Distributed Information Storage and Retrieval in 3D Sensor Networks with General Topologies

Yang Yang, Miao Jin, Yao Zhao, and Hongyi Wu

Abstract—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 research 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.

Index Terms—3D Sensor Networks, Information Storage and Retrieval, in-network, data-centric,

I. INTRODUCTION

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 [1]–[11]. 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) [12], [13]. 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-centric 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.

A. An Overview of Distributed Information Storage and Retrieval Algorithms

Geographical hash table [3], [11], [14]–[16] 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 [17]. 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 1 (a) 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 one simple retrieval curve. This is 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 [1], [4], [18] or with heavy data replication to achieve high probability that the retrieval curve would meet one of the replication curves within the sensor network field [5]. To extend double-ruling scheme to networks with uneven sensor distribution and irregular geometric shapes, landmark-based scheme [9] 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 [19] 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 [10], 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.

B. 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 1 (b). 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 [20], [21]. They map a 3D sensor network to a cube and assign each node a virtual coordinates. Their method requires the network shape topologically equivalent to a cube 1. However, many practical 3D sensor networks are topologically different from a cube. Figure 2(a) shows a 3D sensor network with sensors deployed underwater around an island. The topology of the network is equivalent to a donut with a handle. Figure 2(b) gives another example of a 3D sensor network with multiple coverage holes inside.

Another challenge to achieve 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 schemes, such as GPSR [17]. They require global location information. However, some application scenarios in 3D prohibit the reception of satellite signals by part or all of the sensors, rendering it impossible to solely rely on global navigation systems. Moreover, 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. Even if we assume location information for each sensor node, it is still non-trivial to design a scalable geographic routing scheme that guarantees successful delivery and requires constant storage at each node in general 3D sensor networks.

C. Our Approach

Considering the boundary surface of a 3D volume is always a closed surface, the proposed approach is motivated by a topological concept that any closed surface can be cut open to a topological disk along an appropriate set of edges called a

1 Two shapes topologically equivalent to each other means they can be continuously transformed one into the other. A continuous mapping exists between them.
cut graph of the surface [22]. Two examples given in Figure 3 briefly illustrate the basic idea of our approach. The first example is a 3D volumetric sensor network model with two handles (see Figure 3 (a)). We first detect the boundary nodes and build a triangular structure of the identified boundary surface of the network (see Figure 3 (a)). We then compute a cut graph of the boundary surface of the network which is marked with yellow color in Figure 3 (b). We cut the boundary surface of the network open to a topological disk along the cut graph and then map it to an aligned planar rectangle such that each boundary node of the network is associated with a planar rectangle virtual coordinates (see Figure 3 (c)). Each non-boundary sensor stores the ID of its neighbor nearest to the boundary of the network. A data generator follows the sequence of IDs to the boundary of the network, and then travels along a horizontal line of the virtual planar rectangle and leaves data copies. The two horizontal lines of the virtual planar rectangle marked with blue color shown in Figure 3 (c) correspond to the two real data replication curves of the network shown in Figure 3 (g) marked with the same color. Similarly, a consumer follows the sequence of IDs to the boundary of the network and collects the aggregated data of different types along a vertical line. The vertical line of the virtual planar rectangle marked with red shown in Figure 3 (c) corresponds to the data aggregation curve of the network shown in Figure 3 (g) marked with the same color. The second example is also a 3D sensor network deployed under water (see Figure 3 (d)). We cut the boundary surface of the network open to a topological disk along a computed cut graph (see Figure 3 (e)) and then assign each boundary node of the network a planar rectangle virtual coordinates (see Figure 3 (f)). Similarly, Figure 3 (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.

The rest of this paper is organized as follows: Section II explains in detail the algorithm to compute the cut graph. Section III describes the method to generate the planar rectangle virtual coordinates of boundary nodes of a network.
Section IV discusses a number of implementation issues of the proposed cut graph based double-ruling approach in 3D sensor networks. Section V presents the simulation results and Section VII concludes the paper.

II. CUT GRAPH OF BOUNDARY SURFACE OF THE NETWORK

As briefly introduced in Sec. I, given a sensor network deployed in 3D volume with distance information within one-hop neighborhood only, we detect its boundary nodes and extract a triangular boundary surface of the 3D volume before computing its cut graph. Note that a boundary surface of any 3D volume is a closed surface (i.e., without any holes).

We apply the algorithm proposed in [23] to identify boundary nodes of a 3D volume sensor network. The algorithm is fully distributed with no constraints on communication models. Although the algorithm can tolerate a moderate distance measurement error, we do not require accurate detection of all boundary nodes of the network. We only need a connected triangular structure to approximate the boundary surface of the 3D volume network. We allow some mistakenly detected non-boundary vertices on the triangular structure.

We then apply the algorithm proposed in [24] to extract a triangular structure of the detected boundary nodes of the network. The algorithm takes the connectivity graph of the detected boundary nodes of the network as input. Based on locally estimated distances between nodes within one-hop communication range, a local coordinates system can be constructed. Each edge based on its local coordinates system then computes a weight that measures the number of triangles shared by the edge and the various local neighbor sets for the edge in the initial graph. An iterative algorithm keeps removing edges until each edge in the final graph has weight exactly two - shared by two triangles only. The algorithm is also fully distributed with no constraints on communication models. The constraint of the algorithm is that it assumes that a triangular graph is a sub-graph of the initial connectivity graph of boundary nodes of the network.

Before we introduce the algorithm of computing a cut graph of the triangulated boundary surface of a 3D volume network in Sec. II-B, we give a brief review of some important topological concepts that are necessary to the algorithm in Sec. II-A. We refer the interested reader to [22], [25] for formal definitions and further topological background. Table I summaries the symbols we use in the paper.

![Loops on a genus two surface](image)

Fig. 4. Loops on a genus two surface: $L_1$ is contractible and surface separating; $L_2$ is non-contractible and surface separating; $L_3$ is non-contractible and non-separating.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>a topological disk (i.e., a simply connected domain)</td>
</tr>
<tr>
<td>$\partial D$</td>
<td>the boundary of $D$</td>
</tr>
<tr>
<td>$d$</td>
<td>the average number of one-range neighbors of a sensor node</td>
</tr>
<tr>
<td>$d'$</td>
<td>the average vertex degree of a triangular surface</td>
</tr>
<tr>
<td>$E$</td>
<td>edges of $M$</td>
</tr>
<tr>
<td>$e_{ij}$</td>
<td>an edge belonging to $E$ with two ending vertices $v_i$ and $v_j$</td>
</tr>
<tr>
<td>$F$</td>
<td>triangle faces of $M$</td>
</tr>
<tr>
<td>$f_{jk}$</td>
<td>a triangle face belonging to $F$ with vertices $v_i$, $v_j$, and $v_k$</td>
</tr>
<tr>
<td>$G$</td>
<td>a cut graph of $M$</td>
</tr>
<tr>
<td>$g$</td>
<td>the genus number of $M$</td>
</tr>
<tr>
<td>$K_i$</td>
<td>the discrete Gaussian curvature of $v_i \in V$</td>
</tr>
<tr>
<td>$\tilde{K}_i$</td>
<td>the target Gaussian curvature of $v_i \in V$</td>
</tr>
<tr>
<td>$L$</td>
<td>a loop on $M$</td>
</tr>
<tr>
<td>$l_{ij}$</td>
<td>the length of $e_{ij}$</td>
</tr>
<tr>
<td>$M$</td>
<td>a closed triangular surface</td>
</tr>
<tr>
<td>$m$</td>
<td>the number of the boundary nodes of a 3D network</td>
</tr>
<tr>
<td>$n$</td>
<td>the number of the sensor nodes of a 3D network</td>
</tr>
<tr>
<td>$p_i$</td>
<td>the computed planar rectangle virtual coordinates of a boundary node $v_i$</td>
</tr>
<tr>
<td>$t$</td>
<td>the time</td>
</tr>
<tr>
<td>$u_i$</td>
<td>the logarithm of $\gamma$ of $v_i$</td>
</tr>
<tr>
<td>$V$</td>
<td>vertices of $M$</td>
</tr>
<tr>
<td>$v_i$</td>
<td>a vertex belonging to $V$ with id $i$</td>
</tr>
<tr>
<td>$\theta_{ij}^k$</td>
<td>the corner angle attached to Vertex $v_i$ in Face $f_{ij}$</td>
</tr>
<tr>
<td>$\gamma_i$</td>
<td>the radius of a circle associated with $v_i$</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>the radius function assigned at $M$</td>
</tr>
<tr>
<td>$\phi_{ij}$</td>
<td>weight of $e_{ij}$ (the intersection angle of two circles centered at $v_i$ and $v_j$)</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>the weight function assigned at $M$</td>
</tr>
<tr>
<td>$(\Gamma, \Phi)$</td>
<td>circle packing metric of $M$</td>
</tr>
<tr>
<td>$\delta$</td>
<td>a small constant - the step length computing planar rectangle virtual coordinates</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>the threshold of curvature error computing planar rectangle virtual coordinates</td>
</tr>
</tbody>
</table>

A. 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. 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 4 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 4 with $L_1$ and $L_2$; otherwise it is non-separating as shown in Figure 4 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 $M \setminus (L_1 \cup \cdots \cup L_g)$ 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 hole) 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 of \( D \). Figure 5 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.

### B. Cut Graph of the Boundary Surface of A Network

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 is studied in [26], which proves the problem NP-hard and provides a polynomial-time approximation algorithm. Canonical systems of loops, a system of 2g (g is the genus number of the surface) loops with a single base vertex to cut a surface into a disk, is discussed in [27]. Erickson and Whittlesey [28] provide a greedy algorithm to compute a shortest system of such canonical loops. Shortest cut graph with prescribed vertex set is discussed later in [29].

Instead of adopting the above optimization algorithms, we propose a fully distributed two-step algorithm motivated by that of [30], 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 introduced in Sec. II-B1 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 introduced in Sec. II-B2. For genus zero surfaces, there is no marked edges left after trimming, which provides a convenient way to automatically identify the topology of a 3D sensor network. We discuss the computational cost of the two-step algorithm in Sec. II-B3.

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 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**: 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 \( D_m \) in the proof of Theorem 1 is disconnected. It is impossible that the boundary of a topological disk is disconnected.

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 edges of \( D \) in the worst case can equal to the size of the 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.

**Theorem 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 by the end of the trimming process for a genus zero surface $M$.

**Theorem 4:** After the trimming process, there is no marked edge left for a closed genus zero surface $M$ (i.e., a topological sphere surface).

**Proof:** We first use way of contradiction to prove the cut graph of a genus zero 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 1: at each step $i$ of the propagation, $D_i$ is always a topological disk. So the computed cut graph of a genus zero 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 of a network detected genus zero, the boundary node of the network with the smallest id conducts a simple flooding on the boundary surface of the network to find one boundary node with the longest shortest path measured by hops on the boundary surface. We then cut the surface open to a topological disk surface (i.e., a simply connected domain) along the shortest path between the pair of nodes.

3) **Computational Cost:** The algorithm to compute a cut graph of a boundary surface of a network involves many topological concepts, but the computational complexity of the algorithm is actually low.

The first step of computing the cut graph starts from marking one randomly chosen triangle of the boundary surface of the network and then continues to mark others with the width first way, so each triangle will be visited only once before this step ends. The time complexity of this step is linear to the size of the triangles of the boundary surface. During the growth of the marked triangles, only nodes at the two ends of a marked edge exchange messages. They exchange messages at most twice, the first time when the marked edge shared by only one marked triangle, and the second time when the marked edge shared by two marked triangles. Each edge will be marked only once, so the communication cost is twice of the size of the edges of the boundary surface of the network.

The second step of trimming is to remove those unnecessary marked edges. Both the time complexity and communication cost are determined by the number of marked edges at the end of the first step. The size of the marked edges in the worst case can equal to the size of the triangles of the boundary surface of the network. For genus zero surfaces, denote $m$ the size of the boundary nodes, an extra $O(m \log m)$ cost is spent to find one longest shortest path on the boundary surface of the network after all marked edges are trimmed out.

Given an arbitrary triangular surface, the sizes of its triangles and edges are both linear to the size of the nodes. In summary, for non-genus zero surfaces, both the time complexity and the communication cost of the whole algorithm are linear to the size of the boundary nodes of a network. For genus zero surfaces, we need an extra $O(m \log m)$ cost. Note that the size of the boundary nodes of the network is far less than the size of the network in general.

**III. 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. We introduce briefly discrete surface Ricci flow in Sec. III-A, and then the algorithm in Sec. III-B.

**A. 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 $D = (V, E, F)$ as defined in a similar way in Sec. II-B, We assign each $v_i$ a circle with radius $\gamma_i$ 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 $\phi_{ij}$. 

![Networks](image_url)
We call $\phi_{ij}$ the weight of $e_{ij}$, and denote the weight function $\Phi : E \rightarrow [0, \frac{\pi}{2}]$.  

**Definition 5 (Circle Packing Metric):** [31] A circle packing metric of $D$ includes $\Gamma$ and $\Phi$.

- Denote $l_{ij}$ the length of $e_{ij}$. $l_{ij}$ can be computed from $\gamma_i, \gamma_j$ and $\phi_{ij}$ from the following cosine law:

$$l_{ij}^2 = \gamma_i^2 + \gamma_j^2 + 2\gamma_i \gamma_j \cos \phi_{ij}. \quad (1)$$

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

**Definition 6 (Discrete Gaussian Curvature):** Denote $\theta^j_i$ the corner angle attached to $v_i$ belonging to $f_{ijk}$, $\partial D$ the the boundary of $D$, and $K_r$ the discrete Gaussian curvature at vertex $v_i$. $K_r$ can be computed as the angle deficit at $v_i$:

$$K_i = \begin{cases} 
2\pi - \sum_{f_{ijk} \in \partial D} \theta^j_i, & v_i \notin \partial D; \\
\pi - \sum_{f_{ijk} \in \partial D} \theta^j_i, & v_i \in \partial D.
\end{cases} \quad (2)$$

**Definition 7 (Discrete Surface Ricci Flow):** [32] Denote $(\Gamma_0, \Phi)$ an initial circle packing metric of $M$, $K_r$ and $K_i$ the target and current Gaussian curvatures at $v_i$ respectively, and $t$ the time. Let $u_t$ be the logarithm of $\gamma_i$ for each $v_i$. The discrete surface Ricci flow is defined as:

$$\frac{du(t)}{dt} = (K_i - K_r). \quad (3)$$

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. Chow and Luo proves the convergence of discrete surface Ricci flow in [32]. We refer readers to our previous work [33] for details of discrete surface Ricci flow.

### B. Computing Planar Rectangle Virtual Coordinates

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

We initialize a circle packing metric on $D$ such that each circle associated with a vertex has a unit radius, i.e., $\gamma_i = 1$. Then, $u = \log \gamma_i = 0$ 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 $\theta^j_i$ according to the law of cosines:**

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

4. **Compute current Gaussian curvature $K_i$ for each $v_i$ as Equation 2.**
5. **Denote $\epsilon$ a threshold and set to $1 - 4$. If all $|K_i - K_r| < \epsilon$, the algorithm goes to the next step; otherwise, $u_i = u_i + \delta(K_i - K_r)$, where $\delta$ 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 of the network only needs to exchange information with its direct neighbors when implementing the above algorithm.

### IV. IMPLEMENTATION

**A. 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 of the network 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 horizontal 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.

**B. 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. Note that the boundary surface of the network is only virtually cut open and mapped to a planar rectangle. Once a consumer hits the boundary side of the rectangle, the consumer can cross the boundary side and keep traveling along the line with the same direction. The consumer stops 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.

C. Delivery of Data and Query

As a preprocessing, each of the boundary nodes of the network 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.

D. 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 as discussed in Sec. V-D.

E. 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 6 for the model in Figure 3 (a), determines the time complexity of computing the edge lengths, given by \(-C \log_{\lambda} d'm\) where \( C \) is a constant, \( \epsilon \) is the threshold of curvature error (set to 1e-4 in our implementation), and \( \lambda \) is the step length of each iteration (set to 0.1 in our implementation) [32]. Since each vertex only needs to exchange messages with its one range neighbors at one iteration, the corresponding communication cost is \(-C \log_{\lambda} d'm\) 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., \( O(m) \).

F. Bound on Data Replication and Retrieval Costs

Data replication and retrieval costs, without considering the cost to travel to the nearest boundary node for inner nodes, are bounded by the number of nodes along the horizontal and vertical lines of the virtual planar rectangle respectively.

The algorithm to compute the cut graph in Sec. II starts from one randomly chosen triangle of the boundary surface of a network and propagates to grow the marked triangles in a width-first way until all triangles of the boundary surface have been marked. When cutting the boundary surface open to a topological disk along the computed cut graph, the longest radius of the topological disk is bounded by the longest shortest path of the nodes of the starting triangle on the boundary surface. We then uniformly pick four nodes along the boundary of the topological disk and map the topological disk to a well-aligned planar rectangle with corners the four chosen nodes in Sec. III using Ricci flow. The mapping is a conformal map (i.e., a diffeomorphism) that preserves angles and local shape. Specifically, the mapping preserves both the neighborhood of a boundary node and the relative positions of its neighbors when virtually mapped to a planar rectangle. Therefore, the horizontal and vertical lines of the rectangle should be bounded by the diameter of the topological disk. It is twice of the longest shortest path of the nodes of the starting triangle.

Since we randomly choose the starting triangle to compute the cut graph, the horizontal and vertical lines of the vertical
rectangle should be bounded by twice of the longest shortest path of a pair of nodes on the boundary surface of a network.

V. SIMULATIONS

A set of 3D sensor networks with representative topologies and various sizes are simulated in this work as shown in Figures 2 and 3. The genus numbers of these network models range from genus zero to genus two. One network model is particularly designed to have several inner holes (uncovered regions). We evaluate the performance of the proposed location-free cut graph based double-ruling scheme on these network models. Specifically, we denote the one given in Figure 3 (d) as Network I model. It is genus zero with 4$k$ number of nodes and the average number of neighbors of each node 13.79. We denote the one given in Figure 2 (a) as Network II model. It is genus one with 4$k$ number of nodes, and the average number of neighbors of each node 13.43. We denote the one given in Figure 3 (a) as Network III model. It is genus two with 6$k$ number of nodes and the average number of neighbors of each node 13.67. We denote the one given in Figure 2 (b) as Network IV model. It is genus zero with two inner holes. The size of Network IV model is close to 8$k$ and the average number of neighbors of each node is 13.7181.

Several parameters are particularly important for the performance of a distributed in-network data storage and retrieval scheme. One parameter is the cost to store data and another parameter is the cost to retrieve data. We call them the producer and consumer costs measured by the number of hops traveled to store or retrieve data. In general, there is a balance between the two costs. If one distributed in-network data processing scheme favors data storage, a producer in average will spend less cost than a consumer, and the vice versa. For non double-ruling based distributed-in-network data processing schemes, the number of data types generated inside a network and the frequency of the requests for aggregated data retrieval will greatly increase the consumer cost. Sec. V-A is designed to evaluate the costs of the producer and the consumer with different scenarios. Some scenarios that require frequent aggregated data retrieval are designed to favor double-ruling based schemes, while some scenarios do not.

Traffic load is also a very important parameter to evaluate the performance of a distributed in-network data storage and retrieval scheme. We measure the traffic load on each node by the number of messages passing through it. A balanced traffic load is preferred; otherwise nodes with extremely high traffic load will run out of battery very soon. Sec. V-B is designed to evaluate traffic load with various scenarios. Similarly, some scenarios favor double-ruling based schemes, while some scenarios do not.

It is desirable that a consumer spends a lower cost to retrieve data from a nearby producer than from a far away one. Assume that the distance between a producer and a consumer measured by hops is d, a distance-sensitive retrieval scheme would require that the retrieval cost of the consumer is O(d). Such distance-sensitive retrieval is discussed in Sec. V-C.

The motivation of double-ruling based schemes is to trade storage cost for efficient and successful data retrieval, especially the aggregated data retrieval. To find the balance between storage cost and data retrieval, another parameter - the data storage cost - is evaluated in Sec. V-D.

Different choices of parameters when computing the cut graph in Sec. II and mapping the cut open boundary surface to a planar rectangle in Sec. III may lead to different producer and consumer costs for the same given network. We discuss and evaluate such possibility in Sec. V-E.

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 [20], [21]. 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.

A. Producer and Consumer Costs

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 II 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) Aggregated Data: If there are more than one data type in network, as shown by Figure 7, 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.

Figure 7 clearly shows that the 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. While with the increases of data replication, cut graph based scheme has
a better fault tolerance and a more balanced load distribution across the network as discussed in Sec V-B. The length of the data replication curve of cut graph based scheme is bounded (Sec. VI).

B. Load Distribution

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. We randomly choose the data producer from the network and repeat the tests for ten times and get the average. 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 8 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 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 996 with cut graph scheme for Network III; and a load of 28,237 with GHT scheme and 41,253 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 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.

C. Distance-sensitive Retrieval

It is desirable that a consumer only needs to spend a low travel cost to retrieve data generated from a nearby producer. Such property is called distance sensitivity of retrieval. GHT based schemes in general do not have such property because a consumer travels to a hashed position by the desired data type and the data stored there may be produced by a nearby or a far away producer.

For our scheme, we first consider the case of a pair of consumer and producer close on the boundary surface. They are not necessarily boundary nodes. If they are inner nodes, we measure the hops between the two boundary nodes nearest to the producer node and the consumer node respectively. The following lemma says that if a pair of consumer and producer is close on the boundary surface, the consumer travels a distance less than the one to the producer on the boundary surface to access the data.

Lemma 1: The distance a consumer travels on the boundary surface of a network to hit the data replication curve of a producer is no more than the distance between the consumer and the producer on the boundary surface.

Proof: The mapping we conducted in Sec. III is conformal mapping. It is diffeomorphism preserving the neighborhood of a boundary node and the relative positions of its neighbors when virtually mapped to a planar rectangle. The shortest path between a pair of producer and a consumer on the boundary surfaces can be approximated by a straight line between their nearest boundary nodes on the planar rectangle. A data replication curve is a straight line passing through the boundary node nearest to the producer on plane. The distance between the boundary node nearest to the consumer and the data replication curve is always less than the distance between the pair of the boundary nodes. The two distances equal to each other only when the line between the pair of the boundary nodes is perpendicular to the data replication curve.

Two boundary nodes, one nearest to a producer and the other nearest to a consumer, may be close on the boundary surface but far away on the planar rectangle if they locate exactly on the two sides of the cut graph. Such case won’t affect the distance the consumer travels to retrieve the data, because the boundary surface is only virtually cut open and mapped to a planar rectangle. We design the data retrieval scheme as once a consumer hits the boundary side of the rectangle; the consumer can cross the boundary side and keep traveling along the same line with the same direction.

We randomly select 1000 pairs of producer and consumer on each network model. Figure 10(a) shows the simulation results based on the distances of the pairs of producer and consumer on the boundary surface of a network. It is obvious that the

<table>
<thead>
<tr>
<th>Network</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
<th>Cut Graph GHT</th>
<th>SR-GHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network II</td>
<td>20.3451</td>
<td>22.8416</td>
<td>94.4715</td>
<td>94.4715</td>
<td>25.9980</td>
<td>24.7264</td>
<td>101.8150</td>
<td>101.8150</td>
</tr>
</tbody>
</table>
Fig. 7. Comparison of average consumer costs with the increase of data types in the network.

Fig. 8. Comparison of load distribution with one information producer and one data type in the network.
average consumer cost increases with the distance between the producer and the consumer on the boundary surface of a network. Such trend doesn’t always hold when the distance of a pair of consumer and producer is large. They may be far away on the boundary surface, but the consumer is still close to the data replication curve.

However, if we consider the distance between a pair of producer and consumer in the volume of a network, such distance-sensitive retrieval is not necessarily applied. It is possible that a consumer is geographically close to a producer in the 3D volume of a network, but the consumer has to go a long distance to hit the data replication curve by the producer. Such extreme case happens when the consumer and the producer locate very close to each other but at the two sides of the medial axis of the 3D volume network 3, so their nearest boundary nodes are far away from each other on the boundary surface of the network. We still randomly choose 1000 pairs of producer and consumer on each network model. Figure 10(b) shows the simulation results based on the distances of the pairs of producer and consumer in the 3D volume of a network. The average consumer cost doesn’t show a strong linear correlation with the distance.

Fig. 9. Comparison of load distribution with one hundred information producers and ten data types in the network.

Fig. 10. (a) Consumer costs vs. distance between producer and consumer on boundary surface of a network. (b) Consumer costs vs. distance between producer and consumer in 3D volume of a network.

3The medial axis of a shape is the set of all points that have more than one closest point to the boundary of the shape.
D. Tradeoff Between Storage Cost and Consumer Cost

As discussed in Section IV, 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 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 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.

E. Different Choices of Parameters

As we introduced in Sec. II, the algorithm to compute the cut graph starts from one randomly chosen triangle of the triangular boundary surface of a network. The algorithm marks the triangle and keeps a width-first propagation with the center of the first marked triangle until all triangles of the network have been marked. We assume the algorithm always picks the triangle with the smallest boundary node id as the starting triangle in our previous simulations. To evaluate the effect of the choice of the starting triangle, we first randomly select 1000 pairs of producer and consumer from Network II model. We randomly pick a triangle of the boundary surface as the starting triangle to compute the cut graph and then compute the average producer and consumer costs of the chosen 1000 pairs. The testing is repeated ten times with the same set of pairs of producer and consumer but ten different sets of corner nodes when mapping the cut open boundary surface to a planar rectangle. Table III shows the mean, the standard deviation, and the median of the producer and consumer costs of the ten tests under different sets of corner nodes.

In summary, the different choices of the parameters slightly affect the performance of the proposed cut-graph based double-ruling scheme.

VI. DISCUSSIONS

A. 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 [36] to identification the bottleneck and then segment the network to parts. Double-ruling approach can be applied at individual parts.
B. Network Dynamics

As discussed in Sec. IV-E, 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 do not 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.

C. Network Density

As we mentioned in Sec. II, the algorithm provided in [24] to construct a triangular boundary surface of a network assumes that a triangular graph is a sub-graph of the initial connectivity graph of detected boundary nodes. Such assumption is true only when the node density of the network is not too low. The node densities of the network models we choose in simulations are all around 13. We have no problem to construct such triangular surfaces. In our previous simulations, however, such assumption may not be true when the node density of a network is around or lower than 8.

VII. CONCLUSION

We present a location-free cut graph based double-ruling approach for 3D sensor networks with general topology and geometry shapes. 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.

ACKNOWLEDGEMENTS.

Yang Yang and Miao Jin are partially supported by NSF CCF-1054996, CNS-1018306 and CNS-1320931. Yao Zhao and Hongyi Wu are partially supported by NSF CNS-1018306 and CNS-1320931.

REFERENCES


