Towards Robust Surface Skeleton Extraction and Its Applications in 3D Wireless Sensor Networks

Wenping Liu, *Member, IEEE*, Tianping Deng, Yang Yang, *Student Member, IEEE*, Hongbo Jiang, *Senior Member, IEEE*, Xiaofei Liao, *Member, IEEE*, Jiangchuan Liu, *Senior Member, IEEE*, Bo Li, *Fellow, IEEE*, and Guoyin Jiang

***Abstract*—The 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 deliver 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 coarse-grained surface skeleton, followed by a refining process to generate the fine-grained surface skeleton. Furthermore, we design an efficient updating scheme to react to the network dynamics caused by node failure, insertion, etc. We also investigate the impact of boundary incompleteness and present a scheme to extract the surface skeleton under incomplete boundary. Finally, we apply the extracted surface skeleton to facilitate the design of data storage protocol and curve skeleton extraction algorithm. Extensive simulations show the robustness of the proposed algorithm to shape variation, node density, node distribution, communication radio model and boundary incompleteness, and its effectiveness for data storage and retrieval application with respect to load balancing.**

***Index Terms*—3D sensor networks, curve skeleton, data storage and retrieval, surface skeleton.**

I. Introduction

THE 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) [24], [29] is appealing because 1) it can greatly reduce the communication and energy cost by avoiding frequent in-network flooding for information retrieval [28], and 2) it is very simple yet powerful [10]. GHT names events with keys and hashes the keys into geographic locations; 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 [13] 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 [10] especially in 3D sensor networks where there are arbitrarily large number of perimeters to be explored [36]. If the bounding *box* defining the range of coordinates for hash functions is not properly computed, the imbalance of storage load, together with traffic load, will become more severe [27]. 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

Manuscript received March 12, 2015; revised October 25, 2015; accepted January 01, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. X. Liu. Date of publication January 27, 2016; date of current version December 15, 2016. This work was supported in part by the National Natural Science Foundation of China under Grants 61202460, 61271226, 61272408, 61322210, 61572219, and 61502192; the China Scholarship Council under Grant 201308420188; the National Natural Science Foundation of Hubei Province under Grant 2014CFA040; the International Science and Technology Cooperation Program of China under Grant 2015DFE12860; and the Fundamental Research Funds for the Central Universities under Grant 2015QN073. The work of B. Li was supported in part by a grant from the RGC under Contract 615613, a grant from the NSFC/RGC under Contract N_HKUST610/11, a grant from the NSFC under Contract U1301253, and a grant from ChinaCache Int. Corp. under Contract CCNT12EG01. An earlier version of this work appeared in the Proceedings of ACM MobiHoc 2014. *(Corresponding author: Tianping Deng.)*

W. Liu is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the School of Statistics, Hubei University of Economics, Wuhan 430074, China (e-mail: wenpingliu2009@gmail.com).

T. Deng, Y. Yang, and H. Jiang are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: eitianpingdeng2012@gmail.com; yangyang8795@gmail.com; hongbojiang2004@gmail.com).

X. Liao is with the School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xfliao@hust.edu.cn).

J. Liu is with the School of Computing Science, Simon Fraser University, Vancouver, BC V5A 1S6, Canada (e-mail: jcliu@cs.sfu.ca).

B. Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: bli@ust.hk).

G. Jiang is with the School of Information Management, Hubei University of Economics, Wuhan 430074, China (e-mail: Jiangguoyin@hbue.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2016.2516343

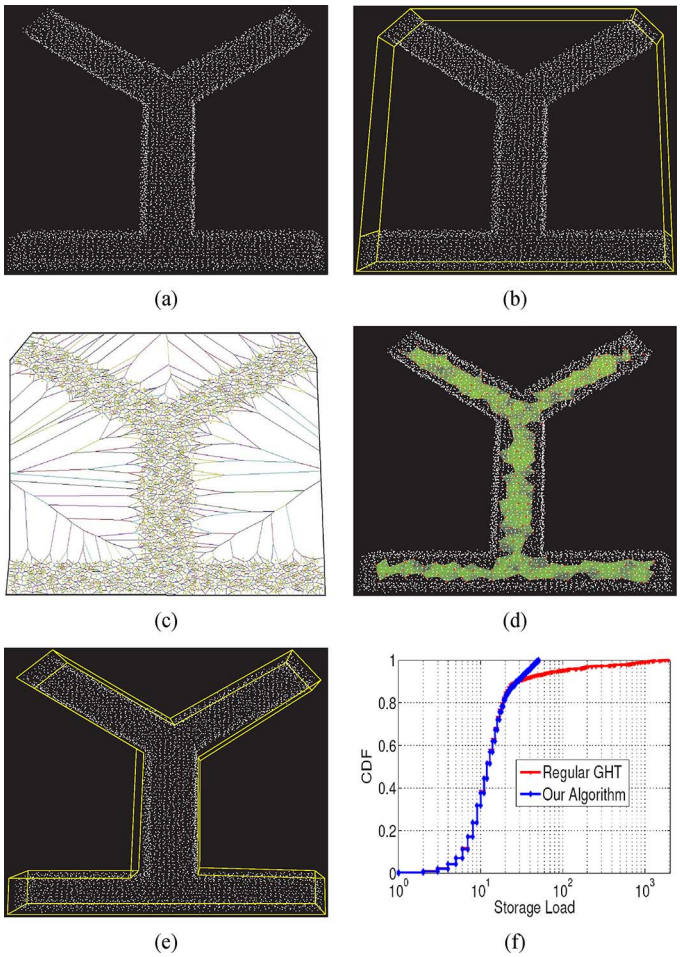


Fig. 1. (a) The Y-shaped 3D network has 12,545 sensors with average node degree 15.36; (b) The bounding box with voids used in [24], [40]; (c) A cross-section of the Voronoi diagram with bounding box in (b); (d) The extracted Surface Skeleton; (e) The bounding box computed based on the Surface Skeleton in (d); (f) Comparison of storage performance with the two bounding boxes.

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 [27], i.e., the larger the volume 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 GHT in [24], [29] and typical segmentation algorithms such as BOTTLENECK in [40], 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 [40]. Please see Fig. 1(f). Similarly, DIM [18] also maps multi-attribute events to the sensing field; the nodes near an empty zone unoccupied by sensors store data mapped to this zone. *Accordingly, to evenly distribute the storage load of sensor nodes, the key for data storage application is to precisely define the bounding box of sensor networks with an arbitrary shape; Fig. 1 shows that the bounding box results could vary greatly with different methods*.

The difficulties of computing the precise bounding box for a complex network mainly stem from the presence of concave

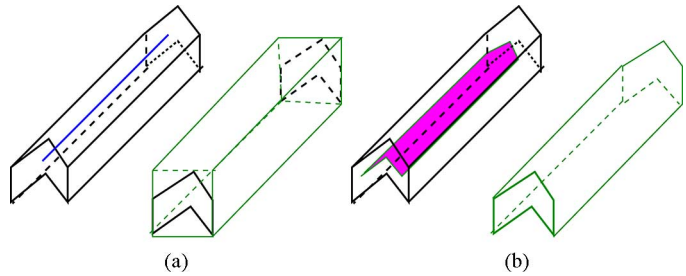


Fig. 2. Bounding box computed based on C-Skeleton and S-Skeleton, respectively. (a) C-Skeleton (blue line in left) and the bounding box (green cubic in right); (b) The S-Skeleton (shaded surface in left) and the bounding box (green polyhedron in right).

valleys (and thus *bridges* between adjacent *peaks*) and/or, holes or tunnels, which often lead 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 line-like skeleton, or *curve skeleton* (referred to as *C-Skeleton*), composed of 1D curves locally symmetric to the object [26]. While C-Skeleton has been used in 3D sensor networks to improve routing performance [21], 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).

A. Challenges

Note that skeleton has also been widely used as an efficient way to facilitate routing [5], segmentation [5], [43] localization [15] and navigation [33], [35], etc., in sensor networks. The usefulness of skeleton has inspired many algorithms for its computation in sensor networks, e.g., [11], [19], [20], [42]. More recently, there are many practical deployments of sensors on 3D environments [34], triggering growing demands for studies on 3D sensor networks. Compared with 2D cases, it is much

more difficult to compute 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. However, the existed algorithms which primarily target at 2D sensor networks and deliver a 1-manifold skeleton composed of 1D curves, cannot be readily applied for 3D cases. For instance, as pointed out in [35], the extension of MAP [5] to 3D sensor networks will potentially eliminate true skeleton nodes, e.g., in a 3D sensor network with bottlenecks; the boundary of a 3D sensor network is typically composed of 2D manifolds, instead of 1D curves, therefore it cannot be decomposed into branches by CASE in [11], nor the boundary node ID can be sequentially grouped to form an interval as in [43]. The distance transform based skeleton computation algorithm as in [19] might work at the first glance, however, it will deliver a surface skeleton with fake holes even in a genus-0 network with bottlenecks where the true skeleton nodes actually have a small distance transform. Further, since the surface skeleton is also a 2-manifold, the connection of the identified skeleton nodes, which are often self-disconnected, to form a meaningful representation which should be homotopic to the original network, is very different from the 2D counterpart. Thus, it faces non-trivial challenges to extend the existing protocols for skeleton extraction in 2D sensor networks to 3D settings.

We are aware that one close work to ours is done by Xia *et al.* [35], 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 [39], which is difficult to obtain when only connectivity information is available and thus small interior holes are not identified [36]. Last but not least, it cannot adapt to network dynamics since the UTC mesh must be constructed in advance. 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.

B. 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. Besides, we use the S-Skeleton as a basis to extract the C-Skeleton of the underlying 3D sensor network. Different from [35], 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 and refining procedure to deliver 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 [35]. 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. As in reality, complete boundary information might not always be easy to obtain, e.g., in sparse networks, we propose a scheme to extract S-Skeleton under incomplete boundary information which is much easier to be identified. Finally, we apply the extracted S-Skeleton for the design of load-balanced data storage protocol and C-Skeleton extraction algorithm in 3D sensor networks. We conduct extensive simulations to show the robustness of the algorithm to shape variation, node density, node distribution, communication radio model, and boundary incompleteness, and its effectiveness for data storage and retrieval with respect to load balance in 3D sensor networks.

The reminder of the paper is structured as follows. In Section II we briefly introduce the motivations and preliminary knowledge of this work, and detail our algorithm in Section III. Section IV is devoted to the applications of the derived S-Skeleton for the design of load-balanced data storage and retrieval protocol, and C-Skeleton extraction algorithm as well. To show the efficiency of the proposed algorithm, we conduct extensive simulations in Section V. In Section VI we present some previous work related to ours, and finally, Section VII concludes the paper.

II. Motivations and Preliminaries

In continuous domain, as mentioned earlier, the S-Skeleton of a 3D object $D \subset R^3$, denoted by $SK(D)$ (A list of notations is given in Table I), 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 $T : D \to \mathbb{R}$ as $T(x) = \min_{y \in \partial\mathcal{D}} d(x, y)$ 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 $x's$ feature points on ∂D. That is, $F(x) = \{y \in \partial D | d(x, y) = T(x)\}$. 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

TABLE I
LIST OF NOTATIONS AND DESCRIPTIONS

Notation	Description
SK(D)	The surface skeleton of an object $D \subset R^3$
$T(x)$	The distance transform of $x \in D$
$d(x, y)$	The distance between point x and y
$F(x)$	The feature transform of $x \in D$
$N_k(p)$	The k-hop neighbors of node p
$\overline{F}(p)$	The extended feature nodes of node p
$\sim_\epsilon$	The geodesic ϵ-equivalence relationship
$\overline{F}_\epsilon(p)$	The ϵ-component set of node p
V_s	The surface skeleton node set
E_s	The link set between pairwise nodes
$List(p)$	The set of nearest boundary nodes of p
$D_p^k(p_1, p_2)$	The set of nodes on the shortest perimeter path between $p_1 \in N_k(p)$ and $p_2 \in N_k(p)$
$c_k(p)$	The k-hop curvature of node p

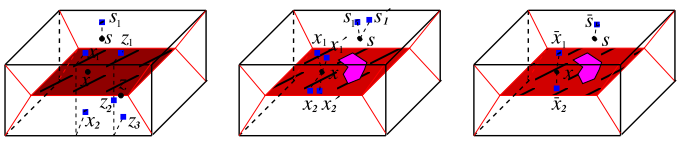


Fig. 3. An illustration of the S-Skeleton of a cubic. There are thirteen sheets formed by the Y-curves and the boundary of the cubic while only the middle sheet is shaded. The solid red lines represent the Y-curves where two sheets meet. (a) Continuous space where the Euclidean distance between points 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.

only curves, which is not considered in this paper. Two sheets may meet along a Y-intersection curve [8] (referred to as *Y-curve*), on which any point has at least three feature points and is thus called 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, cannot be directly applied for discrete 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.

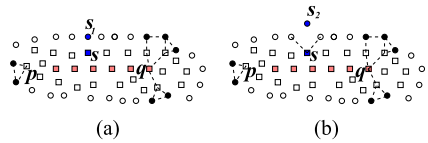


Fig. 4. The principle of skeleton node identification in [43] and its drawback. Boundary nodes are shown by circles, ordinary interior nodes and skeleton nodes are shown by empty rectangles and solid pink rectangles, respectively. (a) The nodes in solid pink rectangles, e.g., q, are skeleton nodes such 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.

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, due to the discrete nature of sensor networks, 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.* [43] 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 the IDs, of its feature nodes; the nodes with only one interval, e.g., p in Fig. 4(a), are ordinary nodes. However, this method still suffers from boundary noise in 2D sensor networks, as mentioned in [19].

Finally, there can be boundary noise, and as well-known, by traditional approach of skeleton extraction, an unwanted skeleton branch needed to be pruned will occur if there is a small bump in the boundary, which cannot be avoided by [43] neither. 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 node IDs of node s in Fig. 4(a) will be split into two intervals in Fig. 4(b), and thus s will identify itself as a skeleton node by the approach in [43]. We propose to solve this problem by introducing an equivalence relation $\sim_\epsilon$ as in [25]:

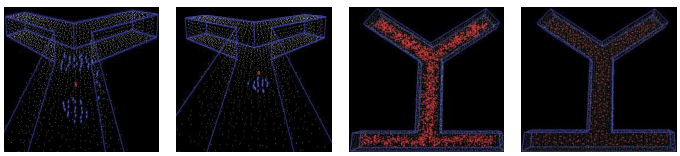
Definition 3: If the geodesic distance between two boundary nodes a, b is no greater than $\epsilon (\geq 0)$, we say that a and b are geodesic ϵ-equivalent, denoted by $a \sim_\epsilon b$.

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 virtual interval which we call as ϵ-*interval*. Thus, a node identifies itself as a skeleton node if and only if it has two or more ϵ-intervals.

With the ϵ-equivalent relationship, 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 I_1 and I_2 have a geodesic distance less than ϵ, they are regarded as one ϵ-interval. Thus, if p has only one interval before bumping, 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. ■



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

Fig. 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 identified S-Skeleton nodes; (d) The maximal independent set of the S-Skeleton nodes.

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 S-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 cannot be directly applied in 3D cases, as here the boundary surfaces are 2-manifolds and the boundary nodes cannot be ordered in sequence to form intervals.

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 $\overline{F}(p)$. However, 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(c). To address this, we first group these extended feature nodes into connected components, mimic of the interval of feature node IDs in the 2D counterpart, 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 shortest path distance between pairwise feature 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 ϵ-components of node p. Obviously, each ϵ-component here serves as a feature point in continuous domain, and thus we can identify an S-Skeleton node in 3D sensor networks as follows.

Proposition 1: For an interior node p, if $|\overline{F}_\epsilon(p)| \geq 2$, then p identifies itself as an S-Skeleton node.

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

III. 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 [12], [16], [17]. Further, we do not assume that the location information of nodes are known, and our work is based on mere connectivity information.

A. S-Skeleton Node Identification

As mentioned in Section II, 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 already received the message from q, it will discard the message):

- If p has not received the flooded message from any boundary node, it appends q to its list of feature nodes (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 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 described in [21] to merge these components into ϵ-component(s). More specifically, each component is firstly assigned an 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 neighbors;
- 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 neighbors;
- 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).

B. Coarse-Grained 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 [14]. 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 [41]. As a result, any pair of independent nodes have a separation greater than one but no greater than three hops, as shown in Fig. 5(d). One advantage of this procedure, as pointed out in [41], 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 [41], and the S-Skeleton (referred to as coarse-grained S-Skeleton), which consists of triangles with edge length of 2 or 3 hops rather than 1 hop, is thus generated.

C. Fine-Grained S-Skeleton Establishment

Note that the coarse-grained S-Skeleton only provides a coarse approximation to the genuine S-Skeleton. Now, we propose a refining scheme to partition each triangle into multiple ones with 1-hop long edges, and deliver a fine-grained S-Skeleton approximating more accurately to the genuine S-Skeleton; as the sampling nodes of genuine S-Skeleton become denser, one can imagine that it can provide a more accurate concave/convex nodes information and thus the computed bounding box could be more tight.

It is noted that there are four kinds of triangles in the coarse-grained S-Skeleton, including equilateral triangles with edge length of either 2 or 3 hops, and isosceles triangles with three edge lengths being either 2, 2 and 3 hops, or 3, 3 and 2 hops. Our objective is to partition each of them into the smallest number of triangles with 1-hop edge length, as more nodes might result in crossing edges when only connectivity information is available, and thus yield a non-planar graph. Below are some key results.

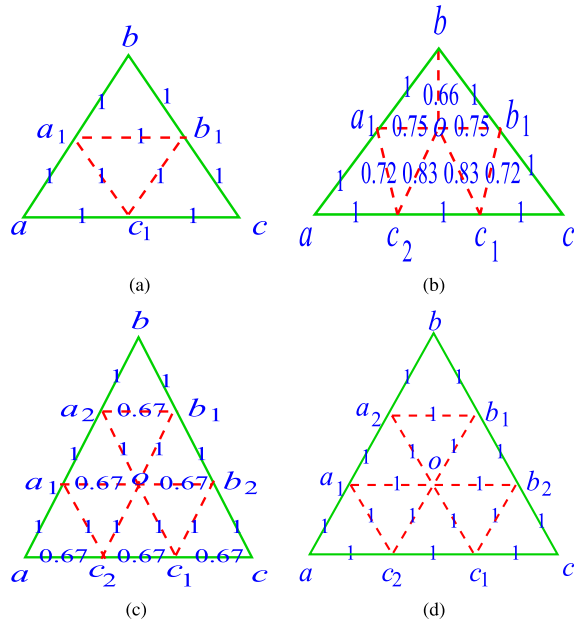


Fig. 6. Illustrative examples. Each value represents an edge length. (a) Four equilateral triangles with 1-unit-long edges. (b) An isosceles triangle with edge lengths being 2, 2, and 3 into 7 isosceles triangles. (c) An isosceles triangle with edge lengths being 3, 3, and 2 into 9 isosceles triangles. (d) Nine equilateral triangles with 1-unit-long edges.

Lemma 1: An equilateral triangle with edge length of 2 units can be decomposed into 4 equilateral triangles with 1-unit long edges, as shown in Fig. 6(a).

Lemma 2: An equilateral triangle with edge length of 3 units can be decomposed into 9 equilateral triangles with 1-unit long edges, as shown in Fig. 6(d).

However, for an isosceles triangle with edge lengths being 2 or 3, we can't partition it into a set of triangles with 1-unit long edge in above way. Fortunately, in this paper we are not pursuing strict equilateral triangles in continuous domain but relax ones in discrete sensor networks. That is, if the edge length is smaller than the communication radio range (no matter how short the edge is), it is regarded as 1 hop, and we also call such a triangle equilateral, with a little abuse of notation. To this end, we can partition an isosceles triangle with edge lengths being 2, 2 and 3 into 7 isosceles triangles, as indicated by Fig. 6(b), and partition an isosceles triangle with edge lengths being 3, 3 and 2 into 9 isosceles triangles, as indicated by Fig. 6(c). Note that the partition method in Fig. 6(c) is the same as that in Fig. 6(d), in order to ensure that each edge is no greater than 1-unit long.

In summary, we have the following:

Theorem 2: The coarse-grained S-Skeleton can be refined to be a fine-grained S-Skeleton with 1-hop long edges for each triangle in above way.

As such, the refinement of the coarse-grained S-Skeleton can be done in a distributed way as follows. First, each triangle is aware of its type by measuring the edge lengths; then the vertices of each triangle initiates a local flooding within the S-Skeleton nodes identified in Section III-A, and accordingly the division points can easily identify themselves.

As a boundary edge only belongs to one triangle, its division points is uniquely determined. For a non-boundary edge, it is relatively complicated as this edge might belong to triangles with two different types. If one is an equilateral triangle with edge length of 2 hops and the other is isosceles with edge lengths being 3, 3, and 2 hops, there should be three division points: two for the isosceles triangle and one for the equilateral triangle. In particular, for the case in Fig. 6(b), an additional node o should identify itself if it's equidistant to two division points of two edges with length of 3, and for the case in Fig. 6(c) (d), the midpoints between division points a_1 and b_2 will also identify themselves. Finally, these identified division points, together with the vertices of triangles, naturally form a set of triangles with edge lengths all being 1 hop, and the fine-grained S-Skeleton is thus generated.

D. Complexity Analysis

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

Proof: The algorithm has three steps: S-Skeleton node identification, coarse-grained S-Skeleton establishment and fine-grained 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 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 [41], [27]. Finally, the refinement of coarse-grained S-Skeleton is only conducted within the identified S-Skeleton nodes, and thus the complexity is at most linear to the network size.

In total, the algorithm has a linear complexity. ■

E. 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 [27]. In other words, nodes/links may come and go [9], 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. More technical details and experimental results can be found in [23].

F. Extension to Incomplete Boundary Case

Note that we have proved in Section II that a small bump in boundary usually does not change the identity of a node being an S-Skeleton node or not, implying that the proposed skeleton node identification scheme is robust to boundary noise. In reality, complete boundary information might be difficult or costly to obtain, e.g., in sparse networks where existing boundary recognition algorithms often do not work very well. To make our algorithm more practical, we now propose an extended version to cover this situation. The key insight is as follows:

Theorem 4: For any node p, let $MIS(p)$ be the maximal independent set of $p's$ feature nodes where the distance between two neighboring feature nodes is no greater than $k < \epsilon$. Then if other genuine feature nodes of p, which do not belong to $MIS(p)$, are not identified as boundary nodes, the S-Skeleton extracted by our algorithm will not change.

Proof: It is noted that our algorithm identifies an S-Skeleton node based on the number of ϵ-components. For an S-Skeleton node p, it has at least two ϵ-components, and the missing of boundary nodes in either component will not result in the decrease of ϵ-component number, and accordingly, the identity of p will not change. On the other hand, for a non-skeleton node q which has only one ϵ-component, if a part of the true boundary nodes not in $MIS(q)$ are not identified, but the rest of the boundary nodes, i.e., $MIS(q)$, still form an ϵ-component because each pairwise neighboring feature nodes in $MIS(q)$ are no more than ϵ hops, and thus q will not change its identity neither. ■

Theorem 4 implies that merely based on a small subset of the boundary information, our algorithm can still extract the genuine S-Skeleton. This property is very useful for sparse networks where complete boundary information is difficult, or costly, to obtain while a part of boundary node can be easily identified based on the neighborhood size. Namely, for any node p, if its r-neighborhood size (i.e., the number of nodes at most r hops away) is locally minimal, say, among its k-hop neighbors (referred to as $N_k(p)$), then p identifies itself as a boundary node. Note that with these identified boundary nodes, we cannot directly use the proposed algorithm to extract the S-Skeleton, since the geodesic distance between feature nodes is not readily computed. One might argue we can simply compute the shortest path between pairwise feature nodes. However, this is problematic since the shortest path will possibly go across the interior of the network, resulting in some S-Skeleton nodes in narrow part of the network not being identified. Further, in complex networks, the concave nodes usually cannot be identified as boundary nodes based on neighborhood size [19]. Even we suppose that the algorithm works in this case, the delivered S-Skeleton will *move* toward the boundary at concave nodes.

To tackle these difficulties, our approach is to construct a maximal independent set of the boundary nodes as follows. First, the nodes with locally minimal r-neighborhood sizes identify themselves as boundary nodes. Then, with these identified boundary nodes, we first construct a maximal independent set MIS_b such that pairwise boundary nodes are at least k hops

away. Theoretically, in such a 2-manifold boundary surface, there are at most $\frac{2\pi k}{k} \approx 6$ points satisfying that each pair of them have a separation no less than k. Thus, in our algorithm, for each boundary node p in MIS_b, its k-hop neighbors are sorted in an ascending order of r-hop neighborhood size; the first node in the sorted list naturally joins to the MIS_b, and the second node also joins if it is k hops away from both p and the first selected node. This process is repeated until there are 6 k-hop neighbors of p in MIS_b. Note that if there are already $l(\geq 1)$ nodes identified as boundary nodes and joining MIS_b, only $6-l$ other k-hop neighbors in the list are needed otherwise the independence will be violated. Eventually, by this way a maximal independent set of the boundary nodes can be obtained while interior nodes are unlikely to join the MIS_b.

With these incomplete boundary nodes, we can identify the S-Skeleton nodes in the following distributed manner. The boundary nodes in MIS_b initiate a flooding message including a hop counter and its neighbors in MIS_b; each node then computes its distance to the boundary nodes, and keeps record of the extended feature nodes, together with their neighboring boundary nodes in MIS_b. Then, it constructs a set of chains based on the neighboring relationship. That is, if p_1 is the neighbor of p_2, and p_2 is that of p_3, and so on, then these nodes form a chain. Afterwards, each node determines its identity based on the number of chains: a node is an S-Skeleton node if it has at least two chains, otherwise it's not. Finally, we triangulate these S-Skeleton nodes in the way described in Section III-B to form a coarse-grained S-Skeleton, followed by a refining process as described in Section III-C to obtain the fine-grained S-Skeleton.

IV. The Applications of the S-Skeleton

A. S-Skeleton Based Data Storage and Retrieval

As mentioned in Section I, the S-Skeleton can be used to define a tight and simple bounding box, which can balance the storage node in GHT [24], [29] and DIM [18].

Note that the bounding box should better be regular as the logical data space is often regular [27]. 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 defined as the following.

Definition 4: A boundary edge of the S-Skeleton is an edge which belongs to one triangle only.

After the triangulation process during the S-Skeleton establishment, the boundary edge identification is rather straightforward. Further, we call the nodes on the boundary edges as 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.

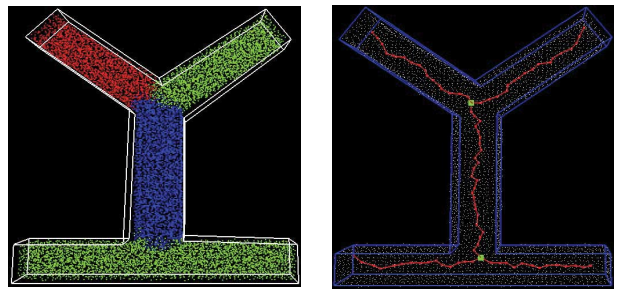
Definition 5: For an S-boundary node p, denote by $N_k(p)$ the nodes at most k hops away from p; let $p_1, p_2 \in N_k(p)$ be two S-boundary nodes such that the shortest paths between any two nodes of p_1, p_2, p will not pass through the S-Skeleton. Denote by $D_p^k(p_1, p_2)$ the set of nodes on the 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 [22]). Then the k-hop curvature of node p, i.e., $c_k(p)$, is defined as

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

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 6: 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. 7(a).



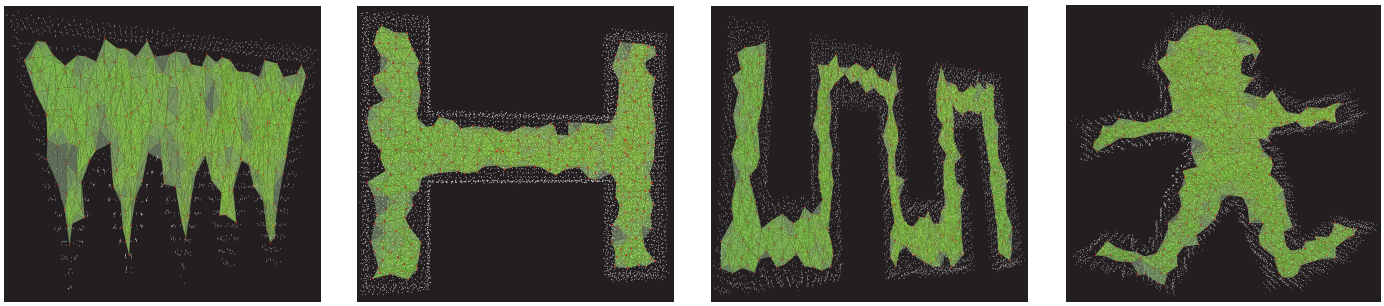
(a) (b)

Fig. 7. Applications of S-Skeleton. (a) The segmentation result and the bounding box; (b) The C-Skeleton.

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], [21], 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 approximately equal volumes. Thus, the storage load of the nodes can be well balanced. We validate the proposed scheme by extensive simulations in Section V.

B. C-Skeleton Extraction

Compared with S-Skeleton, C-Skeleton is a more concise representation of a 3D object, which consists of 1D curves.



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

Fig. 8. The performance of our algorithm under various 3D scenarios. The S-Skeletons are shaded. (a) 5,537 nodes, avg. deg. 15.37; (b) 16,156 nodes, avg. deg. 15.69; (c) 19,313 nodes, avg. deg. 15.78; (d) 15,288 nodes, avg. deg. 15.31. (a) Seabed. (b) H. (c) Snake. (d) Man.

Unfortunately, there lacks of a well-accepted definition for C-Skeleton, and a possible definition is that C-Skeleton is a subset of the S-Skeleton. C-Skeleton has many desirable properties, such as homotopic, invariant under isometric transformations, thin, centered, and robust, etc. [7], and it has been successfully used for routing in 3D sensor networks [21]. Since the S-Skeleton consists of 2-manifolds, one might argue that we can use the skeleton extraction algorithms in 2D networks, e.g., [11], [19], to derive the C-Skeleton of the underlying 3D sensor networks. The undesirability is that these algorithms deliver a self-disconnected skeleton and thus a connecting process is a must to form a meaningful representation. In this subsection, we will propose an efficient way to derive the C-Skeleton based on the established S-Skeleton.

Our approach is distance transform based, but it is different from [19] in that it relies on the distance transform of each edge, instead of one single node.

Definition 7: Let $T(p_1), T(p_2)$ be the distance transform of node p_1 and p_2, respectively. The distance transform of an edge $\langle p_1, p_2\rangle$, denoted by $T(p_1, p_2)$, is defined as the sum of $T(p_1)$ and $T(p_2)$.

In [19], it has been proven that a node is a skeleton node if it has a locally maximal distance transform. Similarly, we can identify a skeleton edge based on its distance transform.

Definition 8: An edge is a skeleton edge if its end nodes are both skeleton nodes.

Theorem 5: For edge $\langle p_1, p_2\rangle$, if the distance transform is locally maximal, then it is a skeleton edge.

Proof: Since the distance transform of edge $\langle p_1, p_2\rangle$ is locally maximal, then there must be at least one node, say p_1, having the locally maximal distance transform, and there is no other neighboring node of p_1 such that its distance transform is larger than p_2, otherwise the edge $\langle p_1, p_2\rangle$ has no locally maximal distance transform. Hence, the line connecting p_1 and p_2 has the steepest slope of the distance field, and thus p_2 is a skeleton node either [30], implying that $\langle p_1, p_2\rangle$ is a skeleton edge. ■

Corollary 1: An edge $\langle p_1, p_2\rangle$ is a skeleton edge if its distance transform is the largest among that of the edges in the triangles edge $\langle p_1, p_2\rangle$ belonging to.

With these results, the extraction of C-Skeleton becomes much easier. Each edge first computes its distance transform, and then determines its identity by judging whether it has the largest distance transform among the edges associated with the same triangle(s). Eventually, the collection of skeleton edges naturally form the C-Skeleton, as shown in Fig. 7(b).

V. Simulations

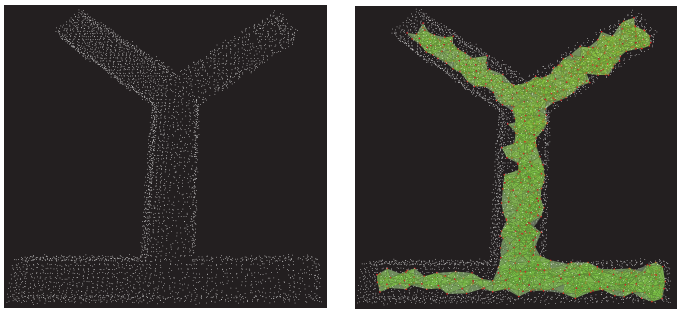
We conduct extensive simulation tests on various 3D scenarios to show the performance of the algorithm. In our simulations, sensor nodes are randomly deployed inside the underlying 3D space, and the boundary nodes are recognized by [12]. The parameter ϵ is set to be one by default. We do not compare with [35] since it requires high node density and nice tetrahedron mesh, as mentioned before, which are not 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, node distribution, communication radio model and boundary incompleteness, and then we show the applicability of the S-Skeleton for data storage.

A. Robustness to Shape Variation

We conduct the proposed algorithm under difference scenarios, namely, Seabed, H, Man, and Snake, as shown in Fig. 8, with network size ranging from 5,537 to 19,313 and average node degree in between 15 and 16. We can obviously see that the derived S-Skeletons in Fig. 8(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 node degree are relatively low for 3D sensor networks, one can imagine that our algorithm can work for small-scale networks.

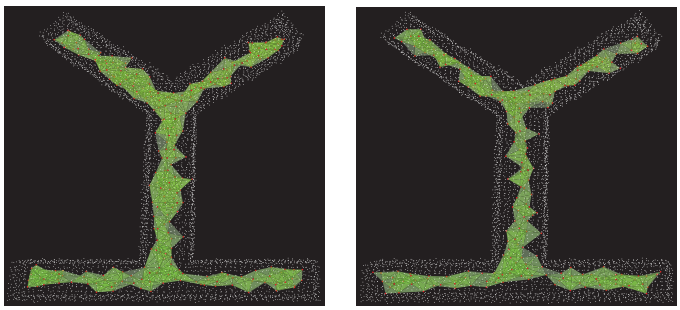
B. Robustness to Node Distribution

To show that our algorithm works for the network with non-uniform distribution, we sample the right part of the network in Fig. 1(a) with probability 0.6 while all nodes in the left part are kept to generate a non-uniformly distributed network, as shown in Fig. 9(a). As the right part is sparser, we enlarge the radio range to keep the network connected. Fig. 9(b) shows the final skeleton. At first glance it might seem anti-intuitive that the skeleton *leans* toward right. This is reasonable, however, since the feature nodes of an interior node are more likely to form more than one component due to the sparsity, and thus more interior nodes in the right part identify themselves



(a) (b)

Fig. 9. The performance of our algorithm under non-uniform network where the right part is a sample from the network in Fig. 5(a) with probability 0.6. (a) The original network; (b) The S-Skeleton.



(a) (b)

Fig. 10. The performance of our algorithm under QUDG. (a) $\alpha = 0.2$, $p = 0.8$; (b) $\alpha = 0.4$, $p = 0.6$.

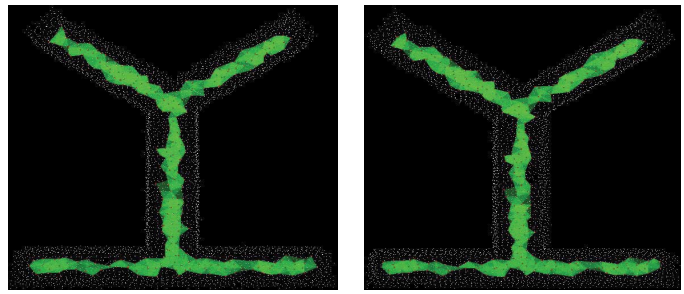
as skeleton nodes. One can address this by combining two components with distance (defined as the minimum distance between pairwise nodes, one in each component) less than a threshold, e.g., 2 hops, as one single component, at the cost of communication overhead.

C. Robustness to Communication Radio Model

We show that our algorithm is insensitive to communication radio model by using Quasi-Unit Disk Graph (QUDG) model. In QUDG, the link between nodes exists if their separation is smaller than $(1-\alpha)R$, and the link exists with probability $p \in (0,1)$ if the separation is in between $(1-\alpha)R$ and R, and no link exists if the separation is larger than R, where $\alpha \in (0,1)$ and p are two system parameters. Fig. 10(a) depicts the S-Skeleton under QUDG where $\alpha = 0.2, p = 0.8$. Compared with the UDG case in Fig. 5(h), it is more *shaggy*. This is because in QUDG, two nodes with a separation smaller than R is not necessary to be neighbors, posing the network topology more irregular. Similarly, the skeleton in Fig. 10(b) is more irregular than that in Fig. 10(a) because the nodes with a separation between $0.6R$ and R have a smaller probability (which is only 0.6) to be neighbors. But overall these two skeletons in Fig. 10 both capture the main topological features of the underlying networks, showing that our algorithm is insensitive to communication radio model.

D. Robustness to Boundary Incompleteness

To investigate the performance of our algorithm under incomplete boundary, we identify a part of boundary nodes based on its neighborhood size and construct a maximal independent set.



(a) (b)

Fig. 11. The performance of our algorithm under incomplete boundary recognized based on the r-hop neighborhood size where $r = 3$. (a) $k = 2, \epsilon = 3$; (b) $k = 3, \epsilon = 4$.

The parameter ϵ is set to be larger than k by one. Please see Fig. 11. We observe that even under such harsh environments with incomplete boundary, our algorithm still achieves a reasonable S-Skeleton, without bending toward the boundary.

E. 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 [24], and the method in [40] (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.

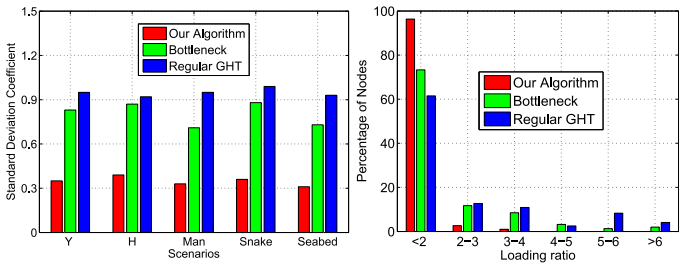
We propose three metrics, namely, *maximum storage load*, *standard deviation coefficient* and *loading ratio of a node*, to quantitatively measure how the storage load spreads across the network. The maximum storage load is closely related to the maximum Voronoi cell volume, and the total storage load is normalized to be 1. The standard deviation coefficient (referred to as *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 II depicts the comparison study on the maximum storage load for the five networks in Figs. 5 and 8. 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. 12(a) presents the sdc distribution for the five networks by these algorithms. Clearly, we can see that our algorithm

TABLE II
COMPARISON STUDY ON MAXIMUM STORAGE LOAD

Algorithm	Y	H	Man	Snake	Seabed
Our algorithm	0.0055	0.0049	0.0056	0.0052	0.0053
Bottleneck	0.0111	0.0107	0.0079	0.0122	0.0067
Regular GHT	0.0124	0.0117	0.0109	0.0122	0.0107



(a) (b)

Fig. 12. The comparison study on storage load where the rectangles from left to right with the same x-axis are obtained by our algorithm, bottleneck and regular GHT, respectively. (a) Standard deviation coefficient distribution; (b) Load ratio distribution.

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 results are undesirable.

Fig. 12(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.

VI. RELATED WORK

Based on the usage of geometric features, existing data-centric storage (DCS for short) protocols can be roughly classified into two categories: non-geometric storage schemes and geometric storage schemes. We first present the previous work on in-network storage, and then we introduce some previous work of skeleton extraction in sensor networks.

A. In-network Data Storage

1) Non-Geometric Schemes: Ratnasamy *et al.* [24], [29] first proposed GHT, a geographic hash table based protocol, where each data item is hashed to a geographic location by using a random hash function; the node (referred to as home nodes) nearest to this location thus stores the data. To this end, GPSR [13] is adopted as a low-level routing scheme to forward the data from the producer to the home node. As sensor nodes are battery-powered and prone to fail, and occasionally some new nodes might be inserted into the network for better coverage performance, or some nodes may be mobile, resulting in a dynamic network topology, in GHT, the home perimeter nodes surrounding the hashed location also maintain a copy of the sensed data to ensure data persistence. A similar work is done by Li *et al.* [18], which is designed to support multi-dimensional range queries in sensor networks where the low-level routing policy is GPSR, yet the distributed index for multi-dimensional data (referred to as DIM) is mapped via a new manner such that the data with similar index are mapped to nearby geographic locations.

Different from GHT, a simple double-ruling based scheme replicates the data item by the producer along the replication curve, instead of the home node in GHT, and the consumer retrieves the sensed data of interest by following the retrieval curve. If the replication curve intersects the retrieval curve, the success of data retrieval can be guaranteed. This method, however, has strict constraint on the network topology: it requires the network be regular, e.g., grid-like structure. Sarkar *et al.* [28] proposed to map the sensor nodes to a sphere, and the sensed data is replicated on the curve passing the consumer and the home node. As there are multiple retrieval curves going through the consumer and intersecting the replication curve, it offers the consumer great flexibility to select one meeting the desired criterion, e.g., balanced load or energy consumption. Clearly, GHT is a sub-case of [28].

The above-mentioned schemes do not incorporate the geometric information of the underlying network and thus have some obvious disadvantages. For instance, in GHT, when the network is irregular-shaped or has holes, boundary nodes will be overloaded, and the approach in [28] will fail if there are multiple holes inside the network. In addition, these schemes are primarily designed for 2D sensor networks, and cannot be directly applied for 3D cases.

2) Geometric Schemes: There are some efforts to use the geometric features for designing a more efficient DCS protocol. Fang *et al.* [10] proposed to partition the network into Voronoi tiles generated by some landmarks, and then used a two-level routing scheme as in [9] to forward data and queries inside and across tiles. It is actually a combination of GHT (across tiles) and double ruling (within each tile). Thus, if the tiles are nicely shaped, the performance can be significantly improved. However, it is rather challenging to select the right

landmarks for achieving that goal in an arbitrarily shaped network. Ye *et al.* [38] studied the problem of data dissemination for wireless sensor networks with mobile sinks, and proposed TTDD, a two-tier data dissemination approach to efficiently forward sensed data to the multiple mobile sinks. With location information of each sensor node except the mobile sinks, in TTDD, a grid structure is generated proactively by the data producer, and the delivery information will be set up at the dissemination nodes nearest to grid points. A query from a mobile sink reaches the producer via two tiers: the lower tier at the local grid square, or cell, of the sink, and the higher tier among the dissemination nodes at grid points. That is, the sink first issues a cell-wided query, and the dissemination nodes of the cell then forward the query to the neighboring dissemination nodes toward the data producer; this process will go on until the query meets either the data producer or the dissemination nodes which have already received the data sensed by the producer, say, upon a request from other mobile sinks. Respecting the fact that in irregular-shaped networks, traditional schemes usually lead to boundary nodes being overloaded, Sarkar *et al.* [27] proposed to map the sensed data to a so-called *covering space* by using Ricci flow and conformal Möbius transforms to "uniformize" the network, which is achieved by first turing an irregular-shaped network into circular, and then *copying* the sensor network to fill up the circular holes. Again, these algorithms are all designed only for 2D sensor networks.

Yang *et al.* [37] conducted the first work on data storage and retrieval in 3D sensor networks. It is motivated by the fact that any closed surface can be cut open to a topological disk by the cut graph of the surface. By using the Ricci flow technique, the topological disk can be mapped to a planar rectangle virtual coordinate, which is a 2D-manifold and has a regular shape such that the traditional double-ruling based scheme applies. For a data producer on the boundary surface, it leaves a copy along the horizontal curve on the planar rectangle; the consumer retrieves the data of interest by following the vertical curve on the planar rectangle. An interior producer needs to follow the sequence of node ID to the nearest boundary node and then replicates its data on the horizontal curve, while an interior consumer retrieves the desired data by first following the sequence of node ID to the nearest boundary node and then moving vertically until it hits the replication curve. As a result, boundary nodes will be overloaded in this scheme.

B. Skeleton Extraction in Sensor Networks

1) Skeleton Extraction: Bruck *et al.* [5] proposed a Medial Axis-based routing Protocol, named MAP. In MAP, the nodes having at least two nearest boundary nodes identify as medial axis nodes. To control boundary noise, unstable medial axis nodes whose nearest boundary nodes are too close are eliminated. Jiang *et al.* [11] proposed to identify convex nodes which can segment the boundaries into branches. A skeleton node is such that it has at least two nearest boundary nodes belonging to different boundary branches. DIST [19] first builds the hop count distance map based on incomplete boundary nodes identified based on the neighborhood size. A node with locally maximal distance transform identifies itself as a skeleton node. Liu *et al.* [20] proposed to construct an index for quantitatively measuring the centredness of a node, and a skeleton node is such that it has a locally maximal index.

While these algorithms target at 2D sensor networks, Xia *et al.* [35] proposed to extract the surface skeleton, which consists of 2D-manifolds and possibly 1D curves, of 3D sensor networks. They first establish the unit tetrahedron cell (UTC) mesh structure of 3D sensor networks. Starting from the boundary surface, a distributed algorithm is proposed to iteratively "peel" off a layer of the UTC mesh structure where the growing pace of the inner boundary surfaces and the shrinking pace of the outer boundary surfaces are the same. The iteration stops when those surfaces meet such that there is no space for further growing and shrinking, and eventually, the medial axis with a set of fully connected faces is obtained. Liu *et al.* [21] proposed a unified framework for line-like skeleton, which consists of 1D curves, extraction in 2D/3D sensor networks. They first present a unified definition for line-like skeleton node in 2D/3D sensor networks, and then proposed a distributed algorithm to identify the line-like skeleton nodes. Since these skeleton nodes are generally disconnected, a metric, named the importance measure of a skeleton node which is monotonic, is proposed such that the skeleton nodes can be connected easily to generate a coarse line-like skeleton. The final skeleton is obtained by conducting a pruning process to delete redundant skeleton branches.

2) The Applications of Skeleton Extraction: Bruck *et al.* [5] used the medial axis as an infrastructure to facilitate routing such that the routing is delivery-guaranteed and traffic load is evenly distributed. Buragohain *et al.* [6] proposed to find a safe route for internal users when emergency happens. Zhu *et al.* [43] proposed to segment an irregular 2D network into nice pieces based on the skeleton, and with the segmentation result the performance of distributed indices and random sampling can be significantly improved. In [15], the skeleton is used for selecting landmarks, based on which a localization scheme is proposed. Li *et al.* [33] exploited the skeleton for road-map construction which guides the movement of users in the sensing field to a safe exit with guaranteed safety. Xia *et al.* [35] proposed a medial-axis based navigation protocol in 3D sensor networks.

VII. Conclusion

We present a connectivity-based, scalable and distributed 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 then apply the surface skeleton to find a tight bounding box, which is then used for load-balanced data storage protocol. Besides, we also propose to extract the C-Skeleton based on the derived S-Skeleton. Extensive simulations show that the proposed algorithm is robust to factors such as shape variation, node density, node distribution, and communication radio model, etc., and effective for data storage application with respect to load balancing.

REFERENCES

[1] D. Andrade, M. Resende, and R. Werneck, "Fast local search for the maximum independent set problem," *J. Heuristics*, vol. 18, no. 4, pp. 525–547, 2012.

[2] N. Azimi, H. Gupta, X. Hou, and J. Gao, "Data preservation under spatial failures in sensor networks," in *Proc. ACM MobiHoc*, 2010, pp. 171–180.

[3] H. Blum, "Biological shape and visual science (Part I)," *Theoret. Biol.*, vol. 38, pp. 205–287, 1973.

[4] S. Bouix and K. Siddiqi, "Divergence-based medial surfaces," in *Proc. Eur. Conf. Comput. Vision*, 2000, pp. 603–618.

[5] J. Bruck, J. Gao, and A. A. Jiang, "MAP: Medial axis based geometric routing in sensor betworks," *Wireless Netw.*, vol. 13, no. 6, pp. 835–853, 2007.

[6] C. Buragohain, D. Agrawal, and S. Suri, "Distributed navigation algorithms for sensor networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–10.

[7] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Trans. Visualiz. Comput. Graphics*, vol. 13, no. 3, pp. 530–548, May–Jun. 2007.

[8] J. Damon, "Global medial structure of regions in $\mathcal{R}^3$," *Geomet. Topol.*, vol. 10, pp. 2385–2429, 2006.

[9] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, "GLIDER: Gradient landmark-based distributed routing for sensor networks," in *Proc. IEEE INFOCOM*, 2005, pp. 339–350.

[10] Q. Fang, J. Gao, and L. J. Guibas, "Landmark-based information storage and retrieval in sensor networks," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.

[11] H. Jiang *et al.*, "Connectivity-based skeleton extraction in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 710–721, May 2010.

[12] H. Jiang, S. Zhang, G. Tan, and C. Wang, "Connectivity-based boundary extraction of large-scale 3D sensor networks: Algorithm and applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 908–918, Apr. 2014.

[13] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM MobiCom*, 2000, pp. 243–254.

[14] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ser. The IBM Research Symposia. New York, NY, USA: Springer, 1972, pp. 85–103.

[15] S. Lederer, Y. Wang, and J. Gao, "Connectivity-based localization of large scale sensor networks with complex shape," *ACM Trans. Sensor Netw.*, vol. 5, no. 4, pp. 31:1–31:32, 2009.

[16] 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. ACM MobiHoc*, 2011, Art. no. 14.

[17] F. Li, C. Zhang, J. Luo, S.-Q. Xin, and Y. He, "LBDP: Localized boundary detection and parametrization for 3D sensor networks," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 567–579, Apr. 2014.

[18] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proc. ACM SenSys*, 2003, pp. 63–75.

[19] W. Liu *et al.*, "Distance transform-based skeleton extraction and its applications in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1763–1772, Sep. 2013.

[20] W. Liu *et al.*, "Connectivity-based and boundary-free skeleton extraction in sensor networks," in *Proc. IEEE ICDCS*, 2012, pp. 52–61.

[21] W. Liu, H. Jiang, Y. Yang, and Z. Jin, "A unified framework for line-like skeleton extraction in 2D/3D sensor networks," in *Proc. IEEE ICNP*, 2013, pp. 1–10.

[22] W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang, "Approximate convex decomposition based localization in wireless sensor networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1853–1861.

[23] W. Liu *et al.*, "Surface skeleton extraction and its application for data storage in 3D sensor networks," in *Proc. ACM MobiHoc*, 2014, pp. 337–346.

[24] S. Ratnasamy *et al.*, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mobile Netw. Appl.*, vol. 8, no. 4, pp. 427–442, 2003.

[25] D. Reniers and A. Telea, "Segmenting simplified surface skeletons," in *Proc. 14th IAPR Int. Conf. Discrete Geomet. Comput. Imagery*, 2008, pp. 262–274.

[26] D. Reniers, J. J. Wijk, and A. Telea, "Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure," *IEEE Trans. Visualiz. Comput. Graphics*, vol. 14, no. 2, pp. 355–368, Mar.–Apr. 2008.

[27] R. Sarkar, W. Zeng, J. Gao, and X. D. Gu, "Covering space for in-network sensor data storage," in *Proc. ACM/IEEE IPSN*, 2010, pp. 232–243.

[28] R. Sarkar, X. Zhu, and J. Gao, "Double rulings for information brokerage in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1902–1915, Dec. 2009.

[29] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *Comput. Commun. Rev.*, vol. 33, no. 1, pp. 137–142, 2003.

[30] H. Talbot and L. Vincent, "Euclidean skeletons and conditional bisectors," in *Proc. SPIE*, 1992, pp. 862–876.

[31] R. Tam and W. Heidrich, "Shape simplification based on the medial axis transform," in *Proc. IEEE Visualiz. Conf.*, 2003, pp. 481–488.

[32] G. Tan, H. Jiang, S. Zhang, Z. Yin, and A.-M. Kermarrec, "Connectivity-based and anchor-free localization in large-scale 2D/3D sensor networks," *ACM Trans. Sensor Netw.*, vol. 10, no. 1, pp. 6:1–6:21, 2013.

[33] J. Wang, Z. Li, M. Li, Y. Liu, and Z. Yang, "Sensor network navigation without locations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1436–1446, Jul. 2013.

[34] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: A wireless sensor network testbed," in *Proc. ACM/IEEE IPSN*, 2005, Art. no. 68.

[35] S. Xia, N. Ding, M. Jin, H. Wu, and Y. Yang, "Medial axis construction and applications in 3D wireless sensor networks," in *Proc. IEEE INFOCOM*, 2013, pp. 305–309.

[36] S. Xia, X. Yin, H. Wu, M. Jin, and X. Gu, "Deterministic greedy routing with guaranteed delivery in 3D wireless sensor networks," in *Proc. ACM MobiHoc*, 2011, Art. no. 1.

[37] Y. Yang, M. Jin, Y. Zhao, and H. Wu, "Distributed information storage and retrieval in 3D sensor networks with general topologies," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1149–1162, Aug. 2015.

[38] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *Proc. ACM MobiCom*, 2002, pp. 148–159.

[39] 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. IEEE INFOCOM*, 2012, pp. 2681–2685.

[40] 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. IEEE SECON*, 2011, pp. 494–502.

[41] H. Zhou, M. Jin, and H. Wu, "A distributed Delaunay triangulation algorithm based on centroidal Voronoi tessellation for wireless sensor networks," in *Proc. ACM MobiHoc*, 2013, pp. 59–68.

[42] D. Zhu *et al.*, "Boundary-free skeleton extraction and its evaluation in sensor networks," *Wireless Netw.*, vol. 21, no. 1, pp. 269–280, 2015.

[43] X. Zhu, R. Sarkar, and J. Gao, "Segmenting a sensor field: Algorithms and applications in network design," *ACM Trans. Sensor Netw.*, vol. 5, no. 2, 2009, Art. no. 12.

Wenping Liu (M'12) received the Ph.D. degree in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 2012. He is a Post-Doctoral Fellow with the School of Computer Science and Technology, Huazhong University of Science and Technology. Meanwhile, he is an Associate Professor with Hubei University of Economics, Wuhan, China. His research interests include mobile computing and wireless networks. He is a member of the IEEE.

Tianping Deng received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 1998, and the M.S. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2007, respectively. He then joined the faculty of Huazhong University of Science and Technology as an Assistant Professor. His research interests include computer networking and wireless communication.

Yang Yang (S'12) received the M.S. degree from Wuhan University of Technology, Wuhan, China, in 2012. He is currently pursing the Ph.D degree in electronic and information engineering at Huazhong University of Science and Technology, Wuhan, China. His research interests include mobile computing and sensor networks. He is a student member of the IEEE.

Hongbo Jiang (M'08–SM'14) received the Ph.D. degree in computer science from Case Western Reserve University, Cleveland, OH, USA, in 2008. He is a Professor with the faculty of Huazhong University of Science and Technology, Wuhan, China. His research concerns computer networking, especially IoTs and mobile computing. He is a Senior Member of the IEEE.

Xiaofei Liao (M'12) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2005. He is a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. He conducts research in the general areas of desktop and system virtualization technology, P2P computing, multimedia streaming, and wireless networks. He is a member of the IEEE.

Jiangchuan Liu (M'03–SM'08) received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology, Hong Kong, in 2003. He is a University Professor with the School of Computing Science, Simon Fraser University, Vancouver, BC, Canada, and an NSERC E.W.R. Steacie Memorial Fellow. His interests include multimedia systems and networks, cloud computing, social networking, online gaming, big data computing, wireless sensor networks, etc. He is a Senior Member of the IEEE.

Bo Li (F'11) received the PhD. degree in electrical and computer engineering from the University of Massachusetts, Amherst, MA, USA, in 1993. He is a a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. His recent research interests include: large-scale content distribution in the Internet, peer-to-peer media streaming, the Internet topology, cloud computing, green computing, and communications. He is a Fellow of the IEEE.

Guoyin Jiang received the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2010. He is an Associate Professor with Hubei University of Economics, Wuhan, China. His research interests include decision support systems, simulation, and e-commerce.