Surface Skeleton Extraction and Its Application for Data Storage in 3D Sensor Networks

Wenping Liu*
Huazhong University of Science and Technology, China
wenpingliu2009@gmail.com

Yang Yang
Huazhong University of Science and Technology, China
yangyang8795@gmail.com

Hongbo Jiang
Huazhong University of Science and Technology, China
hongbojiang2004@gmail.com

Xiaofei Liao
Huazhong University of Science and Technology, China
xfliao@hust.edu.cn

Jiangchuan Liu
Simon Fraser University,Canada
jcliu@cs.sfu.ca

Bo Li
Hong Kong University of Science and Technology, Hong Kong
bli@cs.ust.hk

ABSTRACT

In-network data storage and retrieval are fundamental functions of sensor networks. Among many proposals, geographical hash table (GHT) is perhaps most appealing as it is very simple yet powerful with low communication cost, where the key is to correctly define the bounding box. It is envisioned that the skeleton has the power to facilitate computing a precise bounding box. In existing works, the focus has been on skeleton extraction algorithms targeting for 2D sensor networks, which usually delivers a 1-manifold skeleton consisting of 1D curves. It faces a set of non-trivial challenges when 3D sensor networks are considered, in order to properly extract the surface skeleton composed of a set of 2-manifolds and possibly 1D curves.

In this paper, we study the problem of surface skeleton extraction in 3D sensor networks. We propose a scalable and distributed connectivity-based algorithm to extract the surface skeleton of 3D sensor networks. First, we propose a novel approach to identifying surface skeleton nodes by computing the *extended feature nodes* such that it is robust against boundary noise, etc. We then find the maximal independent set of the identified skeleton nodes and triangulate them to form a compact representation of the 3D sensor network. Furthermore, to react to the dynamics of the sensor networks caused by node failure, insertion, etc., we design an efficient updating scheme to reconstruct the surface skeleton. Finally, we apply the extracted surface skeleton to facilitate the data storage protocol design. Extensive simulations show the robustness of the proposed algorithm to shape variation, node density and network dynamics, and its effectiveness for data storage application with respect to load balancing.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*

Keywords

3D Sensor Networks; Surface Skeleton; Data Storage

1. INTRODUCTION

In-network data storage and retrieval are fundamental functions of sensor networks. While centralized-based schemes suffer from bottleneck at nodes near the sink, distributed in-network data storage is desirable for its scalability and robustness, etc. Among many proposals, geographical hash table (GHT) [26] is appealing because 1) it can greatly reduce the communication and energy cost by avoiding frequent in-network flooding for information retrieval [30], and 2) it is very simple yet powerful [9]. GHT names events with keys and hashes the keys into geographic location; and the sensor node, referred to as *home node*, geographically closest to the hash of its key stores the (*key, value*) pair. GHT then uses GPSR [15] as the low-level routing scheme to greedily forward the data and query packets to the corresponding home node. Upon reaching a local minimum, GPSR adopts the perimeter mode forwarding strategy. Considering the dynamics of sensor networks (e.g., caused by node mobility, insertion or failure due to energy depletion, etc.), GHT proposes to replicate each key-value pair at nodes (referred to as *replica nodes*) on the home perimeter, in order to guarantee data persistency.

Despite its desirable properties, GHT has some disadvantages. For example, inherited from GPSR, in complex networks such as the Y-shaped network shown in Fig. 1(a), the boundary nodes will be overloaded due to the extensive usage of the perimeter forwarding [9] especially in 3D sensor networks where there are arbitrarily large number of perimeters to be explored [35]. If the bounding *box* defining the range of coordinates for hash functions is not properly computed, the imbalance of storage load, together with traffic

*Wenping Liu is also with Hubei University of Economics.

MobiHoc'14, August 11–14, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2632951.2632966.

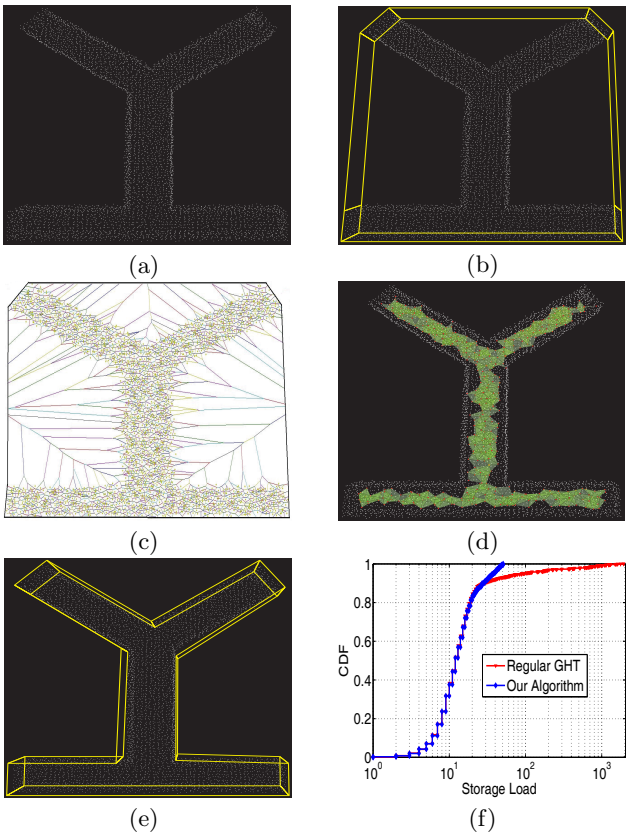


Figure 1: (a) The Y-shaped 3D network has 12,545 sensors with average node degree 15.36; (b) The bounding box with voids used in [26, 37]; (c) A cross-section of the Voronoi diagram with bounding box in (b); (d) The extracted Surface Skeleton; (e) The bounding box computed based on the Surface Skeleton in (d); (f) Comparison of storage performance with the two bounding boxes.

load, will become more severe [29]. This is because by hashing the keys to geographic coordinates, the underlying 3D space is actually divided into a set of Voronoi cells, within each of which there is one and only one sensor node, and thus each sensor node stores the data mapped to its Voronoi cell (see Fig. 1(c)). Obviously, the storage load of a node is proportional to the volume of its cell [29], i.e., the larger the volume of the cell is, the larger the storage and communication cost of the node. And if there exist voids unoccupied by sensors, as shown in Fig. 1(b) which is utilized by Regular GHT solutions such as [26] and typical segmentation algorithms such as [37], all data hashed into the voids will be finally stored at boundary nodes surrounding the voids, and consequently, these boundary nodes (especially when replica nodes on the boundary are used for data persistency) tend to have a higher storage and traffic load [37]. Please see Fig. 1(f). Similarly, DIM [20] also maps multi-attribute event to the sensing field; the nodes near an empty zone unoccupied by any sensor store the data mapped to this zone. *Accordingly, the key for data storage application is to precisely define the bounding box of sensor networks with an arbitrary shape, in order to evenly distribute the storage load of sensor nodes; Fig. 1 shows that the bounding box results could vary greatly with different methods.*

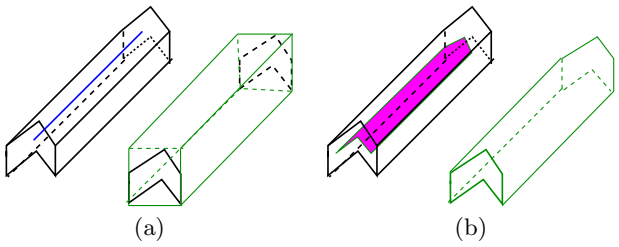


Figure 2: Bounding box computed based on C-Skeleton and S-Skeleton, respectively. (a) C-Skeleton (blue line in left) and the bounding box (green cubic in right); (b) The S-Skeleton (shaded surface in left) and the bounding box (green polyhedron in right).

The difficulties of computing the precise bounding box for a complex network mainly stem from the presence of concave valleys (and thus *bridges* between adjacent *peaks*) and/or, holes or tunnels, which often leads to the failure of finding a tight bound of the network based on convex hull, as shown in Fig. 1(b). At first glance, it seems quite intuitive to locate concave nodes by computing the concavity. Unfortunately, without coordinate information, it is rather difficult to calculate the *depth* of the concave valley, a traditional way to compute concavity in computer graphics, while the neighborhood size based algorithm and CATL [32] do not work well in long-and-narrow networks, making it a challenge to compute concave nodes in 3D sensor networks. On the other hand, the surface skeleton (referred to as *S-Skeleton*) of a 3D object is a generic and compact representation of the underlying object which can preserve the object's genus and the topological features very well [4]. In continuous 3D space, S-Skeleton, also called *medial surface* or *medial axis*, of a 3D object is defined as the interior points with at least two nearest boundary points, as shown in Fig. 2(b). Also we notice that another kind of skeletons of 3D objects is *curve-skeleton* (referred to as *C-Skeleton*), composed of 1D curves locally symmetric to the object [28]. While C-Skeleton has been used in 3D sensor networks to improve routing performance [24], inherently it is not a proper way to compute a tight bounding box; Fig. 2(a) shows an example. Between above two definitions of skeleton in 3D sensor networks, we emphasize that typically, the presence of a concave valley (or hole, or tunnel) will incur a *bent* S-Skeleton. Thus, if the location where the S-Skeleton deforms is identified, then the concave valley can be easily located, and accordingly, the computed bounding box is supposed to be tight and precise, as shown in Fig. 2(b). That is, we envision that S-Skeleton of a 3D sensor network, as shown in Fig. 1(d), can efficiently facilitate defining a tight and precise bounding box, as shown in Fig. 1(e), and balancing the storage loads, as shown in Fig. 1(f).

Related work. Skeleton has been widely used as an efficient way to facilitate routing [5], segmentation [40],localization [17] and navigation [19, 34], etc., in sensor networks. The usefulness of skeleton has inspired many algorithms for its computation in sensor networks, e.g., [12, 13, 22, 23, 39]. However, these algorithms primarily target at 2D sensor networks, and deliver a 1-manifold skeleton composed of 1D curves. More recently, there are many practical deployments of sensors on 3D environments [7, 33], triggering growing demands for studies on 3D sensor networks. Compared with 2D cases, it is much more difficult to compute

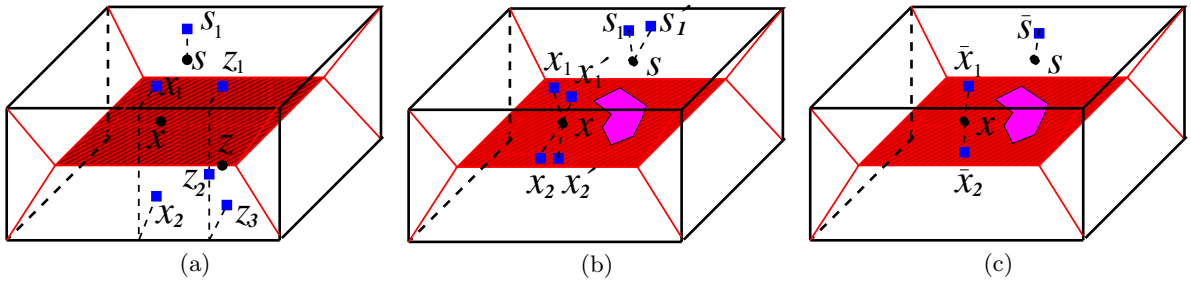


Figure 3: An illustration of the S-Skeleton of a cubic. There are thirteen sheets formed by the Y-curves and the boundary of the cubic while only the middle sheet is shaded. The solid red lines represent the Y-curves where two sheets meet. (a) Continuous space where the Euclidean distance between points are given. The feature points of x, z, s are shown by solid rectangles. Points x is an ordinary S-Skeleton point, z is a Y-curve point (and is also a junction point), and s is not an S-Skeleton point; (b) The distance between nodes are measured by integers (e.g., in hops). Node x in the shaded sheet should be an S-Skeleton node while s shouldn't. However, the non-skeleton node s has two extended feature nodes. The void bounded by the polygon in the shaded sheet is generated because of the even width of the cubic; (c) Node x has two connected components $\overline{x}_1, \overline{x}_2$ while s has only one.

the S-Skeleton of 3D objects [31], because the S-Skeleton of a 3D object consists of 2-manifolds (or skeletal sheets), and possibly 1D curves. Thus, it faces non-trivial challenges to extend the existing protocols to 3D settings.

We are aware that one close work to ours is done by Xia *et al.* [34], aiming at constructing the medial axis for navigation and data storage in 3D sensor networks. They first establish the unit tetrahedron cell (UTC) mesh structure and then iteratively "peel" off a layer of the UTC mesh structure to yield the medial axis. Principally, such a morphological thinning based method, attempting to realize Blum's grassfire model [3], is very sensitive to the distance metric and typically fails to accurately localize medial surface points [4], and it is also sensitive to boundary noise. That is, a small change in boundary surface may result in a considerable change in the surface skeleton. As such, a post-processing operation for pruning spurious branches is needed. Further, the unit tetrahedron cell (UTC) structure requires high node density and nice tetrahedron mesh [36], which is difficult to obtain when only connectivity information is available and thus small interior holes are not identified [35]. Last but not least, it cannot adapt to network dynamics since the UTC mesh must be constructed in advanced. When the network topology changes due to node failure or insertion, etc., the reconstruction of UTC mesh is costly owing to its high time and message complexity.

Our contributions. In this paper, we study the problem of S-Skeleton extraction in 3D sensor networks, and propose a connectivity-based, scalable and distributed S-Skeleton extraction algorithm which is robust again boundary noise and node density, etc., and can quickly react to the network dynamics. Whereafter, based on the extracted S-Skeleton, we propose the method to find the tight and precise bounding box followed by the solution for load-balanced data storage. Different from [34], we do not require any special structure like unit tetrahedron cell, which is difficult to obtain in a 3D network, especially with low node density. In our work, each node identifies itself as an S-Skeleton node by computing the *extended feature nodes* instead of the exact feature nodes, due to the discreteness of sensor networks and the presence of boundary noise. Then, the maximal independent set of the identified S-Skeleton nodes is constructed, followed by a triangulation procedure to form a compact representation of the underlying 3D sensor network. The merits of our S-Skeleton extraction algorithm are that it is robust against boundary noise and does not suffer from low node density, and thus can be applied in more generic cases than [34]. Further, to react to the network dynamics caused by node failure or insertion, etc., for the first time we propose an efficient updating scheme to reconstruct the S-Skeleton. Finally, we apply the extracted S-Skeleton for the design of load-balanced data storage protocol in 3D sensor networks. We conduct extensive simulations to show the robustness of the algorithm to shape variation, node density, network dynamics, and so on.

The reminder of the paper is structured as follows. In Section 2 we briefly introduce the motivations and preliminary knowledge of this work, and detail our algorithm in Section 3. Section 4 is devoted to the application of the S-Skeleton. To show the efficiency of the proposed algorithm, we conduct extensive simulations in Section 5. Finally, Section 6 concludes the paper.

2. MOTIVATIONS AND PRELIMINARIES

In continuous domain, as mentioned earlier, the S-Skeleton of a 3D object $D \subset R^3$, denoted by $SK(D)$, is a collection of the interior points having more than one nearest boundary point (referred to as *feature point*). More formally, we first define the distance transform $DT : D \to \mathbb{R}$ as $DT(x) = \min_{y \in \partial\mathcal{D}} d(x, y)$ if $x \in \mathcal{D} \backslash \partial D$ and 0 otherwise, where ∂D is the boundary surface of D and $d(x, y)$ is the (Euclidean) distance of point x to y. Further, we define the feature transform $F : D \to \mathbb{P}(\partial D)$ where $\mathbb{P}$ is the power set, assigning to each point $x \in D$ the set of feature points on ∂D to x. That is, $F(x) = \{y \in \partial D | d(x, y) = DT(x)\}$. Since an S-Skeleton point has more than one feature point, its feature size should be larger than one, and if a point has only one feature point, it must be a non-skeleton point. Thus, there is an easy way to identify an S-Skeleton point based on the feature size. Formally, we have

DEFINITION 1. *A point x is an S-Skeleton point if its feature size $|F(x)| \geq 2$.*

Based on Definition 1, we can deliver the S-Skeleton composed of 2-manifolds, or *sheets*, and possibly 1D curves. Fig. 3 (a) shows an example of the S-Skeleton of a cubic. In degenerated cases like a cylinder, the S-Skeleton may contain only curves, which is quite unusual and thus not

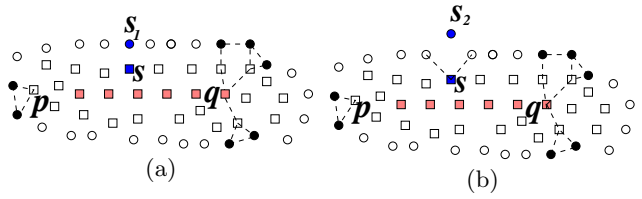


Figure 4: The principle of skeleton node identification in [40] and its drawback. Boundary nodes are shown by circles, ordinary interior nodes and skeleton nodes are shown by empty rectangles and solid rectangles, respectively. (a) The nodes in solid pink circles, e.g., q, are skeleton nodes such that their feature nodes form more than one interval, and p is not a skeleton node since there is only one interval of feature nodes; (b) The small bump in the boundary (shown by the solid blue circle) generates a new skeleton node s indicated by the solid blue rectangle.

considered in this paper. Two sheets may meet along a Y-intersection curve [6], called *Y-curve*, or Medial Scaffold [10, 11], on which any point has at least three feature points and is referred to as a *Y-curve point*. Hence, a sheet is bounded by some Y-curves and possibly the boundary of the object. In particular, we also call as *junction points* the Y-curve points where more than two sheets intersect.

Definition 1, however, can not be directly applied for sensor networks where the coordinate information of each sensor is often costly to acquire and in most scenarios, we can only use connectivity information. Many factors, such as the rounding error of distance between nodes, low node density, boundary noise, etc., pose great challenges to identify skeleton nodes in sensor networks. We first take a look at the challenges and present the solutions in 2D sensor networks, and then we extend these solutions to the 3D counterparts.

First, due to the rounding error of distance between nodes, low node density or even "width", etc., many interior nodes may only have one nearest boundary node (referred to as *feature node*), potentially degrading the performance of skeleton extraction. In order to tackle this problem, we can compute the *extended feature nodes* defined below.

Definition 2. *For an interior node p having a minimum hop count distance k to the boundary, its extended feature nodes of p, denoted by $\overline{F}(p)$, are the boundary nodes which are k or $k+1$ hops to p.*

Second, there are many unstable nodes (e.g., p in Fig. 4(a)) having two or more close feature nodes. As a result, the delivered skeleton will contain too many spurious branches, especially when the extended feature nodes are introduced. An approach to tackle this problem is proposed by Zhu *et al.* [40] where a node identifies itself as a skeleton node (e.g., q in Fig. 4(a)) if it has two or more intervals, i.e., ordered and consecutive sequences, of feature nodes; the nodes with only one interval, e.g., p in Fig. 4(a), are ordinary nodes. Note that this approach is invalid for 3D sensor networks as the boundary nodes cannot be ordered to form intervals. Besides, it suffers from boundary noise in 2D sensor networks, as mentioned in [21, 22], and more greatly in 3D cases. That is, a small bump in the boundary will cause an unwanted skeleton branch needed to be pruned. For instance, when the boundary node s_1 in Fig. 4(a) *moves* to s_2 in Fig. 4(b), the only one consecutive sequence of feature nodes of node s in Fig. 4(a) will be divided into two intervals in Fig. 4(b), and thus s will identify itself as a skeleton node. The solution to this problem is to introduce an equivalence relation $\sim_\epsilon$ as in [27]:

Definition 3. *If the geodesic distance between two boundary nodes a, b is less than $\epsilon(>0)$, we say that a and b are geodesic ϵ-equivalent, denoted by $a \sim_\epsilon b$.*

Similarly, for each node p, if the minimum of the geodesic distances between nodes belonging to two intervals I_1, I_2 is less than ϵ, we say that the two intervals are geodesic ϵ-equivalent, and can be treated as one *ϵ-interval*. A node identifies itself as a skeleton node if and only if it has two or more ϵ-intervals. Clearly, we have

Theorem 1. *A small bump in the boundary, separating an interval I of p into two intervals I_1, I_2 with geodesic distance less than ϵ, does not change the identity of p.*

Proof. Since the two intervals I_1, I_2 have a geodesic distance less than ϵ, they are regarded as one ϵ-interval. Thus, if p has only one interval before the bump, then after the bump p still has only one ϵ-interval and remains a non-skeleton node; otherwise, p keeps its identity as a skeleton node, which proves the claim. □

Theorem 1 shows that such a process of skeleton node identification is robust to boundary noise. One can easily prove that it is also robust to many other factors, such as low node density or node failure, etc.

Note that these challenges will become more severe in 3D environments. That is, some true S-Skeleton nodes may not identify themselves as skeleton nodes due to even "width", etc. Consequently, there might be holes in the skeletal sheets, as shown in Fig. 3 (b), resulting in that the S-Skeleton does not preserve the network's genus. At the same time, there might be more unstable S-Skeleton nodes and thus many unwanted faces. Further, it is noted that the solutions to these problems in 2D sensor networks can not be directly applied in 3D cases, as here the boundary surfaces are 2-manifolds and the boundary nodes cannot be ordered in sequence, as mentioned earlier.

Now we extend the solutions in 2D sensor networks to 3D cases. Observe that in 2D sensor networks, the boundary nodes in a consecutive sequence must be connected, thus an interval of boundary nodes implies a connected component of boundary nodes. Hence, in 3D sensor networks, we can identify an S-Skeleton node based on the number of connected components. More specifically, for each node p, we compute its extended feature nodes $\overline{F}(p)$. Clearly, even for an ordinary interior node, it may have more than two extended feature nodes and any two of them are not necessarily neighboring, as shown in Fig. 5 (b). To address this, we first group these extended feature nodes into connected components, as shown in Fig. 3 (c), and then use equivalence relation $\sim_\epsilon$ to merge these connected components into bigger component(s); two components are ϵ-equivalent if their distance (defined as the minimal geodesic distance between pairwise nodes belonging to different components) is less than ϵ, and form a virtual component called *ϵ-component*. Denote by $\overline{F}_\epsilon(p)$ the set of virtual components of node p. Obviously, each ϵ-component here serves as the feature point in the continuous domain. As such, we can identify an S-Skeleton node as follows.

Proposition 2. *For an interior node p, if $|\overline{F}_\epsilon(p)| \geq 2$, then p identifies itself as an S-Skeleton node.*

By Proposition 2, we can identify the S-Skeleton nodes based on only one parameter ϵ, which is used to avoid suffering from rounding noise, boundary noise, and node density, etc. Clearly, for Definition 1, $\epsilon = 0$, and only the exact feature points are used. That is, the skeleton nodes identified based on Proposition 2 is a subset of those identified based on Definition 1, while Proposition 2 can yield a more stable result in 3D sensor networks, which is validated by extensive simulations in Section 5. With the identified S-Skeleton nodes, we perform a triangulation procedure to form a compact representation, i.e., the S-Skeleton, of a 3D sensor network, which will be detailed in next section.

3. ALGORITHMS

In this section, we present the details of our algorithm for S-Skeleton extraction in 3D sensor networks. Since boundary recognition is out of the scope of the paper, we thus assume that boundary nodes have already been recognized, e.g., by [14, 18]. Further, we do not assume that the location information of nodes are known, and our work is based on mere connectivity information.

3.1 S-Skeleton Node Identification

As mentioned in Section 2, an interior node is an S-Skeleton node if it has at least two ϵ-components formed by extended feature nodes. Thus, the first step of our algorithm is to identify the extended feature nodes of each interior node. This can be done in a distributed fashion as follows. Each boundary node issues a flooding within the network at roughly the same time. The flooded message includes the ID of the boundary node, and a counter, which is set to be zero by default, to indicate the distance of a node to the origin of the flooded message. When receiving a flooded message from a boundary node, say q, each node p executes the following rules:

- If p has yet not received the flooded message from any boundary node, it appends q to the list of nearest boundary nodes of p (denoted by $List(p)$), increases the counter by one and forwards the updated message to its neighbors;
- else if the distance of p to q is equal to, or larger by one than, the minimal distance of p to the nodes in $List(p)$, p keeps record of the boundary node q, increases the counter by one and forwards the updated message to its neighbors;
- else p simply discards the arrived message.

Consequently, each interior node keeps record of its extended feature nodes and the distance (in hops) to the feature nodes.

Subsequently, these extended feature nodes issue a limited flooding to construct connected component(s), followed by a hop-by-hop expansion process as given in [24] to merge these components into ϵ-component(s). More specifically, each component is firstly assigned a unique identifier, and then each extended feature node initiates a flooding message including its identifier and a counter (initialized to be zero), which indicates how far (in hops) the message has travelled. When a boundary node p receives a flooded message from a boundary node q which has been assigned an identifier and has a counter no greater than ϵ, it executes the following rules:

- if p has no identifier, then p will be assigned the same identifier as $q's$, increase the counter by one, and forward the updated message to its neighboring boundary nodes;
- else if p and q have different identifiers, and the sum of the counters of p and q is less than ϵ, then p increases the counter by one, and forwards the updated message to its neighboring boundary nodes;
- else p simply discards the arrived message from q.

This way, the communication cost of ϵ-component construction can be very low. An interior node with more than one ϵ-component identifies itself as an S-Skeleton node, as shown in Fig. 5(a); and the interior node with only one component is a non-skeleton node, as shown in Fig. 5(b). Finally, Fig. 5(c) draws the identified S-Skeleton nodes of the Y-shaped network in Fig. 1(a).

3.2 S-Skeleton Establishment

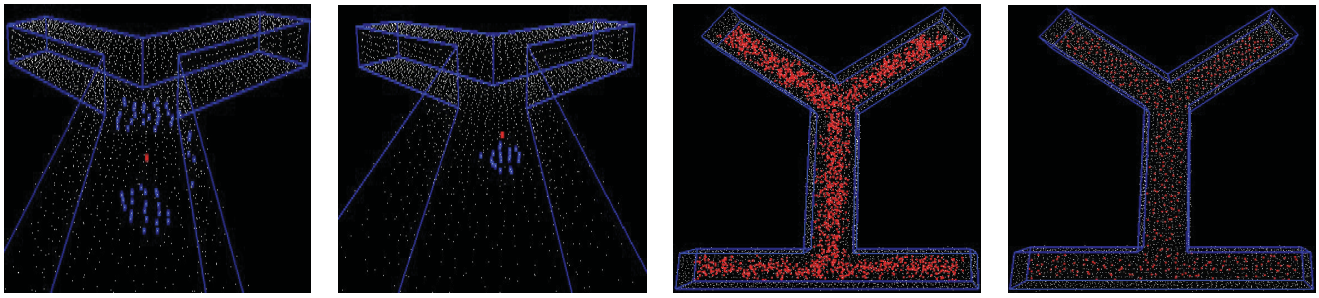
With the identified S-Skeleton nodes, we now connect them to form a set of 2-manifolds, i.e., the S-Skeleton. Note that the identification of an S-Skeleton node is based on the extended feature nodes, in order to guarantee that the true S-Skeleton nodes will not be ignored. Unfortunately, this may incur that some S-Skeleton nodes are redundant in a given scenario, e.g., with parallel boundaries, which brings a non-trivial challenge to construct the S-Skeleton. To address this issue, we propose to construct the maximal independent set of the S-Skeleton nodes, followed by triangulating them to form the S-Skeleton.

Given the undirected S-Skeleton graph $G_s = (V_s, E_s)$ where V_s denotes the set of the identified S-Skeleton nodes, and E_s is the set of links between S-Skeleton nodes, an independent set is a subset $V_s' \in V_s$ such that no nodes in V_s' are adjacent; and a maximum independent set is a maximum-cardinality independent set [1]. We will not compute the maximum independent set of V_s, which is an NP-hard problem [16]. Instead, we find the maximal independent set, which is an independent set where no node can be inserted without violating the independence. Since the S-Skeleton consists of 2-manifolds, the maximal independent set of the S-Skeleton nodes can be similarly constructed in a distributed fashion as given in [38]. As a result, any pair of independent nodes have a separation greater than one and smaller than three hops, as shown in Fig. 5 (d). One advantage of this procedure, as pointed out in [38], is to maintain the independent nodes uniformly distributed. Clearly, the independent nodes serves as sites which decompose the S-Skeleton nodes into Voronoi cells. With the Voronoi diagram, we can easily obtain its dual, the Delaunay triangulation, e.g., by the method in [38], and the S-Skeleton is thus generated, as shown in Fig. 1(d).

3.3 Complexity Analysis

THEOREM 3. *The proposed S-Skeleton extraction algorithm has a linear time and message complexity.*

PROOF. The algorithm has two steps: S-Skeleton node identification and S-Skeleton establishment. During the first step, each interior node only forwards a small number of packets, and thus in total the flooding from boundary nodes



(a) (b) (c) (d)

Figure 5: An illustration. (a) An S-Skeleton node (shown by the big red rectangle) with three feature node components (in blue); (b) A non-skeleton node (shown by the big red rectangle) with one feature node component; (c) The S-Skeleton nodes; (d) The maximal independent set of the S-Skeleton nodes.

incurs an $O(N)$(N is the number of sensors) time and message complexity. To compute the number of the connected components of the extended feature nodes, for each node, the algorithm only incurs at most $O(N_f)$ time and message complexity, and thus $O(N_f N)$ for all nodes, where $N_f (\ll N)$ denotes the largest number of extended feature nodes among all interior nodes. For the second step, the construction of the maximal independent set and the triangulation both incur a linear time and message complexity [38, 29]. In total, the algorithm has a linear complexity. □

3.4 Network Dynamics

As well known, wireless sensors networks are resource-constrained, and many factors can cause their failures. For instance, sensor nodes are fragile and may fail due to energy depletion or destruction by external events (e.g., natural disasters, adversarial attacks, etc. [2]), and congestion may also cause packet loss. Besides, occasionally some sensors may be added to the network for a better performance [29]. In other words, nodes/links may come and go [8], which consequently incurs the dynamics in the network topology. To show the performance of our algorithm under such harsh environments, we consider the S-Skeleton reconstruction problem for dynamic networks. When the network topology changes, the algorithm does not necessarily compute the S-Skeleton from scratch, which is otherwise time/message costly. Instead, as will be proven, the dynamic of the network topology will cause a local impact on the S-Skeleton, and thus only a local operation on the reconstruction of the S-Skeleton is conducted to speed up the reconstruction process.

We first present some results in continuous 3D space.

LEMMA 4. *A local deformation on the boundary surface of a 3D space only poses a local impact on the S-Skeleton .*

PROOF. Let x be an arbitrary S-Skeleton point. Then, x has at least two feature points. Suppose that the boundary surface deforms at a local surface, say, from LS to LS'. For an S-Skeleton point x, let $T(x), T'(x)$ denote its distance transform before and after deformation, respectively. Generally, there are four cases:

1) **Case 1.** $T(x) = T'(x)$ and $F(x) \bigcap LS = \emptyset$. This means that no feature point of x in LS, and the deformation does not change $x's$ existing feature points because they are still at the minimum distance to x. Although some new feature point may lie in LS', but x has more than one feature point, and thus is still an S-Skeleton point.

2) **Case 2.** $T(x) = T'(x)$ and $F(x) \bigcap LS \neq \emptyset$. This means that at least one feature point lie in LS, and after the deformation, the feature size $|F(x)|$ decreases. If $|F(x) \setminus \{F(x) \bigcap LS\}| \geq 2$, then x is still an S-Skeleton point; otherwise, it is not.

3) **Case 3.** $T'(x) < T(x)$. This means that after deformation, x has the minimum distance to LS', and the new feature points $F'(x) = \{y \in LS' | d(x,y) = T'(x) = min_{y \in LS'} d(x,y)\}$. If $|F'(x)| \geq 2$, then x is still an S-Skeleton node; otherwise, it is not.

4) **Case 4.** $T'(x) > T(x)$. This means that all feature points of x lie in LS, and the boundary deforms such that the distance of the new boundary LS' to x is larger than $T(x)$. Clearly, x should re-identify its identity since after deformation, it is unclear whether $|F(x)| \geq 2$ or not.

In summary, if the S-Skeleton point x has at least one feature point in LS, and/or the deformation will change the distance transform of x, then x may re-identify its identity; otherwise, x remains an S-Skeleton point. Clearly, the affected S-Skeleton points are all within a small distance to LS or LS', thus the local deformation on the boundary surface will only have a local impact on the S-Skeleton. □

LEMMA 5. *The emergence of a new hole or tunnel has only a local impact on the S-Skeleton.*

PROOF. Clearly, the emergence of a new hole or tunnel will generate a new boundary. For an S-Skeleton point x, some feature points may disappear, while new feature points may appear at the new boundary. Thus, similar to Lemma 4, only the S-Skeleton points closest to the new boundary should re-identify their identities because of the change of distance transform, and thus the impact of the emergence of a new hole or tunnel on the S-Skeleton is local. □

Similarly, we have

LEMMA 6. *The disappearance of a hole or tunnel has only a local impact on the S-Skeleton.*

Lemma 4 to 6 imply that in sensor networks, the failure of boundary nodes [1] and interior nodes [2], and the insertion

[1]It will incur neighboring interior nodes to be new boundary nodes, and is thus a discrete analog of the boundary deformation in continuous space.

[2]It is a discrete analog of the emergence of a new hole or tunnel in continuous space.

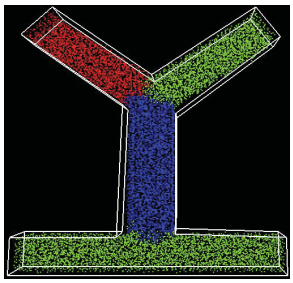


Figure 6: The segmentation result and the bounding box.

of new nodes into a void for better coverage [3] only have a local impact on the delivered S-Skeleton. Thus, we have

THEOREM 7. *The network dynamics caused by node failure or insertion have a local impact on the S-Skeleton.*

It can be easily shown that a non-skeleton node might become an S-Skeleton node due to the network dynamics such that the connectivity of the S-Skeleton is maintained and the genus can be preserved. Clearly, such a node must be near where the topology changes. Roughly speaking, if the distance transform of a node changes, its identity may change as well. Thus, we can narrow the S-Skeleton reconstruction down to the nodes whose distance transform change and the nodes whose feature nodes fail. When new S-Skeleton node identify themselves, and at the same time *old* S-Skeleton nodes give up their identities, we can easily reconstruct the S-Skeleton. Note that we do not necessarily reconstruct the S-Skeleton from scratch. In fact, we have

THEOREM 8. *The network dynamics affect S-Skeleton reconstruction locally.*

PROOF. When an old S-Skeleton node p fails, only the triangles containing p need to be reconstructed; if there is a new S-Skeleton node q nearest to an S-Skeleton node p, then q will not join the independent set otherwise the independence is violated, and there is no need to update the triangles. Hence, the impact of network dynamics on the S-Skeleton reconstruction is local. □

As such, only a local triangulation is conducted to form an updated S-Skeleton.

4. THE APPLICATION OF THE S-SKELETON FOR DATA STORAGE

As mentioned in Section 1, the S-Skeleton can be used to define a tight and simple bounding box, which can balance the storage node in GHT [26] and DIM [20].

Clearly, the bounding box should better be regular as the logical data space is often regular [29]. As such, our objective is to identify a small number of *critical boundary nodes* used to define a tight and simple bounding box. To that end, we first identify the boundary edge of the S-Skeleton which is not shared by any other triangles. The nodes on the boundary edges are boundary nodes (referred to as *S-boundary nodes*) of the S-Skeleton. Then we compute for each S-boundary node p its k-hop curvature, denoted by $c_k(p)$, which is defined below.

DEFINITION 4. *For an S-boundary node p, denote by $N_k(p)$ the nodes at most k hops away from p; let $p_1, p_2 \in N_k(p)$ be two S-boundary nodes such that the shortest paths between any two nodes of p_1, p_2, p will not pass through the S-Skeleton. Denote by $D_p^k(p_1, p_2)$ the set of nodes on the shortest path between p_1 and p_2 using the nodes in $N_k(p)$ (When $N_k(p)$ is disconnected, some auxiliary nodes can be used as in [25]). Then the k-hop curvature of node p, i.e., $c_k(p)$, is defined as*

$$c_k(p) = \frac{|D_p^k(p_1, p_2)| - 1}{\pi \times k} \quad (1)$$

With the k-hop curvature, we define the concave/convex node of the S-Skeleton (referred to as S-concave/S-convex node), as follows.

DEFINITION 5. *For the given $\delta_1, \delta_2 \in (0, 1)$ and k, an S-boundary node p is an S-convex node if $c_k(p) < 1 - \delta_1$, or an S-concave node if $c_k(p) > 1 + \delta_2$.*

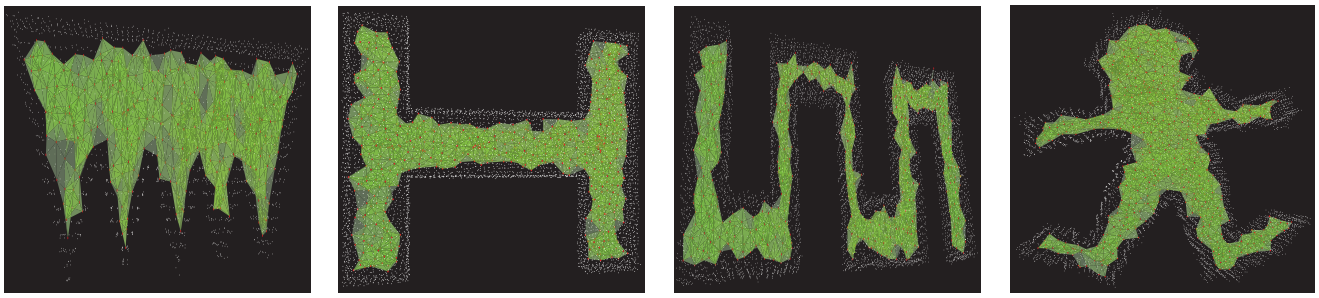
Based on the identified S-concave nodes, the S-Skeleton can be decomposed into regular pieces by connecting nearby S-concave nodes. Then, S-Skeleton nodes flood within the network, and the nodes nearest to the same regular piece of the S-Skeleton naturally form a connected component. At the same time, each boundary node computes its distance (referred to as *local feature size*) to the S-Skeleton. The boundary nodes, whose nearest S-Skeleton nodes are S-convex/S-concave nodes and their local feature sizes are locally maximal, identify themselves as *critical convex/concave nodes*. As such, the bounding box is obtained where the vertices of the bounding box are critical convex nodes and critical concave nodes, and interestingly, the network is decomposed into regular components, as shown in Fig. 6.

With the computed bounding box, we first map each data item x produced by a node to a geographic location g in the sensing space bounded by the bounding box via a random hash function h, i.e., $h(x) = g$. Then the node nearest to the location g, referred to as home node, stores the data. Finally, similar to [5, 22, 24], we adopt the S-Skeleton based routing protocol as a low-level routing scheme for the producer to forward the data to the home node such that the traffic load is evenly distributed and the delivery can be guaranteed. For the robustness to node failures, we let the nodes *surrounding* the location g, referred to as *replica nodes*, also store the data. As the bounding box is tight, the sensing space can be divided into Voronoi cells with almost equally volume. Thus, the storage load of the nodes can be well balanced. We validate the proposed scheme by extensive simulations in Section 5.

5. SIMULATIONS

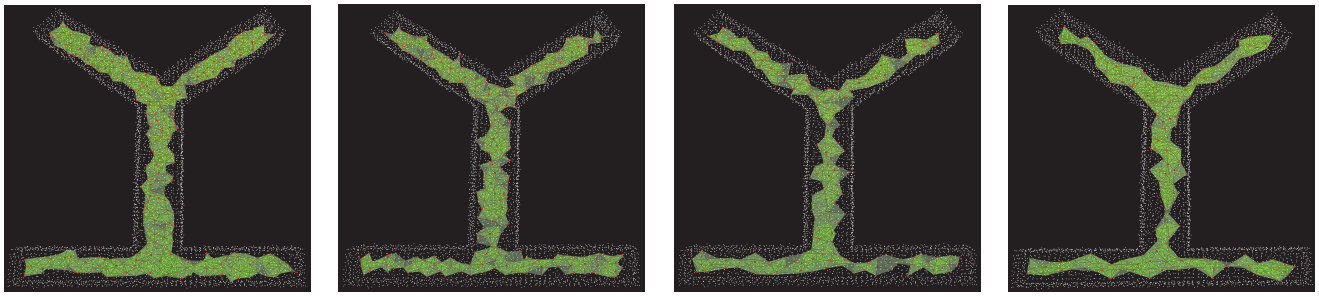
We conduct extensive simulation tests on various 3D scenarios to show the performance of the algorithm. In our simulations, sensor nodes are randomly deployed inside the underlying 3D space, and the boundary nodes are recognized by [14]. The parameter ϵ is set to be one by default. We do not compare with [34] since it requires high node density and nice tetrahedron mesh, as mentioned before, which are not

[3] If the insertion incurs the disappearance of a hole or tunnel, then it is a discrete analog of the disappearance of a hole or tunnel in continuous space; otherwise, a hole or tunnel shrinks, which is a discrete analog of the boundary deformation in continuous space.



(a) Seabed (b) H (c) Snake (d) Man

Figure 7: The performance under various 3D scenarios. The S-Skeletons are shaded. (a) 5,537 nodes, average node degree 15.37; (b) 16,156 nodes, average node degree 15.69; (c) 19,313 nodes, average node degree 15.78; (d) 15,288 nodes, average node degree 15.31.



(a) (b) (c) (d)

Figure 8: The performance under different node densities. (a) avg.deg 13.12; (b) avg.deg 17.38; (c) avg.deg 21.64; (d) avg.deg 24.76.

available in our settings where the average node degree are all in between 15 and 16, pretty low for 3D sensor networks. We first examine the robustness of the algorithm to shape variation and node density, and then test the performance under dynamic networks. Finally, we show the applicability of the S-Skeleton for data storage.

Robustness to shape variation. We conduct the proposed algorithm under difference scenarios, namely, Seabed, H, Man, and snake, as shown in Fig. 7, with network size ranging from 5,537 to 19,313 and average node degree in between 15 and 16. We can clearly see that the derived S-Skeletons in Fig. 7(a) to (d) correctly capture the main topological features, e.g., the irregularity, of the underlying networks, showing that our algorithm is robust to shape variation. Since the average node degree are all relatively low for 3D sensor networks, one can imagine that our algorithm can work for small-scale networks as well.

Robustness to node density. We vary the communication radio range to generate four networks with different node densities. Please see Fig. 8. Fig. 8(a) shows the result under low density, where the average node degree is only 13.12, below which the network becomes disconnected. We can observe that the extracted S-Skeleton correctly captures the salient features of the network. From Fig. 8(b) to (d), we can see that with the increasing of node density, the S-Skeleton becomes *thinner*, because less and less S-Skeleton nodes are joined to the maximal independent set, without violating the independence, for constructing the S-Skeleton. But overall our algorithm delivers a very stable result and the irregularity of the underlying network is correctly captured by the S-Skeleton, showing that our algorithm is robust to node density.

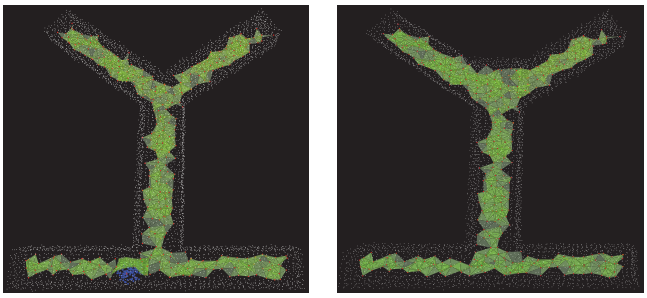
Algorithm	Y	H	Man	Snake	Seabed
Our algorithm	0.0058	0.0053	0.0057	0.0053	0.0057
Bottleneck	0.0111	0.0107	0.0079	0.0122	0.0067
Regular GHT	0.0124	0.0117	0.0109	0.0122	0.0107

Table 1: Comparison study on maximum storage load.

Reaction to network dynamics. We next show the performance of our algorithm under dynamic networks by allowing node failure and insertion. In Fig. 9(a), the blue nodes are *dead*, e.g., due to energy depletion. Compared with Fig. 1(c), the reconstructed S-Skeleton correctly reflects such a dynamic behavior by *moving* away from these failed nodes. In Fig. 9(b), some nodes are added to the network in the upper *valley* of the network, and there the S-Skeleton accordingly becomes *fatter* than Fig. 1(c) while the other remains the same.

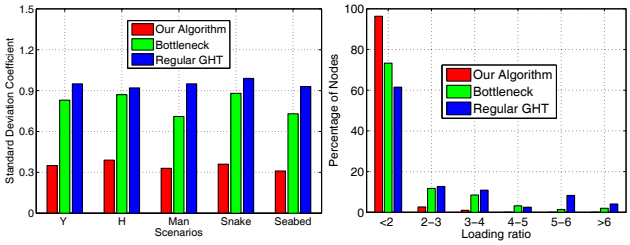
Application for data storage. We apply the S-Skeleton to compute a tight and precise bounding box. As the storage of each node is proportional to the volume of the Voronoi cell generated by the node, we expect that our bounding box can guarantee a balanced storage load. To show its advantage, we compare our algorithm with Regular GHT [26], and the method in [37] (referred to as *Bottleneck*) which segments a 3D network by identifying bottlenecks, and then computes the bounding box of the network. No replica nodes are considered in the comparison study.

We propose three metrics, namely, *maximum storage load*, *standard deviation coefficient* and *loading ratio of a node*, to quantitatively measure how the storage load spreads across



(a) (b)

Figure 9: The performance of the algorithm under dynamic networks. (a) The new S-Skeleton when some nodes (shown in blue) fail; (b) The new S-Skeleton when some nodes are inserted into the upper valley.



(a) (b)

Figure 10: The comparison study on storage load. (a) Standard deviation coefficient distribution; (b) Load ratio distribution.

the network. The maximum storage load is closely related to the maximum Voronoi cell volume, and the total storage load is normalized to be 1. The standard deviation coefficient (referred to *sdc*) of storage load, is defined as the ratio of the standard deviation of the storage load of nodes to their average. A smaller sdc means the loads are more evenly distributed while a larger sdc indicates that the distribution is more uneven, namely, the storage load is more imbalanced. The loading ratio of a node is the ratio of the storage load of a node to the average storage load of the network.

Table 1 depicts the comparison study on the maximum storage load for the five networks in Fig. 5 and Fig. 7. We observe that our algorithm outperforms the other two algorithms in the five networks. Especially, for Y, H and Snake-shaped networks, Bottleneck produces near the same maximum storage load as GHT since no bottlenecks are correctly identified and thus the bounding box is not accurately computed. In Man and Seabed-shaped networks, Bottleneck performs better than GHT because it identifies some bottlenecks and thus the bounding box is tighter than GHT, but Bottleneck performs worse than our algorithm since it ignores some concave points, e.g., at two *knees* of the Man-shaped network, resulting in an imprecise bounding box.

Fig. 10(a) presents the sdc distribution for the five networks by these algorithms. Clearly, we can see that our algorithm yields the smallest sdc for all networks, because the bounding box computed based on S-Skeleton tightly bound the network and no large voids exist; Regular GHT produces the worst result since the computed bounding box based on convex hull incurs large voids, resulting in that the boundary nodes are inevitably overloaded. As for Bottleneck, it identifies a few bottlenecks in the Man and Seabed-shaped networks, and thus the quality of the bounding boxes is fair. That is the reason why the sdc is relatively small. In the other three networks, however, there is no bottleneck identified and thus the result is undesirable.

Fig. 10(b) describes the load ratio distribution of the nodes in the investigated five networks by the three algorithms. As expected, our algorithm produces a near-ideal result, observing that over ninety percent of nodes have a loading ratio smaller than 2. Since the bounding box computed by our algorithm is tight and precise, the Vonoroi cell of each node has almost equal volume, and hence storage load can be evenly distributed. The result by Bottleneck is again fair where about seventy percent of nodes have a small load. But there do exist nodes having a large loading ratio (greater than 6), because some *concave points* are not identified. Thus, the computed bounding box possibly incurs *bridges* and voids. As a result, the nodes on the *pockets*, i.e., the boundary nodes surrounding the voids, are heavily loaded. Regular GHT produces a long-tailed loading ratio distribution where only near sixty percent of nodes have a small load ratio. Besides, five percent of nodes, mainly on the boundary of voids caused by the poorly computed bounding box, are overloaded. Overall, the load ratio distribution is skewed, implying that the storage load by Regular GHT is highly unbalanced. In summary, our algorithm outperforms the other two algorithms in terms of load balance.

6. CONCLUSION

We present a connectivity-based surface skeleton extraction algorithm in 3D sensor networks. The identification of skeleton node is based on the computation of the extended feature nodes such that it is robust against boundary noise, node density, and so on. To react to the dynamics of the sensor network caused by node failure or insertion, etc., we propose an efficient updating scheme to reconstruct the surface skeleton. We finally apply the surface skeleton to find a tight bounding box, which is then used for load-balanced data storage protocol. Extensive simulations are conducted to validate the performance of the proposed algorithm.

7. ACKNOWLEDGEMENTS

The authors thank the reviewers for their comments. This work was supported in part by the National Natural Science Foundation of China under Grant 61202460 and Grant 61271226; by the China Postdoctoral Science Foundation under Grant 2014M552044; by the Fok Ying Tung Education Foundation under Grant 132036; by the CCF-Tencent Foundation under Grant CCF-TencentAGR20130101; by the Fundamental Research Funds for the Central Universities under Grant 2014XJGH003, Grant 2014QN158 and Grant 2014QN164; by the Hong Kong Scholars Program under Grant XJ2012019; and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). The corresponding author is Hongbo Jiang.

8. REFERENCES

[1] D. Andrade, M. Resende, and R. Werneck. Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18(4):525–547, 2012.

[2] N. Azimi, H. Gupta, X. Hou, and J. Gao. Data preservation under spatial failures in sensor networks. In *Proc. of ACM MOBIHOC*, 2010.

[3] H. Blum. Biological shape and visual science (part i). *Theoretical Biology*, 38:205–287, 1973.

[4] S. Bouix and K. Siddiqi. Divergence-based medial surfaces. In *Proc. of European Conference on Computer Vision*, 2000.

[5] J. Bruck, J. Gao, and A. A. Jiang. Map: Medial axis based geometric routing in sensor betworks. In *Proc. of ACM MOBICOM*, 2005.

[6] J. DAMON. Global medial structure of regions in r3. *Geometry and Topology*, 10:2385–2429, 2006.

[7] M. Doddavenkatappa, M. Chan, and A. Ananda. Indriya: a low-cost, 3d wireless sensor network testbed. In *Proc. of ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities(TRIDENTCOM)*, 2011.

[8] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. Glider: Gradient landmark-based distributed routing for sensor networks. In *Proc. of IEEE INFOCOM*, 2005.

[9] Q. Fang, J. Gao, and L. J. Guibas. Landmark-based information storage and retrieval in sensor networks. In *Proc. of IEEE INFOCOM*, 2006.

[10] F.F.Leymarie and B.B.Kimia. Computation of the shock scaffold for unorganized point clouds in 3d. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[11] F.F.Leymarie and B.B.Kimia. The medial scaffold of 3d unorganized point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):313–330, 2007.

[12] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, and W. Liu. Case: Connectivity-based skeleton extraction in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2009.

[13] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, and W. Liu. Connectivity-based skeleton extraction in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(5):710–721, 2010.

[14] H. Jiang, S. Zhang, G. Tan, and C. Wang. Cabet: Connectivity-based boundary extraction of large-scale 3d sensor networks. In *Proc. of IEEE INFOCOM*, 2011.

[15] B. Karp and H. Kung. Gpsr: Greedy perimeter stateles routing for wireless networks. In *Proc. of ACM MOBICOM*, 2000.

[16] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. 1972.

[17] S. Lederer, Y. Wang, and J. Gao. Connectivity-based localization of large scale sensor networks with complex shape. In *Proc. of IEEE INFOCOM*, 2008.

[18] F. Li, J. Luo, C. Zhang, S. Xin, and Y. He. Unfold: uniform fast on-line boundary detection for dynamic 3d wireless sensor networks. In *Proc. of ACM MOBIHOC*, 2011.

[19] M. Li, Y. Liu, J. Wang, and Z. Yang. Sensor network navigation without locations. In *Proc. of IEEE INFOCOM*, 2009.

[20] X. Li, Y. J. Kim, R. Govindan, and W. Hong. Multi-dimensional range queries in sensor networks. In *Proc. of ACM SenSys*, 2003.

[21] W. Liu, H. Jiang, X. Bai, G. Tan, C. Wang, and K. Cai. Distance transform-based skeleton extraction and its applications in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(9):1763–1772, 2013.

[22] W. Liu, H. Jiang, X. Bai, G. Tan, C. Wang, W. Liu, and K. Cai. Skeleton extraction from incomplete boundaries in sensor networks based on distance transform. In *Proc. of IEEE ICDCS*, 2012.

[23] W. Liu, H. Jiang, C. Wang, C. Liu, Y. Yang, W. Liu, and B. Li. Connectivity-based and boundary-free skeleton extraction in sensor networks. In *Proc. of IEEE ICDCS*, 2012.

[24] W. Liu, H. Jiang, Y. Yang, and Z. jin. A unified framework for line-like skeleton extraction in 2d/3d sensor networks. In *Proc. of IEEE ICNP*, 2013.

[25] W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang. Approximate convex decomposition based localization in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2012.

[26] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mobile Networks and Applications*, 8(4):427–442, 2003.

[27] D. Reniers and A. Telea. Segmenting simplified surface skeletons. In *Proc. of the 14th IAPR International Conference on Discrete Geometry for Computer Imagery*, 2008.

[28] D. Reniers, J. J. Wijk, and A.Telea. Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):355–368, 2008.

[29] R. Sarkar, W. Zeng, J. Gao, and X. D. Gu. Covering space for in-network sensor data storage. In *Proc. of ACM/IEEE IPSN*, 2010.

[30] R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. *IEEE/ACM Transactions on Networking*, 17(6):1902–1915, 2009.

[31] R. Tam and W. Heidrich. Shape simplification based on the medial axis transform. In *Proc. of IEEE Visualization Conference*, 2003.

[32] G. Tan, H. Jiang, S. Zhang, and A. Kermarrec. Connectivity-based and anchor-free localization in large-scale 2d/3d sensor networks. In *Proc. of ACM MOBIHOC*, 2010.

[33] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: a wireless sensor network testbed. In *Proc. of ACM/IEEE IPSN*, 2005.

[34] S. Xia, N. Ding, M. Jin, H. Wu, and Y. Yang. Medial axis construction and applications in 3D wireless sensor networks. In *Proc. of IEEE INFOCOM, mini-conference*, 2013.

[35] 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 ACM MOBIHOC*, 2011.

[36] X. Yu, X. Yin, W. Han, J. Gao, and X. D. Gu. Scalable routing in 3D high genus sensor networks using graph embedding. In *Proc. of IEEE INFOCOM, mini-conference*, 2012.

[37] H. Zhou, N. Ding, M. Jin, S. Xia, and H. Wu. Distributed algorithms for bottleneck identification and segmentation in 3D wireless sensor networks. In *Proc. of IEEE SECON*, 2011.

[38] H. Zhou, M. Jin, and H. Wu. A distributed delaunay triangulation algorithm based on centroidal voronoi tessellation for wireless sensor networks. In *Proc. of ACM MOBIHOC*, 2013.

[39] D. Zhu, Q. Tao, J. Xing, Y. Wang, W. Liu, and H. Jiang. The extraction and evaluation of skeleton in sensor networks. In *Proc. of IEEE Ninth International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2013.

[40] X. Zhu, R. Sarkar, and J. Gao. Shape segmentation and applications in sensor networks. In *Proc. of IEEE INFOCOM*, 2007.