Geometry in Wireless Sensor Networks
In-network Information Processing and Localization

A Dissertation

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Yang Yang

Fall 2013

Geometry in Wireless Sensor Networks
In-network Information Processing and Localization

Yang Yang

APPROVED:

DEDICATION

This dissertation is dedicated to my father, Xiaomin Yang and my mother, Yanwei Yu.

ACKNOWLEDGMENTS

The work described here could not have been finished without help and support from others. The following words cannot express the depth of my appreciation. First, I would like to express my gratitude to my advisor, Dr. Miao Jin. Without her outstanding guidance and continuous support throughout my doctoral studies, I could not have finished this work. I also would like to thank Dr. Hong-Yi Wu, Dr. Nian-Feng Tzeng and Dr. Magdy Bayoumi for their valuable supervision of my academic study and research. Second, I would like to thank all of the CACS faculty for the wonderful knowledge they taught in classes and all of the CACS staff for their help in my academic career. I also would like to thank all of my friends and colleagues for their help, especially Ning Ding, Su Xia, Zhe Cui, Zhixun Zhou, Hongyu Zhou, Xiyue Xiang, Buri Ban, Zhipeng Yang, Yao Zhao, Yang Liu, Ting Ning, Yanyan Han, Rajesh Prasad, Ruochi Zhang, Yi Wang and Ran Zhang. Finally, I would like to give thanks to my parents for their help, encouragement, and sacrifice. They always supported me without considering themselves. Without their support, I would not be able to reach such a great milestone in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Wireless Sensor Network

Wireless sensors are devices equipped with different types of modules, such as temperature, odorimeter, barometers, light and so on. These sensors can be used to collect the measured data from the environments and communicate with each other through the wireless interfaces. Some of them could be very small and simple and some of them could be computational powerful. These wireless sensors can form a network and work with each other to solve various problems. Nowadays, wireless sensors can be adopted in many areas. For example, they can be used underwater for detection, they can be distributed in the forest to monitor the environments, they can be deployed on the mountain to study the activity of the volcanoes, or they can even be equipped in the human body to monitor the blood pressure. Wireless sensor networks have experienced an explosive growth in recent years. In comparison with earlier computer communication systems, the unique and intrinsic challenge in sensor networking is distributed and scalable computation and communication.

1.2 Geometry in Wireless Sensor Networks

It is obvious that the location of sensors could be on a two-dimensional plane, a three-dimensional ocean or a three-dimensional mountain. For the communication models, we can extract some graphs from the sensor network communication models such like the unit disk graph. For sensing model, the signal of sensors could be broadcast or directional such that the sensing model of sensors could be a disk or a sector. In fact, geometry is everywhere in the wireless sensor network field. As long as we could identify some geometric problems from the sensor network studies, we could probably provide some efficient algorithms and useful tools from the geometry world to help the network issues.

CHAPTER 2: DISTRIBUTED INFORMATION STORAGE AND RETRIEVAL IN GENERAL 3D WIRELESS SENSOR NETWORKS

Distributed in-network data-centric processing aims to reduce energy consumed for communication and establish a self-contained data storage, retrieval, aggregation, and query sensor system which focuses more on the data itself rather than the identities of the individual network nodes. Double-ruling based schemes support efficient in-network data-centric information storage and retrieval, especially for aggregated data, since all data with different types generated in a network can be conveniently retrieved along any single retrieval curve. Previous double-ruling based research focuses on two-dimensional (2D) wireless sensor networks where a 2D planar setting is assumed. With increasing interests in deploying wireless sensors in three-dimensional (3D) space for various applications, it is urgent yet fundamentally challenging to design double-ruling based approach in general 3D sensor networks because double-ruling based schemes in general have much harder geometric constraints than other distributed in-network data-centric processing schemes.

In this chapter, we propose a geographic location free double-ruling based approach for general 3D sensor networks with possibly complicated topology and geometric shapes. Without the knowledge of the geographic location and the distance bound, a query simply travels along a simple curve with the guaranteed success to retrieve aggregated data through time and space with one or different types across the network. Extensive simulations and comparisons show the proposed scheme with low cost and a balanced traffic load.

2.1 Background

Wireless sensor networks have experienced an explosive growth in recent years. In comparison with earlier computer communication systems, the unique and intrinsic challenge in sensor networking is distributed and scalable computation and communication. In particular, an individual sensor is highly resource-constrained, with extremely limited computing, storage, and communication capacities. On the other hand, however, the target applications often require large-scale deployment where the amount of data generated, stored and transmitted in the network grow proportionally with the network size. This dilemma renders the conventional sensor networking strategy that intends to transmit all sensor data to an external server impractical. To this end, distributed in-network data-centric storage and retrieval have been extensively discussed in the literature. The new paradigm of distributed in-network data-centric processing focuses more on data themselves rather than the identities of individual sensor nodes. Data are uniquely named and data processing is achieved using data names instead of network addresses, aiming to establish a self-contained data acquisition, storage, retrieval, and query system.

While a two-dimensional (2D) planar setting has been assumed in most earlier studies of in-network data storage and retrieval, there have been increasing interests in deploying wireless sensors in three-dimensional (3D) space for such applications as underwater reconnaissance and atmospheric monitoring. Several explorative 3D sensor network testbeds have been developed recently (either in space or underwater). Although they are all in relatively small size, we foresee large-scale deployment will soon be demanded in the near future.

This research focuses on in-network data-centric information storage and retrieval in large-scale three-dimensional (3D) sensor networks. We first summarize existing in-network data-cebtric storage and retrieval algorithms for two-dimensional (2D) networks, and then discuss the challenges in 3D networks, followed by an overview of our proposed approach.

2.1.1 Overview of Distributed Information Storage and Retrieval Algorithms

Geographical hash table is a general approach for in-network data-centric storage and retrieval. A basic geographical hash table based scheme hashes a datum by its type into geographic coordinates and stores at the sensor node geographically nearest to such coordinates. Queries apply the same hash table with the desired type to retrieve data from the storage node. Delivery of the data is implemented by geographic routing, such as GPSR. To reduce bottleneck at the hash nodes and improve data survivability under node failure, a geographical hash table based scheme applies a structured replication with multiple mirrors scattered in the network. Structured replication reduces the cost of storage but increases the cost of queries.

Different from geographical hash table based schemes, a double-ruling based scheme works as follows. A datum (or a pointer to the datum) is duplicated along a curve called replication curve, and a query travels along another curve called retrieval curve. Successful retrieval is guaranteed if the retrieval curve intersects the replication curve. A simple double-ruling scheme on a planar grid is illustrated in Figure 2.1 where nodes are located at lattice points. The replication curves follow the horizontal lines and the retrieval curves follow the vertical lines. By traveling along a vertical line, a data query, called information consumer, can always find the requested data generated by an information producer.

Double-ruling based schemes support efficient data retrieval, since all data with different types generated in a network can be conveniently retrieved along any retrieval curve. This is

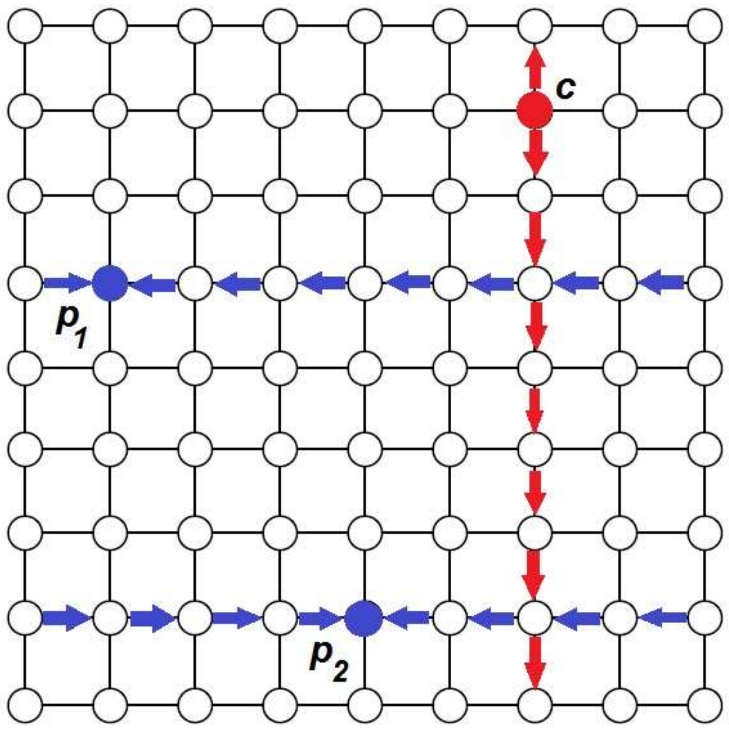


Figure 2.1: A simple double-ruling scheme on a 2D grid sensor network.

in a sharp contrast to geographical hash table based schemes where an information consumer has to visit multiple nodes scattered in the network to collect data with different types hashed to various locations. Moreover, with modestly increased data replication, a double-ruling based scheme has well balanced load across the network, while nodes near the hashed location suffer much higher traffic load than others in a geographical hash table based scheme. A double-ruling based scheme also has better fault tolerance against geographically concentrated node failure by replicating data on nodes that are uncorrelated with node proximity.

Double-ruling based schemes achieve all the desired properties at the cost of more data duplication and much stronger geometric constraints on the shape of a sensor network than geographical hash table based schemes. Previous double-ruling based schemes either assume networks with 2D grid shape or with heavy data replication to achieve high probability that the retrieval curve would meet one of the replication curves within the sensor network. To extend double-ruling scheme to networks with uneven sensor distribution and irregular geometric shapes, landmark-based scheme is proposed to partition the sensor field into tiles. GHT is adopted at the tile level, i.e., a data type is hashed to a tile instead of a single node. Inside each tile, a double-ruling scheme is applied to ensure the intersection of a retrieval path and a replication path. Later, a location-free double-ruling scheme is introduced in the year of 2007 based on boundary recognition and the computation of the respective gradient fields. To improve the flexibility of retrieval, a spherical projection-based double-ruling scheme is proposed in the year of 2006, where a planar network is mapped to a sphere based on the inverse of stereographic projection. Both the replication and retrieval curves are great circles such that a retrieval curve always intersects all other replication circles.

2.1.2 Challenges in 3D Networks

Although double-ruling has shown highly effective for distributed information storage and retrieval in 2D sensor networks, it cannot be efficiently applied in 3D networks. A naive double-ruling based scheme in 3D sensor networks is shown in Figure 2.2. In such a 3D grid-based cube-shape sensor network, data replication and retrieval are along the horizontal and vertical planes respectively, such that a retrieval plane intersects all replication planes. Besides an extremely high cost of data replication, such 3D grid-based double-ruling scheme requires a network with a regular cube shape and uniform node distribution. Recently, a volumetric parametrization based double-ruling scheme is introduced in LuoSECON 2012. They map a 3D sensor network to a cube and assign each node a virtual coordinates. Their method require the network shape topologically equivalent to a cube. However, many practical 3D sensor networks are topologically different from a cube. Figure 2.3(a) shows a

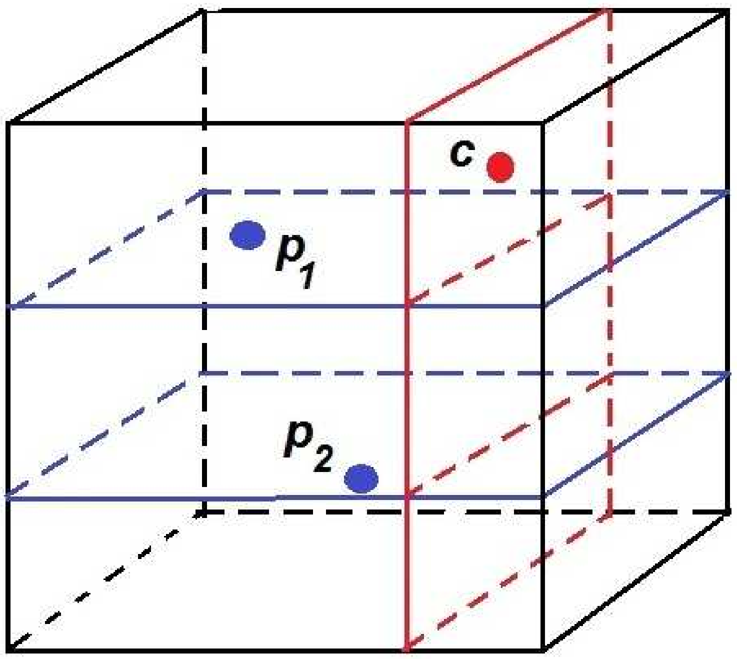


Figure 2.2: A simple double-ruling scheme on a 3D grid sensor network.

3D sensor network with sensors deployed underwater around an island. The topology of the network is equivalent to a donut with a handle as shown in Figure 2.3(b). Figure 2.3(c) gives another example of a 3D sensor network with multiple coverage holes inside.

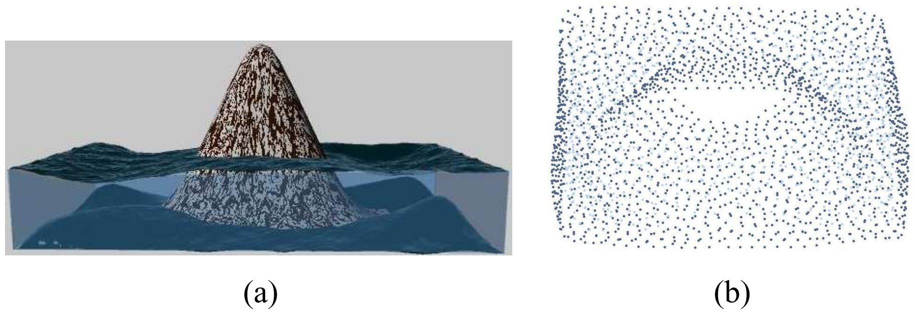


Figure 2.3: A 3D sensor network model.

Another challenge of double-ruling in 3D sensor networks is the delivery of data and query to the mapped geo-localizations for in-network data storage and retrieval. Previous GHT and double-ruling based schemes on 2D sensor networks rely on geographical routing

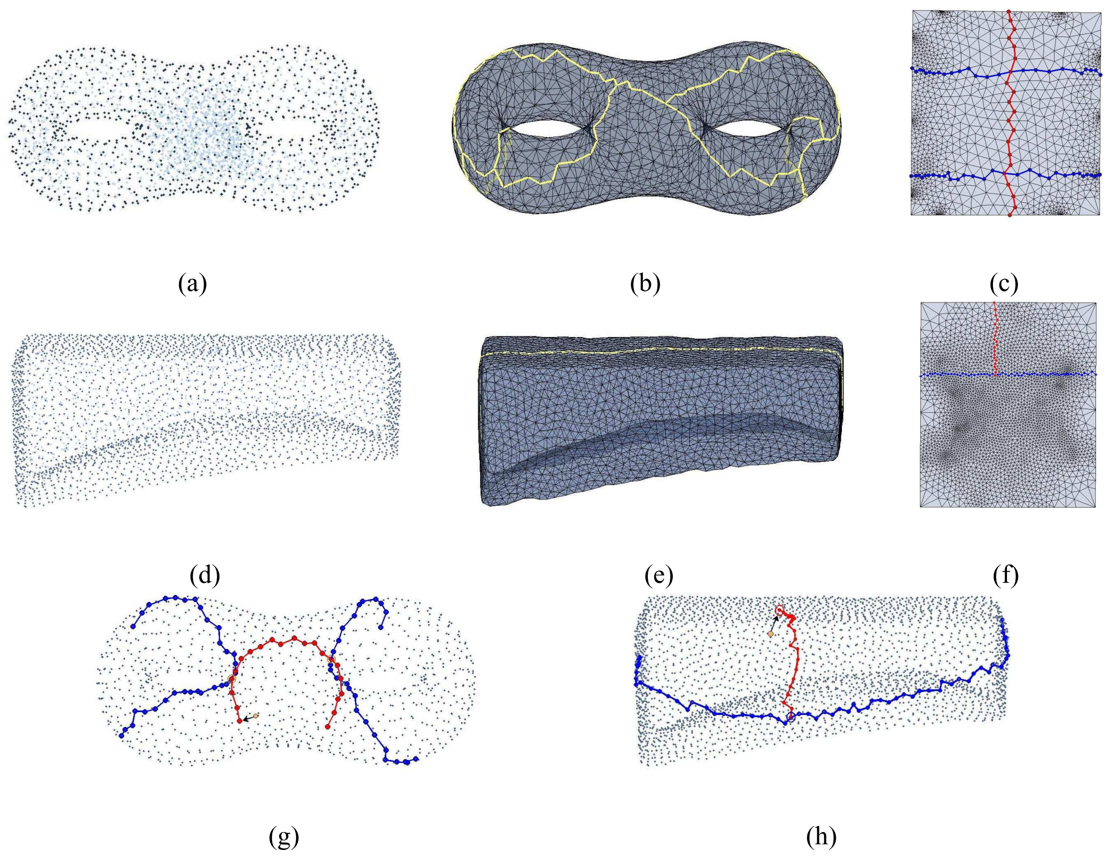


Figure 2.4: Network Models with different topologies.

schemes, such as GPSR. They require location information. However the cost to equip GPS receiver at each node are greatly exacerbated in a 3D sensor network due to the dramatically increased sensor quantity in order to cover a 3D space compared with its 2D counterpart. Some application scenarios even prohibit the reception of satellite signals by part or all of the sensors, rendering it impossible to solely rely on global navigation systems. Moreover, geographic routing with local information only that can guarantee constant storage at each node is non-trivial and even impossible in general 3D sensor networks.

2.1.3 Our Approach

The proposed approach is motivated by a topology concept that any closed surface can be cut open to a topological disk along an appropriate set of edges called a cut graph of the surface. Two examples given in Figure 2.4 briefly illustrate the basic idea of our approach. The first example is a 3D volumetric sensor network model with two handles (see Figure 2.4 (a)). We first detect its boundary nodes and build a triangular structure of the identified boundary surface (see Figure 2.4 (a)). We also compute a cut graph of the boundary surface which is marked with yellow color in Figure 2.4 (b). We cut the boundary surface open to a topological disk along the cut graph and then map it to an aligned planar rectangle such that each boundary node is associated with a planar rectangle virtual coordinates (see Figure 2.4 (c)). Each non-boundary sensor stores the ID of its neighbor nearest to the boundary. A data generator follows the sequence of IDs to the boundary, and then travels along a horizontal line of the virtual planar rectangle and leaves data copies. The two horizontal lines marked with blue shown in Figure 2.4 (c) correspond to the two data replication curves shown in Figure 2.4 (g) marked with the same color. similarly, a consumer follows the sequence of IDs to the boundary and collects the aggregated data of different types along a vertical line. The vertical line marked with red shown in Figure 2.4 (c) corresponds to the data aggregation curve shown in Figure 2.4 (g) marked with the same color. The second example is also a 3D sensor network deployed under water (see Figure 2.4 (d)). We cut its boundary surface open to a topological disk along a computed cut graph (see Figure 2.4 (e)) and then assign each boundary node a planar rectangle virtual coordinates (see Figure 2.4 (f)). Similarly, Figure 2.4 (h) gives a data query example.

The proposed cut graph based double-ruling approach works for 3D sensor networks with general topology and geometry shapes. Without the knowledge of the geographic location and the distance bound, the success of data retrieval is always guaranteed because a pair of horizontal and vertical lines surely intersect. Retrieval of aggregated data through time and space with different types is also guaranteed. A consumer travels along a vertical line and then collects all desired information in the network because the vertical line intersects all horizontal lines - replication curves of the network. The proposed algorithm has no constraints on communication models and is fully distributed. Each node only needs to exchange information with its direct neighbors. The amount of extra information stored at individual nodes is constant and small. Simulation results show the proposed approach with low cost and a balanced traffic load.

2.2 Cut Graph of Boundary Surface

Given a sensor network deployed in 3D volume, we first need to detect its boundary nodes and extract a triangulated boundary surface of the 3D volume. Note that the boundary surface of a 3D volume is a closed surface (i.e., without any holes). We apply an existing boundary detection algorithm to identify boundary nodes of a 3D volume sensor network and the algorithm proposed in triangulation to extract a triangular structure of the detected boundary nodes. Both algorithms are fully distributed, with no constraints on communication models. They require no GPS information, but distance information within one-hop neighborhood only. Although both methods can tolerate distance measurement errors, we don't require accurate detection of all boundary nodes, since we only need a connected triangular structure to approximate the boundary surface of the 3D volume network. We even allow some mistakenly detected non-boundary vertices on the triangular structure.

2.2.1 Topological Background

A surface is orientable if it has two distinct sides. General surfaces in real world are orientable surfaces. A loop is a continuous function of the circle on surface. Two loops are homotopic if there is a continuous deformation from one loop onto the other on surface.

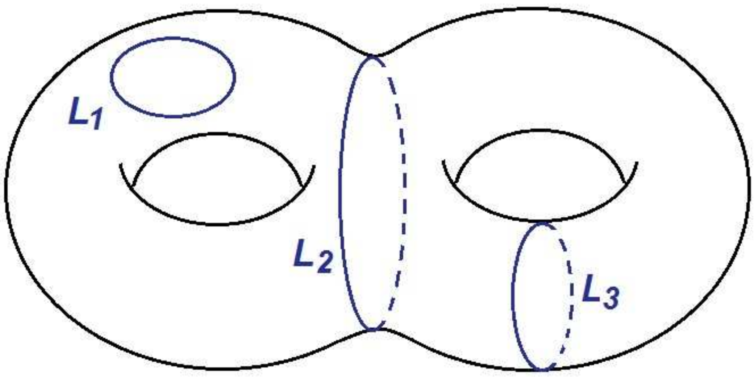
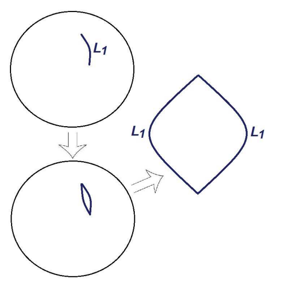


Figure 2.5: Loops on a genus two eight surface.

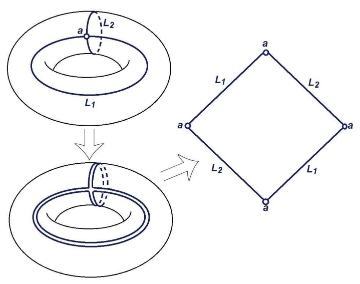
Denote M a connected and orientable surface embedded in 3D, and L a loop on M. L is contractible if it is homotopic to a constant (a loop which can shrink to a single point) as shown in Figure 2.5 with L_1; otherwise it is non-contractible. L is surface separating if it can be expressed as the symmetric difference of boundaries of topological disks embedded in surface as shown in Figure 2.5 with L_1 and L_2; otherwise it is non-separating as shown in Figure 2.5 with L_3. Any non-separating loop is a non-contractible loop; similarly, any contractible loop is a separating loop.

The genus of M is the maximum number of disjoint non-separating loops $L_1, L_2, \cdots, L_g$ in M; that is, any L_i and L_j have no topological intersection if $i \neq j$, and $M \backslash (L_1 \cup \cdots L_g)$ is connected. The genus number is the most basic topology information of a surface and equals to the number of handles. For example, a disk and a sphere have a genus 0, and a torus has a genus 1.

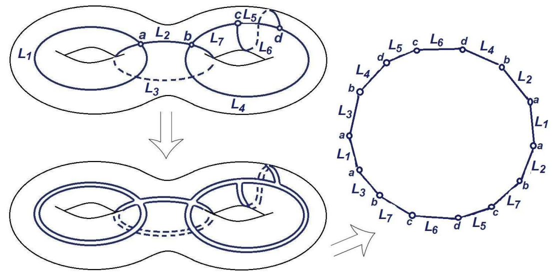
Any closed surface M (e.g., a surface without a boundary) can be opened into a topological disk D (e.g., a surface with one boundary) by cutting along an appropriate set of edges called cut graph. Denote G a cut graph of M. Each edge of G appears twice on the boundary of D, and we can obtain M by gluing together these corresponding boundary edges



(a) Network I.



(b) Network II.



(c) Network III.

Figure 2.6: Cut graph on different models.

of D. Figure 2.6 shows cut graphs of a genus 0, a genus 1, and a genus 2 surfaces respectively. The three closed surfaces are cut open to topological disks along the given cut graphs. Denote g the genus number of M, the number of base loops of a cut graph is $2g$.

2.2.2 Cut Graph of Boundary Surface

The triangulated boundary surface of a 3D sensor network is connected, orientable, and closed. We abuse the symbol M to denote a connected and orientable triangulated surface embedded in 3D. Specifically, we denote $M = (V, E, F)$ a triangulated surface embedded in $3D$, consisting of vertices V, edges E, and triangle faces F. Denote $v_i \in V$ a vertex with id i; $e_{ij} \in E$ an edge with two ending vertices v_i and v_j; $f_{ijk} \in F$ a triangle face with vertices v_i, v_j, and v_k.

The problem of computing the shortest cut graph of a general topology surface proves the problem NP-hard and provides a polynomial-time approximation algorithm. Canonical systems of loops, a system of $4g$ (g is the genus number of the surface) loops with a single base vertex to cut a surface into a disk. Erickson and Whittlesey provide a greedy algorithm to compute a shortest system of such canonical loops.

Instead of adopting the above optimization algorithms, we propose a fully distributed two-step algorithm, aiming to balance the size of the cut graph measured by the number of edges and the computation cost and the communication cost measured by the number of exchanged messages. The basic idea of the first step is to grow triangles with the width first way. At each step of the growing, all the marked triangles always form a topological disk and the marked edges form the boundary of the disk. After all the triangles have been marked, we can cut the closed surface into a topological disk along the marked edges. The size of the cut graph can be largely reduced by trimming away those unnecessarily marked edges at the second step for trimming. For topological sphere surfaces, there is no marked edges left after trimming, which provides a convenient way to automatically identify the topology of a 3D sensor network.

2.2.2.1 Computing the Cut Graph

The algorithm starts from one randomly chosen triangle f_{ijk} of M, which can be the one with the smallest node id. f_{ijk} marks itself and its three edges e_{ij}, e_{jk}, and e_{ki}. Each of the marked edges checks whether it is shared by two marked triangles. For example, edge e_{ij} finds its neighboring triangle f_{jil} unmarked. e_{ij} then removes mark from itself but adds mark on triangle f_{jil} and edges e_{il} and e_{lj}. Note that it is possible that e_{il} or e_{lj} may have been marked already. The propagation algorithm stops when all the triangles of M have been marked. Let all the marked edges be G, which form a cut graph of M. We prove the correctness of the algorithm as follows.

Theorem 2.1. *A closed surface M can be cut into a topological disk D along the cut graph G computed by the above algorithm.*

Proof. We can prove it by way of induction. The algorithm starts from one triangle of M with its edges marked. This single triangle is topologically equivalent to a disk with boundary edges marked. We denote it as D_1. After the $i-1$ steps, $i-1$ triangles have been marked and added. We have D_{i-1}. Suppose D_{i-1} is a topological disk with boundary edges marked. At the step i, an unmarked triangle is identified, which means this triangle does not belong to D_{i-1} but must share at least one of the boundary edges of D_{i-1}. By adding this triangle into D_{i-1} and updating the marked edges as the way described by the algorithm, the newly formed D_i is still a topological disk. Marked edges form the boundary of D_i. When all the triangles of M have been marked, D_m has included all the triangles and is still a topological disk with its boundary edges marked. The surface M can then be cut open to a topological disk $D = D_m$ along the marked edges by splitting each marked edge and its two ending vertices to two. ∎

Theorem 2.2. *The cut graph computed by the above algorithm is connected.*

Proof. This can be easily proved by way of contradiction. If the computed cut graph is disconnected, then the boundary of D_m is disconnected. It is impossible that the boundary of a topological disk is disconnected. ∎

2.2.2.2 Trimming

If we cut a closed surface M open along the cut graph computed by the above propagation algorithm, the boundary of the topological disk surface D would be extremely zigzagged. The size of the boundary can be $\Omega(m)$, where m is the number of triangles of M. The reason is that in the worst case the size of the marked edges is increased by one each time when one triangle of M is marked.

To control the size of the cut graph, we need to trim away those unnecessarily marked edges. Marked edges forming non-segmenting loops of M are necessary because the loops correspond to the cut open of each handle. While for those marked and dangling tree edges which do not belong to or connect any loops, M can still be cut open to a topological disk after removing them from the cut graph.

The algorithm of trimming is straightforward. Each marked edge checks its two ending vertices whether they connect to other marked edges. If one of its two ending vertices does not connect to any other marked edges, the edge is identified as a dangling tree edge and can be unmarked - removed from the cut graph. The unmarked edge will then send messages to its neighboring marked edges through the other ending vertex. Its neighbors then conduct the same checking when receiving the message. The trimming process stops when there is no marked, dangling tree edges. Note that it is impossible that the two ending vertices of a marked edge do not connect to any other marked edges, because we proved that the marked edges are connected in Theorem 2.2.

Theorem 2.3. *The removal of marked, dangling tree edges does not disconnect the cut graph.*

Proof. Each dangling tree edge connects to G with only one ending vertex when it is removed. So the removal of dangling tree edges does not disconnect G. ∎

We have the following Theorem which said that all marked edges will be unmarked at the end of the trimming process for topologically sphere surface M.

Theorem 2.4. *After the trimming process, there is no marked edge left for a topological sphere surface M ($g = 0$).*

Proof. We first use way of contradiction to prove the cut graph of a topological sphere surface M computed by the above propagation algorithm is a tree. Considering the fact that M is topologically equivalent to a sphere, every loop on M is contractible, therefore surface separating, If there exist a loop in the computed cut graph of M, the loop will separate M to two disconnected parts. This contradicts what we have proved in Theorem 2.1: at each step i of the propagation, D_i is always a topological disk. So the computed cut graph of a topological sphere surface M should be a tree without loops. Each marked edge will be identified as dangling tree edge and removed eventually by the proposed trimming algorithm. There will be no marked edges left at the end of the trimming process. ∎

For a boundary surface detected topologically equivalent to a sphere, we conduct a simple flooding on the boundary surface to find a pair of boundary nodes with the longest shortest path (hops) on the surface. We then cut the surface open along the shortest path between the pair of nodes.

2.3 Generating Planar Rectangle Virtual Coordinates

After we virtually cut the outside boundary surface of a network M to a topological disk D along the computed cut graph. We then apply discrete surface Ricci flow to compute the planar rectangle virtual coordinates of boundary nodes.

2.3.1 Discrete Surface Ricci Flow

To briefly introduce the concept of discrete surface Ricci flow, we start from the definitions of circle packing metric and discrete Gaussian curvature. Given a topological disk triangulated surface $D = (V, E, F)$, We assign each v_i a circle with radius γ_i and denote the radius function $\Gamma : V \to \mathbb{R}^+$. For each edge e_{ij}, the two circles at v_i and v_j intersect with an acute angle ϕ_{ij}. We call ϕ_{ij} the *weight* of e_{ij}, and denote the weight function $\Phi : E \to [0, \frac{\pi}{2}]$.

Definition 2.1 (Circle Packing Metric)**.** A circle packing metric of D includes Γ and Φ.

Denote l_{ij} the length of e_{ij}. l_{ij} can be computed from γ_i, γ_j and ϕ_{ij} from the following cosine law:

$$l_{ij}^2 = \gamma_i^2 + \gamma_j^2 + 2\gamma_i\gamma_j \cos\phi_{ij}. \tag{2.1}$$

Discrete Gaussian curvature measures how curved a discrete surface is embedded in $\mathbb{R}^3$.

Definition 2.2 (Discrete Gaussian Curvature)**.** Denote θ_i^{jk} the corner angle attached to v_i belonging to f_{ijk}, ∂D the the boundary of D, and K_i the discrete Gaussian curvature at vertex v_i. K_i can be computed as the angle deficit at v_i:

$$K_i = \begin{cases} 2\pi - \sum\limits_{f_{ijk}\in F} \theta_i^{jk}, & v_i \notin \partial D; \\ \pi - \sum\limits_{f_{ijk}\in F} \theta_i^{jk}, & v_i \in \partial D. \end{cases} \tag{2.2}$$

Discrete surface Ricci flow deforms the initial circle packing metric such that the final circle packing metric induces edge lengths satisfying the target Gaussian curvatures.

2.3.2 Computing Planar Rectangle Virtual Coordinates

A planar rectangle has zero Gaussian curvature everywhere except its four corner points with Gaussian curvature $\frac{1}{2}\pi$. So we uniformly pick four vertices along the boundary of the topological disk triangulated surface D and assign their target Gaussian curvatures: $\overline{k_i} = \frac{1}{2}\pi$. For all other vertices of D, we assign: $\overline{k_i} = 0$.

We initialize a circle packing metric on D such that each circle associated with a vertex has a unit radius, i.e., $\gamma_i = 1, u_i = \log \gamma_i = 0$ for each v_i, and $\phi_{ij} = \frac{\pi}{2}$ for each e_{ij}. Discrete surface Ricci flow deforms the circle packing metric on D such that the induced edge lengths from final circle packing metric satisfy our pre-defined Gaussian curvatures. Isometric embedding of D on plane based on the computed edge lengths generates a planar rectangle mapping of D. The mapping is diffeomorphism that provides planar rectangle virtual coordinates for vertices of D. The detail of the algorithm is as follows:

1. Initialization of circle packing metric: For each v_i, $u_i = 0$. For each e_{ij}, $\phi_{ij} = \frac{\pi}{2}$,

2. Compute edge length for each e_{ij}: $l_{ij} = e^{u_i} + e^{u_j}$;

3. Compute each θ_i^{jk} according to the law of cosines:

$$\theta_i^{jk} = cos^{-1} \frac{l_{ij}^2 + l_{ki}^2 - l_{jk}^2}{2 l_{ij} l_{ki}};$$

4. Compute current Gaussian curvature k_i for each v_i as Equation 3.2.

5. Denote ϵ a threshold and set to $1e-4$. If all $|\overline{k_i} - k_i| < \epsilon$, the algorithm goes to the next step; otherwise, $u_i = u_i + \delta(\overline{k_i} - k_i)$, where δ is a small constant and set to 0.1, and the algorithm goes back to step 2.

6. Isometric embedding: Denote p_i the planar coordinates of each v_i. Start from a boundary edge e_{ij} with v_i one of the four chosen boundary vertices: we assign $p_i = (0, 0), p_j = (l_{ij}, 0)$. In a breadth first search way, if f_{ijk} has exactly two vertices (e.g., v_i and v_j) with planar coordinates (e.g., p_i and p_j), compute p_k as one intersection point of two circles centered at p_i and p_j with radii l_{ik} and l_{jk} respectively, and satisfying $(p_k - p_i) \times (p_j - p_k) > 0$. Repeat the above process until every vertex has its planar coordinates. The planar rectangle is automatically aligned with x-axis.

Note that each boundary node only needs to exchange information with its direct neighbors when implementing the above algorithm.

2.4 Implementation

In this section, we implement every part of the proposed algorithm by using our own simulators.

2.4.1 Data Replication

Since a network is location free, we let each non-boundary node store the ID of its neighbor nearest to the boundary of the network. A producer follows a sequence of nodes to the nearest boundary node denoted as p. The boundary surface of the network has been mapped to a virtual planar rectangle, so each boundary node has a planar rectangle virtual coordinates. We assume a data replication curve is along a horizonal line of the virtual planar rectangle. The horizontal line through p is unique, solely determined by the y coordinate of the planar rectangle virtual coordinates of p. The producer travels and leaves pointers or copies of the data at nodes along the line with two directions - one with the increased and the other with the decreased x values. At each step, the producer simply checks the planar rectangle virtual coordinates of its one range neighbors and chooses the one with the closest distance to the line and along the current direction. Once finishing data replication, the producer turns back and follows the reversed path back.

2.4.2 Data Retrieval

Without the aware of the knowledge of the producer's location and the distance, a consumer follows a sequence of nodes to the nearest boundary node denoted as p. We assume a data retrieval curve is along a vertical line of the virtual planar rectangle. A vertical line passing through p is determined solely by the x coordinate of the planar rectangle virtual coordinates of p. The consumer simply travels along the line with two directions - one with the increased and the other with the decreased y values. At each step, similarly, the producer simply checks the planar rectangle virtual coordinates of its one range neighbors and chooses the one with the closest distance to the line and along the current direction. The consumer can stop as soon as it hits the replication curve of its desired data. If there are multiple producers and different types of data, the consumer travels along a full vertical line to collect all the aggregated data in the network. Once data has been collected, the consumer turns back and follows the reversed path back.

2.4.3 Delivery of Data and Query

As a preprocessing, each of the boundary nodes sends messages recording its minimum hop count to boundary (initialized to zero) to its neighbors. A non-boundary node receives a message and compares with its current record (initialized to infinity). If the received count has more than one hop count less, the node updates its current one and records the ID of its neighbor sending this message. The node also updates the count of the message and then sends to its neighbors. Otherwise, the node simply discards the message. When there is no message in the network, each of the non-boundary nodes of the network has recorded the ID of its neighbor nearest to boundary. It is then straightforward for a producer or a consumer to travel along the shortest path to the boundary according to the sequences of IDs.

2.4.4 Storage

We have very limited information stored at the nodes of the network. For each of the non-boundary nodes, it only stores the ID of its neighbor nearest to boundary; for each of the boundary nodes, it stores the computed planar rectangle virtual coordinates. For the data replication, we can leave copies of data on either all the nodes along the replication curve; or just a small portion of nodes sampled along the replication curve, which is a trade off between the storage cost and the retrieval cost.

2.4.5 Time Complexity and Communication Cost

We measure the communication cost by the number of messages. Denote the size of all the nodes of a network as n, the size of its boundary nodes as m. We summarize the time complexity and the communication cost of the major steps of the cut graph based double-ruling scheme.

Both the time complexity and the communication cost to compute the cut graph are linear to the size of the boundary nodes of the network, $O(m)$, including the trimming step.

We apply discrete Ricci flow to compute the planar rectangle virtual coordinates. The number of iterations, as shown by Figure 2.7 for the model in Figure 2.4 (a), determines the time complexity of computing the edge lengths, given by $-C\frac{\log \epsilon}{\lambda}$ where C is a constant, ϵ is the threshold of curvature error (set to 1e-4 in our implementation), and λ is the step length of

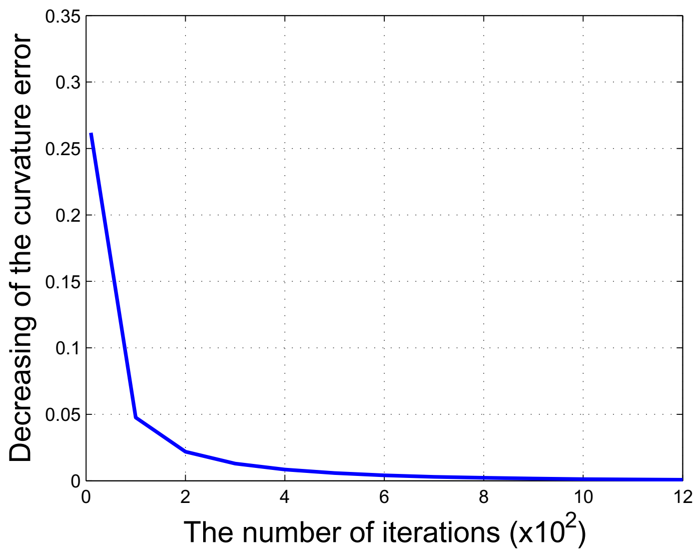


Figure 2.7: The convergence rate of discrete Ricci flow.

each iteration (set to 0.1 in our implementation). Since each vertex only needs to exchange messages with its one range neighbors at one iteration, the corresponding communication cost is $-C\dfrac{\log \epsilon}{\lambda}d'm$ where d' is the average vertex degree of the triangulated boundary surface. With the computed edge lengths, both the time complexity and the communication cost of computing the planar rectangle embedding are linear to the size of the boundary nodes, i.e., $O(m)$.

Let the boundary nodes of the network synchronize among themselves and start to send messages recording the minimum hop count to boundary to its neighbors at roughly the same time, the total communication cost for each non-boundary node to record the ID of its neighbor nearest to boundary is $O(dn)$ where d is the average number of neighbors of each node. The time complexity is $O(n)$.

2.5 Simulations

We evaluate the performance of the proposed location-free cut graph based double-ruling scheme in 3D sensor networks given in Figures 2.3 and 2.4. Specifically, we denote the one given in Figure 2.4 (e) as Network I with 4369 number of nodes and the average number of neighbors of each node 13.79, the one given in Figure 2.3 (b) as Network II with 4298 number of nodes and the average number of neighbors of each node 13.43, the one given in Figure 2.4 (a) as Network III with 6194 number of nodes and the average number of neighbors of each node 13.67, and the one given in Figure 2.4 (a) as Network IV with two inner holes and the average number of neighbors of each node 13.7181. Producer and consumer costs are measured by the number of hop counts they travel to store or retrieve data. Traffic load on each node is measured by the number of messages passing through it. In our simulations, each node has equal probability to be a producer or a consumer.

Note that there are very limited algorithms to compare with because all previous double-ruling based schemes can't work on 3D sensor networks with general topology and geometry shapes including the naive 3D grid-based double-ruling scheme and the volumetric parametrization based double-ruling scheme. Hashing based schemes can tolerate different topology but require Geographic information. Our implementation of the GHT scheme for comparison has actually considered geographic information to design the hash function and stored heavy routing information on each node (shortest path tree rooted at each node) to guarantee the routing path a shortest one from the producer and the consumer to the hashed location, and hence all "improved GHT" approaches won't help to achieve better performance in our comparison.

Table 2.1: Comparison of average producer and consumer costs of single type of data

	Network I			Network II			Network III			Network IV		
	Cut Graph	GHT	SR-GHT	Cut Graph	GHT	SR-GHT	Cut Graph	GHT	SR-GHT	Cut Graph	GHT	SR-GHT
Producer cost	69.3331	22.5738	19.4178	77.6277	23.6468	16.0400	78.1474	24.6926	13.7566	146.7712	41.0066	24.23068
Consumer cost	19.5381	22.8416	94.4715	18.2786	23.8186	93.0521	25.3367	24.7264	101.8150	24.796	41.2531	157.83

2.5.1 Producer and Consumer Costs

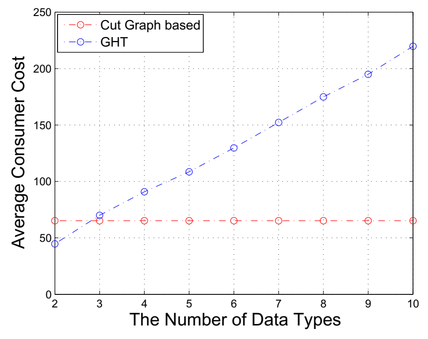
Producers stand for the information producers who generate different types of data. Consumers stand for the information consumers who want to retrieval the data.

2.5.1.1 Single Type of Data

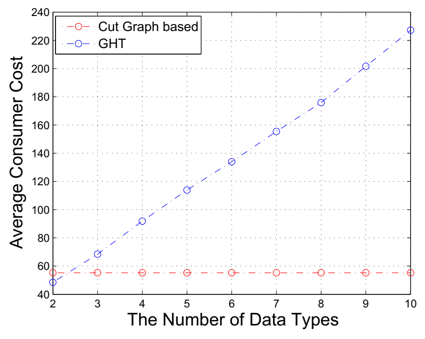
We compare the proposed scheme with GHT one with and without structured replication. For GHT with structured replication (SR-GHT), we apply 1 level hierarchy with extra 3 mirror points scattered in network to store the nearby data. Table 2.1 lists the average producer and consumer costs with one type of data generated in network. For cut graph based scheme, the producer cost is the highest and the consumer cost is the lowest; a producer needs to travel and leave copies of data along the whole replication curve while a consumer can stop immediately when its retrieval curve intersects the replication curve. For SR-GHT scheme, on the contrary, the producer cost is the lowest and the consumer cost is the highest; a producer can store data at the closest location, but a consumer has to travel to both the hashed location and its three mirror points to collect data.

2.5.1.2 Aggregated Data

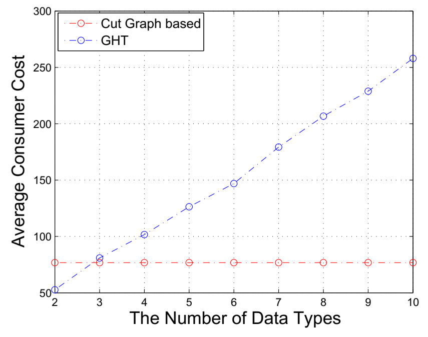
If there are more than one data type in network, the consumer cost of cut graph based scheme is fixed; a consumer collects all different types of data by simply moving along a retrieval line. While the consumer cost of GHT scheme increases proportional to the number of data types; a consumer has to travel to different hashed locations for different types of data. Note that the cost of GHT scheme may decrease because we simply take a round trip to each hashed location in our implementation. But to find a minimum tour to visit all of the locations is the traveling salesman problem, which is NP-hard. The producer cost does not change for either cut graph based and GHT schemes with the increase of data types.



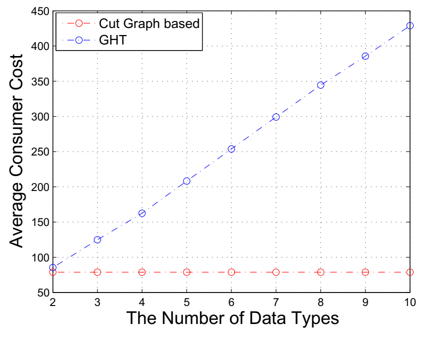
(a) Network I.



(b) Network II.



(c) Network III.

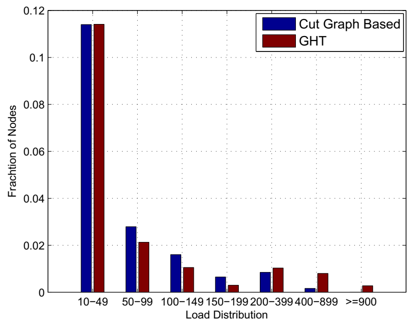


(d) Network IV.

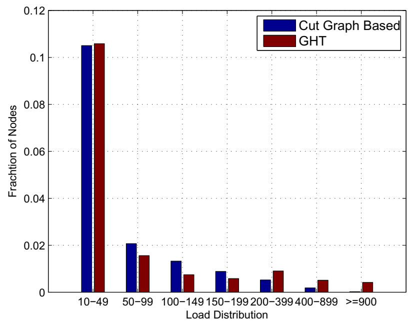
Figure 2.8: Average consumer costs with the increase of data types.

Figure 2.8 clearly shows that cut graph based scheme performs the best for retrieval of multiple types of data generated in network. When there is only one type of data in network, cut graph based scheme and GHT schemes have a tradeoff between the producer and consumer costs.

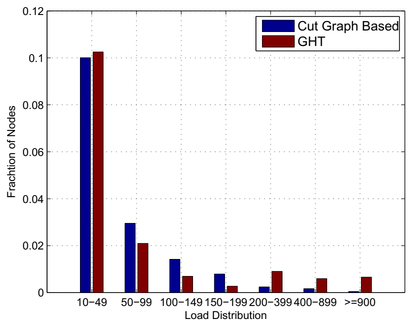
2.5.2 Load Distribution



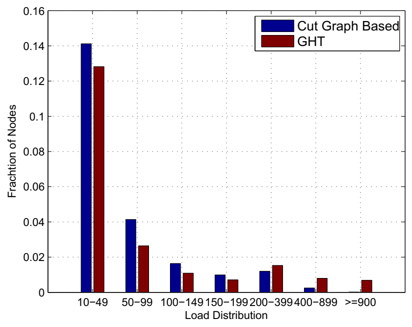
(a) Network I.



(b) Network II.



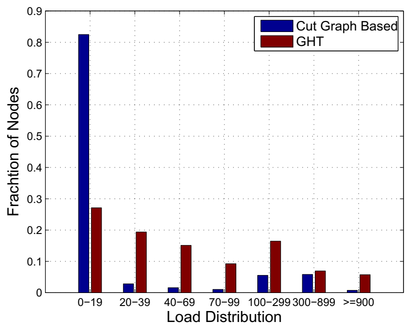
(c) Network III.



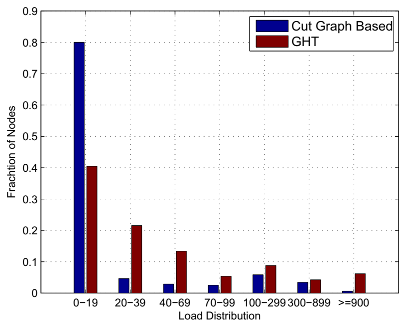
(d) Network IV.

Figure 2.9: Load distribution with one producer and one data type.

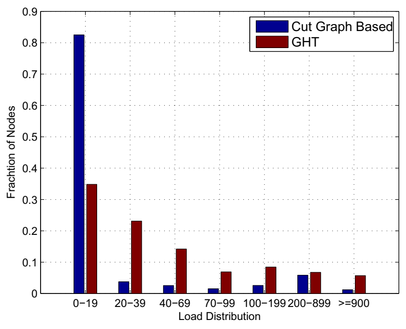
We simulate different scenarios to evaluate the load distribution of cut graph based scheme and compare with GHT scheme. The first scenario is one data producer with one data type in a network. Each node in the network has equal probability to request for data. For both GHT and cut graph based schemes, the load on the majority of the nodes are within a small number. Specifically, the load on roughly 83% of nodes in Network I, roughly 84% of nodes in Network II and Network III, and roughly 79% of nodes in Network IV are below 10. Figure 2.9 shows the distribution of high traffic load on the remaining nodes. For GHT scheme, nodes near the hashed location suffer much higher traffic load; while for cut graph



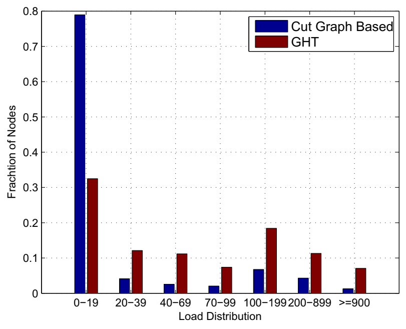
(a) Network I.



(b) Network II.



(c) Network III.



(d) Network IV.

Figure 2.10: Load distribution with one hundred producers and ten data types.

based scheme, boundary nodes take a little bit more traffic load since the load has been evenly distributed among the boundary. The node suffering the highest traffic has a load of 4368 with GHT scheme and 813 with cut graph scheme for Network I; a load of 4297 with GHT scheme and 140 with cut graph scheme for Network II; a load of 6193 with GHT scheme and 981 with cut graph scheme for Network III: and a load of 14077 with GHT scheme and 996 with cut graph scheme for Network IV.

The second scenario is one hundred data producers with ten data types in a network. We randomly choose the data producers from the network. Each node in the network has equal probability to request for aggregated data. Figure 2.10 shows the distribution of the total traffic load of data storage and retrieval. For GHT scheme, a data consumer has to travel a long path to collect different types of data scattered in a network, which generates high traffic load in the network; while for cut graph based scheme, a data consumer has fixed cost for aggregated data retrieval so that the majority of the traffic load of the network is still low. The node suffering the highest traffic has a load of 10211 with GHT scheme and 1542 with cut graph scheme for Network I; a load of 10540 with GHT scheme and 1404 with cut graph scheme for Network II; a load of 22935 with GHT scheme and 1614 with cut graph scheme for Network III: and a load of 48074 with GHT scheme and 3778 with cut graph scheme for Network III.

2.5.3 Tradeoff Between Storage Cost and Consumer Cost

As discussed in Section 2.4, it is a tradeoff between the storage cost and the consumer cost for the proposed cut graph-based double-ruling approach for information storage and retrieval in general 3D sensor networks. Figure 2.11 shows clearly that the average consumer cost drops dramatically when the percentage of nodes stored with a copy of the data along the data replication curve increases from 10% to 40%, and then decreases slowly when the

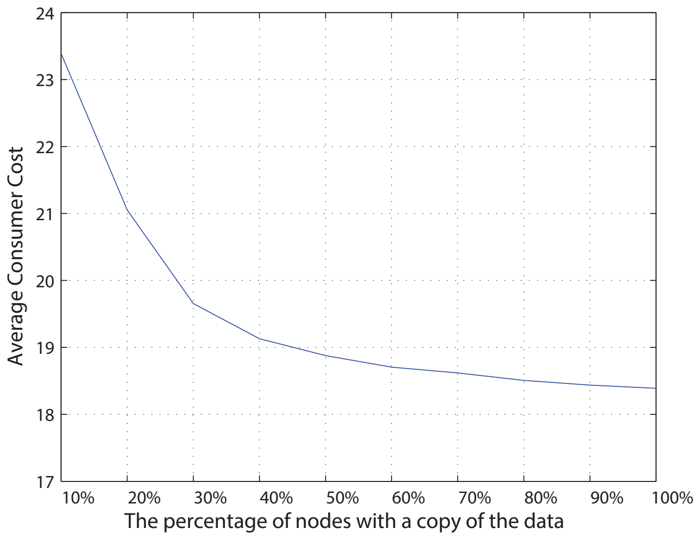


Figure 2.11: Tradeoff between storage cost and consumer cost.

percentage of nodes stored with a copy of the data is over 50% for Network II model. A balance between the storage cost and the consumer cost would be to store copies of data at half of the nodes along the data replication curve.

2.6 Discussions

In this section, we have three different scenarios to show how they could effect the overall performance of our algorithm.

2.6.1 Network Model

The proposed solution doesn't require any global position information of a network. It has no constraints on communication models of the network either. The network only requires one-hop neighborhood distance information to detect boundary nodes and construct a triangular structure at the pre-processing step.

The proposed cut-graph computation algorithm is independent of the complexity and irregularity of a 3D volume where a set of sensor are deployed, because the boundary surface of a 3D volume is always a closed surface. The algorithm cuts a closed surface with any geometric shape or topology to a topological disk and then virtually map it to a planar rectangle. It is possible that a 3D network degenerates. One example is that a 3D volume sensor network degenerates to a 3D surface network. The algorithm can simply be applied to the surface network directly which is not necessarily closed. Another example is that part of a 3D network degenerates to a single line, neither a volume nor a surface. We can apply 3D network segmentation algorithm to identification the bottleneck and then segment the network to parts. Double-ruling approach can be applied at individual parts.

2.6.2 Network Dynamics

As discussed in Sec. 2.4.5 , both the time complexity and communication cost of the proposed cut graph-based double-ruling approach are dominated by computing the planar rectangle virtual coordinates of the boundary surface using discrete surface Ricci flow. For a network with possible nodes' failures, we don't need to restart the computation of Ricci flow each time a sensor node runs out of its battery. We only need to replace a dead boundary node with its nearest active sensor node. The process can be triggered by nodes with dead communication to one common node. They conduct a local flooding to find one node nearest to the dead one. Note that this new one is not necessarily a boundary node. Denote this new node v_i. v_i initializes its $\gamma_i = 1, u_i = \log \gamma_i = 0$. v_i and its direct neighbors recompute the weights of edges neighboring to v_i. Discrete surface Ricci flow continues till the convergence.

2.6.3 Distance Sensitivity

It is preferred that the retrieval cost of a consumer for one type of data is related with its distance to the location where the data is produced. Such distance sensitivity is however not guaranteed in our algorithm. It is possible that one consumer is geographically close to one producer in a 3D volume network, but the consumer has to go a longer distance to hit the data replication curve by the producer. Such case can happen when the consumer and the producer locate at two sides of the medial axis of the 3D volume network. The reverse is also possible that the consumer can travel a shorter distance to hit the data replication curve than its real distance to the producer.

2.7 Summary

We present a location-free cut graph based double-ruling approach for 3D sensor networks with general topology. An information consumer simply travels along a simple curve with the guaranteed success to retrieve aggregated data through time and space with different types across the network. We conduct extensive simulations and comparisons that further show the proposed approach with low cost and a balanced traffic load.

CHAPTER 3: 3D SURFACE LOCALIZATION WITH TERRAIN MODEL

The majority of current research on sensor network localization focuses on wireless sensor networks deployed on two dimensional (2D) plane or in three dimensional (3D) space, very few on 3D surface. However, many real world applications require large-scale sensor networks deployed on the surface of a complex 3D terrain. Compared with planar and 3D network localizations, surface network localization exists unique and fundamental hardness.

In this research, we explore 3D surface network localization with terrain model. A digital terrain model (DTM), available to public with a variable resolution up to one meter, is a 3D representation of a terrain's surface. It is commonly built using remote sensing technology or from land surveying and can be easily converted to a triangular mesh. Given a sensor network deployed on the surface of a 3D terrain with one-hop distance information available, we can extract a triangular mesh from the connectivity graph of the network. The constraint that the sensors must be on the known 3D terrain's surface ensures that the triangular meshes of the network and the DTM of the terrain's surface approximate the same geometric shape and overlap. We propose a fully distributed algorithm to construct a well-aligned mapping between the two triangular meshes. Based on this mapping, each sensor node of the network can easily locate reference grid points from the DTM to calculate its own geographic location. We carry out extensive simulations under various scenarios to evaluate the overall performance of the proposed localization algorithm. We also discuss the possibility of 3D surface network localization with mere connectivity and the results are promising.

3.1 Background

A variety of applications in wireless sensor networks require geographic locations of sensor nodes. Instead of equipping each sensor node with a high cost localization hardware such as GPS receiver, different localization algorithms and protocols have been proposed that allow the sensor nodes to derive their own locations.

Current localization research focuses on sensor networks deployed on two-dimensional (2D) plane or in three-dimensional (3D) space. They take distance information as input, and then search the solution space to find coordinates of sensor nodes that preserve the distance matrix as much as possible. Distance between adjacent sensors can be measured by received signal strength (RSS) or time difference of arrival (TDOA), or simply approximated by one-hop radio range. For remote sensors, their distance can be approximated by hop counts of the shortest path.

In real-world applications, many large-scale sensor networks are deployed over complex terrains, such as the volcano monitoring project and ZebraNet. Localization of a network deployed over a 3D surface exists a unique hardness compared with the well-studied localization of a network in 2D or 3D space. Specifically, due to limited radio range, the distance between two remote sensors deployed over a 3D surface can only be approximated by their surface distance, the length of the shortest path between them on the surface. Such surface distance is different from the 3D Euclidean distance of two nodes. A localization algorithm doesn't exist for a network deployed over a 3D surface with surface distance information only, even if we assume accurate range distance measurement available. One intuitive example is that a piece of paper can be rolled to different shapes, but distance between any pair of points on the paper doesn't change. With pure surface distance information, we can never figure out the current shape of the paper. We can also learn the hardness of localization of a network deployed over a 3D surface from differential geometry. Consider that the distance information of a sensor network deployed over a 3D surface approximates the distance information of the surface, there exists no unique embedding in 3D within rigid motions for a general surface with distance only.

In some previous papers, authors assume each sensor node can measure not only distances between its neighboring nodes but also its own height information. They require a sensor network is deployed on a surface with single-value property - any two points on the surface have different projections on plane. Such property ensures that they can project the network deployed over a 3D surface to 2D plane by removing z coordinate without ambiguity. They apply existing 2D network localization method on the projected one to compute the x and y coordinates of each sensor node, and then add the height information back as the z coordinate.

Later, a cut-and-sew algorithm is proposed to generalize the localization algorithm from single-value surfaces to general surfaces. The algorithm takes a divide-and-conquer approach by partitioning a general 3D surface network into a minimal set of single-value patches. Each single-value patch can be localized individually, and then all single-value patches are merged into a unified coordinates system.

However, integrating height measurement into every sensor of a network is not always practical and affordable, especially for a large-scale sensor network. The motivation of this work is to explore the possibility of localization of a network deployed over surfaces with one-hop distance information only or even just mere connectivity, if we have the information of the deployed terrain surface.

3.1.1 Our Approach

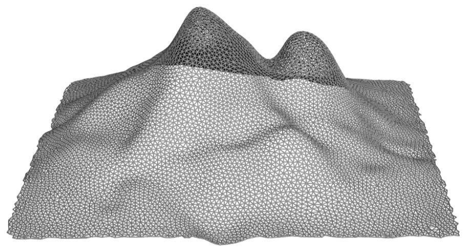
A 3D representation of a terrain's surface is called a digital terrain model (DTM). DTMs are commonly built using remote sensing technology or from land surveying, and are available to public with a variable resolution up to one meter. For example, the Shuttle Radar Topography Mission (SRTM) is a high-resolution digital topographic database that provides DTM data for North and South America with high accuracy and dense coverage. It is expected that acquisitions from radar satellites TerraSAR-X and Tan DEM-X will be available in 2014 to provide a uniform global coverage of DTMs up to 5 m absolute height accuracy at 10 m grid spacing.

A DTM is represented by a grid of squares, where the longitude, latitude, and altitude (i.e., 3D coordinates) of all grid points are known. It is straightforward to convert the gird into a triangulation, e.g., by simply connecting a diagonal of each square. Therefore a triangular mesh of the DTM of a terrain surface can be available before we deploy a sensor network on it. On the other hand, given a wireless sensor network deployed on a terrain surface with one-hop distance information available, a simple distributed algorithm can extract a refined triangular mesh from the network connectivity graph. Vertices of the triangular mesh are the set of sensor nodes. An edge between two neighboring vertices indicates the communication link between the two sensors. The constraint that the sensors must be on the known 3D terrain surface ensures that the triangular mesh of the DTM of the terrain surface overlaps with the triangular mesh extracted from the network connectivity graph. The question is how the latter can be localized in reference to the former.

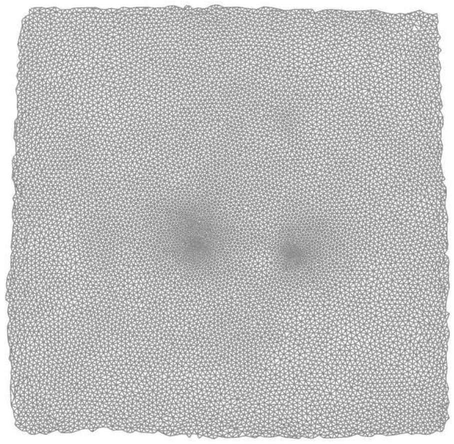
The proposed approach is based on surface conformal structure. Conformal structure is an intrinsic geometric structure of surfaces, determined by surface distance. Conformal structure can tolerate a small local deformation of a surface, so the conformal structure of a surface is consistent even if the surface is approximated by different triangulations with various densities. Surfaces sharing the same conformal structure exist conformal mapping between them. A conformal mapping is a one-to-one and continuous mapping which preserves angles and local shape.

The triangular mesh of the DTM of a terrain surface and the triangular mesh extracted from the connectivity graph of a network deployed over the terrain surface approximate the geometric shape of the same terrain surface. Theoretically, the two triangular meshes share the same conformal structure. We can construct a well-aligned conformal mapping between them. Based on this mapping, each sensor node of the network can easily locate reference grid points of the DTM to calculate its own location.

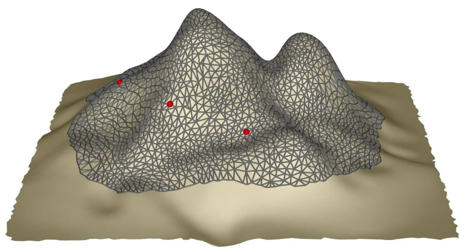
Fig. 3.1 illustrates the basic idea. Fig. 3.1 (a) shows the triangular mesh of the DTM of a terrain surface. Fig. 3.1 (c) shows the triangular mesh extracted from the connectivity graph of a network deployed over the terrain surface. We first compute two conformal mappings, denoted as f_1 and f_2 respectively, to map the two triangular meshes to plane as shown in Figs. 3.1 (b) and (d) respectively. Such mapping exists based on Riemanns theorem that a topological disk surface can be mapped to plane through a conformal mapping. However, the two mapped triangular meshes on plane are not aligned. Three anchor nodes marked with red as shown in Fig. 3.1 (c) are deployed with the network to provide the reference for alignment. Based on the positions of the three anchor nodes, We construct another conformal mapping, denoted as f_3, to align the mapped network triangular mesh with the mapped DTM triangular mesh on plane. Combining the three mappings, $f_1^{-1} \circ f_3 \circ f_2$, induces a well-aligned conformal mapping between the two triangular meshes shown in Fig. 3.1 (a) and (d)



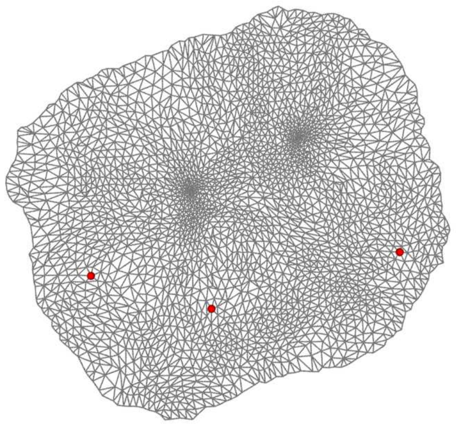
(a)



(b)



(c)



(d)

Figure 3.1: DTM and Sensor Network Triangulation Mesh.

respectively. Based on the well-aligned mapping, each sensor node of the network, a vertex of the network triangular mesh, simply locates its nearest grid points, vertices of the DTM triangular mesh, to calculate its own geographic location. Note that the proposed localization algorithm, theoretically speaking, only requires three anchor nodes for a network with thousands or even tens of thousands of sensor nodes.

3.2 Theoretical Knowledge

Before giving the details of the proposed surface network localization algorithm, we introduce briefly the background knowledge necessary to the algorithm. Specifically, we introduce the concept of discrete conformal mapping and discrete surface Ricci flow, a tool we apply to compute discrete conformal mapping of a triangular mesh from 3D to 2D plane. We then introduce M$\ddot{o}$bius Transformation, a tool we apply to align two planar triangular meshes.

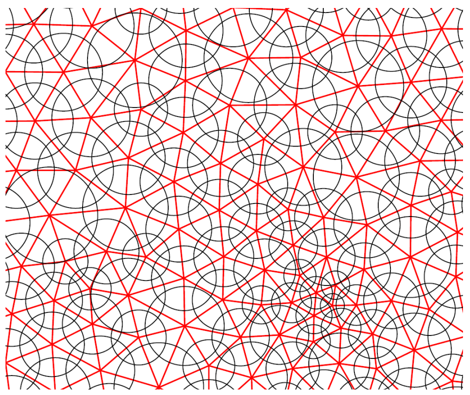
3.2.1 Discrete Conformal Mapping

Intuitively speaking, a conformal mapping is a one-to-one and continuous mapping that maps infinitesimal circles to infinitesimal circles and preserves the intersection angles among the infinitesimal circles.

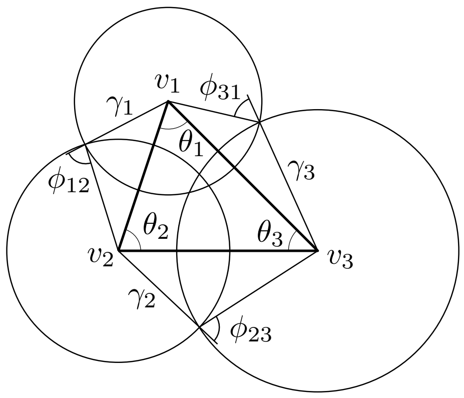
In discrete setting, we denote $M = (V, E, F)$ a connected triangular mesh embedded in $\mathbb{R}^3$, consisting of vertices (V), edges (E), and triangle faces (F). Specifically, we denote $v_i \in V$ a vertex with ID i; $e_{ij} \in E$ an edge with two ending vertices v_i and v_j; $f_{ijk} \in F$ a triangle face with vertices v_i, v_j, and v_k. A boundary edge is defined as an edge shared by one triangle face only. The two ending vertices of a boundary edge are defined as boundary vertices. A non-boundary edge is shared by two triangular faces.

If we use circles with finite radii to approximate infinitesimal circles, we can approximate conformal mapping in discrete setting. It is called circle packing metric. We assign each v_i a circle and denote γ_i its radius. The radius function is $\Gamma : V \to \mathbb{R}^+$. The two circles at v_i and v_j of edge e_{ij} intersect with an acute angle, denoted as ϕ_{ij} and called the *weight* on the edge. The edge weight function is then $\Phi : E \to [0, \frac{\pi}{2}]$.

Denote l_{ij} the edge length of e_{ij}. l_{ij} can be computed from the circle radii of the two



(a)



(b)

Figure 3.2: Circle Packing Metric.

ending vertices γ_i, γ_j and its weight ϕ_{ij} from the cosine law:

$$l_{ij}{}^2 = \gamma_i{}^2 + \gamma_j{}^2 + 2\gamma_i\gamma_j \cos\phi_{ij}. \tag{3.1}$$

Definition 3.1 (Circle Packing Metric)**.** The circle packing metric of a discrete surface M includes the circle radius function and the edge weight function.

From the definition of conformal mapping, a conformal mapping on a discrete surface with circle packing metric modifies the vertex radii, and preserves the edge weights.

3.2.2 Discrete Surface Ricci Flow

Richard Hamilton first introduced Ricci flow in his seminal work in 1982. Chow and Luo proved a general existence and convergence theorem for discrete Ricci flow on surfaces. Jin et al. later provided a series of computational algorithms for discrete Ricci flow on surfaces.

Before we introduce the definition of discrete surface Ricci flow, we need to give definitions of discrete metric and discrete Gaussian curvature first.

Definition 3.2 (Discrete Metric)**.** A discrete metric on M is a function $l : E \to \mathbb{R}^+$ on the set of edges, assigning to each edge $e_{ij} \in E$ a positive number l_{ij} such that all triangles satisfy the triangle inequalities $f_{ijk} \in F$: $l_{ij} + l_{jk} > l_{ki}$.

Edge lengths of M satisfy the triangle inequalities, so they are sufficient to define a discrete metric on M.

Definition 3.3 (Discrete Gaussian Curvature)**.** Denote θ_i^{jk} the corner angle attached to Vertex v_i in Face f_{ijk}, and ∂M the boundary of M, the discrete Gaussian curvature K_i on $v_i \in V$ is defined as the angle deficit at v_i:

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk}\in F} \theta_i^{ij}, & v_i \notin \partial M, \\ \pi - \sum_{f_{ijk}\in F} \theta_i^{jk}, & v_i \in \partial M. \end{cases} \tag{3.2}$$

Since we can compute corner angles directly from edge lengths, a discrete metric solely determines the discrete Gaussian curvature of M.

Definition 3.4 (Discrete Surface Ricci Flow)**.** Discrete surface Ricci flow continuously deforms the circle packing metric of M according to the difference between the current and target Gaussian curvatures in a heat-like diffusion process, and converges when the difference is less then a threshold. The final circle packing metric induces a metric which satisfies the target Gaussian curvatures, and is conformal to the original surface metric.

3.2.3 Mobius Transformation

A complex number $z = a + bi$ defined on a complex plane can be simply considered as a point $p(a, b)$ on plane, where a and b are x and y coordinates of Point p respectively.

Definition 3.5 (M$\ddot{o}$bius Transformation)**.** A M$\ddot{o}$bius transformation is a conformal map between complex plane to itself, represented as:

$$f(z) = \frac{az + b}{cz + d}, \tag{3.3}$$

where a, b, c, d are complex numbers, satisfying $ad - bc = 1$.

If a M$\ddot{o}$bius transformation maps four distinct complex numbers z_1, z_2, z_3, z_4 to four distinct complex numbers w_1, w_2, w_3, w_4 respectively, i.e., four distinct planar points are mapped to another four distinct planar points, the M$\ddot{o}$bius transformation keeps their cross-ratio invariant, represented as:

$$\frac{(z_1 - z_3)(z_2 - z_4)}{(z_2 - z_3)(z_1 - z_4)} = \frac{(w_1 - w_3)(w_2 - w_4)}{(w_2 - w_3)(w_1 - w_4)}. \tag{3.4}$$

Note that all operations including addition, subtraction, multiplication, and division are all defined on complex numbers.

3.3 Surface Network Localization

Given a wireless sensor network deployed on a terrain surface with one-hop distance information available, we can adopt our previous work to extract a refined triangular mesh from the network connectivity graph. Vertices of the triangular mesh are the set of sensor nodes. An edge between two neighboring vertices indicates the communication link between the two sensors. Note the algorithm is fully distributed and has no constraint on communication models.

Denote M_1 the triangular mesh of the DTM of the terrain surface and M_2 the triangular mesh extracted from the connectivity graph of the network. The proposed three-step localization algorithm is fully distributed. We explain each step in detail, specifically, conformal mapping of both M_1 and M_2 to plane, alignment of mapped M_1 and M_2 on plane, and localization of vertices of M_2.

3.3.1 Conformal Mapping to Plane

Given a triangular mesh $M = (V, E, F)$ embedded in $\mathbb{R}^3$, we apply discrete surface Ricci flow to conformally map M to plane. Denote the mapping $f : M \to D \in \mathbb{R}^2$. The mapping result is stored at each v_i as a complex number (i.e., $z = x + yi$), and (x, y) serves as the planar coordinates of v_i.

One fact to consider when designing the mapping algorithm is that the boundary shape of a large-scale sensor network can be random and complicated, and the mapping result should be independent of the boundary shape. So we apply discrete surface Ricci flow with the following free-boundary condition: we assign the target Gaussian curvatures of all non-boundary vertices to zero, and discrete surface Ricci flow only deforms the circle radii of non-boundary vertices. Discrete surface Ricci flow converges when the target Gaussian curvatures of non-boundary vertices equal to zero. i.e., flat. Note that boundary vertices are ending vertices of boundary edges. Boundary edges of M can be easily detected according to the definition that they are shared by only one triangle face.

We first construct an initial circle packing metric (Γ_0, Φ) of M based on edge lengths. Denote γ_i^{jk} the corner radius associated with corner angle θ_i^{jk}. Each v_i computes its corner radii as:

$$\gamma_i^{jk} = \frac{l_{ki} + l_{ij} - l_{jk}}{2},$$

where l_{ij}, l_{jk}, l_{ki} represent the distance measurements of edges e_{ij}, e_{jk}, e_{ki}, respectively. Then v_i computes its initial circle radius γ_i by averaging its corner radii:

$$\gamma_i = \frac{1}{m} \sum_{f_{ijk} \in F} \gamma_i^{jk},$$

where m is the number of the adjacent faces to v_i (i.e., the vertex degree of v_i). For each edge e_{ij}, we compute its edge weight ϕ_{ij}, i.e., the intersection angle of the two circles centered at v_i and v_j with radii γ_i and γ_j respectively based on the Euclidean cosine law:

$$\cos \phi_{ij} = \frac{l_{ij}^2 - \gamma_i^2 - \gamma_j^2}{2\gamma_i\gamma_j}.$$

With the constructed initial circle packing metric, in each iteration of discrete surface Ricci flow, only non-boundary vertices update their circle radii. Specifically, each non-boundary v_i exchanges its current $u_i = log\gamma_i$ with its direct neighbors and updates its adjacent edge lengths $\{l_{ij}|e_{ij} \in E\}$ according to Eqn. 3.1. With the updated edge lengths, v_i computes its corner angles $\{\theta_i^{jk}|f_{ijk} \in F\}$ according to the inverse $\cos$ law:

$$\theta_i^{jk} = \cos^{-1} \frac{l_{ki}^2 + l_{ij}^2 - l_{jk}^2}{2l_{ki}^2 l_{ij}^2}.$$

Then v_i computes its current discrete Gaussian curvature K_i as the excess of the total angle sum at v_i (Eqn. 3.2). If for every non-boundary v_i, the difference between its target Gaussian curvature $\bar{K}_i$ that is set to zero and current Gaussian curvature K_i is less than a threshold (we set to $1e - 5$ in our experiments), discrete surface Ricci flow converges. Otherwise, each non-boundary v_i updates its u_i: $u_i = u_i + \delta(\bar{K}_i - K_i)$, where δ is the step length (we set to 0.05 in our experiments).

When discrete surface Ricci flow converges, we can stop the iterations. Each edge e_{ij} updates its length according to the final circle radii $\gamma_i = e^{u_i}$ and $\gamma_j = e^{u_j}$ and the stored edge weight ϕ_{ij}:

$$l_{ij} = \sqrt{{\gamma_i}^2 + {\gamma_j}^2 + 2\gamma_i\gamma_j \cos\phi_{ij}}.$$

With the computed edge lengths, we can embed M to plane. For simplicity, we let the vertex with the smallest ID (denoted as v_0) initiate the embedding process. Its planar coordinates are set to $(0, 0)$. Then it arbitrarily selects one of its direct neighbors, e.g., v_j, and sets the planar coordinates of v_j to $(0, l_{ij})$. For vertex v_k, adjacent to both v_i and v_j, it calculates the intersection points of the two circles centered at v_i and v_j with radii l_{ik} and l_{jk}, respectively. Then, v_j chooses one of the intersection points that satisfies $(v_j - v_i) \times (v_k - v_i) > 0$ as its planar coordinates. The procedure continues until all vertices of M have their planar coordinates.

Note that we can pre-compute the conformal mapping of M_1 to plane and then pre-load the mapping result to sensor nodes before the deployment of a network.

3.3.2 Alignment

Denote f_1 and f_2 the mappings that conformally map M_1 and M_2 to planar regions D_1 and D_2 respectively. We need to construct another conformal mapping that aligns D_2 with D_1 on plane.

Eqn. 3.4 provides a natural alignment of two planar regions based on three pairs of anchor points. Denote f a M$\ddot{o}$bius transformation that maps the planar region D_1 with three distinct points z_1, z_2, z_3 to the planar region D_2 with three distinct points w_1, w_2, w_3. Particularly, z_1, z_2, z_3 are mapped to w_1, w_2, w_3 respectively. We use complex numbers to represent points on plane. Assume we use z_{ij} to denote $z_i - z_j$, and w_{ij} to denote $w_i - w_j$, f can be represented in a closed form from Eqn. 3.4,

$$f(z) = \frac{w_2(z - z_1)z_{23}w_{12} - (z - z_2)z_{13}w_{23}w_1}{(z - z_1)z_{23}w_{12} - (z - z_2)z_{13}w_{23}}. \tag{3.5}$$

Before we continue the alignment algorithm, we give a brief introduction of *Barycentric coordinates*. They provide a convenient way to interpolate a function on triangles as long as the function's value is known at vertices. Let's consider a function f defined on a triangle f_{abc} with $f(v_a)$, $f(v_b)$, and $f(v_c)$ known. Denote $Area|f_{abc}|$ the area of triangle f_{abc}. The function value of any point p located inside this triangle can be written as a weighted sum of the function value at the three vertices:

$$f(p) = t_1 f(v_a) + t_2 f(v_b) + t_3 f(v_c),$$

where $t_1 = \dfrac{Area|f_{pbc}|}{Area|f_{abc}|}$, $t_2 = \dfrac{Area|f_{pca}|}{Area|f_{abc}|}$, and $t_3 = \dfrac{Area|f_{pab}|}{Area|f_{abc}|}$. It is obvious that t_1, t_2, and t_3

are subject to the constraint $t_1 + t_2 + t_3 = 1$. t_1, t_2, and t_3 are called Barycentric Coordinates of Point p on f_{ijk}.

Assume three anchor nodes - sensor nodes equipped with GPS - are randomly deployed with other sensors. Each anchor node is assigned planar coordinates, e.g., mapped to plane by f_2. Denote the planar point of an anchor node mapped by f_2 with a complex numbers $z_i (1 \leq i \leq 3)$.

Each anchor node then checks its stored M_1 or simply sends a request with its known geographic position to a server to locate three nearest grid points of the DTM, denoted as v_i, v_j, and v_k. Since M_1 and M_2 are not perfectly overlap in general, the anchor node does not necessarily locate inside $f_{ijk} \in M_1$. We compute the projection point of the anchor node to f_{ijk}. The projection point is the closest point of M_1 to the anchor node. Since f_1 is a continuous and one-to-one mapping, we can compute the planar coordinates of the projection point mapped by f_1 based on the planar coordinates of v_i, v_j, and v_k. Specifically, denote (t_1, t_2, t_3) the Barycentric Coordinates of the projection point on f_{ijk}, $f_1(v_i)$, $f_1(v_j)$, and $f_1(v_k)$ the planar coordinates of v_i, v_j, and v_k mapped by f_1, the planar coordinates of the projection point mapped by f_1 is: $t_1 f_1(v_i) + t_2 f_1(v_j) + t_3 f_1(v_k)$. Denote the planar point of the projection point mapped by f_1 with a complex number w_i($1 \leq i \leq 3$).

Each anchor node conducts a flooding to send out its z_i and w_i to the whole network. When receiving the three pairs of planar coordinates, a non-anchor node $v_i \in M_2$ simply plugs them and its planar coordinates by f_2 into Eqn. 3.5. The computed one is the aligned planar coordinates of the sensor node.

3.3.3 Localization

With the aligned planar coordinates, each sensor node locates three nearest grid points on plane. Denote v_i, v_j, and v_k the three nearest grid points on plane. The mapped planar point of the sensor node locates inside the planar triangle $f_{ijk} \in M_1$. Denote (t_i, t_j, t_k) the Barycentric Coordinates of the mapped planar point of the sensor node on f_{ijk}. The 3D geographic coordinates of the sensor node can be computed as $t_i p(v_i) + t_j p(v_j) + t_k p(v_k)$, where $p(v_i)$, $p(v_j)$, and $p(v_k)$ are the 3D geographic coordinates of v_j, v_k, and v_l respectively.

3.3.4 Time Complexity and Communication Cost

Assume we measure the communication cost by the number of exchanged messages. Both the time complexity and communication cost of the proposed localization algorithm are dominated by the step to compute conformal mapping. The time complexity of discrete surface Ricci flow is measured by the number of iterations. The time complexity and communication cost of planar embedding based on computed edge lengths by discrete surface Ricci flow are linear to the size of the network.

The time complexity and communication cost of the other steps of the localization algorithm are either linear to the size of the network or constant complexity.

A special note is that we don't need to compute the conformal mapping of M_1 each time. We only need to compute it once before we start to deploy a network, and then pre-load only the mapping data related with the FoI (Field of Interest) to sensor nodes if they have sufficient storage. Otherwise, a server may be designated to keep the DTM database.

3.4 Discussion

In this section, we have three different scenarios to show how they could effect the performance of our localization result.

3.4.1 The Size of Anchor Nodes

Theoretically speaking, the proposed localization algorithm requires only three anchor nodes to align two triangular meshes on plane, even one triangular mesh is extracted from the connectivity graph of a network with thousands or even tens of thousands of sensor nodes. If there are more then three anchor nodes deployed with the network, we can apply the least-square conformal mapping method instead of M$\ddot{o}$bius transformation to incorporate all anchor nodes into the alignment to improve the localization accuracy.

Fig. 3.3 shows one example. For a network with size 2.6k deployed on a 3D surface as shown in Fig. 3.1(d), the localization error of the network decreases with the increased number of anchor nodes. Compared with M$\ddot{o}$bius transformation based alignment introduced in Sec. 3.3.2 , least-square conformal mapping based alignment is more flexible to take anchor nodes into alignment. But from the other side, least-square conformal mapping method is centralized with high computational complexity.

3.4.2 Anchor Node Free

As we introduced in Sec. 3.2.1 , conformal mapping maps infinitesimal circles to infinitesimal circles, so locally conformal mapping introduces no distortion, only scaling. Such scaling is called conformal factor. conformal factor at v_i can be approximated as the ratio of the triangle areas in $3D$ and mapped in $2D$ plane of all f_{ijk} incident to v_i,

$$cf(v_i) = \frac{\sum_{f_{ijk}\in F} Area_{3D}|f_{ijk}|}{\sum_{f_{ijk}\in F} Area_{2D}|f_{ijk}|}.$$

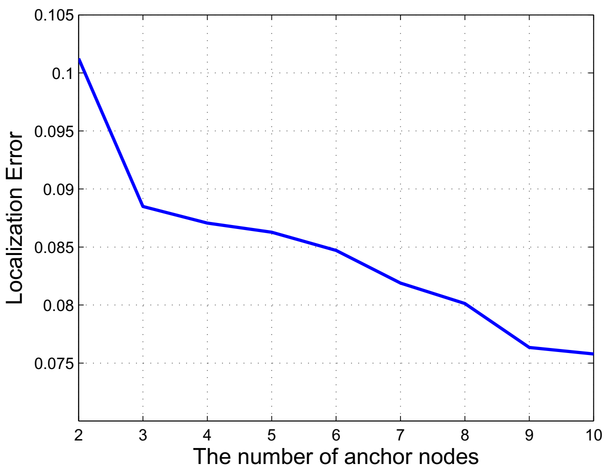


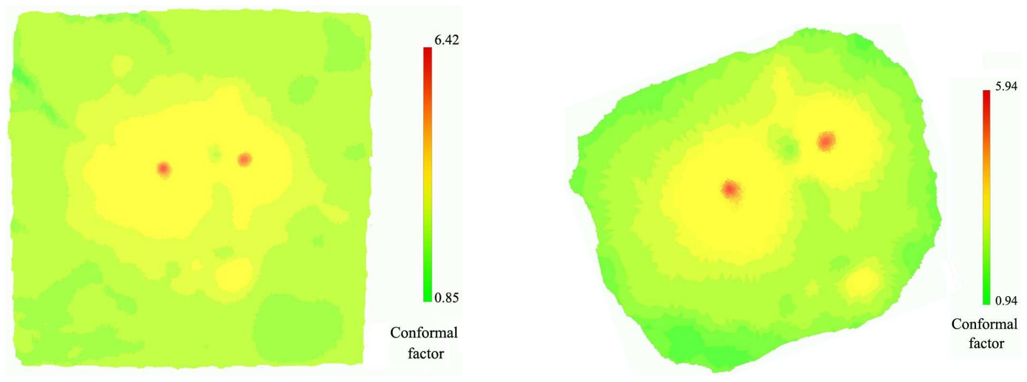
Figure 3.3: Localization error decreases with the increased number of anchor nodes.

Conformal factor at the peak of a terrain surface is usually huge. We can demonstrate the fact by an extreme case. Suppose we have a long and open tube-shape can and we conformally map it to plane. The center of the bottom of the can is mapped to the origin. No matter what conformal mapping we construct, conformal factor increases exponentially fast as the mapped point on plane close to the origin.

Based on the fact, vertices of M_2, i.e., sensor nodes, with the highest conformal factors are around the peaks of a terrain surface. We can apply them as anchor nodes for alignment. Assume the network shown in Fig. 3.1(d) is anchor node free. We compute conformal factors of the triangular meshes of the DTM and the network and use colors to encode them at the mapped planar regions shown in Fig. 3.4. It is obvious that areas marked with red represent the regions with high conformal factors. We pick one vertex with the highest conformal factor for each red marked region. Suppose we pick v_1 and v_2 for the triangular mesh of the network, v_3 and v_4 for the triangular mesh of the DTM. Suppose v_1 shares a similar conformal factor with V_3. v_1 simply determines its 3D coordinates the same as v_3. Similarly, v_2 determines its

3D coordinates the same as v_4.

Note that if the shape of a mountain region is extremely complicated, conformal factors may identify wrong pairs of nodes between M_1 and M_2. The anchor free localization method is not stable in that case.



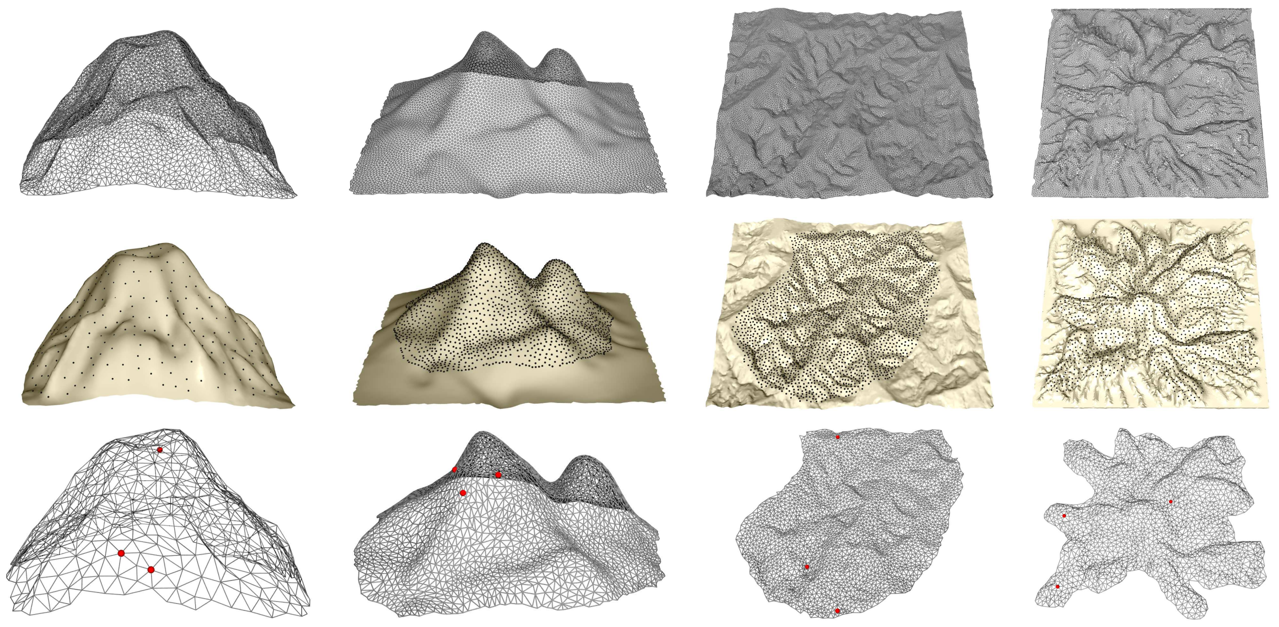
(a) (b)

Figure 3.4: Color coding with conformal factors on plane.

3.4.3 Connectivity Only

When range distance measurement is not available, we can still extract a sparse triangular mesh from a network connectivity graph. A simple landmark-based algorithm uniformly selects a subset of nodes in a distributed way and denotes them as landmarks, such that any two neighboring landmarks are approximately a fixed K hops away ($K \geq 6$). The dual of a discrete Voronoi diagram with generators the set of landmarks forms a triangulation. Vertices of the triangulation is the set of landmarks. Edge between two neighboring vertices is a shortest path between the two landmarks. We simply assume the edge length of the triangulation a unit one, and then apply exactly the same localization algorithm for landmark nodes.

A non-landmark node, denoted as n_i, finds its three nearest landmarks, denoted as v_1, v_2,



(a) DTM I (b) DTM II (c) DTM III (d) DTM IV

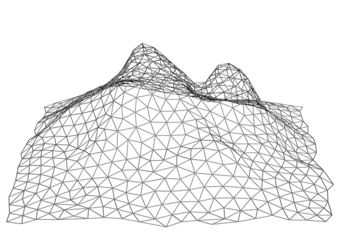
Figure 3.5: Localization results for different network models.

v_3 with computed 3D coordinates $p(v_1)$,$p(v_2)$, and $p(v_3)$ respectively. Denote d_1, d_2, and d_3 the shortest distances (hop counts) of node n_i to the three landmarks v_1, v_2, v_3 respectively. Then node n_i computes its 3D coordinates $p(n_i)$ simply by minimizing the mean square error among the distances:

$$\sum_{j=1}^{3}(|p(n_i) - p(v_j)| - d_j)^2. \tag{3.6}$$

3.5 Simulations

Fig. 3.5 shows a set of DTMs of representative terrain surfaces, on which wireless sensor networks are randomly deployed (see the black points on these terrain surfaces). The sizes of the sensor networks deployed on DTM I, II, III, and IV, are 0.5k, 2.6k, 3k, and 2k respectively. We carry out extensive simulations under various scenarios to evaluate the overall performance of the proposed algorithm with different factors such as the positions of


(c) DTM II

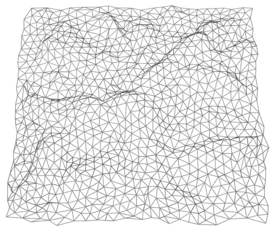

(d) DTM IV

Figure 3.6: The same set of DTMs as shown in Fig. 3.5 with very low resolutions.

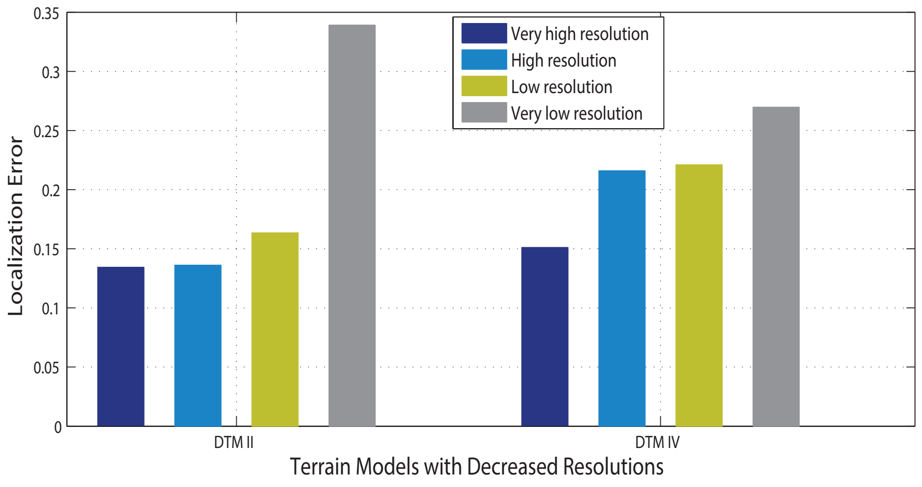


Figure 3.7: Localization results with different DTM resolutions.

the three anchor nodes, the resolution of a DTM, and the one-hop distance measurement error. We compute the localization error as the ratio of the average node distance error (all sensors in the network) and the average transmission range.

3.5.1 Terrain Models with Different Resolutions

To evaluate the impact of the resolution of a DTM, we compute the localization errors of a network deployed on a terrain surface with four different resolutions of the DTM. The resolution of the highest one is almost twenty times of the resolution of the lowest one. Fig. 3.5(b) and (d) show the two DTMs of our testing with very high resolutions, and

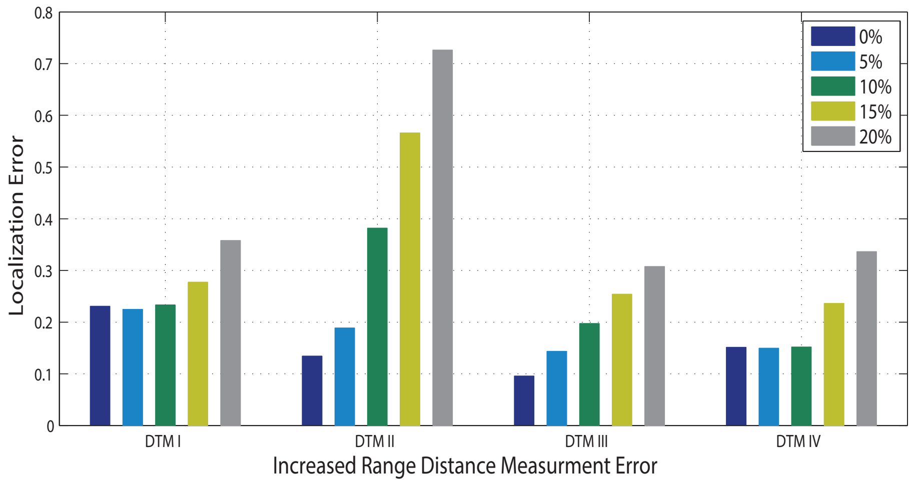


Figure 3.8: Localization error increases with one-hop distance measurement.

Fig. 3.6(a) and (b) show the same two DTMs with very low resolutions. The results given in Fig. 3.7 show that the resolution of a DTM has a small impact on the performance of the localization algorithm unless it is too low.

3.5.2 Networks with Measurement Errors

We have also evaluated our algorithm when errors are introduced in one-hop distance measurement. For each network, we choose the set of anchor nodes that gives the median localization error based on the repeated tests. Fig. 3.8 shows that the localization algorithm is sensitive to measurement error. So for a network with potentially large measurement errors, we select uniformly a set of landmark nodes such that each landmark node has one hop distance to its landmark neighbors, i.e., a Voronoi diagram with a small and constant cell size, and then build a triangular mesh from the chosen landmark nodes with edge length approximately the averaged transmission range. Similar as connectivity based surface localization discussed in Sec. 3.4.3 , we localize the landmark nodes first and then other non-landmark nodes.

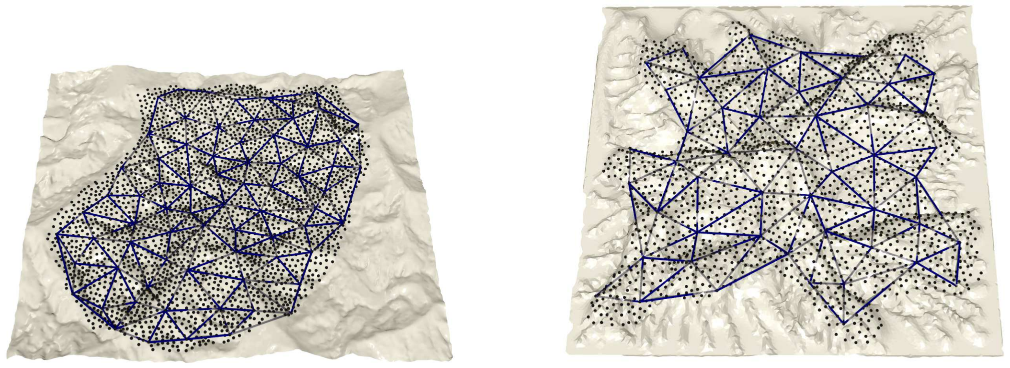

(a) (b)

Figure 3.9: Networks with Connectivity Information Only.

3.5.3 Networks with Connectivity Information Only

As we discussed in Sec. 3.4.3 , we uniformly select a subset of nodes marked as landmark nodes and build a sparse triangulation for a network with mere connectivity. Each vertex is a landmark node and each edge has an approximately constant length. Fig. 3.9 shows the sparse triangular meshes generated from the network with size $3k$ deployed on DTM III and the network with size $2k$ deployed on DTM IV. The localization errors for landmark nodes of the two networks are 0.2037 and 0.2610 respectively.

3.5.4 The Convergence Time

We carry out experiments to test the number of iterations of discrete surface Ricci flow required for convergence. Fig. 3.10 shows the convergence rates of discrete surface Ricci flow on network with size $3k$ deployed on DTM III and network with size $2k$ deployed on DTM IV. As we can pre-compute the planar conformal mapping of the DTM triangulation of a terrain surface, we can apply Newtons numerical method to compute the solution of discrete surface Ricci flow. The computation of this centralized method is very efficient with less than ten iterations in a few seconds for a triangulation with $10k$ size.

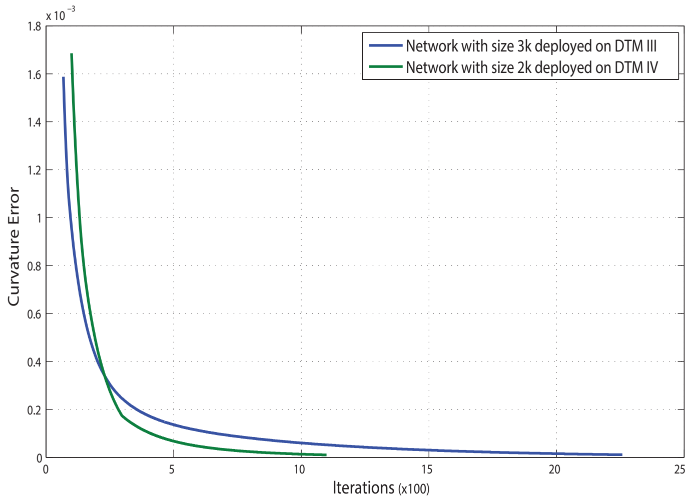


Figure 3.10: The convergence rate of the discrete surface Ricci flow algorithm.

3.6 Summary

In this paper we have proposed a fully distributed algorithm to localize a wireless sensor network deployed on the surface of a complex 3D terrain. Our algorithm constructs a well-aligned mapping between the triangular mesh of the DTM of the terrain surface and the triangular mesh extracted from the connectivity graph of the network deployed on the terrain surface. Based on the mapping, each sensor node of the network can easily locate reference grid points from the DTM to calculate its own geographic location. We have carried out extensive simulations under various scenarios to evaluate the overall performance of the proposed algorithm with different factors. We have also discussed the possibility of 3D surface network localization with connectivity only with promising results.

CHAPTER 4: CONCLUSIONS

In this dissertation, firstly, I have proposed a routing scheme for 3D wireless networks, which is based on Harmonic Volumetric Embedding (HVE). More specifically, my proposed solution is based on a unit tetrahedron cell (UTC) mesh structure. I propose a distributed algorithm to realize volumetric harmonic mapping of the UTC mesh under spherical boundary condition. It is a one-to-one map that yields virtual coordinates for each node in the network. Since a boundary has been mapped to a sphere, node-based greedy routing is always successful thereon. At the same time, I exploits the UTC mesh to develop a face-based greedy routing algorithm, and prove its success at internal nodes. To route a data packet to its destination, face-based and node-based greedy routing algorithms are employed alternately at internal and boundary UTCs, respectively. For networks with multiple internal holes, a segmentation and *tunnel*-based routing strategy is proposed on top of *VHM* to support global end-to-end routing.

As far as I know, *this is the first work that realizes truly deterministic routing with constant-bounded storage and computation in general 3D sensor networks*. To make a local routing decision, each node only needs to store virtual coordinates of itself and its neighbors, and a routing table with a size bounded by the number of internal holes.

Secondly, I focus on large-scale 3D wireless sensor networks. Accordingly, I have proposed a Bubble Routing scheme, which does not require the global coordinates of sensors (such as GPS). The sensor distribution can be either uniform or non-uniform. The sensor deployment may be constrained by obstacles and terrain variations, thus resulting in complex network shapes with multiple interior holes and concave exterior boundary. The key contribution of our proposed scheme is to preprocess the global information via a distributed algorithm, such that a sensor only needs to store a minimum amount of information to make correct and efficient local routing decisions, thus achieving scalable routing with guaranteed delivery.

More specifically, my proposed routing protocol is outlined below. First, a 3D sensor network is segmented into a set of hollow spherical cells (HSCs), one for each interior hole. The boundary of an HSC is called a Hollow Spherical Bubble (HSB). To enable greedy routing inside an HSC, a continuous and one-to-one mapping is performed to map the HSB to a virtual sphere, which guarantees greedy forwarding between any two nodes thereon. Such a mapping is nontrivial. Moreover, virtual trees grow from the HSB toward the inside of the HSC. They are employed to guide routing between two nodes to largely parallel to the greedy path between their projections on the HSB. This design ensures greedy routing in an HSC and achieves load balancing at the same time. A global routing table is created to route packets across HSCs. Its size is small and bounded by the number of interior holes.

Finally, I consider a sensor network deployed in a 3D space with one or multiple internal holes, where sensors cannot be deployed. I show that local minima are always on the boundaries of holes if nodal density is high, and there exists a DISCO algorithm to support routing between any pair of points in a strong-connected 3D network. I propose a distributed and deterministic algorithm with constant storage, communication and computation overhead, dubbed trace-routing, to escape from local minima. Trace-routing is triggered when a packet reaches a local minimum. It constructs a virtual cutting plane that contains the local minimum and the destination and intersects the corresponding boundary surface to yield a trace. The trace is a closed loop that can be computed locally with constant overhead. The packet is routed along such a trace, thus deterministically moving out of the local minimum.

The proposed trace-routing algorithm does not demand preprocessing of the global network information, neither does it require establishing or maintaining a global data structure, which often consumes a storage space proportional to the network size and needs frequent update due to network dynamics. I further prove the trace-routing algorithm guarantees data delivery in a strong-connected 3D network under both continuous and discrete settings. In a nutshell, trace-routing supports DISCO, thus highly efficient in large-scale 3D sensor networks. It is worth mentioning that the proposed trace-routing algorithm and related discussions and proofs do not rely on any particular communication model (such as the unit ball graph model or quasi-unit ball graph model). Only a maximum radio range is assumed, which is generally known in practical sensor networks.

REFERENCES

[1] J. Allred, A. B. Hasan, S. Panichsakul, W. Pisano, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni, “SensorFlock: An Airborne Wireless Sensor Network of Micro-Air Vehicles,” in *Proc. of SenSys*, pp. 117–129, 2007.

[2] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, “Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications,” *IEEE Network*, vol. 20, no. 3, pp. 12–18, 2006.

[3] X. Bai, C. Zhang, D. Xuan, J. Teng, and W. Jia, “Low-Connectivity and Full-Coverage Three Dimensional Networks,” in *Proc. of MobiHOC*, pp. 145–154, 2009.

[4] X. Bai, C. Zhang, D. Xuan, and W. Jia, “Full-Coverage and K-Connectivity (K=14, 6) Three Dimensional Networks,” in *Proc. of INFOCOM*, pp. 388–396, 2009.

[5] C. Liu and J. Wu, “Efficient Geometric Routing in Three Dimensional Ad Hoc Networks,” in *Proc. of INFOCOM*, pp. 2751–2755, 2009.

[6] T. F. G. Kao and J. Opatmy, “Position-Based Routing on 3D Geometric Graphs in Mobile Ad Hoc Networks,” in *Proc. of The 17th Canadian Conference on Computational Geometry*, pp. 88–91, 2005.

[7] J. Opatrny, A. Abdallah, and T. Fevens, “Randomized 3D Position-based Routing Algorithms for Ad-hoc Networks,” in *Proc. of Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, pp. 1–8, 2006.

[8] R. Flury and R. Wattenhofer, “Randomized 3D Geographic Routing,” in *Proc. of INFOCOM*, pp. 834–842, 2008.

[9] F. Li, S. Chen, Y. Wang, and J. Chen, “Load Balancing Routing in Three Dimensional Wireless Networks,” in *Proc. of ICC*, pp. 3073–3077, 2008.

[10] J. Zhou, Y. Chen, B. Leong, and P. Sundaramoorthy, “Practical 3D Geographic Routing for Wireless Sensor Networks,” in *Proc. of SenSys*, pp. 337–350, 2010.

[11] D. Pompili, T. Melodia, and I. F. Akyildiz, “Routing Algorithms for Delay-insensitive and Delay-sensitive Applications in Underwater Sensor Networks,” in *Proc. of MobiCom*, pp. 298–309, 2006.

[12] W. Cheng, A. Y. Teymorian, L. Ma, X. Cheng, X. Lu, and Z. Lu, “Underwater localization in sparse 3d acoustic sensor networks,” in *Proc. of INFOCOM*, pp. 798–806, 2008.

[13] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, “Routing with Guaranteed Delivery in Ad Hoc Wireless Networks,” in *Proc. of Third Workshop Discrete Algorithms and Methods for Mobile Computing and Communications*, pp. 48–55, 1999.

[14] B. Karp and H. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks,” in *Proc. of MobiCom*, pp. 1–12, 2001.

[15] E. Kranakis, H. Singh, and J. Urrutia, “Compass Routing on Geometric Networks,” in *Proc. of Canadian Conference on Computational Geometry (CCCG)*, pp. 51–54, 1999.

[16] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, “Geometric Ad-hoc Routing: Theory and Practice,” in *Proc. of The 22nd ACM Symposium on the Principles of Distributed Computing*, pp. 63–72, 2003.

[17] F. Kuhn, R. Wattenhofer, and A. Zollinger, “Worst-case Optimal and Average-case Efficient Geometric Ad-hoc Routing,” in *Proc. of MobiHOC*, pp. 267–278, 2003.

[18] B. L. S. Mitra and B. Liskov, “Path Vector Face Routing: Geographic Routing with Local Face Information,” in *Proc. of ICNP*, pp. 147–158, 2005.

[19] H. Frey and I. Stojmenovic, “On Delivery Guarantees of Face and Combined Greedy-face Routing in Ad Hoc and Sensor Networks,” in *Proc. of MobiCom*, pp. 390–401, 2006.

[20] G. Tan, M. Bertier, and A.-M. Kermarrec, “Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks,” in *Proc. of INFOCOM*, pp. 1719–1727, 2009.

[21] S. Durocher, D. Kirkpatrick, and L. Narayanan, “On Routing with Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks,” in *Proc. of International Conference on Distributed Computing and Networking*, pp. 546–557, 2008.

[22] S. Xia, X. Yin, H. Wu, M. Jin, and X. Gu, “Deterministic Greedy Routing with Guaranteed Delivery in 3D Wireless Sensor Networks,” in *Proc. of MobiHoc*, pp. 1–10, 2011.

[23] H. Zhou and et. al., “Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks,” in *Proc. of ICDCS*, pp. 744–753, 2010.

Yang, Yang. Bachelor of Science, Northwestern Polytechnical University, Summer 2005;
Master of Science, University of Louisiana at Lafayette, Spring 2011; Doctor of
Philosophy, University of Louisiana at Lafayette, Fall 2013
Major: Computer Science
Title of Dissertation: Geometry in Wireless Sensor Networks In-network Information
Processing and Localization
Dissertation Director: Dr. Miao Jin
Pages in Dissertation: 72; Words in Abstract: 274

ABSTRACT

Firstly, I propose a geographic location free double-ruling based approach for general 3D sensor networks with possibly complicated topology and geometric shapes. Without the knowledge of the geographic location and the distance bound, a query simply travels along a simple curve with the guaranteed success to retrieve aggregated data through time and space with one or different types across the network. Extensive simulations and comparisons show the proposed scheme with low cost and a balanced traffic load.

Secondly, I explore 3D surface network localization with terrain model. A digital terrain model (DTM), available to public with a variable resolution up to one meter, is a 3D representation of a terrain's surface. It is commonly built using remote sensing technology or from land surveying and can be easily converted to a triangular mesh. Given a sensor network deployed on the surface of a 3D terrain with one-hop distance information available, we can extract a triangular mesh from the connectivity graph of the network. The constraint that the sensors must be on the known 3D terrain's surface ensures that the triangular meshes of the network and the DTM of the terrain's surface approximate the same geometric shape and overlap. I propose a fully distributed algorithm to construct a well-aligned mapping between the two triangular meshes. Based on this mapping, each sensor node of the network can easily locate reference grid points from the DTM to calculate its own geographic location. I carry

out extensive simulations under various scenarios to evaluate the overall performance of the proposed localization algorithm. The possibility of 3D surface network localization with mere connectivity and the results are promising is also discussed.

BIOGRAPHICAL SKETCH

Yang Yang received his Bachelor’s in Computer Science from the Northwestern Polytechnical University in China in 2005 and his Master of Science in Computer Science from the University of Louisiana at Lafayette in 2011. He has been working toward a Doctor of Philosophy in Computer Science with The Center for Advanced Computer Studies (CACS) at the University of Louisiana at Lafayette.

His current research interests include computational geometry and wireless sensor networks.