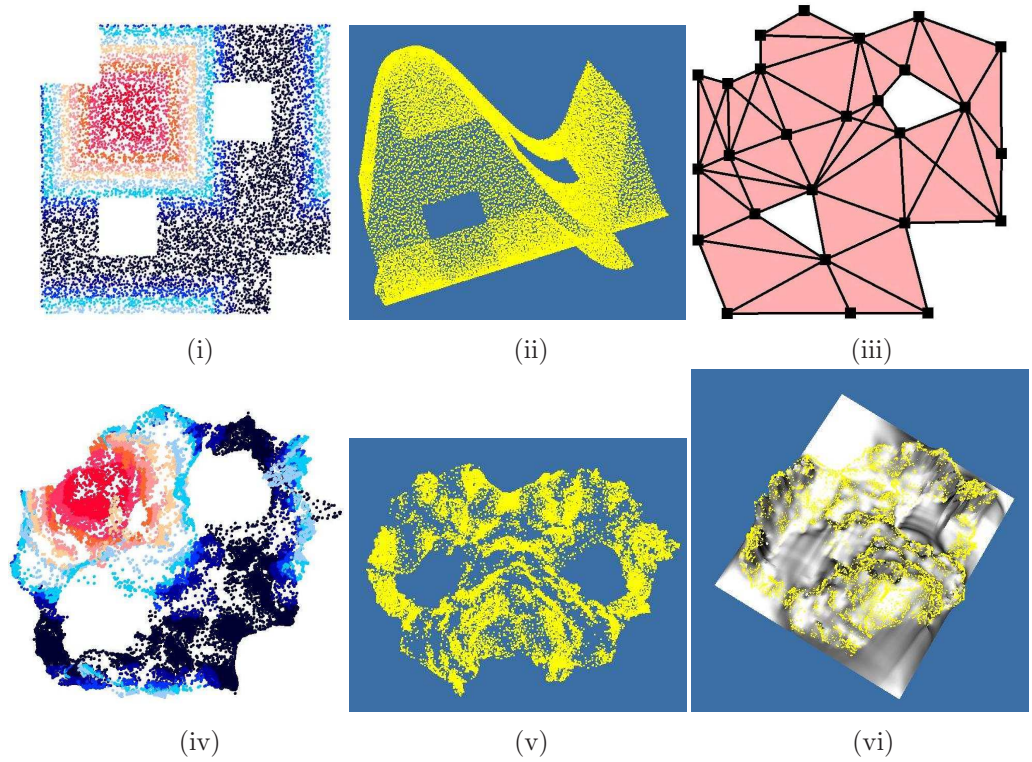


Figure 4.13. Two holes with undulating surface. (i) shows the ground truth flatted to the plane where red denotes 'high' points and blue 'low' points (ii) shows the ground truth in 3D, (iii) shows the delaunay complex embedded (iv) shows all the nodes embedded projected onto the 2D plane (v) shows all the nodes embedded in 3D (vi) shows a surface draped over the embedded nodes giving a clear view of the result.

Lemma 4.8.1 (Lipschitz continuity). *The geometric feature size of any manifold with boundary $\mathcal{S} \subseteq \mathbb{R}^3$ is 1-Lipschitz: $GFS(x) \leq GFS(y) + d(x, y)$ for any $x, y \in \mathbb{R}^3$.*

Proof: We first show that the Euclidean local feature size is 1-Lipschitz. The proof follows from triangle inequality. Suppose that point p is the closest point of y on the Euclidean medial axis of $\mathcal{S}$. Then $d(p, y) = ELFS(x)$. Thus, $ELFS(x) \leq d(p, x) \leq d(p, y) + d(x, y) = ELFS(y) + d(x, y)$. Similarly, the geodesic local feature size is also 1-Lipschitz, with the same proof.

Now we show that the homotopy feature size is 1-Lipschitz. Suppose the homotopy feature size for a point x is r. That is, for all $t \leq r$ the ball $B_t(x) \cap \mathcal{S}$ is contractible. Now, for a point y, if its homotopy feature size is $r' > r + d(x, y)$. We take $t = r' - d(x, y)$, $t > r$. $B_{r'}(y) \cap \mathcal{S}$ contains $B_t(x) \cap \mathcal{S}$. This means that any cycle in $B_t(x) \cap \mathcal{S}$ is contractible – if otherwise this non-contractible cycle is also inside $B_{r'}(y) \cap \mathcal{S}$, which will be a contradiction. Thus $HFS(x) \leq HFS(y) + d(x, y)$.

Putting everything together, $GFS(x) = \min\{ELFS(x), GLFS(x), HFS(x)\} \leq \min\{ELFS(x), GLFS(x), HFS(x)\} + d(x, y) \leq GFS(y) + d(x, y)$. □

In the following we will use two distance measures, the Euclidean distance and the geodesic distance between any two points $p, q \in S$.

For a surface S, we denote the *Euclidean medial axis* as the collection of points in $\mathbb{R}^3$ that has two or more closest points in S, with the distance measured by the Euclidean distance. We also denote the *geodesic medial axis* as the collection of points in S that has two or more closest points in ∂S, with the distance measured by the geodesic distance on S. For example, a cylinder's Euclidean medial axis is its axis and its geodesic medial axis is the circle along the midpoint of two boundary circles. See Figure 4.14. At any point $p \in S$, we define its *Euclidean/geodesic local feature size* $ELFS(p)/GLFS(p)$ as the Euclidean/geodesic distance to the Euclidean/geodesic medial axis of S. We also define *homotopy feature size* $HFS(p)$ at a point p as half of the length of shortest non-contractible cycle (i.e., enclosing one or multiple holes). The *generic feature size* is defined

as $FS(p) = \min\{ELFS(p), GLFS(p), HFS(p)\}$. Intuitively, the Euclidean local feature size captures how much a surface bends near p. The homotopy local feature size captures how close p is to the hole boundaries.

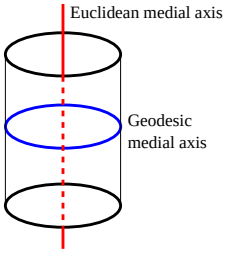


Figure 4.14. A cylinder's Euclidean medial axis is its axis (in red) and its geodesic medial axis is the circle along the midpoint of two boundary circles (in blue).

For any point $p \in S$, we denote its *homotopy feature size* $HFS(p)$ as the largest radius r such that $B_r(p)$ and $B_r(p) \cap \partial\mathcal{S}$ are both contractible.

From the above Lemma, for any point p, its homotopy feature size must be no greater than its geodesic feature size.

Corollary 4.8.2. *$HFS(p) \leq GFS(p)$, for any $p \in \mathcal{S}$.*

Proof: Assume otherwise. There is a point $p \in \mathcal{S}$ such that $HFS(p) > GFS(p)$. Take $HFS(p) > r > GFS(p)$. The ball $B_r(p)$ contains a point on the medial axis. Thus $B_r(p) \cap \partial\mathcal{S}$ is not connected. This contradicts with the fact that r is smaller than $HFS(p)$. □

Proof: As the combinatorial Delaunay complex is the Cěch complex of the Voronoi cells, the theorem follows immediately from the Cěch Theorem [13]. Recall the definition of the Cěch complex. Given a collection of sets $\mathcal{U} = \{V(u), \forall u \in S\}$, the *Cěch complex* is the abstract simplicial complex whose k-simplices correspond to nonempty intersections of $k+1$ distinct elements of $\mathcal{U}$. The Cěch Theorem says that if the sets and all non-empty finite intersections are contractible, then the union $\cup_u V(u)$ has the same homotopy type as the Cěch complex. In our case, the Cěch complex is the Delaunay complex $DS(S)$, the union of the Voronoi cells is $\mathcal{R}$. Thus the claim is true. □

Theorem 4.8.3. *A Voronoi edge $VE(u, v)$ cannot form a loop with the two landmarks u, v on the same side.*

Proof: Assume otherwise that the Voronoi edge contains a loop L with u, v on the outside. See Figure 4.15. We take u' as the closest point on L to u and

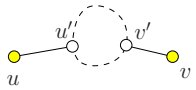


Figure 4.15. Two landmarks on same side.

v' the closest point on L to v. We argue that $u' = v'$. If otherwise, we have $d(u', u) < d(v', u)$ as well as $d(v', v) < d(u', v)$. Since L is part of the Voronoi edge of u, v, $d(u, p) = d(u, p)$ for any point $p \in L$. This implies that $d(u', u) < d(v', u) = d(v', v) < d(u', v) = d(u', u)$. This implies a contradiction. □

In particular,

1. Any Voronoi cell does not have one or multiple holes (either a real network hole or another Voronoi cell) in its interior. Remark that any Voronoi cell $V(u)$ must be connected as the nodes all have paths to u.
2. The Voronoi edge of any two landmarks is either empty or contractible.
3. There are at most one Voronoi vertex for any three landmarks.

The following claim is true for geodesic medial axis. The proof for a surface $\mathcal{S}$ in the plane is given in chapter 2. Actually the same proof works for a surface $\mathcal{S}$ in $\mathbb{R}^3$.

Lemma 4.8.4. *Given a ball B containing at least two points on $\partial\mathcal{S}$, for each connected component of $B \cap \mathcal{S}$, either it contains a point on the inner medial axis, or its intersection with $\partial\mathcal{S}$ is connected.*

The following Lemma about Euclidean local feature size is useful in later proofs.

Lemma 4.8.5 ([23]). *Given a Euclidean ball B containing at least two points on $\mathcal{S}$, for each connected component of $B \cap \mathcal{S}$, either it contains a point on the Euclidean medial axis, or one of the following is true: (i) $B \cap \mathcal{S}$ is not a topological disk; (ii) $\partial(B \cap \mathcal{S})$ is not a topological circle.*

Proof: If B is not tangent to $\mathcal{S}$ we can shrink the ball until it is tangent to $\mathcal{S}$ at some point x. Then we shrink the ball and keep it to be tangent to x, until B is tangent to a point y and does not include any other points of $\mathcal{S}$ inside. We denote this ball as B'. At this point the center of B' is on the Euclidean medial axis. Notice that B' is included in B'. Thus B contains a point on the medial axis. □

Part II

Sensor Network Location Certification

Chapter 5

Collaborative Location Certification

5.1 Overview

In many instances localization of the network is not enough, we must also ensure that the location that a node reports itself to be ins in fact truthful. Thus, a robust network must require such information be uncompromised, lest a few faulty or malicious nodes will have a deleterious effect on the entire network.

We introduce a protocol that validates the truthfulness of location information associated with event reports. The protocol relies on the collaborative interaction of the network nodes to find compromised parties. Since each node is an active participant in the network and spends a substantial amount of time and resources relaying messages for others, it automatically has some knowledge of the activity within the network. This knowledge can be put to good use in spotting anomalous behavior. The workload and detection ability is thus distributed across the network, to avoid a single point of failure and gracefully degrade with increasing number of compromised nodes.

To achieve this purpose, at an overview level, nodes in the network (compactly) record summaries of routing paths taken by packets through the network. Upon

receiving a packet, nodes examine whether their route matches a historically expected behavior by packets from the same claimed location. A belief about the correctness of this location claim of this packet is then created and propagated to the data sink, either as part of this packet or later on, in an out of band fashion. The attached beliefs will be used by the authorized packet evaluators PEs (sinks or authorized intermediate relay nodes) to certify the truthfulness of the packets' location information.

We show by simulations that the belief rating has a strong correlation with the deviation of the source's real location from its claimed location. Thus if a node lies about its location, the farther away it claims to be from its real location, the more likely the packets will be identified.

The memory limitations of sensor nodes require light-weight protocols both in terms of memory and power usage. Accordingly, we developed a path metric and a compact way to express path trajectories, by using locality-sensitive hashing [45]. This metric captures the fact that packets from sources with incorrectly claimed locations are likely to have path trajectories deviating from previously observed traffic paths.

The key advantage of our solution lies in its collaborative nature and the involvement of the network in a community of trust. A single malicious or faulty node will is unlikely to take over the entire network and cause significant damage. To better understand the challenge of the problem and the rationale of our approach, we also outline the following alternative straight-forward schemes.

- **Immediate neighbor detection.** An immediate neighbor (p) of the malicious node could detect that it is not in the region it claims, because this region is out of the communication range of p. This scheme is vulnerable to multiple (two) adversarial colluding nodes; the adversary directly communicating with p would not actually be lying about its location. Moreover, in general the problem is more significant, because an adversary can "create" a whole set of fake nodes $b_1, b_2, \cdots, b_k$ where the distance between any b_i and b_j is within the communication radius R, and the distance between b_k and the true node is more than R. Even by looking at the distances between

all the nodes on the path and ensuring that the distance between any two consecutive nodes is short, an honest node cannot determine if there is a node lying about its location.

- **Distance from straight line trajectory.** A simple idea for a scenario where the sensor nodes are densely and evenly distributed throughout the region is to compute how far the current node lies off the straight line trajectory between the source node and the sink. Based on how far away from the direct path this node lies should the belief rating be made. However, this mechanism would fail in the case of irregularities in the network. Specifically, when there are holes in the network, or when routing paths do not follow straight line paths, a node may well be placed far from the straight line from source to destination — also observed in real experiments [83]. As shown later, our path metric is adaptive to these variations in traffic patterns. We capture the case when a packet follows a path not "similar" to the expected ones, in which case the packet will be tagged with a poor belief rating.

This part is structured as follows. In section 5.2 we overview related work on location certification and secure localization algorithms. We introduce the sensor model and the adversary model in section 5.3. In section 5.4 we discuss collaborative location certification, and the belief rating generation and propagation. Both the metric and the detection ability are evaluated by simulations in section 5.8.

5.2 Related Work

Secure localization has been studied by a number of groups. For example, the SeRLoc protocol [55] tackles secure localization by using specially equipped "locator" nodes that emit powerful beacon signals through the networks. Depending on the beacons a node hears, it computes the "center of gravity" of the overlapping regions to determine its location. The locator devices are assumed to be tamper proof. Elsewhere [57] robust statistics are used to improve the resilience of

an anchor-based localization algorithm in a hostile environment where the nodes may receive false information from the neighbors.

Even though the nodes can obtain their locations correctly by these secure localization protocols or extra secure location support such as GPS, they do not prevent a node from lying about its location and generating event reports with a false location claim. A few existing protocols have tackled this problem for inter-node settings, mainly using fine grained timing analysis of signal travel times, under the assumption of high accuracy clocks available on sensors.

Capkun and Hubaux [17, 18] introduced Verifiable Multilateration (VM), to prevent a node from lying about its own position, and Verifiable Time Difference of Arrival (VTDOA), to stop an adversary from influencing the reported position of a true node. VM uses 3 anchor nodes that surround the unknown node. Transmissions are with RF signals that travel at the speed of light, therefore claimant can only pretend to be further away from any one anchor, but not closer (since it can't make its transmissions go faster than light). To claim a greater distance to one anchor node, it must claim a small distance to another node, which is impossible since trilateration is used. VTDOA compares the TDOA and ToF distance estimations to prevent a malicious node from jamming the signal of a true node and replaying it from another location. This technique requires that the anchor nodes are synchronized with each other (and the claimant node). The Echo [68] protocol uses a multi-part handshake between some "verifying" nodes and the claimant using RF and ultrasound to guarantee that the node is in an asserted area. It does not however pin-point the precise location of a node, it only verifies that it is within a rough region, and it requires that the verifiers be within the region in question. Waters and Felten [79] present a similar scheme which uses tamper-resistant devices.

Timing analysis typically requires highly accurate clocks that may not be realistic in networks with inexpensive hardware, e.g., cheap sensor nodes. For example, the VM scheme requires accurate synchronization (maximum clock difference of 1ns) [17, 18]. The use of ultrasound relaxes a little the stringent requirement on synchronization, but requires an additional hardware artifact. Moreover, these

anchor-based schemes typically assume authorized and trustworthy anchor nodes and often require a sufficient number of anchors that cover all the sensor nodes, which may not be practical when the deployment and the operation of the sensor network are in hostile territory.

The scheme we propose in this thesis does not assume any anchor nodes, nor any special hardware such as ultrasound transmitters. We take a different approach and establish, by the participation of all the sensor nodes, a collaborative community that certifies the locations of packets routed through the network. We can thus make use of the traffic pattern, brought by the message relaying, at little or no extra cost.

5.3 Model

In this section we discuss the considered adversarial and deployment models.

5.3.1 Adversary

Of concern here is a malicious, powerful adversary with strong incentives to capture and compromise sensors for the purpose of altering the sensor data flow, e.g., by inserting false data and event reports and eventually influencing decision making process at the base station. For such an adversary, pure denial of service (DOS) attacks that aim to disable sensors and parts of the network are only of marginal interest and will not be considered here. For DOS attacks, [20, 80] offer techniques to address these issues. Multi-path forwarding [49] alleviates the problem of malicious relay nodes dropping legitimate reports. Also, by using a cache to store the signatures of recently forwarded reports we can prevent against the same packet from being replayed [46, 81].

In particular, the mechanisms introduced here provide correctness assurances of node location claims in the process of event reporting. They prevent a compromised node from generating illicit event reports for locations other than its own. This is important because by compromising "easy target" sensors (say, sensors on

the perimeter of the field that's easier to access), the adversary should not be able to impact data flows associated with other ("premium target") regions of the network. To achieve this goal, data routed through the network will be "tagged" by participating nodes with "belief" ratings, collaboratively assessing the probability that the claimed source location is indeed correct. We call this process *location certification*.

To circumvent location certification (e.g., for the purpose of injecting fake event reports referencing remote, out of reach locations) an adversary could attempt to: (i) favorably modify certificates for its own fake data (e.g., by altering the associated belief ratings), or (ii) unfavorably alter certificates of legitimate traffic. The probability of success of such attempts is naturally related to the density of compromised nodes in the network. The ability of success adapts gracefully to the density of compromised nodes and the solution can operate even in the presence of a large number of adversarial nodes, as validated through simulations.

For illustration purposes, we first consider an adversary that only attempts to maliciously claim a different location in its event reports (but does not maliciously alter belief ratings of other packets it routes). We then discuss additional security issues in section 5.5.

5.3.2 Deployment and Routing

We focus in this thesis primarily on monitoring networks in which the sensors collect information of interest and send data/event reports to a base station (data sink) for post-processing and analysis. Immediately after deployment, for an initial short period, the network is assumed free from any adversarial presence. Since our location certification procedure is based on using past history of network routes, we must assume that the original history is initially "clean". The better the history data is in terms of "cleanliness", the better the location certification will perform.

Since history data is used to predict future network behavior, we assume that the network is to some degree consistent in its routing behavior. If strong routing

patterns are exhibited, i.e., all packets from the same source are typically routed along similar paths, then location certification will perform well. If the legitimate routing behavior is completely random (say packets from the same source take arbitrary routing paths) then our protocol won't work well. Naturally, geometric routing maintains a high degree of consistency in routing patterns, so we generally will speak in terms of a geometric routing protocol, and this is the routing scheme used in our simulations. However, geometric routing is not a prerequisite for the protocol.

5.4 Certification Algorithm

In this section we detail the main components of the location certification protocol.

5.4.1 Solution Overview

At an overview level, the proposed solution unfolds as follows. Immediately after deployment, network secure localization protocols [55] allow sensors to acquire location information that is to be later used in event reports. Existing research achieves this by assuming a largely un-compromised network for a short amount of time after deployment. We believe this is reasonable, especially if we consider the minimal time and resource requirements for corrupting even an individual sensor. In other words, even in the presence of an adversary with immediate physical access to a sensor node, some amount of time (e.g. minutes) will be required to locate and compromise the sensor internals and software.

Once the network becomes fully operational, sensors will start generating event reports associated with their respective location. A compromised sensor could then attempt to generate illicit event reports for locations other than its own. To defeat such an adversary, nodes along the path from the source to destination will attach "belief" ratings to passing data packets, quantifying the correctness probability of the claimed source locations. Informally, beliefs are a function of observed past traffic patterns in conjunction with the claimed source location.

Upon receiving packets with routing information deviating from expected traffic patterns, nodes will have the opportunity to propagate negative belief ratings associated with these packets. The negative beliefs reflect the appearance of an anomaly in the routing pattern.

Thus this scheme is able to detect both the case in which the routing pattern is altered by an adversary (a compromised node lies about its location, or other routing attacks such as wormhole attacks [44,63,67]) and the case of node failures — a large fraction of nodes run out of battery power or get physically destroyed by adversaries causing significant routing pattern changes. A node which rarely participated in the data collection operation will get a low confidence, which is reasonable as the network collectively has little or no information to decide whether it is a legitimate node.

We re-iterate that this solution assumes *the network is initially free from adversaries* for a short period of time. If a large number of compromised nodes is present at the start and they are able to generate arbitrary traffic patterns then collaborative certification will be less effective. In a typical deployment there is often a short period of time which is more than enough for our scheme to collect enough history traffic data. With this limitation in mind, we believe that the novelty in our scheme lies in the compact and efficient way of summarizing the history traffic pattern and the ability of using the history to verify the correctness of future packets.

5.4.2 Strawman's Book-keeping.

Before proceeding, to illustrate, we first discuss an extremely simple book-keeping mechanism, the understanding of which will motivate our final solution. As part of the routing protocol, each sensor will maintain a history and normalized count of each previously seen source-destination pair for routed packets. New incoming packets from rarely seen sources will then be considered more suspicious and associated with a low rating.

While this scheme is extremely simple and scalable, it presents certain limitations, in particular in its ability to detect deviations in full routes (as opposed to endpoints). If a node does route information between the claimed origin and destination, then the packet from an adversary claiming to be from a different location will be considered fair game. To achieve a better detection accuracy, more information about information flow is required in the belief rating construction.

5.4.3 Inter-Path Distance Metric

Accordingly, we explore first how to compare packet routes efficiently in a meaningful way. Based on the sequence of nodes a packet has visited, we derive a trajectory of a packet by the piece-wise linear curve connecting the intermediate nodes in their visited order.

We define a distance metric that measures how far two trajectories are. The distance is designed such that fake claimed locations for packets will result in large distances between real and expected trajectories. There are many generic ways to measure the distance between two curves in space, such as Hausdorff distance and Frechét distance. Here we design a measure well suited to our problem at hand.

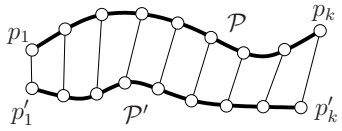


Figure 5.1. The distance metric between two paths $\mathcal{P}, \mathcal{P}'$. In this figure we adopt a uniform parametrization and the samples are placed uniformly on the paths.

Given a trajectory $\mathcal{P}$ (a curve in the plane), we adopt a parametrization (e.g., uniform, but other parameterizations may also be used, as will be shown later) and take k samples $\{p_1, p_2, \cdots, p_k\}$, on $\mathcal{P}$. We define the distance between two paths $\mathcal{P}, \mathcal{P}'$ as $\pi(\mathcal{P}, \mathcal{P}') = \sum_{i=1}^{k} ||p_i - p'_i||^2$, the sum of squared distance between corresponding sample points. From a different viewpoint if a path $\mathcal{P}$ is considered a point in $2k$ dimensional Euclidean space $p = (p_1, p_2, \cdots, p_k)$ (each point p_i is a point on 2D), the distance between two paths is the squared ℓ_2 norm of

their corresponding representative $2k$-dim points. In the following we will see this observation become very useful in the design of a succinct data structure that summarizes the relative distances of a set of paths by a set of points on a 1-dimensional line.

Essentially each node will keep a compact structure (to be explained in section 5.6) that summarizes the trajectories of the packets that go through it. Once a new packet arrives, the current trajectory is compared against past trajectories of packets from the same source or nearby. See Figure 5.2 for an example where an adversarial node s sends a message that goes through a node u, but claims it is at location s'. The path taken by the packet, $\mathcal{P}$, is different from the path it should have taken if it is indeed generated from a node at s' (shown as the dashed path $\mathcal{P}'$). This is exactly the type of discrepancies are be captured by our path metric. We use a parametrization scheme that samples uniformly in each hop. If a path has m hops, then each hop is sampled uniformly k times to yield a total of $m \times k$ sample points.

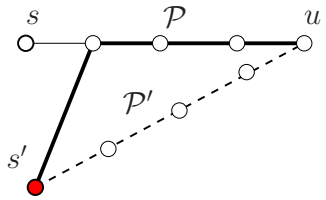


Figure 5.2. The real path $\mathcal{P}$ taken by a packet from s is different from the path $\mathcal{P}'$ it should have taken if it were generated from the claimed location s'.

5.4.4 Locality Sensitive Hashing

The memory foot-print of full path history on sensors would be too large. Consequently, we adopt locality sensitive hashing [45] a mechanism perfectly suited to compress such data and represent each path by a single value. The distance between two paths then becomes the distance between their compressed values. In general, locality sensitive hashing takes points in high dimensional space and maps them to 1D such that the Euclidean distances between them are roughly

preserved. Recall that each path can be considered as a point in $2k$-dimensional space, which is then hashed to a point in 1D such that the distance between any two paths is correlated to the distance between their corresponding 1D points.

Locality-sensitive hashing makes use of the properties of stable distributions. A stable distribution [21] is a distribution where the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where $X_1 \ldots X_n$ are i.i.d. variables from that distribution. It is known that Gaussian distribution is stable for ℓ_2 norm. This means that if we represent a path $\mathcal{P}$ in our network by a vector in $2k$-dimensional space $v = (p_1, p_2, \cdots, p_k)$ and generate a random vector a (with each element chosen uniformly randomly from a Gaussian distribution $\sigma(0, 1)$), of the same dimension, then taking the dot product of the two vectors, $a \cdot v$, results in a scalar distributed as $||v||_2 X$, where $|| \cdot ||_2$ is the ℓ_2 norm, and X is a random variable with Gaussian distribution $\sigma(0, 1)$. It follows that for two vectors (v_1, v_2) the distance between their hash values $|a \cdot v_1 - a \cdot v_2|$ is distributed as $||v_1 - v_2||_2 X$ where X is a random variable of a Gaussian distribution. Therefore, if we have a vector v_i, which represents a path in our network, we can generate a scalar value from it (by taking the dot product with a) that still maintains the property that its distance from another scalar generated by another vector v_2 is correlated to the original "distance" between v_1 and v_2 as we previously defined. A hash function that uses random variables of a stable distribution to map high-dimensional vectors to 1D points satisfies the above definition of locality sensitivity.

Upon receiving a packet with observed trajectory vector v, each sensor will use locality sensitive hashing to store only the hashed value $h(v) = a \cdot v$ together with the location where this packet was generated. For a new packet that claims to be from the same region, the hashed value of the new packet is compared with the hashed values in the past history. A belief rating directly proportional to the difference of these two values is then generated.

In practice, to reduce overhead, a packet will only carry the position of the last node visited and the hash value computed by that node. A new node will update the packet with its own location and the new hash which is computed as the sum

of the old hash plus this node's $h(v)$ value. This yields similar results to the method described above, while significantly reducing communication overhead.

In section 5.8.1 we show how the use of locality sensitive hashing in the design of belief ratings results in a strong correlation between the distance a node is from a claimed location and the resulting belief.

5.4.5 Belief Generation

By the property of locality sensitive hashing, packets taking paths similar or identical to each other tend to cluster together on the real line, while packets coming from unknown regions or following a highly irregular path map to different points. To express a correctness belief about the claimed location of incoming packets, their associated hash values are thus compared with the expected value ranges of nodes originating from the same region. We explain in section 5.6 how the hash values of different packets are stored and compared, and how we group hash values based on the path's origin.

Note that when the network is first deployed nodes do not have any history about the network. We allow the beliefs to evolve gradually over the lifetime of the network by having the belief rating be comprised of 2 values: one is the actual *belief rating*, the second is the *belief confidence*. Intuitively, the rating captures how well a new packet matches previous history data, while the confidence measures how much history data a node has at its disposal. If the confidence is low than the belief value is less useful. Initially, in a new network, the confidence values will be low, and the beliefs ratings will be of limited value. But as nodes accumulate history, belief ratings will be given with higher confidence. The interplay between belief values and confidence values is complex and actual thresholds of what is considered a "good" rating or "bad" rating would depend on the particular network and/or application. These can be adjusted depending on the circumstances. The data structure we use captures both components of the belief rating. See section 5.6.

5.5 Security

We now describe the additional measures that must be taken to ensure that an adversary, or a group of adversaries working together, does not tamper with the belief ratings or hash values of packets passing through it. We want to ensure that (i) individual belief values are not tampered with in transit, (ii) packets containing incriminating ratings cannot be distinguished from other traffic, (iii) new fake belief values are not added to bad-mouth a packet or improve its rating, and (iv) existing belief ratings are not removed from packets.

5.5.1 Semantic Security

To ensure the above, we first require that each sensor be associated with a unique, public identifier (e.g., MAC) and with a secret, unique symmetric encryption key, known only to a very small set of authorized, un-compromised parties such as the data sink or a few intermediary relays, called packet evaluators, `PE`s. This is a reasonable, practical assumption to be found in existing research [16].

This key can then be used for communication between the sensor and the `PE`s. Such communication however, we require to be deployed using any semantically secure encryption cipher [40] (e.g., any cipher running in CTR mode). Semantic security is necessary to prevent an adversary of correlating encrypted fields in the current packet with fields of previously seen ones (e.g., if they represent the same value). Our solution does not depend on the deployed encryption mechanism. Symmetric key cryptography has been chosen over public key crypto, due to the computation-limited platform assumed. While details are out of scope here, we note that more powerful mechanisms can be devised using asymmetric key primitives. Such mechanisms would allow optimized, in-network location claim evaluation and packet filtering, effectively reducing overhead induced by compromised traffic.

5.5.2 Secure Belief Propagation

Upon generating a belief b (composed of a rating and confidence, see section 5.4.5) and hash for the current packet, a sensor i will encrypt it using its shared symmetric key with the sink k_i and append the result $E_{k_i}(b)$ to the packet. Requirements (i) and (ii) above are naturally handled. To ensure (iii) and (iv) we propose to use a cryptographic digest[1] chain constructs. Specifically, upon propagating a new belief b with the current packet, a sensor i will perform two operations. First it will encrypt the belief as above $E_{k_i}(b)$. Second, it will update the packet's digest chain value c as follows:

$$c_{new} \leftarrow H(c_{old}|H(b)),$$

where H is a cryptographic digest, and '|' denotes concatenation. At the PEs end, each packet's digest chain can be reconstructed and verified upon decrypting all beliefs. To selectively remove a rating from the packet, a malicious adversary is faced with having to correctly reconstruct a new digest chain for the remaining beliefs. This, however, will require the decryption of those beliefs (using secrets not in the possession of the adversary). Thus (iii) and (iv) are handled. An adversary can still attach a bad belief rating, i.e., bad-mouth a packet. But this is essentially denial of service attack in which the relay adversary can simply drop the packet from the data stream. As message digests over small amounts of data are extremely fast, the induced overhead is minimal.

To summarize, a node receiving a packet will append an encrypted belief rating, update the digest chain as mentioned above and deliver the packet to the next hop. The way that nodes attach belief ratings can be adaptive to the network scenario and desired detection ability. We now introduce two belief generation

[1]We use "cryptographic digest" to denote a cryptographic hash function, so as to avoid any confusion with the locality sensitive hash constructs. We note that the collision resistance of such a hash function is not paramount here, given that the adversary would have a hard time finding meaningful collisions within a few minutes (and for a large enough number of packets to become meaningful), before the packet is due at the sink. We assume the hash function outputs 6 bytes.

and propagation methods.

Stochastic Propagation. An immediate and natural extension to the above propagation mechanism is the use of a stochastic behavior reducing communication requirements while gracefully degrading security assurance levels. Specifically, sensors express beliefs only with a certain probability $p_s \in (0, 1]$. Upon receiving an incoming packet, a node will uniform randomly decide whether to generate and propagate its current belief. Naturally, p_s will now determine the efficiency of the solution; lower values will result in less beliefs propagated but also lower communication overhead. Nevertheless, this will not immediately result in a beneficial environment for a malicious adversary, as there is a certain favorable asymmetry to be considered. An adversary will not be able to determine which sensors decide to express beliefs about a given packet. Statistically, over a larger number of packets, even small values for p_s will result in the detection of consistent malicious behavior. This is validated through simulations.

Out-of-Band Propagation (OOBP). Probabilistic, semantically secure encryption was used to prevent adversaries from understanding and correlating beliefs generated by the same party about different packets. Nevertheless, an adversary can still selectively drop (a few) packets, just below the threshold of being detected by traditional denial of service prevention algorithms [80]. A simple solution can be deployed to detect and prevent this behavior.

Informally this solution proceeds as follows. Each sensor maintains a window W of yet-to-be-propagated beliefs. The belief generated for a new incoming packet will not be propagated as part of the current packet, but instead will be placed in W and its place will be taken by another (older) belief from W. To maximize network usage and probabilities of arrival, the replacement choice will need to be performed considering such things as its age in the window, its packet priority (if any), its type (e.g., bad beliefs should be propagated faster) etc. At the receiving side, the `PE` will maintain a similar window for each incoming packet for which not all expected (or a sufficient number of) beliefs have been received. In the event of network delays or corruptions, the `PE` may decide to timeout while waiting and

accept or reject based on the currently received beliefs.

The benefits of such a belief propagation mechanism are multi-fold. To selectively drop packets while remaining undetected, a malicious adversary would need to eliminate all beliefs pertaining to these packets. The semantically secure nature of encryption and the fact that they are propagated out of band, render this a difficult problem. The adversary has little incentive to remove belief ratings on a packet (i.e., attack (iv)) as those beliefs are not necessarily certifying this packet. Its hardness increases naturally with larger sizes of the window $|W|$ – due to increasing un-predictability of the belief propagation process. For $|W| = 1$, the mechanism gracefully converges to the above described protocol. Moreover, as the window size is node-specific it can be set dynamically per-sensor, considering memory constraints. Space constraints prevent further details here.

5.6 Storage

We now introduce the data structure used to store history data within each node, namely a dynamic quadtree sub-dividing based on the amount of contained history data — nodes with high densities of data will further subdivide to achieve finer history "resolution". Each sensor p organizes the paths observed in a quadtree grouped by the sources of the paths, to exploit their spatial correlation. Intuitively the paths taken by packets from nearby sources passing through p should be similar. Thus even if p sees a packet for the first time from node x, p can test the path against paths in its history originated from the neighborhood of x.

In effect, this divides the area of the network into regions, and use the history of each region to form beliefs on individual nodes within. The sensor field is recursively divided into quadrants as needed. For each quad, we store the mean of all the hash values of paths originating from that region and their standard deviation. The partitioning of the quad is controlled by the standard deviation and the number of hashed values inside. When the standard deviation is larger than a threshold, the quad is further partitioned.

The standard deviation of the hash values inside a quad measures how similar

paths originating from a particular region are. Thus the standard deviation naturally controls the granularity of the quadtree partitioning. We don't maintain the hashed values inside each quad, but only store their mean μ and standard deviation σ. Specifically, suppose there are m hashed values $x_1, x_2, \cdots, x_m$ inside a quad, then

$$\mu_m = \frac{1}{m}\sum_{i=1}^{m} x_i, \sigma_m = \sqrt{\frac{\sum_{i=1}^{m}(x_i - \mu_m)^2}{m-1}}$$

This dramatically reduces the storage required, proportional to the diversity of past traffic, relatively independent of the number of packets that go through p.

Whenever a new packet with trajectory v and old hash a arrives, the new hash value $h(v, a)$ is computed. The new packet is then compared with the mean μ and standard deviation σ of the quad where the claimed source falls in. A belief rating that the packet is from a source within that quad is computed as the probability that $h(v)$ is a sample from a Gaussian distribution with mean μ and variance σ. If the belief rating is "good" (user defined), the hash value is included in the history data. It is to be inserted into the appropriate quadtree node with the standard deviation and mean of that quad node updated. Note that the mean and standard deviation can be updated without requiring the original hash values. Suppose the new hashed value is x_{m+1}. Then the mean is updated as

$$\mu_{m+1} \leftarrow \frac{m \cdot \mu_m + x_{m+1}}{m+1}$$

and the standard deviation is updated as

$$\sigma_{m+1} \leftarrow \sqrt{\frac{1}{m}[\sigma_m^2(m-1) + x_{m+1}^2] + \mu_m^2 - \mu_{m+1}^2}$$

In the training phase of the protocol, the hash values are kept and the quadtree node further sub-divides itself if its standard deviation becomes too large. After the training phase the hash values are discarded and only the mean and standard deviation of each quad are kept.

As we alluded to in section 5.4.5, the quad size (i.e. depth in the tree) determines the confidence of a belief rating. A large quad size usually means we have little data from a particular region. Therefore, we have less confidence in the belief rating we assign. Conversely, a small size means that we have more knowledge, and can assign a belief with greater confidence.

In our implementation, the belief rating itself is the number of standard deviations a new hash value differs from the mean of hash values previously seen. Typically, a hash value 4 standard deviations from the mean results in a very poor rating. The confidence value we use is the depth of the quad node used in computing the belief. The deeper a node in the tree is the more fine-grained information about the area in question we have, thus a higher confidence in our beliefs. Of course, other strategies can be employed here.

To summarize, the data structure is a quadtree with each quad recording the mean and standard deviation of the hashed values for a particular region of the network. We do not keep the entire path observed in the history, not even the hashed values. Once a new packet comes in, we compare it against the history, generate a belief, and update the history.

5.7 Overhead Analysis

Communication. The number of belief ratings that is propagated with packets is network specific, but in usual scenarios we estimate 5-6 such values (5-6 bytes total). Other protocol-related information each packet carries is the location of the previous node on the path (2 bytes), the cryptographic digest used to protect the belief ratings (6 bytes) and the hash value computed by the previous node on the path (2 bytes). This would yield a total of about 15 bytes. Given that TinyOS packets in modern applications range anywhere from 36 to 100 bytes [3], this is certainly acceptable, especially in hostile deployment scenarios where such assurances are required.

In practice we further reduce the overhead by attaching beliefs in a probabilistic way. Only an adaptively small fraction of packets are randomly selected to

carry beliefs. Alternately, only a small fraction of relay nodes attach belief ratings. Moreover, in a streaming application scenario, spanning multiple packets, only the first (header) packet in a data stream will need to carry this additional information—thus amortizing the overhead over the entire sequence.

Storage. The storage overhead is determined by the granularity of history traffic patterns. As we do not explicitly store all the hashed values of packets that visit a node, but rather only the mean and standard deviation of the values inside each quad, this results in a limited overhead. The amount of data a particular node stores is dependent on a its location within the network. A node near the center might be involved with messages passing through from multiple directions, while a node on the periphery will see data originating from only a few directions. This impacts the degree to which a node is able to group hashes of similar areas together.

For example, a maximum depth of the quadtree of 4 proved sufficient in our simulations to provide a partition of $1/256th$ of the network, a very detailed division of the sensor field. For a node at the center of the network potentially a full quadtree of depth 4 will be needed, with $4^4 = 256$ leaf nodes. A node on the periphery with only detailed information on half of the network, and no detail on the other half may have only $2 * 4^3 = 128$ leaf nodes. Similarly when nodes only route data to and from a few (1 or 2) sinks, the quadtrees of all nodes won't have more than 128 partitions. Each leaf node requires 3 bytes to store the mean (1 byte), standard deviation (1 byte), and the number of hashed values (1 byte). For a quadtree with 128 leaves, this requires 386 bytes of memory. Moreover, we can further reduce this overhead by adaptively considering lower depths in (un-interesting sub-branches of) the quadtrees, depending on available memory.

5.8 Experimental Results

We validated our model by simulation with networks of various sizes, ranging from 50 to 1000 nodes. We considered geographic routing (GPSR [50]) to route messages between nodes and data sinks. We note however that our solution is not

hard-coded to GPSR and works with other routing protocols. To model link and node failures, links and nodes are set to go offline at various times – links were active only 85% of the time and nodes 95% on average. Unless where otherwise noted, we used a single sink located at the center of the network, at $(x_{w/2}, y_{w/2})$, where w is the width of the network.

The network was first trained to understand the traffic pattern, by assuming a short interval of uncompromised traffic. In this interval, nodes normally pass messages to the sink, while observing routed traffic and building the hash history. We were then able to test the effect of moving one node to another location, and compare how the distance moved relates to our ability to detect it.

5.8.1 Model Validation

We first evaluated the path metric and the utility of locality sensitive hashing. As we expect, the further away the claimed location of a node is from its true location, the worst its belief rating. To this end, nodes' claimed locations are varied with respect to their true location and the behavior of the hash values is observed. Figure 5.3 illustrates that the further away the claimed location of a node is, the greater the change to its hash value. The figure plots the change in hash value with respect to the change in path "distance". The x-axis shows the difference between the honest path and dishonest path by computing their "distance" as previously defined — by summing the distances between all the sample points of the 2 paths. For this simulation a network of 50×50 was used with a communication radius of 10 and a sampling rate of 100. A node was randomly selected at distance at least d from the sink, d being the width of the network. False locations were acquired by randomly choosing a direction and calculating the coordinates of the new claimed location, based on the magnitude of the claim. The strong linear correlation shows that our distance measure between paths truthfully captures the severity of the wrong location claim.

Using hash values, each node along the path forms a belief as to the honesty of the packet that comes its way, by comparing its hash value with the hash values

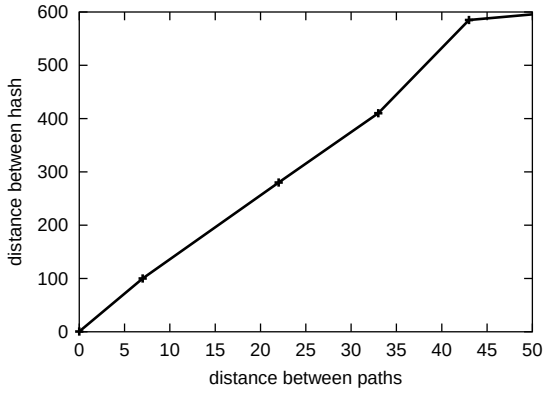


Figure 5.3. As the distance between claimed and true location increases so does the difference between resulting hash values.

in the history for the region. The beliefs formed will be slightly different for all nodes on the route. Normally the beliefs are stronger near the start of the path because the change in the path is more significant at the earlier nodes than those further away; in other words, since the start nodes are looking at a shorter path the percentage of the claim is greater.

Figure 5.4 shows the actual belief ratings formed by each node along a 6 hop path from source to sink. Values are shown for when the adversary claims to be at positions from 0 to 35 units away away from its real location (or 0 to 4 hops as the nodes have a radius of 10) in a network of 50×50. As can be seen, the belief quickly drops once the adversary makes a claim more than 1 hop away.

5.8.2 Parameter Fine-tuning

There are a number of parameters used in our protocol that interestingly have little or no effect on the overall belief rating generated. The power of using locality sensitive hashing with history data overwhelms other network factors.

We mentioned how the hash function requires a random vector of size $2k$ where k is the degree of parametrization of the path ("sampling parameter"). Figure 5.5 shows that even a low sampling parameter will result in quite accurate belief ratings. There is no clear distinction between a value k ranging from 20 to 2000,

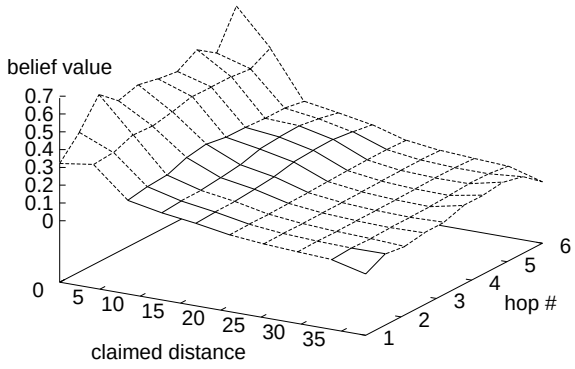


Figure 5.4. The beliefs generated by each node along the path, as the adversary claimed distance increases.

only the sampling parameter of 10 appears insufficient. This is important, because, by using a lower parametrization we reduce the overhead of the hash function computation. This graph only gives a snapshot of what one node "believes". A node on the path from source to sink was arbitrarily selected (in this case the node at hop 4) and it's beliefs were plotted, this being the reason for the routing irregularities seen in the figure.

In addition, we have found that network size and density do not noticeably impact the efficacy of our certification system. Again, this is because the hash values give overwhelmingly good indicators as to the honesty of a node, regardless of network specifics. Figure 5.6 shows belief ratings for three typical network topologies, namely (i) an area of 50×50, $n = 100$, $R = 10$, $deg = 10$, (ii) 100×100, $n = 500$, $R = 10$, $deg = 14$, (iii) 500×500, $n = 1000$, $R = 40$, $deg = 17$ — where n is the number of nodes, R the communication radius and deg the average node degree.

The set of simulations indicate a number of interesting observations that are useful for parameter tuning in practice.

- For sufficiently good belief ratings, only 5 nodes on the relay path are required to attach their beliefs. This substantially reduces communication

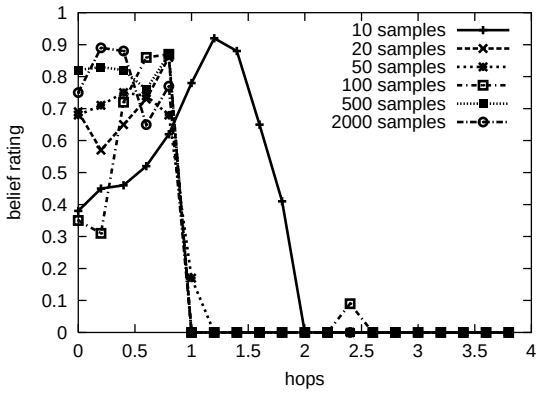


Figure 5.5. Belief values (as a function of hop-distance of claimed vs. true location) using various samplings of the paths. This shows that we can reduce the overheads of hash function computation by using a lower parametrization.

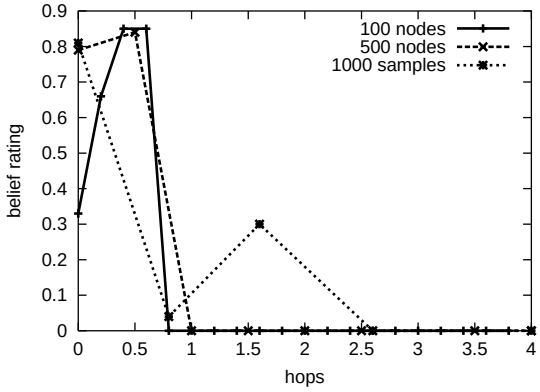


Figure 5.6. Beliefs under generated by the following networks: (i)50×50, $n = 100$, $R = 10$, (ii) 100×100, $n = 500$, $R = 10$ (iii) 500×500, $n = 1000$, $R = 40$

overhead and validates the stochastic belief propagation as only a small fraction of nodes need to participate.

- The number of samples (in hashing) for a path can be as low as 20.
- With the above parameters, the belief rating of a packet from a source node that claimed to be at a location as close as a single hop away, decreases quickly toward zero.

5.8.3 Detection of malicious claims

Figure 5.7 illustrates the percentage of packets accepted by the data sink after examining their belief values and comparing them with a "threshold value" for acceptance. Specifically, the sink will require the average of the lowest 3 beliefs on the path to be above the threshold to be accepted. The graph also shows the percentage of honest nodes accepted (the distance between claimed location and the true location is 0). Having a high threshold of 80% is too strict, and some honest packets are therefore incorrectly dropped. For this simulation we incorporated variability into the routing pattern by having only 85% of the links be active at any given time (thus the routes taken by the packets from the same source vary).

The results show that having only a small percentage of nodes attach beliefs is sufficient for a surprisingly strong detection ability. For example, by just having 5 beliefs attached, the detection accuracy is 95%.

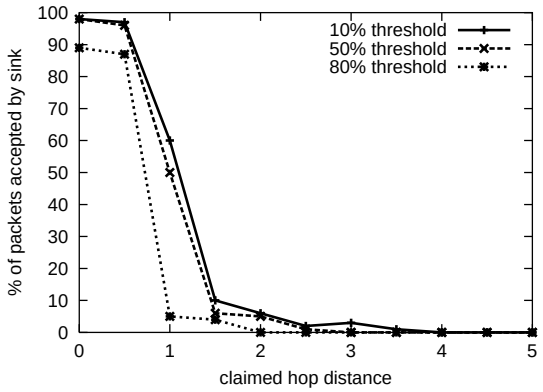


Figure 5.7. The percentage of the number of packets accepted by the sink node with respect to distance claim of node in terms of the number of hops away it is from true location. The beliefs received at the sink must be above the given threshold value to be accepted.

Part III

Conclusion and Future Work

Chapter 6

Conclusion

We have presented here a couple of instances where effective solutions to tough problems for sensors networks can be devised using the collective involvement and intelligence of the network. We are aware that individually the sensors are primitive entities, but as a whole the network is quite powerful and smart. When designing solutions to tough problems we should not think in terms of how can individual limited sensors accomplish the task at hand, but rather how can we take advantage of the collective power and intelligence of the network. All to often, with the former mindset, solutions rely on the use of some external entity that has more capabilities than a typical node. But this approach is inelegant and increases the complexities of the algorithm and, as a consequence, leads to complications in network operation.

Often network-wide solutions are avoided because of the overhead cost of radio transmissions among nodes. However, when such schemes are actually devised we find the overhead is very reasonable, requiring only a constant number of transmissions.

We demonstrate both for localization and for location verification that by instituting a hierarchical structure for collaboration we can design effective algorithms. With localization we devise a scheme of performing the task iteratively. It is iterative in that first a small subset of 'landmark' nodes are embedded and then the entire network; and it is iterative in the actual selection of the landmark nodes,

selecting 2 initially and then incrementally selecting more as deemed necessary. Using such an approach leads to new ideas and results in the geometric properties of such networks and offers guarantees on its performance.

Similarly, with regard to location certification, no special tamper-proof hardware is used to ensure honesty in location reporting. Nor are special designated nodes with long-range radios to monitor righteous network activity used. Rather all nodes work together to detect malicious activity in their midst.

With this philosophy we can build better algorithms for the massively distributed computer systems of the future.

Chapter 7

Future Work

In general the theme of this work is to consider the network as one large entity and address challenges by using the collective knowledge and capabilities of the network as a whole.

With regard to localization there a few possible directions to explore in a 3D environment. In the 3D version we make the assumption that the nodes lie on a monotonic surface; that is, any vertical line will intersect the terrain at most once. This guarantees that by always embedding a component so that it does not lie above or below another component we will stay true to the real representation. However, in theory the algorithm can still be applied to non-monotonic surfaces, by first dividing the network into monotonic segments. Then each segment can be embedded independently and then finally the segments can be glues together. There are multiple possibilities how to carry this out in practice. We may imagine a scenario where nodes have a long-range line-of-sight detection abilities (perhaps) by using radar and can thereby detect if it lies directly above or below other nodes in the network.

Also for the 3D version we assume that each node knows its height information using an altimeter. While such altitude sensor are cheap and can be equipped on each node, it would be interesting to explore how to achieve similar results by equipping a small subset of the nodes with altimeters. For instance we may only require the landmark nodes to have altimeters thereby allowing us to construct an

accurate frame with which to embed the rest of the network. In order to ensure that each landmark has an altimeter, we can modify the algorithm that so that when a candidate node is selected to be a landmark, the closest node to it with an altimeter is designated to be a landmark.

As far as embedding is concerned there are many research problems left open. While our work gives theoretical guarantees on bounding the landmark density and proving that in they would result in a rigid delaunay complex to make embedding possible, there are still practical issues that must be dealt with regard to how to deal with the fact that the distance measurements are only approximations so they distort the final result. How to deal with such distortions is not entirely clear. Firstly, it is clear that distortions arise from a number of different parameters such as the network density and the communication model. And for each of these parameters there is an optimal density of landmarks which may or may not be known when the localization algorithm is put into practice. Understanding the interplay between all these factors is left for further study. Secondly, there are various approaches on how to mitigate the effect of the rough hop-count estimates. One idea may be to run the embedding algorithm from different starting points and then overlaying the results. This circumvents the issue of very bad results from the cumulative effect of mild distortions propagating through the network. Other options may include incorporating various rubberbanding schemes to distribute errors evenly throughout the graph.

With regard to landmark selection algorithm there is a great degree of variability in the order of the conditions to check for in making a new landmark. Depending on which conditions are checked for first could effect the end result in the number of landmarks selected. Also, how many nodes should be made into a landmark simultaneously? This then leads into a number of practical issues when running this algorithm on a real distributed network where a primal goal is to minimize the number of transmissions. The question now is how to organize the algorithm so that the number of transmissions is minimized while still keeping the efficacy of the algorithm unharmed.

Bibliography

[1] Greenorbs project. http://greenorbs.org/.

[2] H. S. Abdelsalam and S. Olariu. A 3d-localization and terrain modeling technique for wireless sensor networks. In *FOWANC '09: Proceedings of the 2nd ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing*, pages 37–46, New York, NY, USA, 2009. ACM.

[3] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. *Mobile Data Management (MDM)*, 2007.

[4] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60:125–135, 1998.

[5] N. Amenta and R. K. Kolluri. Accurate and efficient unions of balls. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, pages 119–128, New York, NY, USA, 2000. ACM.

[6] B. D. O. Anderson, P. N. Belhumeur, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, and Y. R. Yang. Graphical properties of easily localizable sensor networks. *Wireless Networks*, 2007.

[7] J. Aspnes, D. Goldenberg, and Y. R. Yang. On the computational complexity of sensor network localization. In *The First International Workshop on*

Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS), pages 32–44, 2004.

[8] M. Badoiu, E. D. Demaine, M. T. Hajiaghayi, and P. Indyk. Low-dimensional embedding with extra information. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 320–329, New York, NY, USA, 2004. ACM Press.

[9] M. H. Bateni, E. D. Demaine, M. T. Hajiaghayi, and M. Moharrami. Plane embeddings of planar graph metrics. *Discrete Comput. Geom.*, 38(3):615–637, 2007.

[10] A. R. Berg and T. Jordán. A proof of connelly's conjecture on 3-connected generic cycles. 2002.

[11] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 46–54, 2004.

[12] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 46–54. ACM Press, 2004.

[13] R. Bott and L. Tu. *Differential Forms in Algebraic Topology*. Springer-Verlag, 1982.

[14] J. Bruck, J. Gao, and A. Jiang. MAP: Medial axis based geometric routing in sensor networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 88–102, August 2005.

[15] S. Cabello, E. D. Demaine, and G. Rote. Planar embeddings of graphs with specified edge lengths. *Journal of Graph Algorithms and Applications*, 11(1):259–276, 2007.

[16] S. A. Camtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical Report TR-05-07, Rensselaer Polytechnic Institute, March 2005.

[17] S. Capkun and J.-P. Hubaux. Securing position and distance verification in wireless networks. Technical Report EPFL/IC/2004-43, 2004.

[18] S. Capkun and J. P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proceedings of Infocom*, Miami, FL, USA, March 2005.

[19] G. Carlsson and V. de Silva. Topological approximation by small simplicial complexes. In *Proceedings of the Symposium on Point-Based Graphics*, June 2004.

[20] S. Cheung, B. Dutertre, and U. Lindqvist. Detecting denial-of-service attacks against wireless sensor networks. Technical Report Technical Report SRI-CSL-05-01, Computer Science Laboratory, SRI International, May 2005.

[21] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.

[22] V. de Silva. A weak definition of Delaunay triangulation. Technical report, Stanford University, October 2003.

[23] T. K. Dey. *Curve and Surface Reconstruction.* Cambridge University Press, 2007.

[24] H. Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge Univ. Press, 2001.

[25] J. Elson. *Time Synchronization in Wireless Sensor Networks.* PhD thesis, University of California, Los Angeles, May 2003.

[26] T. Eren, D. Goldenberg, W. Whitley, Y. Yang, S. Morse, B. Anderson, and P. Belhumeur. Rigidity, computation, and randomization of network localization. In *Proceedings of IEEE INFOCOM*, 2004.

[27] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *Proc. of the 24th Conference of the IEEE Communication Society (INFOCOM)*, volume 1, pages 339–350, March 2005.

[28] Q. Fang, J. Gao, and L. J. Guibas. Landmark-based information storage and retrieval in sensor networks. In *The 25th Conference of the IEEE Communication Society (INFOCOM'06)*, April 2006.

[29] S. P. Fekete, M. Kaufmann, A. Kröller, and N. Lehmann. A new approach for boundary recognition in geometric sensor networks. In *Proceedings 17th Canadian Conference on Computational Geometry*, pages 82–85, 2005.

[30] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2004.

[31] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.

[32] S. Funke. Topological hole detection in wireless sensor networks and its applications. In *DIALM-POMC '05: Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, pages 44–53, New York, NY, USA, 2005. ACM Press.

[33] S. Funke and C. Klein. Hole detection or: "how much geometry hides in connectivity?". In *Proc. ACM Symposium on Computational Geometry (SCG)*, 2006.

[34] S. Funke and N. Milosavljević. Guaranteed-delivery geographic routing under uncertain node locations. In *Proceedings of IEEE INFOCOM 2007*, 2007.

[35] S. Funke and N. Milosavljevic. Network sketching or: "how much geometry hides in connectivity? - part II". In *18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[36] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, 2003.

[37] J. Gao, L. Guibas, S. Oudot, and Y. Wang. Geodesic delaunay triangulation and witness complex in the plane. In *Proc. 18th ACM-SIAM Sympos. on Discrete Algorithms*, 2008.

[38] D. Goldenberg, P. Bihler, M. Cao, J. Fang, B. D. Anderson, A. S. Morse, and Y. R. Yang. Localization in sparse networks using sweeps. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 110–121, 2006.

[39] D. Goldenberg, A. Krishnamurthy, W. Maness, Y. R. Yang, A. Young, A. S. Morse, A. Savvides, and B. Anderson. Network localization in partially localizable networks. In *Proceedings of IEEE INFOCOM*, 2005.

[40] O. Goldreich. *Foundations of Cryptography.* Cambridge University Press, 2001.

[41] J. E. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity.* Graduate Studies in Math., AMS, 1993.

[42] B. Hendrickson. Conditions for unique graph realizations. *SIAM J. Comput.*, 21(1):65–84, 1992.

[43] A. Howard, M. J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, October 2001.

[44] Y. C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM*, 2003.

[45] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM Press.

[46] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. pages 56–67, 2000.

[47] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *J. Comput. Phys.*, 137(2):346–365, 1997.

[48] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.

[49] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003.

[50] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[51] S. G. Kobourov, A. Efrat, D. Forrester, and A. Iyer. Force-directed approaches to sensor network localization. In *8th Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2006.

[52] A. Kroeller, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *to appear in SODA2006*, 2005.

[53] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha. Resilient localization for sensor networks in outdoor environments. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 643–652, Washington, DC, USA, 2005. IEEE Computer Society.

[54] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engineering Math.*, (4):331–340, 1970.

[55] L. Lazos and R. Poovendran. SeRLoc: secure range-independent localization for wireless sensor networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 21–30, New York, NY, USA, 2004. ACM Press.

[56] M. Li and Y. Liu. Rendered path: range-free localization in anisotropic sensor networks with holes. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 51–62, New York, NY, USA, 2007. ACM.

[57] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, pages 91–98, 2005.

[58] H. Luo, Z. Guo, W. Dong, F. Hong, and Y. Zhao. Ldb: Localization with directional beacons for sparse 3d underwater acoustic sensor networks. *Journal of Networks*, 5(1), 2010.

[59] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *ACM SenSys 2009*, November 2009.

[60] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *SenSys '04: Proceedings of the*

2nd international conference on Embedded networked sensor systems, pages 50–61, New York, NY, USA, 2004. ACM Press.

[61] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *IEEE GLOBECOM*, pages 2926–2931, 2001.

[62] D. Niculescu and B. Nath. *Ad hoc* positioning system (APS) using AOA. In *IEEE INFOCOM*, volume 22(1), pages 1734–1743, 2003.

[63] P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, 2002.

[64] N. Priyantha, A. Chakraborthy, and H. Balakrishnan. The cricket location-support system. In *In Proc. of MOBICOM*, Boston, MA, USA, August 2000.

[65] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. Technical Report TR-892, MIT LCS, 2003.

[66] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM Press, 2003.

[67] K. Sanzgiri, B. Dahill, B. Levine, and E. Belding-Royer. A secure routing protocol for ad hoc networks. In *International Conference on Network Protocols (ICNP)*, November 2002.

[68] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *In Proc. of WISE 2003*, 2003.

[69] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in *ad-hoc* networks of sensors. In *Proc. 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001)*, pages 166–179, Rome, Italy, July 2001. ACM Press.

[70] A. Savvides, H. Park, and M. B. Srivastava. The n-hop multilateration primitive for node localization problems. *Mob. Netw. Appl.*, 8(4):443–451, 2003.

[71] A. Savvides and M. B. Strivastava. Distributed fine-grained localization in ad-hoc networks. *IEEE Transactions of Mobile Computing*, 2003.

[72] J. B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.

[73] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 201–212, 2003.

[74] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 201–212, New York, NY, USA, 2003. ACM Press.

[75] A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[76] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22):2319–323, December 2000.

[77] W. T. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 13(52):743–768, 1963.

[78] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. manuscript.

[79] B. Waters and E. Felten. Secure, private proofs of location. Technical Report TR-667-03, Princeton University, January 2003.

[80] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.

[81] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 28, Washington, DC, USA, 2003. IEEE Computer Society.

[82] L. Zhang, X. Zhou, and Q. Cheng. Landscape-3d; a robust localization scheme for sensor networks over complex 3d terrains. *Local Computer Networks, Annual IEEE Conference on*, 0:239–246, 2006.

[83] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138, New York, NY, USA, 2004. ACM Press.

[84] X. Zhu, R. Sarkar, and J. Gao. Shape segmentation and applications in sensor networks. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, 2007.