# A Unified Framework for Line-Like Skeleton Extraction in 2D/3D Sensor Networks

Wenping Liu, *Member, IEEE*, Hongbo Jiang, *Senior Member, IEEE*, Yang Yang, *Student Member, IEEE*, Xiaofei Liao, *Member, IEEE*, Hongzhi Lin, and Zemeng Jin

**Abstract—**In sensor networks, skeleton extraction has emerged as an appealing approach to support many applications such as load-balanced routing and location-free segmentation. While significant advances have been made for 2D cases, so far skeleton extraction for 3D sensor networks has not been thoroughly studied. In this paper, we conduct the first work of a unified framework providing a connectivity-based and distributed solution for *line-like* skeleton extraction in both 2D and 3D sensor networks. We highlight its practice as: 1) it has linear time/message complexity; 2) it provides reasonable skeleton results when the network has low node density; 3) the obtained skeletons are robust to shape variations, node densities, boundary noise and communication radio model. In addition, to confirm the effectiveness of the line-like skeleton, a 3D routing scheme is derived based on the extracted skeleton, which achieves balanced traffic load, guaranteed delivery, as well as low stretch factor.

**Index Terms—**2D/3D sensor networks, connectivity-based, line-like skeleton

## 1 Introduction

Topological feature of sensor networks can greatly affect the design of point-to-point routing, data gathering scheme, and so on. Skeleton (or medial axis) is an important infrastructure of sensor networks as it can accurately capture the topological features of the network. A variety of applications, such as routing [4], navigation [6], localization [16], segmentation [29], etc., can benefit from this abstraction of the underlying geometric environment.

The potential applicability of skeleton naturally inspired many researchers to study its computation in sensor networks. Bruck et al. [4] proposed an algorithm named MAP which identifies medial axis nodes that have at least two nearest boundary nodes which are not too close. However, MAP is sensitive to boundary noise. As such, Zhu et al. [29] proposed to identify skeleton nodes based on the interval of the nearest boundary nodes, and Jiang et al. [13] designed an algorithm to locate a skeleton node if it has two or more nearest boundary nodes on different branches, trying to alleviate the effect of boundary noise. With an input of incomplete boundaries, DIST [17], [18] conducts skeleton extraction by first building the hop count distance transform. Each node then identifies itself by comparing the hop count transform of itself and its neighbors. Liu et al. [19] proposed a boundary-free framework for skeleton extraction which identifies as skeleton nodes the nodes with a locally maximal index computed based on the neighborhood size of each node.

Despite the success on 2D environments, so far there has been no line-like skeleton extraction designed for 3D sensor networks. Previous solutions mostly assume a 2D space, explicitly or implicitly, and exploit the network's 2D geometric properties. Such properties often take a very different form or are more difficult to utilize in 3D spaces (see Fig. 1). For example, connecting the skeleton nodes to form a line-like skeleton in a flooding-based manner is the basic idea of the schemes in [5], [13], [18], [19]. It is generally more difficult to connect nodes in 3D space than to connect nodes on the plane. With only connectivity information, in most scenarios, e.g., networks with parallel boundaries as shown in Fig. 2a, the algorithm using flow complex [29] (referred to as naive algorithm) will fail at local maxima, incurring a self-disconnected skeleton, see Fig. 2b. Besides, in principle, simply extending these 2D solutions to 3D settings will result in a *surface skeleton* (or medial surface) [1] composed of a series of 2D manifolds (also called skeletal sheets), as shown in Fig. 1a. This representation of the skeleton is computationally expensive and often demands a great amount of storage space at nodes, and thus is infeasible for resource-constrained sensor networks. Furthermore, while surface skeleton has been used for navigation in 3D sensor networks [27], however, when the surface skeleton is used to facilitate routing (that is, using the skeleton as a reference path, the packet is always forwarded along the direction "parallel" to the skeleton if possible), the routing path can be considerably long. For instance, as shown in Fig. 1c, points $p$ and $q$, located on the opposite sides of the surface skeleton, have the same distance $h$ to the nearest surface skeleton point (shown by the solid blue circle). The packet from point $p$ to $q$ is forwarded along the direction indicated by the green curve, and thus suffers a long path to delivery.

- *W. Liu, H. Jiang, Y. Yang, H. Lin and Z. Jin are with School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, HuBei, China. E-mail:{wenpingliu2009, hongbojiang2004, yangyang8795, eihongzhilin2012, jinzemeng}@gmail.com.*
- *X. Liao is with the School of Computer, Huazhong University of Science and Technology, Wuhan, HuBei, China. E-mail: xfliao@hust.edu.cn.*

*Manuscript received 5 Sep. 2013; revised 21 Mar. 2014; accepted 25 Mar. 2014. Date of publication 13 Apr. 2014; date of current version 8 Apr. 2015. Recommended for acceptance by I. Stojmenovic.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TC.2014.2317184*

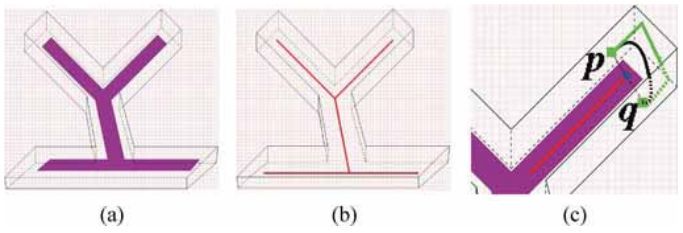


Fig. 1. Surface skeleton, line-like skeleton and their applications in routing. (a) The surface skeleton (indicated by the shaded two-manifolds); (b) The line-like skeleton (red curve); (c) Points $p$ and $q$ have the same nearest skeleton point shown by the solid blue circle. The routing path from $p$ to $q$ generated by the routing protocol on top of the line-like skeleton is indicated by the dark black curve, and the routing path generated by the surface skeleton based routing protocol is indicated by the green curve.

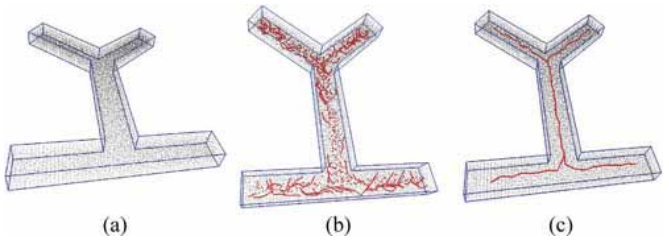


Fig. 2. Skeleton extraction in a Y-shaped 3D sensor network with 12,545 nodes and avg.deg 22.53. (a) Original network. Boundary nodes are marked in blue; (b) The skeleton extracted by the naive method; (c) The skeleton by our algorithm.

In the literature of computer graphics, an alternative skeletal representation of a 3D object is a line-like skeleton [3], as shown in Fig. 1b, which is a 1D curve possibly with branches. The line-like skeleton, also referred to as curve-skeleton, centerline, or inverse-kinematic skeleton, is a subset of the surface skeleton. Compared with surface skeleton, line-like skeleton has lower dimensionality, and is easier to compute and occupies less storage space. This kind of line-like skeletons in both 2D and 3D environments, satisfies desirable properties: homotopic to the original object, centered, connected, robust, hierarchical, etc. As detailed in Section 4.6, the routing protocol on top of the line-like skeleton is delivery-guaranteed and has low stretch factor. Fig. 1c shows an example. The packet from $p$ to $q$ is forwarded along the circle with the radius $h$ centered at the skeleton point (shown by the solid blue circle). It can be found that the routing path by the line-like skeleton based routing protocol has smaller length than the path by the surface skeleton based routing protocol.[1]

As such, in this paper, our objective is to extract the line-like skeleton of a 2D/3D sensor network with reliance on mere connectivity information. To that end, we first derive a unified definition of line-like skeleton points in both 2D and 3D settings. Then, in practice, to obtain the skeleton of a sensor network, for each node, we first identify its *feature nodes* and construct a set of connected components, and thus identify it as a skeleton node when the connected components can be connected to form at least one loop (for 3D sensor networks) or curve (for 2D sensor networks) such that the boundary can be decomposed into two or more connected components accordingly. To connect the identified skeleton nodes, we propose importance measure of skeleton nodes, based on which a skeleton tree is constructed. Finally using the proposed branch similarity, we measure and trim the redundant skeleton branches, as shown in Fig. 2c. To the best of our knowledge, this is the first work on skeleton extraction in 3D networks, and our contribution is a novel skeleton extraction solution feasible for both 2D/3D networks. Our algorithm is desirably distributed, connectivity-based, scalable, and robust to boundary noise, etc. In addition, in order to demonstrate the effectiveness of the extracted skeleton, we propose a line-like skeleton based 3D routing scheme, which achieves balanced traffic load, guaranteed delivery, as well as low stretch factor.

The remainder of the paper is organized as follows. We briefly introduce related work in Section 2, and in Section 3 we present the theoretical foundations of this work. Section 4 is devoted to the skeleton extraction algorithm in 2D/3D sensor networks. We demonstrate the effectiveness of the algorithm in Section 5, and conclude the paper in Section 6.

## 2 Related Work

In this section, we briefly introduce some representative studies on skeleton extraction and its applications in sensor networks, respectively.

*Skeleton extraction.* Assuming complete boundaries, Bruck et al. [4] proposed MAP, a medial axis-based routing protocol. MAP identifies medial axis nodes that have at least two nearest boundary nodes, and eliminates unstable medial axis nodes whose nearest boundary nodes are not nearby, e.g., with a distance larger than a certain hops. However, MAP is still sensitive to boundary noise and possibly delivers a medial axis with spurious branches. To address this problem, Zhu et al. [29], [30] proposed to identify skeleton nodes based on the interval of the nearest boundary nodes, trying to alleviate the effect of boundary noise; Jiang et al. [13] proposed an algorithm, entitled CASE, to identify corner nodes and accordingly segment the boundaries into branches; an interior node identifies itself as a skeleton node if it has two or more nearest boundary nodes located on different branches. By posing the threshold value on the corner node, a multi-scale skeleton can be easily obtained. Different from the above algorithms which require complete boundary information, Liu et al. [17], [18] proposed to identify incomplete boundaries based on the neighborhood-size of each node, and then build the hop count distance transform. A skeleton node is such that its hop count transform is locally maximal. The merit of Dist is that it can be used in sparse networks where complete boundary information is often difficult to obtain and thus above algorithms do not work well. When the network is complex, e.g., with concave nodes, however, the obtained skeleton will bend toward or even cross the boundaries, due to the boundary incompleteness. As such, Liu et al. [19]

1. Formally, we can give a brief explanation. First the surface skeleton is a two-manifold, that is, for any line-like skeleton point $v$ in the two-manifold, there exists a small value $r > 0$ such that its open $r$-neighbors (the points having a distance less than $r$ to the point $v$) are also in the two-manifold. Then, for any point $u$ which has a distance $h$ to the neighbors of point $v$, the distance between $u$ and $v$ is $\sqrt{r^2 + h^2} > h$. That is, the circle centered at the line-like skeleton point with radius $h$ is totally included in a region bounded by the routing path generated by the surface skeleton based routing protocol, which suffices to confirm the observation.

TABLE 1
List of Key Notations

| Notation | Definition |
|---|---|
| $D$ | A 2D or 3D object. |
| $S(D)$ | The surface skeleton of D. |
| $T(x)$ | The distance transform of $x$. |
| $F(x)$ | The feature transform of $x$. |
| $f(x)$ | The geodesic distance function of point $x$. |
| $\mathcal{CP}(\mathcal{D})$ | The core point of D. |
| $\nabla T$ | The gradient field of D. |
| $V(x)$ | The flow vector field of $x$. |
| $\rho(p), \tau(p)$ | The importance measure of $p$ in continuous space and sensor networks, respectively. |
| $Sim(B(p))$ | Branch similarity of branch B(p) of root $p$. |

proposed a framework for skeleton extraction which, rather than relying on boundary information, is based on purely neighborhood size of each node. For each node, it first constructs an index to quantitatively measure its centredness, and the node with locally maximal index identifies itself as a skeleton node. Note that the above-mentioned algorithms target only for 2D sensor networks. For 3D sensor networks, Xia et al. [27] proposed a surface skeleton extraction in 3D sensor networks. First, the unit tetrahedron cell (UTC) mesh structure of 3D sensor networks is established. Then, starting from the boundary surface, one layer of the UTC mesh structure is iteratively "peeled" off where the growing pace of the inner boundary surfaces and the shrinking pace of the outer boundary surfaces are the same. The iteration continues until there is no space for further growing and shrinking, and eventually, the surface skeleton, composed of 2-manifolds, is delivered.

*The applications of skeleton extraction.* Bruck et al. [4] designed a medial axis based routing protocol which is delivery-guaranteed and load-balanced. Buragohain et al. [6] proposed to compute a safe navigation path based on a sparse graph, named skeleton, for internal users under the presence of dangerous events. Zhu et al. [29] decomposed an irregular 2D network into nice pieces based on the skeleton, and showed that it can improve the performance of distributed indices and random sampling. In [16], landmarks are selected according to the distance to the skeleton, and then a distributed localization scheme is proposed. Wang et al. [25] used the skeleton to build a road-map which navigates the users in the sensing field to a safe exit with guaranteed safety. Xia et al. [27] used the medial axis to improve the performance of navigation and data storage in 3D sensor networks.

## 3 Theoretical Foundations

In this section, we first derive a unified definition for line-like skeleton points in both 2D and 3D continuous domains. And then we propose a metric to measure the degree to which a point reflects the details of the underlying object. The usage of this metric, as will be shown in Section 4, is to serve as a guidance to connect the identified skeleton points. For simplicity, we only focus on the case of the objects with simple shape, and later in Section 4.4 we will discuss how to adapt it to the case with complex shape.

### 3.1 Line-Like Skeleton Point Definition

Let $D \subset \mathbb{R}^k$ be a $k$-dimensional$(k = 2, 3)$ object bounded by a connected $(k-1)$-manifold $\partial D$ (A list of key notations can be found in Table 1). Especially, for a 3D object, we assume its boundary surface to be smooth($C^\infty$), compact and without boundary. We define the (euclidean) distance transform $T : D \to \mathbb{R}$ as $T(x \in D) = \min_{y \in \partial D} d_E(x, y)$ for $x \notin \partial D$ and 0 otherwise, where $d_E(x, y)$ is the (euclidean) distance of point $x$ to $y$; let $F : D \to \mathbb{P}(\partial D)$ be a feature transform assigning to each point $x \in D$ the set of the nearest points (namely, *feature points*) on $\partial D$ to $x$, where $\mathbb{P}(S)$ denotes the power set, i. e., the set of all nonempty subsets of $S$. Then, we formally have $F(x) = \{z \in \partial D | d_E(x, z) = T(x)\}$.

We start with the surface skeleton of $D \subset \mathbb{R}^3$, denoted by $S(D)$, followed by our definition for the line-like skeleton point. Formally, $S(D) = \{x \in D | \#\{F(x)\} \geq 2\}$. Clearly, $S(D)$ can be decomposed into two subsets: $S_1(D)$ and $S_2(D)$, where $S_1(D) = \{x \in D | \#\{F(x)\} = 2\}$, and $S_2(D) = \{x \in D | \#\{F(x)\} > 2\}$. We call a point in $S_1(D)$ and $S_2(D)$ as a *generic* and *non-generic skeleton point*, respectively. We first study the properties of a generic skeleton point, and then show that a non-generic skeleton point also has the same properties.

For any point $x \in S_1(D)$, let $x_a$ and $x_b$ be its two feature points; let $X$ be the set of the feature points, i.e., $X = \{(x_a, x_b) | x \in S_1(D)\}$. Define the feature distance function $F_d : X \to \mathbb{R}$, and $f = F_d \circ F$, then $f$ is a mapping from $S_1(D)$ to the one-dimensional space $\mathbb{R}$, i.e., for a point $x \in S_1(D)$, $f(x)$ is the geodesic distance (the distance along the surface rather than through the space) between its feature points $x_a$ and $x_b$; we call $f$ the *geodesic distance function* (GDF). Specially, for a point with only one feature point, we define its geodesic distance function as 0. Clearly, for each point $x \in S_1(D)$, if there are two globally geodesic shortest paths between $x_a$ and $x_b$, they must have equal length. As such, $f$ is well defined on $S_1(D)$ and $D$.

Since we have assumed the boundary to be smooth, it can be shown by using a similar technique as in [2] that $S_1(D)$ is a smooth manifold, and $F : D \to \mathbb{P}(\partial D)$ is differentiable [8]. Define $\Gamma : S_1(D) \to \mathbb{R}^2$ as the local coordinate function. Then $\Gamma$ is a diffeomorphism, respecting the fact that $S_1(D)$ is a smooth manifold. Further, we define $\Psi : \mathbb{R}^2 \to \mathbb{R}$ such that for each point $x \in D$, we have $\Psi(\Gamma(x)) = f(x)$. As such, from the perspective of differential geometry, $f$ is differentiable at point $x$ if and only if $\Psi$ is differentiable at $\Gamma(x)$.

Next we first introduce some results about geodesic distance function and geodesic shortest path, and then we derive a unified definition of the line-like skeleton for 2D/3D objects.

**Lemma 1.** *The function $f$ has no local minimum.*

**Proof.** We only prove this for 2D objects since a 3D object with genus 0 can be *cut* into slices with boundary parallel to the gradient field $\nabla T$ [22], and each interior point falls in only one slice; hence its feature points are all on the boundary of the 2D slice. To that end, we only need to prove that for any point $y$ having two feature points $y_a$ and $y_b$, there is a neighboring point $x$ satisfying that the geodesic distance $f(x)$ between its two feature points $x_a, x_b$ is less than $f(y)$. Please see Fig. 3a. Clearly, we can always find a point $x$ such that one of its feature points, e. g., $x_a$ is on the geodesic shortest path between $y_a$ and $y_b$. We prove by contradiction that $x_b$ is also on the geodesic shortest path between $y_a$ and $y_b$. If $x_b$ is not on the geodesic

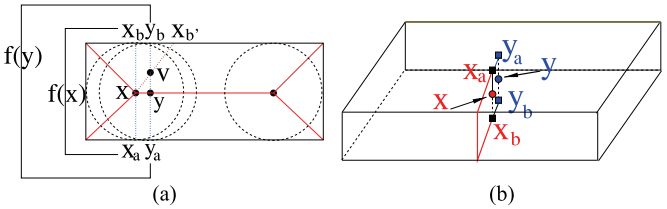


Fig. 3. An illustration for Lemma 1 and Theorem 4.

shortest path between $y_a$ and $y_b$, e.g., the feature point $x_b$ moves to point $x_{b'}$, then the chord $xx_{b'}$ will intersect with the chord $yy_b$, say at a non-skeleton point $v$. That is, the point $v$ stays on two chords, contradicting the fact that each non-skeleton point stays on a unique chord, which has already been proved by [4]. Therefore, $x_b$ is also on the geodesic shortest path between $y_a$ and $y_b$, and the geodesic distance between $x_a$ and $x_b$ is less than that between $y_a$ and $y_b$. That is, $f(x) < f(y)$ which proves the claim. □

**Lemma 2.** *The geodesic distance function $f$ is not differentiable everywhere.*

**Proof.** We only prove this for 2D object $D$. Assume that $x$ is a skeleton point, $x_a$ and $x_b$ are its two feature points, then we have $f(x) > 0$. For each point $y_1, y_2$ on the two chords $xx_a, xx_b$, it has been proven in [4] that $x_a$ (or $x_b$) is the unique feature point of $y_1$ (or $y_2$). That is, $f(y_1) = f(y_2) = 0$. It naturally follows that $f$ is not differentiable at the point $x$. In a similar way, we can prove that the geodesic distance function $f$ is not differentiable at any skeleton point. □

Lemma 2 says that the geodesic distance function $f$ is not differentiable at skeleton points. Next we will prove that the singularities of $f$, i.e., the points where $f$ is not differentiable, has a Lebesgue measure zero.

**Definition 1.** *A function $g : \mathbb{R}^k \to \mathbb{R}$ is locally Lipschitz continuous (also referred to as locally $K$-Lipschitz) if for any point $x \in \mathbb{R}^k$, there exists a real constant $K \geq 0$ and some $\epsilon > 0$ such that, for every point $y \in \mathbb{R}^k$ satisfying $|g(x) - g(y)| < \epsilon$, the following inequality holds:*

$$|g(x) - g(y)| \leq K\|x - y\|. \tag{1}$$

Obviously, the (euclidean) distance function $F$ is locally Lipschitz continuous since, for any two points $x, y \in \partial D$, $F(x) \leq F(y) + 1 \cdot \|x - y\|$ always holds, and

**Lemma 3.** *$\Psi$ is locally Lipschitz continuous.*

With Lemmas 1, 2 and 3, according to Rademacher's theorem [10], the singularities of $f$ has a Legesgue measure zero, implying that the singularities of $f$ consists of 1D curve, namely, the line-like skeleton [8]. In other words, a point $v$ is a line-like skeleton point if the function $f$ is not differentiable at point $v$.

To compute the singularities of the function $f$, however, is cumbersome and impractical for sensor networks. Fortunately, we have the following theorem:

**Theorem 4.** *For any non-singular point of $f$, there is only one geodesic shortest path; and for each singular point, there are two geodesic shortest paths between its feature points, decomposing the boundary into connected components.*

**Proof.** For 2D objects, it is obvious that the geodesic shortest path between any two feature points of a singular point of $f$ segments the boundary into two parts. We now prove that this is the case for 3D objects. We prove by contradiction that for a singular point $x$, there are more than one geodesic shortest paths between its two feature points $x_a$ and $x_b$. Suppose that there is only one geodesic shortest path between its feature points, please see Fig. 3b. This implies that there exists a neighboring point, say $y$, such that $y's$ geodesic distance between its feature points $y_a$ and $y_b$ is larger than that of $x$, namely, $f(y) > f(x)$. According to Lemma 1, there exists a point $y'$ such that $f(y') < f(x)$. When the points $y, y'$ approach to point $x$, the feature distances $f(y), f(y')$ converge to $f(x)$, that is, the function $f$ is smooth at point $x$, which contradicts that $x$ is a singular point of $f$. Since there are two geodesic shortest paths between $x_a$ and $x_b$, they should have the same length and form at least one loop. Accordingly, the boundary of the object is decomposed into connected components. □

Theorem 4 provides a simple way for skeleton point identification: if point $v$ is such that the geodesic shortest paths between its feature points form a *feature loop*, decomposing the boundary into components, then point $v$ is a skeleton point; otherwise, it is a non-skeleton point, as shown in Fig. 6a.

Next we consider the non-generic skeleton in $S_2(D)$, i.e., the points with three or more feature points. It happens at the intersection of $n(\geq 3)$ 2-manifolds with boundaries in $S_1(D)$. It is easy to show that the neighbors of any point $x \in S_2(D)$ do not necessarily belong to $S_2(D)$, that is, $S_2(D)$ is not a 2-manifold. Therefore it either consists of isolated points or forms 1D curve. Clearly, for point $x \in S_2(D)$, it has more than two geodesic shortest paths between its feature points, which form at least one loop and separate the boundary into connected components.

Note that a skeleton point of 2D objects also has the same property, i.e., the shortest paths between its feature points form a curve decomposing the boundary into branches. As such, we now present a unified definition of the line-like skeleton in 2D/3D space as follows:

**Definition 2.** *A point is a* line-like skeleton point *if and only if the geodesic shortest paths between its feature points decompose the boundary into two or more connected components.*

Following this definition, we can identify the skeleton points, without reliance on any parameters. However, it is noted that these identified skeleton points is not self-connected. A straightforward method is to use flow complex [7], [29], however, there can be many local maxima, resulting a set of disconnected skeleton arcs, as mentioned in Section 1. Thus, it faces a non-trivial challenge to connect these skeleton points into a meaningful representation of the underlying 2D/3D object.

### 3.2 Importance Measure of Line-Like Skeleton Point

To connect the identified skeleton points, in this section, we propose the *importance measure* associated with the line-like skeleton points, based on which a skeleton tree can be easily constructed.

#### 3.2.1 Importance Measure in 2D

Let $D$ has no holes. Its skeleton $S(D)$ can be defined as the singularities of distance transform $T : D \to \mathbb{R}$. Let $\nabla T$ be the gradient field of $D$. Obviously, $\nabla T$ is undefined on $S(D)$. Conventionally, each boundary point only flows to $S(D)$ following the direction of $\nabla T$, so the delivered skeleton is self-disconnected. To address this, we define a flow vector field $\mathbf{V}$ such that after flowing to $S(D)$, the point keeps flowing along the skeleton until it reaches a unique point of $D$, called *core point* $\mathcal{CP}(\mathcal{D})$, and deliver a self-connected skeleton. Specifically, for non-skeleton point $z$, similar with [29], we define the driver $d(z)$ as its unique feature point. For a skeleton point, however, we define its driver as the neighboring skeleton point with smallest GDF, instead of the sink itself in [29], to achieve a self-connected skeleton. For $\mathcal{CP}(\mathcal{D})$, the driver is itself. Accordingly, we define the flow $\mathbf{V}$ as $\mathbf{V}(z) = \frac{z-d(z)}{|z-dz|}$ for $z \neq d(z)$ and 0 otherwise.

Governed by the flow $V$, each non-skeleton point will first flow to $S(D)$ and then keep moving along $S(D)$ toward the core point $\mathcal{CP}(\mathcal{D})$. Since $D$ has no holes, its skeleton $S(D)$ is a tree-like structure. As such, each skeleton point $p$ can divide it into two sub-trees and accordingly, the flows passing through a skeleton point $p$ will decompose the object into connected components, and the origins of the flows also decompose the boundary $\partial D$ into connected components, say $C_1(p), C_2(p), \ldots, C_l(p)(l \geq 2)$. As such,

**Definition 3.** *We define the Importance Measure of $p$, denoted by $\rho(p)$, as follows*

$$\rho(p) = 1 - \max_i\{\lambda(C_i(p))\}/\lambda(\partial D), \tag{2}$$

*where $\lambda(\cdot) : \Omega \to \mathbb{R}$ is a Lebesgue measure assigning a real value to a subset of $n$-dimensional space $\Omega$.*

Intuitively, the importance measure of a point describes how many *trajectories* flow to this point. Since the core point is a skeleton point where all trajectories "sink" into, it has the largest importance measure.

**Lemma 5.** *Each object has only one core point.*

**Theorem 6.** *The importance measure $\rho(p)$ is a monotonic function with the distance from $p$ to $\mathcal{CP}(\mathcal{D})$.*

**Proof.** Since the unique core point $\mathcal{CP}(\mathcal{D})$ is a skeleton point where all trajectories "sink" into, for any skeleton point $p$ with a distance $d(p)(>0)$ to $\mathcal{CP}(\mathcal{D})$, the amount of trajectories flowing to $p$ is smaller than that of $\mathcal{CP}(\mathcal{D})$. Assume $q$ is a skeleton point at the same side of $p$ to $\mathcal{CP}(\mathcal{D})$. Without loss of generality, assume $d(q) > d(p)$. Clearly, the trajectories flowing to $q$ will surely flow to $p$ but not vice versa. That is, the importance measure of $p$ is larger than that of $q$, which proves the claim. □

Theorem 6 implies that the closer a skeleton point is to the core point, the larger its importance measure, and there only exists one local (also global) maximum, i.e., $\mathcal{CP}(\mathcal{D})$, resulting in a self-connected skeleton. As such, it offers a rule to generate a skeleton tree.

#### 3.2.2 Importance Measure in 3D

We can easily extend the 2D model to a 3D object $D$ such that a monotonic importance measure is applicable for extracting a robust and self-connected line-like skeleton for the 3D object $D$.

Similar with 2D model, our goal is to construct a flow vector field $\mathbf{V}$ which can guarantee that each non-skeleton point first flows to the line-like skeleton following the direction of the gradient field $\nabla T$(where $T$ is a distance transform) and then keeps moving along the line-like skeleton toward a unique core point $\mathcal{CP}(\mathcal{D}) \in S(D)$, thereby a self-connected skeleton can be obtained. Compared with 2D model, however, defining $\mathbf{V}$ on a point in a 3D object is more challenging since an improper flow vector field might generate a surface skeleton which contains 2-manifold sheets.

Our solution is a divide-and-conquer strategy. More specifically, we first *cut* the 3D object $D$ into slices, and then apply our 2D model on these slices to define the flow vector field $\mathbf{V}$. Let $D_s$ be a slice with boundary $\partial D_s \subset \partial D$, and $\mathcal{CP}(D_s)$ be its core point, called *local core point*. Within each slice $D_s$, we can define the flow vector field $\mathbf{V}_s$ using our 2D model. Namely, for each non-skeleton point $z \in D_s$, the driver $d(z)$ is its unique feature point in $\partial D_s$, and the driver of a skeleton point is the neighboring skeleton point with the largest GDF. Note that we cannot define the driver of the local core point as itself, otherwise the local core points of these slices are not self-connected. To address this issue, note that each slice divides the boundary into components, we call the local core point of the slice which decomposes the boundary surface into two halves as the *global core point*, and define the driver of each local core point as the neighboring local core point closest to the global core point. As for the global core point, the driver is itself. As such, the flow vector field $V$ can be constructed accordingly. That is, for each point $z \in D$, $\mathbf{V}(z) = \frac{z-d(z)}{|z-dz|}$ for $z \neq d(z)$ and 0 otherwise.

The slices should be carefully chosen such that its normal is vertical to the gradient field $\nabla T$, and thereby the trajectories originating on each slice boundary remain in the slice. Note that here the slice boundary $\partial D_s$ decomposes the boundary $\partial D$ into components, reminiscent of the geodesic shortest paths between feature points of a line-like skeleton point $p$ that form at least one feature loop and decompose the boundary into connected components, where $p$ corresponds to the local core point. As such, the feature loop provides a good approximation to the slice boundary, and the nodes nearest to the same slice boundary can be regarded as a slice. We thus define the importance measure $\rho(p)$ of the skeleton point in the same way as in the 2D model by Equ. (2). Again, The global core point $\mathcal{CP}(\mathcal{D})$ is where all points sink into and thus has the largest importance measure. In addition, the trajectories flow to the global core point along the direction of monotonically increasing importance measure. The closer a skeleton point(a local core point) is to the global core point, the more details of the object it reflects.

In summary, for both 2D and 3D objects, the line-like skeleton point has a unified definition and has the common property: the (geodesic) shortest paths between its feature points decompose the boundary into connected components $C_1(p), C_2(p), \ldots, C_l(p)(l \geq 2)$, which can be used for measuring the importance of the skeleton point by the Equ. (2) and thus guiding the skeleton tree construction for achieving a self-connected skeleton.

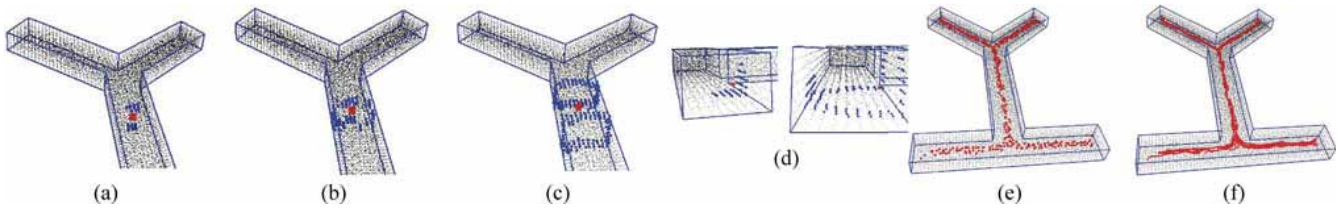


Fig. 4. Line-like skeleton extraction in Fig. 2. (a) An interior node (marked in red) and its feature nodes (marked in blue); (b) Feature loop nodes (marked in blue); (c) The dilated path; (d) The feature nodes (left) and the dilated path (right) of a non-skeleton node; (e) Skeleton nodes; (f) Skeleton tree.

## 4 Algorithm

In this section, we adapt the aforementioned principle of skeleton extraction in continuous domain to the discrete analog. Here boundary information is taken as an input; there are many boundary extraction algorithms in 2D/3D sensor networks, e.g., [9], [15], [26], [28]. We first present the outline of our algorithm, followed by the details of each step. Since the 2D cases are quite similar, we limit the details on 3D spaces here.

1) *Skeleton node identification.* The node, which satisfies that the geodesic shortest paths (for 3D sensor networks) or shortest paths (for 2D sensor networks) between feature nodes decompose the boundary into 2 or more connected components, marks itself as a skeleton node, as shown in Figs. 4e and 5c.

2) *Importance measure computation and skeleton tree construction.* The *importance measure* of each skeleton node is then derived from the computation of the number of nodes in the connected components. After that, each skeleton node chooses the neighboring skeleton node with the largest importance measure as the parent node, and accordingly, a skeleton tree is constructed following the direction of the monotonically increasing importance measure to derive a self-connected skeleton, as shown in Figs. 4f and 5d.

3) *Refinement.* The skeleton tree may contain redundant skeleton branches, owing to the discrete nature of sensor networks. We trim the skeleton tree according to the proposed *branch similarity*. As a result, the final skeleton is obtained, as shown in Figs. 2c and 5e.

### 4.1 Skeleton Node Identification

To identify the feature node(s) of interior nodes, the boundary nodes initiate controlled floods inside the network; each interior node is associated with a boundary tree rooted at one of its feature node and keeps record of the feature nodes (shown in Fig. 4a). Note that this flooding process only causes overall $O(N)$ message cost where $N$ is the network size as each node just forwards the first flooding message it receives. In our algorithm, if an interior node has a minimum hop count distance $k$ to the boundary, the boundary nodes that are $k+1$ hops away are also regarded as the feature nodes, respecting the fact that in special scenarios (e.g., a box-shaped network with even width, or sparse networks), some skeleton nodes possibly have only one feature node.

Based on these feature nodes, we mainly address how to identify skeleton nodes in 3D sensor networks (as for 2D cases, the method is quite similar). Note that there is no need for every interior node to determine its identity, which otherwise incurs a large communication cost. Instead, similar with [18], only the leaf nodes of the boundary trees conduct the skeleton node identification process. In continuous domain, as highlighted in Definition 2 (see Section 3.1), at each skeleton node, the geodesic shortest paths will form at least one loop, separating the boundary surface into two or more parts, as shown in Fig. 6a, which guides our steps to identify skeleton nodes in 3D sensor networks as follows:

1) *Constructing and connecting the feature components.* For any leaf node, each feature node issues a controlled-flooding to construct a set of connected components, each of which is assigned a unique identifier. To connect these components, a controlled hop-by-hop expansion process is conducted for achieving a low cost. More specifically, each feature node $p$ broadcasts a message including its identifier, and $p's$ neighboring boundary nodes (or any $p's$ neighboring nodes in 2D networks) which has not been assigned any identifier, are assigned the same identifier. Such process is conducted repeatedly until two boundary nodes (or any nodes in 2D networks) with different identifiers meet. This way, the geodesic shortest paths between feature components can be built. Without ambiguity, the feature nodes and the nodes on the geodesic shortest paths are henceforth referred to as feature nodes, see Fig. 4b.

2) *Feature loop identification.* In continuous domain, as mentioned in Section 3.1, for a line-like skeleton point in 3D objects, the geodesic shortest paths between its feature points form at least one feature loop, while the set of the geodesic shortest paths between the feature points of a non-skeleton point is only a non-loop curve. As such, to identify whether a leaf node is a line-like skeleton node, the presence of the feature loop(s) is a key. One might argue that feature loop identification can be simply done as follows: any feature node floods within the feature nodes to build a shortest path tree, if there are two neighboring feature nodes having no common ancestor but the root, then the feature nodes form a loop. For a network with large node density, this might not be problematic; in a sparse network, however, this method does not work. That is, there might be fake loops. Please see Fig. 6b as an example. To deal with this, each feature node first floods within a small scope, i.e., its two-hop neighbors. Those nodes that are at most 2-hops away from these feature nodes then naturally form a dilated path, as shown in Fig. 6c. If the boundary of the dilated path is more than one closed curve, i.e., the nodes that are two hops away from the feature nodes (as shown in Fig. 4c) cannot form one connected component, then there is at least one genuine feature loop, and the corresponding interior node $p$ identifies itself as a skeleton node; otherwise, $p$ is not a skeleton node, as shown in Fig. 4d. Note that this can be easily done in a distributed fashion. Fig. 4e shows the skeleton nodes of a 3D network.

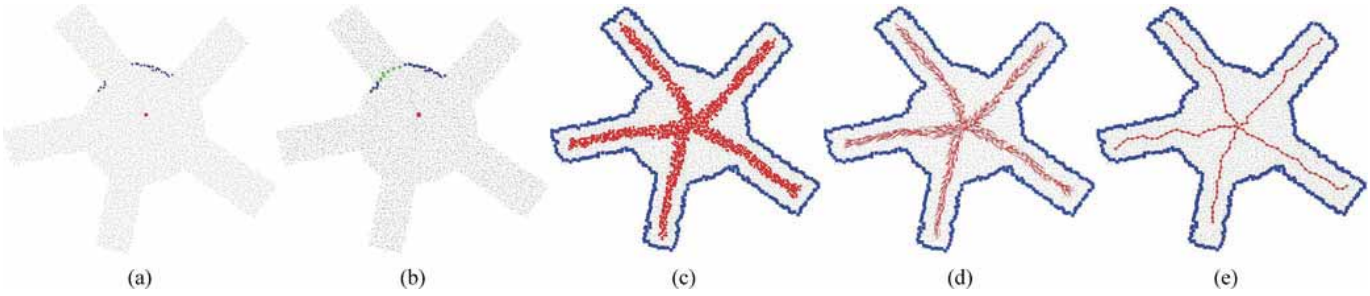


Fig. 5. Skeleton extraction in a sun-shaped 2D sensor network which has 5,217 nodes with average degree 12.27. (a) A skeleton node (marked in black) and its feature nodes (marked in red); (b) Shortest path (marked in green) between connected components; (c) Skeleton nodes; (d) Skeleton tree; (e) Final skeleton.

### 4.2 Importance Measure Computation and Skeleton Tree Construction

In order to connect the identified skeleton nodes to form a meaningful line-like representation of the original network, guided by Definition 3 in Section 3.2, each skeleton node is assigned an *importance measure*, a metric measuring how important the skeleton node is to reflect the details of the network. Based on the importance measure, a skeleton tree can be constructed such that a self-connected skeleton is generated.

Note that in 3D sensor networks, the feature loop of a skeleton node decomposes the boundary surface into a set of connected components. In 2D sensor networks without holes, the shortest paths between the feature points of a skeleton node also decompose the boundary into branches. We call the feature loop (in 3D networks) or geodesic path (in 2D networks) as *Feature Component*. For a skeleton node $p$, denote by $C_1(p), C_2(p), \ldots, C_l(p)$ the connected components of the boundary divided by the feature component of $p$, satisfying that $|C_1(p)| \leq |C_2(p)| \leq \cdots \leq |C_l(p)|$. Then the importance measure of $p$, denoted by $\tau(p)$ is given as follows:

$$\tau(p) = 1 - |C_l(p)|/|C|, \tag{3}$$

where $C$ is the set of boundary nodes. Generally, the importance measure of $p$ ranges from 0 to 1.

After each skeleton node computes its importance measure, a skeleton tree can be constructed to connect these skeleton nodes. According to Theorem 6, the skeleton tree is constructed in a greedy manner, namely, each skeleton node selects the neighboring node with the largest importance measure as its parent node, which finally forms a skeleton tree rooted at the so-called *core node* (the counterpart of core point in continuous domain), mimicking the flow in continuous domain as described in Section 3.2. Obviously, the core node, where all skeleton nodes "sink" into, has the largest importance measure. Fig. 4f shows the result of skeleton tree construction. Note that there are redundant skeleton branches, owing to the discrete nature of sensor networks and thus many nodes being identified as skeleton nodes. As such, most of them are *similar* with each other, i.e., they have some common feature nodes. In the next section, we will conduct a pruning process to deliver a line-like skeleton.

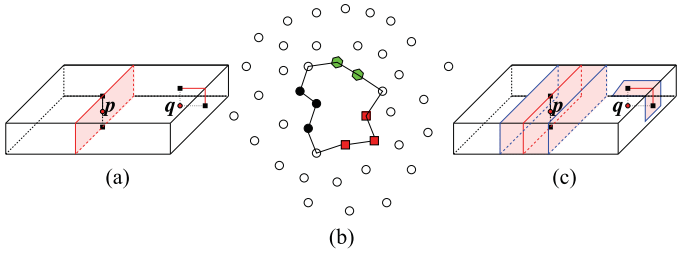


Fig. 6. Feature loop identification. (a) Feature loop curve and non-loop curve; (b) Fake feature loop. The solid nodes are the feature nodes (the feature nodes with the same shape are in a connected component) of an interior node and the empty circles are other boundary nodes. The loop does not decompose the boundary surface into pieces, therefore it is a fake loop; (c) The dilated paths (blue curve) is the set of points having the same distance to the feature loop (red curve). The boundary of the dilated path of $p, q$ have 2 and 1 closed curve respectively, therefore $p$ identifies itself a line-like skeleton point, and $q$ is a non-skeleton point.

### 4.3 Refinement

As mentioned, the skeleton tree may have unwanted skeleton branches, and thus refinement is needed for a good skeleton. To this end, we propose a metric named *branch similarity*, based on which we can obtain a simplified skeleton without unwanted skeleton branches.

We first present the definition of branch similarity.

**Definition 4.** *For two skeleton branches $B_1, B_2$ with a common parent node $p$, let $L_1, L_2$ denote the depths of the branches $B_1, B_2$ respectively. For two nodes $p_1^i, p_2^i$ on the branches $B_1, B_2$, which are both $i(\leq L = \min\{L_1, L_2\})$ hops from node $p$, define $I(p_1^i, p_2^i)$ to be 1 if $p_1^i, p_2^i$ are neighboring, and 0 otherwise. Then the branch similarities between $B_1$ and $B_2$ of $B_1$, and $B_2$, denoted by $Sim(B_1|B_2)$, and $Sim(B_2|B_1)$ respectively, are defined as:*

$$\begin{aligned} Sim(B_1|B_2) &= \sum_{i=1}^{L} I(p_1^i, p_2^i)/L_1 Sim(B_2|B_1) \\ &= \sum_{i=1}^{L} I(p_1^i, p_2^i)/L_2. \end{aligned} \tag{4}$$

*If the parent node $p$ has $I(>0)$ children nodes that generate $I$ skeleton branches accordingly, then the branch similarity of the branch $B(p)$, including $p$ and the nodes on the $I$ branches, is defined as*

$$Sim(B(p)) = \max_{i,j \leq I, i \neq j} \{Sim(B_j|B_i)\}. \tag{5}$$

Fig. 7 shows the principle of computing branch similarity. With the branch similarities, we now simplify the skeleton tree in an iterative fashion. Initially, we eliminate those skeleton branches with branch similarity of 1.0. If all

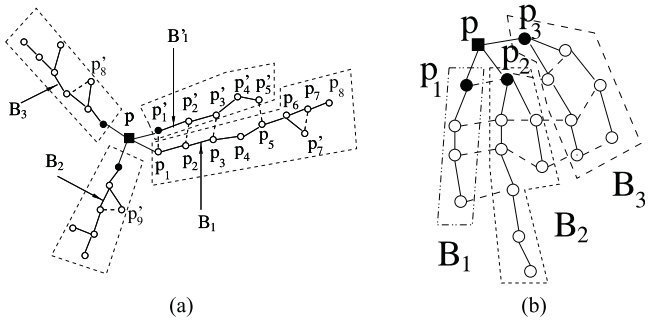


Fig. 7. Skeleton branch similarity. The parent node $p$(shown by solid rectangle) has several children nodes (shown by solid circles) and thereby skeleton branches (bounded by dashed polygons), and the hollow circle are ordinary skeleton nodes. The solid line between two skeleton nodes indicates they are on the same skeleton branch, and the dashed line indicates they are neighboring nodes. (a) Branch $B_1$ and $B_1'$ have 4 pairs of neighboring skeleton nodes, and the skeleton branch similarity of $B_1$ and $B_1'$ are $Sim(B_1'|B_1) = \frac{4}{5} = 0.8$ and $Sim(B_1|B_1') = \frac{4}{9} = 0.44$, respectively; (b) Parent node $p$ has three skeleton branches. $Sim(B_1|B_2) = 1.0$, $Sim(B_2|B_1) = \frac{4}{6} = 0.66, Sim(B_2|B_3) = \frac{2}{6} = 0.33$, $Sim(B_3|B_2) = \frac{3}{6} = 0.5$. Therefore, $Sim(B_1) = 1.0, Sim(B_2) = 0.66$, $Sim(B_3) = 0.5$.

skeleton branches of a parent node have a branch similarity is no less than the given value, then all branches but the one with the largest length is kept. Next, for any given step size $\delta > 0$, the branches with branch similarity larger than $1.0 - \delta$ will be trimmed, followed by trimming the branches with branch similarity larger than $1.0 - 2 \times \delta$, and so on. The iteration is terminated when no more skeleton branches could be deleted, and the final skeleton is thus generated, please see Fig. 2c. Note that our algorithm has no dependency on the step size $\delta$. The extracted skeleton is computed based on the importance measure of each skeleton node, which can reflect the detail of the network on a different spatial scale. A node with large importance measure typically represents the large-scale detail of the network, and the node with small one, possible caused by boundary noise, represents the small-scale detail. By setting the threshold value on importance measure, we can obtain multi-scale and nested skeletons, i.e., the skeleton generated by imposing a small threshold includes the skeleton by a large threshold. Please see Fig. 8.

### 4.4 The Case for Objects with Complex Shape

So far we only concentrate our algorithm on the networks with simple shape. For scenarios with complex shapes (e.g., with tunnels, see Fig. 9a), the origins of the flows passing through a skeleton point could not decompose the boundary into connected components such that the proposed algorithm is no longer valid. To address this issue, the feature loop with the largest number of boundary nodes is first identified, and then the nodes on the feature loop floods simultaneously within boundaries. The nodes, who have the same hop count distance to the feature loop, will form at least one loop. If we treat each loop as a dummy node, then after the flooding, there can be many dummy nodes forming a dummy tree, and the tree will fork at one side of a tunnel and meet at the other side. Next we *cut* the network from the place where the dummy tree meets to form a cross section which can decompose the network (and the boundary) into components, then the proposed algorithm is still applicable (see Fig. 9b).

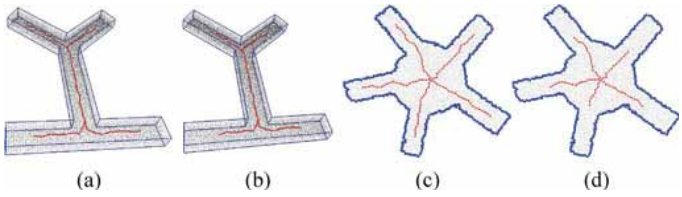


Fig. 8. Multi-scale skeletons by setting the threshold $\tau$ on importance measure. (a) $\tau = 0.1$; (b) $\tau = 0.2$; (c) $\tau = 0.2$; (d) $\tau = 0.3$.

### 4.5 Algorithm Complexity

**Theorem 7.** *Our algorithm has a linear time and message complexity.*

**Proof.** Our algorithm has the following steps: skeleton node identification, importance measure computation, skeleton tree construction, and refinement. We prove the claim by discussing the time and message complexity of our algorithm for 3D networks only, and as for 2D cases, it is quite similar.

*Skeleton node identification*. This process involves feature node identification, geodesic shortest paths construction and feature loop identification. Feature node identification often incurs a time and message complexity of $O(1)$. During the construction of geodesic shortest paths, our strategy is a hop-by-hop expansion. For each node, this often demands a small time and message complexity, which is about $O(1)$, and thus at most $O(N)$ in total (In our algorithm, only a small part of interior nodes conducts the skeleton node identification). The feature loop identification is limited within the dilated paths which are, say, the nodes two-hops away from the geodesic shortest paths, and accordingly, the time and message complexity will be very low, which is about $O(1)$, and thus $O(N)$ in total.

*Importance measure computation and skeleton tree construction*. Let $N_s(D), N_b(D)$ be the number of line-like skeleton nodes and boundary nodes, respectively. The computation of importance measure incurs a time and message complexity of $O(N_s(D) * N_b(D))$, which are about $O(N)$. After each skeleton node computes its importance measure, the skeleton tree is constructed in a greedy manner, which incurs a time and message complexity of $O(N_s(D))$.

*Refinement*. The refinement process is conducted within the identified skeleton nodes, and thus the time and message complexity is $O(N_s(D))$.

In summary, the time and message complexity of our algorithm are both linear to the network size. □

### 4.6 Line-Like Skeleton Based 3D Routing Protocol

Generally, the evaluation of a good skeleton is application-oriented, and no common metric is well accepted yet. In sensor networks, routing is the fundamental function. To quantitatively evaluate the performance of our algorithm, we thus design a delivery-guaranteed, low stretch factor and load-balanced routing protocol based on the extracted line-like skeleton. We are aware that MAP [4] has conducted a line-like skeleton based 2D routing protocol. However, MAP is not applicable for 3D analog, since its naming scheme potentially limits its applicability within 2D sensor networks only. In this paper, we are interested in the application of line-like skeleton in routing for 3D sensor networks.

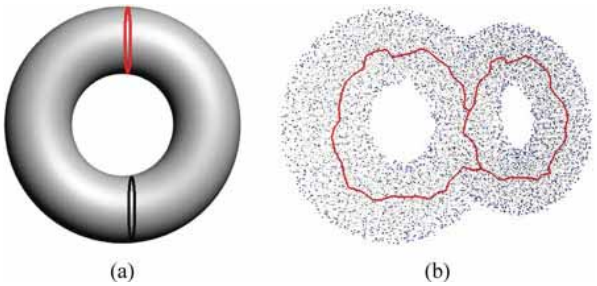


Fig. 9. Our algorithm under complex scenario. (a) The tree rooted at a dummy node indicated by the red eclipse will meet at the dummy node indicated by the black eclipse; (b) The skeleton of double-torus shaped network.

After the extraction of the skeleton, we construct a so-called *skeleton graph*, where each vertex corresponds to a critical skeleton node, which is either an end skeleton node (i.e., a skeleton node having only one neighboring skeleton node) or a joint skeleton node (i.e., a skeleton node having more than two neighboring skeleton nodes); and an edge between two vertices represents that the shortest path between the corresponding two skeleton nodes does not pass through other critical skeleton node. Then we let each skeleton node store skeleton information, including its neighboring skeleton nodes and which edge it is in. Our skeleton-based 3D routing protocol has two major steps:

*Preprocessing phase.* First, skeleton nodes flood simultaneously within the network to build a shortest path forest, where each skeleton node is a root of a shortest path tree (referred to as a skeleton tree). Each node is associated with one and only one skeleton tree, and keeps record of its hop count distance to its root, and the skeleton graph as well. We denote by $r(p)$ the root of node $p$, $h(p)$ the hop count distance of $p$ to $r(p)$. We call the set of nodes that have equal distance $h'$ to the same skeleton node as contour set, denoted by $CS(h')$; and the nodes having equal distance $h'$ to the skeleton are referred to as level set, denoted by $LS(h')$. Obviously, $CS(h') \subset LS(h')$ for any positive $h'$. Next, the source node then figures out the reference path using the skeleton information it stores, from the root of the source to that of the destination.

*Routing phase.* The message from the source $S$ is forwarded to the destination $T$ as follows:

- if $r(S) = r(T)$ and $h(S) = h(T)$, then $S$ forwards the message to its neighbor $q$ on $CS(r(S))$;
- else if $r(S) = r(T)$ and $h(S) \neq h(T)$ (without loss of generality, we assume $h(S) > h(T)$), $S$ forwards the message to its neighbor $q$ such that $h(q) < h(S)$;
- otherwise, $S$ forwards the message to a node $q$ such that $r(q)$ is closer to $r(T)$ than $r(S)$ is to $r(T)$, $q$ severs as an intermediate.

Note that there might be more than one intermediate node based on the above-mentioned policy. The one with the least load is selected as the intermediate node to achieve balanced load. Such process is conducted repeatedly until the message reaches the destination. This way, the delivery of the packet can be guaranteed. Fig. 10 depicts an illustrative example.

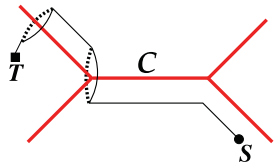


Fig. 10. Routing illustration. The source node $S$ is shown by a circle, and the destination $T$ is shown by a rectangular. The red curve shows the skeleton $\mathcal{C}$, and the black curve is the routing path. The circles around the skeleton are contour sets.

## 5 Performance Evaluation

To demonstrate the efficiency of the proposed algorithm, we conduct extensive simulations on different 2D/3D scenarios, and compare the proposed algorithm with the existing algorithms. Specifically, for 3D sensor networks, we compare with the naive algorithm using traditional flow complex. Note that the flow complex based algorithm suffers from local maxima and incurs a self-disconnected skeleton, as mentioned earlier in Section 1. For fair comparison, similar with our algorithm, the naive algorithm first identifies skeleton nodes by using flow complex, and then selects the skeleton node with the largest distance transform as a root to build a skeleton tree, and finally the skeleton tree is refined based on branch similarities. And for 2D sensor networks, we compare our algorithm with MAP [4] and CASE [13], both of which require complete boundaries.

Besides, we also study the performance of our algorithm under different node densities, and the performance of the proposed algorithm under different scales of boundary noise is also examined. Further, we show the robustness of our algorithm to communication model by using quasi-unit disk graph (QUDG) and log-normal communication radio model. Finally, we adopt the extracted line-like skeleton as an infrastructure to facilitate 3D routing, such that by evaluating the quality of routing paths, we can quantitatively measure the *goodness* of the extracted line-like skeleton.

### 5.1 Simulation Setup

In our simulations, nodes are randomly deployed in the sensing field. The communication model is unit disk graph by default. That is, a link between two nodes exists if their separation is less than the communication radio range. The threshold for importance measure is user-defined and in our algorithm, it is set to be 0.02 unless otherwise stated.

In routing experiments, we randomly choose 20,000 pairs of source and destination nodes. Since our algorithm guarantees delivery, considering the limitation of storage space and computation complexity, we only compared our line-like skeleton based routing protocol with surface skeleton based routing protocol, naive algorithm based routing protocol, and with Random-Walk [11] as well. The surface skeleton based routing protocol and naive algorithm based routing protocol work in the same way as the line-like skeleton based routing protocol. More specifically, each node computes its hop count distance to the skeleton via flooding from the skeleton nodes, and then keeps track of the root skeleton node; at the same time, each node stores the skeleton information, including the node IDs of each skeleton node and its neighbors, for the purpose of guiding the packet's

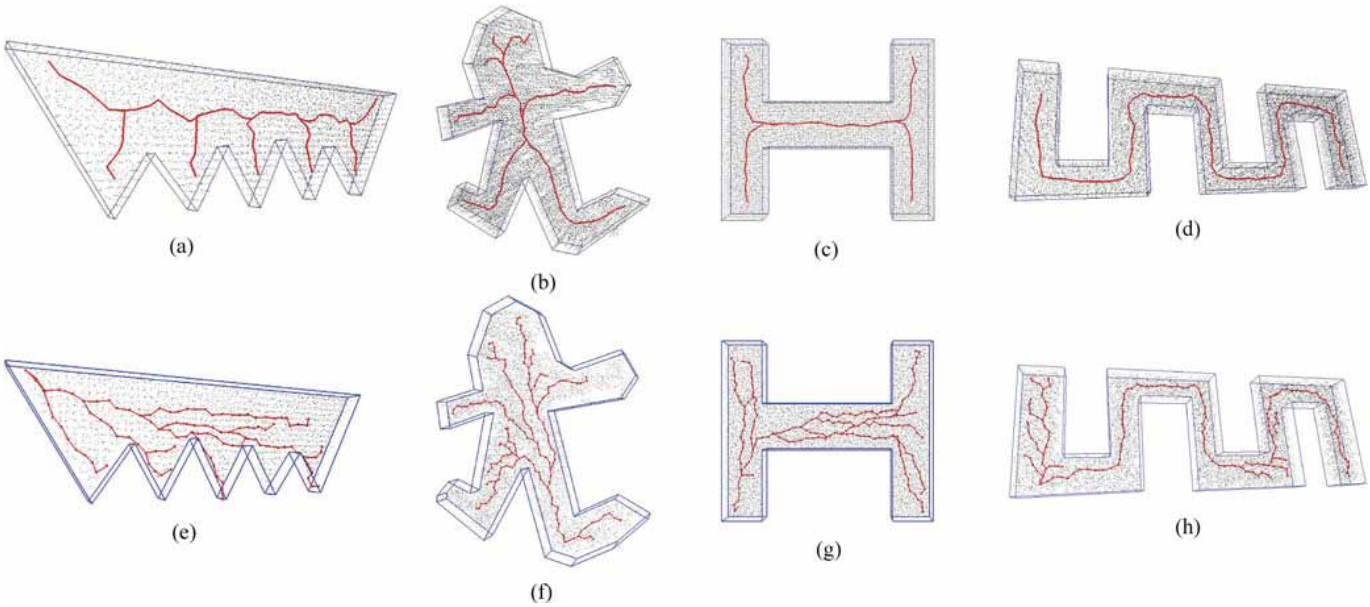


Fig. 11. More results. Row 1: our algorithm; row 2: naive algorithm using flow complex. (a) Seabed, 5,537 nodes, avg. deg. 22.35; (b) Man, 15,288 nodes, avg. deg. 20.78; (c) H, 16,156 nodes, avg. deg. 22.47; (d) Snake, 19,313 nodes, avg. deg. 19.84.

forwarding. The surface skeleton based routing protocol and naive algorithm based routing protocol also have two phases: during the preprocessing phase, the source node finds the reference path in the skeleton nodes, followed by forwarding the packet to its neighbors in the same way as the line-like skeleton based routing protocol does. Random-Walk [11] is a truly greedy routing scheme, where the packet is greedily forwarded to the destination. Upon reaching a local minimum, a random walk is used on a local sphere structure to help it escape from the local minimum.

## 5.2 Simulation Results

### 5.2.1 Performance under Different Scenarios

Fig. 11 illustrates the results of our algorithm and naive algorithm on a variety of scenarios, namely, Seabed (Figs. 11a and 11e), Man (Figs. 11b and 11f), H (Figs. 11c and 11g) and Snake (Figs. 11d and 11h), with the number of nodes ranging from 5,537 to 19,313. In Figs. 11a and 11e, the Seabed-shaped network has 5,537 nodes with five deep *valleys*, which are unlikely caused by boundary noise. Consequently, the extracted skeleton by our algorithm (see Fig. 11a) has five branches to reflect the details of the network, accurately capturing the main topological feature of the original network. While the skeleton by naive algorithm (see Fig. 11e) has many "parallel branches" because it identifies too many skeleton nodes due to the presence of parallel boundaries, and the refinement process has no guarantee to achieve a good skeleton.

Figs. 11b and 11f describes the extracted skeleton of a Man-shaped network by our algorithm and naive algorithm, respectively. Note that there are five *logical* parts (i.e., head, two arms and two feet). Clearly, our algorithm delivers a medially-placed line-like skeleton and reflects these topological features very well, outperforming naive algorithm. Note that in Fig. 11b, three skeleton branches occur in the head part because there the network is comparatively sparser, incurring many nodes have a feature loop that decomposes the boundary into connected components, and hence identify themselves as skeleton nodes. By setting an appropriate threshold for the importance measure, these skeleton nodes could be deleted, at the expense of other skeleton nodes at the end of arms and feet being disregards as skeleton nodes.

We can see the similar results in Figs. 11c and 11g and Figs. 11d and 11h. Figs. 11c and 11g illustrates the line-like skeleton of an H-shaped network by our algorithm and naive algorithm. Obviously, the extracted skeleton by algorithm is better which is well centered and homotopic to the underlying environment. In Figs. 11d and 11h, there is a long and narrow corridor, but we still achieve a line-like skeleton that captures the network topology very well, outperforming naive algorithm.

Fig. 12 depicts the comparison results of MAP, CASE and the proposed algorithm under 2D scenarios. We can clearly see that the skeletons generated by MAP (shown in Fig. 12a) have many spurious branches due to the existence of unstable skeleton nodes, even though the skeleton nodes with the geodesic distance (in hops) between the nearest boundary nodes less than 4 have already been regarded as unstable skeleton nodes and thus eliminated, implying that MAP is sensitive to the boundary noise; and CASE ignores some important skeleton branches, as shown in Fig. 12b, since the true skeleton nodes, whose nearest boundary nodes locate at the same boundary branch due to the improperly chosen threshold of the corner node, did not identify themselves as skeleton nodes. Our algorithm produces the best skeletons (shown in Fig. 12c) which accurately capture the salient topological features of the underlying networks, outperforming the other two schemes.

### 5.2.2 Robustness to Boundary Noise

To investigate the robustness of our algorithm to boundary noise, we add to the boundary nodes of the 3D network in Fig. 4 some random perturbations following a normal distribution with 0 mean and standard deviations in terms of the times of the communication radio range, as shown in

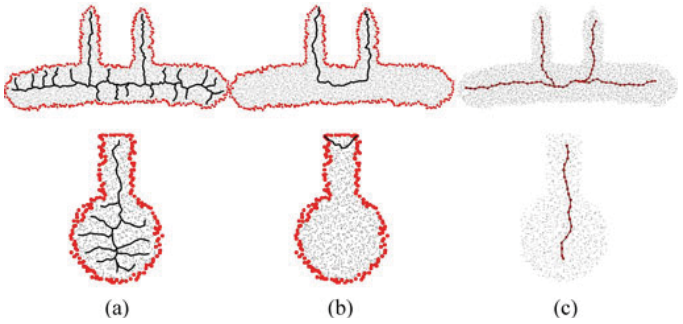


Fig. 12. Skeleton extraction under different 2D scenarios. Row 1: terminal-shaped network; row 2: bat-shaped network. (a) MAP [4]; (b) CASE [13]; (c) Our algorithm.

Fig. 13. We can clearly see that with the increasing scale of the boundary noise, the extracted skeletons exhibit more and more irregularities, but the major topological features of the original network are still accurately reflected by the extracted skeletons, implying that our algorithm is robust to boundary noise.

### 5.2.3 Performance under Different Node Densities

To evaluate the impact of node density on the performance of the proposed algorithm, we vary the communication radio range of the 3D network in Fig. 4 and a 2D network to generate four networks with different average node degrees, respectively. Fig. 14 describes the extracted skeletons of the corresponding networks. In Fig. 14a, the average node degree is only 14.53, which is very sparse for a 3D network. The extracted skeleton by our algorithm, however, captures very well the topological features of the underlying network, except that the skeleton here is rougher, due to the low node density. From Figs. 14b to 14d, we observe that, with the increasing of the node density, our algorithm extracts smoother skeletons with less skeleton nodes that accurately reflect the topological features. This happens because the network becomes *denser* and the hop count distance provides a better approximation to the euclidean distance. Accordingly, the extracted skeleton is more and more approximate to the real line-like skeleton. We see reasonable results for the 2D network in Figs. 14e to 14h, implying that our algorithm is insensitive to the node density and can always achieve stable results for both 2D and 3D sensor networks.

### 5.2.4 Line-Like Skeleton Based 3D Routing Protocol

To measure the quality of routing paths, we exploit two metrics, i.e., the CDF (cumulative density function) of the load of sensors (in terms of routes involved in the routing experiments), and the average stretch factor, which is defined as the average of the ratio of the routing path length to the shortest path length.

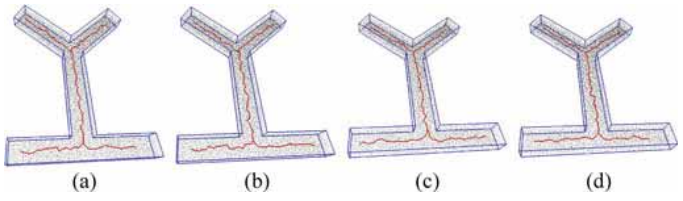


Fig. 13. The impact of boundary noise on the skeleton extraction of the 3D network in Fig. 4. The boundary noise are generated by adding to the boundary nodes random perturbations following a normal distribution with 0 mean and standard deviations in terms of the times of the communication radio range $r$. (a) 0.054$r$; (b) 0.072$r$; (c) 0.090$r$; (d) 0.108$r$.

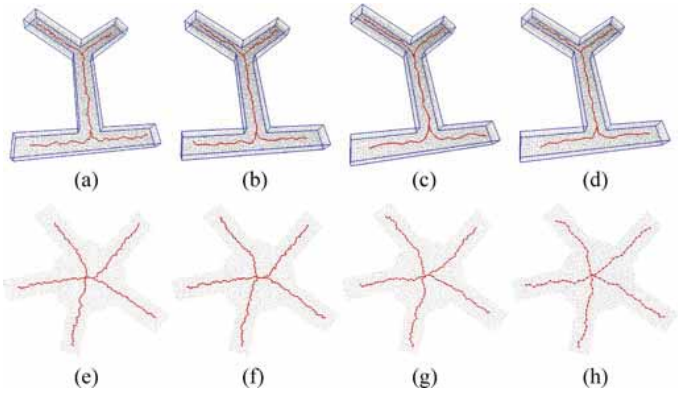


Fig. 14. The performance of the proposed algorithm under different node densities. Row 1: the 3D network in Fig. 4; row 2: a 2D network with 5,217 nodes. (a) avg. deg. 14.53; (b) avg. deg. 16.61; (c) avg. deg. 18.12; (d) avg. deg. 20.09; (e) avg. deg. 5.65; (f) avg. deg. 10.12; (g) avg. deg. 17.15; (h) avg. deg. 2.42.

Fig. 15a exhibits the CDFs of the load of sensors by our line-like skeleton based 3D routing protocol (the red curve), the surface skeleton based 3D routing protocol (the magenta curve), naive algorithm based routing protocol (the blue curve) and Random-Walk (the green curve). Clearly, the traffic load by our line-like skeleton based routing protocol is well balanced, with around 90 percent of sensors involved in less than 50 routes, and less than 5 percent of sensors involved in more than 100 routes, but all sensors are involved in less than 550 routes. The traffic load by surface skeleton based routing protocol is worse, with around 80 percent of sensors involved in less than 50 routes, and about 12 percent of sensors involved in more than 100 routes. It is worth noting that some sensors are involved in more than 600 routes. By looking at the log file, we find that this happens because many routes *flood* to the sensor nodes 1 hop away from the surface skeleton nodes, and consequently, these nodes are inevitably overloaded. The load by naive algorithm based routing protocol is comparable with surface skeleton based routing protocol since the former has many branches which incurs many packets routing along the contour set (called "rotation") , and the nodes near the junction of the skeleton are heavily loaded. As mentioned earlier, upon reaching local minima, Random-Walk searches along the boundaries to escape from the local minimum; obviously, the boundary nodes will be heavily loaded, which is confirmed by our experiments, where only

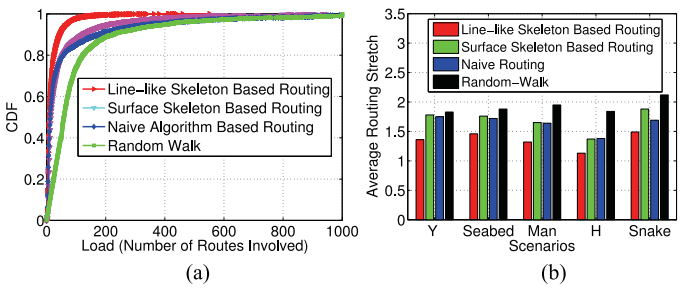


Fig. 15. (a) CDF of load of sensors; (b) average stretch factors.

40 percent around of sensors are involved in less than 50 routes, and more than 5 percent of sensors are in more than 200 routes. Overall, our line-like skeleton based routing protocol outperforms the other three schemes in terms of load balance.

We next move to the comparison of these routing schemes in term of the average stretch factors, shown in Fig. 15b. In the tested five scenarios, the average stretch factors of our line-like skeleton based routing protocol are all below 1.5. Note that the lowest one is only about 1.13 for H-shaped network in Fig. 11c, because in our line-like skeleton based routing protocol, the packet is forwarded parallel to the line-like skeleton and then make a rotation if necessary. The more complex the network is, the more rotation steps are conducted. The line-like skeleton of the H-shaped network in Fig. 11c is relatively simple, and thus low stretch factor is achieved. Similarly, the average stretch factor of snake-shape in Fig. 11d is the largest because more rotation step is conducted. The average stretch factor by surface-skeleton based routing is higher than our routing protocol, ranging from 1.37 to 1.88, as here the routing path has to be vertical to the normal of each two-manifold sheet. With the presence of redundant skeleton branches, the average stretch factors by naive algorithm based routing are all between those by line-like skeleton based routing and surface-skeleton based routing. As for Random-Walk, to escape from local minima on a local sphere structure will significantly increase the routing path length. The snake-shape in Fig. 11d is so complex that the average stretch factor of Random-Walk is over 2.0; and the other four average stretch factors by Random-Walk are all greater than 1.5. In summary, our line-like skeleton based routing algorithm has the lowest average stretch factor, and outperforms the other schemes.

## 6 Conclusion

We have proposed a unified framework for the line-like skeleton extraction in 2D/3D sensor networks. It is fully distributed, scalable, with reliance on mere connectivity information. On top of the line-like skeleton we have designed a routing protocol for 3D sensor networks which achieves balanced traffic load, guaranteed delivery, as well as low stretch factor. To the best of our knowledge, this is the first work on skeleton extraction in 3D sensor networks. Particularly, we conduct the first step of formalizing a unified and "good" skeleton for both 2D and 3D sensor networks.

In the future we will study its uses in applications, such as segmentation [14], [21], data collection [12] and storage [23], localization [24], and navigation [27], etc., which have attracted the attention of many researchers.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61073147, Grant 61173120, Grant 61202460 and Grant 61271226; and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). An earlier version of this work appeared as [20]. The corresponding author of this paper is Hongbo Jiang.

## References

[1] N. Amenta, M. Bern, and M. Kamvysselis, "A new voronoi-based surface reconstruction algorithm," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Techn.*, 1998, pp. 415–421.

[2] D. Attali, J.-D. Boissonnat, and A. Lieutier, "Complexity of the delaunay triangulation of points on surfaces: The smooth case," in *Proc. 19th Annu. Symp. Comput. Geom.*, 2003, pp. 201–210.

[3] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, "Skeleton extraction by mesh contraction," in *Proc. ACM SIGGRAPH*, 2008.

[4] J. Bruck, J. Gao, and A. A. Jiang, "MAP: Medial axis based geometric routing in sensor networks," in *Proc. 11th Annu. Int. Conf. Mobile Comput. Netw.*, 2005, pp. 88–102.

[5] J. Bruck, J. Gao, and A. A. Jiang, "MAP: Medial axis based geometric routing in sensor networks," *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 Conf. Comput. Commun.*, 2006, pp. 1–10.

[7] T. K. Dey, J. Giesen, and S. Goswami, "Shape segmentation and matching with flow discretization," in *Proc. Workshop Algorithms Data Struct.*, 2003, pp. 25–36.

[8] T. K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function," in *Proc. 4th Eurograph. Symp. Geom. Process.*, 2006, pp. 143–152.

[9] D. Dong, Y. Liu, and X. Liao, "Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods," in *Proc. 10th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2009, pp. 135–144.

[10] H. Federer, *Geometric Measure Theory (Classics in Mathematics)*. New York, NY, USA: Springer, 1996.

[11] R. Flury and R. Wattenhofer, "Randomized 3D geographic routing," in *Proc. IEEE 27th Conf. Comput. Commun.*, 2008.

[12] H. Jiang, S. Jin, and C. Wang, "Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 6, pp. 1064–1071, Jun. 2011.

[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 Trans. Parallel Distrib. Syst.*, vol. 21, no. 5, pp. 710–721, May 2010.

[14] H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang, "CONSEL: Connectivity-based segmentation in large-scale 2D/3D sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2012, pp. 2086–2094.

[15] H. Jiang, S. Zhang, G. Tan, and C. Wang, "CABET: Connectivity-based boundary extraction of large-scale 3d sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2011, pp. 784–792.

[16] S. Lederer, Y. Wang, and J. Gao, "Connectivity-based localization of large scale sensor networks with complex shape," in *Proc. IEEE 27th Conf. Comput. Commun.*, 2008.

[17] 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 Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1763–1772, Sep. 2013.

[18] 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. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 42–51.

[19] 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. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 52–61.

[20] 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 Int. Conf. Netw. Protocol*, 2013, pp. 1–10.

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

[22] 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. Vis. Comput. Graph.*, vol. 14, no. 2, pp. 355–368, Mar. 2008.

[23] 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.

[24] G. Tan, H. Jiang, S. Zhang, and A.-M. Kermarrec, "Connectivity-based and anchor-free localization in large-scale 2D/3D sensor networks," in *Proc. 11th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2010, pp. 191–200.
[25] 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.
[26] Y. Wang, J. Gao, and J. S. B. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proc. 12th Annu. Int. Conf. Mobile Comput. Netw.*, 2006.
[27] S. Xia, N. Ding, M. Jin, H. Wu, and Y. Yang, "Medial axis construction and applications in 3D wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2013, pp. 305–309.
[28] H. Zhou, Y. Wu, and M. Jin, "A robust boundary detection algorithm based on connectivity only for 3D wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun*, 2012, pp. 1602–1610.
[29] X. Zhu, R. Sarkar, and J. Gao, "Shape segmentation and applications in sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2007, pp. 1838–1846.
[30] X. Zhu, R. Sarkar, and J. Gao, "Segmenting a sensor field: Algorithms and applications in network design," *ACM Trans. Sens. Netw.*, vol. 5, no. 2, pp. 12:1–12:32, 2009.

**Wenping Liu** received the PhD degree from the Huazhong University of Science and Technology in 2012. He is currently an associate professor in the Hubei University of Economics, China. His research interests include topological recognition and its applications in sensor networks. He is a member of the IEEE.

**Hongbo Jiang** received the BS and MS degrees from the Huazhong University of Science and Technology, China and the PhD degree from Case Western Reserve University in 2008. After that he joined the faculty of Huazhong University of Science and Technology where he is currently a full professor. His research concerns computer networking, especially algorithms and architectures for high-performance networks and wireless networks. He is a senior member of the IEEE.

**Yang Yang** received the BE and ME degrees from the Wuhan University of Technology, China. He is currently working toward the PhD degree in the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China. His research interests include algorithms in wireless networks. He is a student member of the IEEE.

**Xiaofei Liao** received the PhD degree in computer science and engineering from the Huazhong University of Science and Technology (HUST), China, in 2005. He is currently a professor in the School of Computer Science and Engineering at HUST. He was a reviewer for many conferences and journal papers. His research interests are in the areas of system software, P2P system, cluster computing and streaming services. He is a member of the IEEE and the IEEE Computer society.

**Hongzhi Lin** received the PhD degree from the Huazhong University of Science and Technology in 2008. He is currently an assistant professor in Huazhong University of Science and Technology. His research interests include algorithms in wireless and sensor networks.

**Zemeng Jin** is currently working toward the undergraduate degree in the Huazhong University of Science and Technology, China. His research interests include algorithms in sensor networks.