Skeleton Extraction from Incomplete Boundaries in Sensor Networks based on Distance Transform

Wenping Liu[1,4] Hongbo Jiang[1] Xiang Bai[1] Guang Tan[2] Chonggang Wang[3] Wenyu Liu[1] Kechao Cai[1]

[1]Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[3]InterDigital Communications, PA, 19406, [4]Hubei University of Economics, China
[1]{wenpingliu2009,hongbojiang2004,xiang.bai,caikechao}@gmail.com, liuwy@mail.hust.edu.cn, [2]guang.tan@siat.ac.cn, [3]cgwang@ieee.org

***Abstract*—We study the problem of skeleton extraction for large-scale sensor networks using only connectivity information. Existing solutions for this problem heavily depend on an algorithm that can accurately detect network boundaries. This dependence may seriously affect the effectiveness of skeleton extraction. For example, in low density networks, boundary detection algorithms normally do not work well, potentially leading to an incorrect skeleton being generated.**

This paper proposes a novel approach, named DIST, to skeleton extraction from incomplete boundaries using the idea of *distance transform*, a concept in the computer graphics area. The main contribution is a distributed and low-cost algorithm that produces accurate network skeletons without requiring that the boundaries be complete or tight. The algorithm first establishes the network's distance transform – the hop distance of each node to the network's boundaries. Based on this, some *critical skeleton nodes* are identified. Next, a set of *skeleton arcs* are generated by controlled flooding; connecting these skeleton arcs then gives us a coarse skeleton. The algorithm finally refines the coarse skeleton by building shortest path trees, followed by a prune phase. The obtained skeletons are robust to boundary noise and shape variations.

***Index Terms*—Sensor networks, skeleton, distance transform, incomplete boundaries.**

I. Introduction

The distribution of the sensors and overall network topology are imperative to a variety of applications such as data routing, localization, and path planning, etc. Often the geographical locations and the node deployments may vary greatly, affected by factors such as obstacles, deployment randomness, and holes. One primitive representing the network topology is *skeleton*, also known as *medial axis*.

Skeleton extraction has been extensively studied in the computer vision [9] and computer graphics [28] areas for many years, where the skeleton is an important descriptor containing both the topological and geometrical properties of objects. In wireless sensor networks, the skeleton information of the sensor network can greatly improve the performance of routing [5], [6], location service [21], [30], segmentation [33], [20], [25], and navigation [7] algorithms. Some applications, such as network diagnosis [26], environment monitoring in coal mines [23], [24], etc., can also benefit from the skeleton information. Our approach to skeleton extraction for a sensor network is to 1) find the skeleton nodes and 2) connect them in a proper way. Despite extensive work on this subject, skeleton extraction continues to be a challenge when only connectivity information is available.

In the computer vision area, skeleton is defined by the grass-fire model [1] or the locus of the centers of maximal disk [2]. However, due to the discrete nature of wireless sensor networks, these definitions do not immediately lead to the identification of skeleton nodes. In a discrete network, the distance between nodes has to use hop distance instead of Euclidean distance. Often this results in noise, e.g., inaccurate estimation of boundaries and skeletons when the Euclidean distances between neighbors vary significantly. Besides, even if the skeleton nodes are known, connecting them to form a connected skeleton is still nontrivial, since the extracted skeleton nodes are usually insufficient to form a connected skeleton.

Prior Work: In the literature a number of techniques have been proposed to extract skeleton in sensor networks. Although existing connectivity-based skeleton extraction algorithms [5], [6], [18], [19] prove to be very effective, their main drawback is the heavy dependency on an accurate boundary detection algorithm, often with complete boundaries. As a result, this limits the algorithms' applicability in some situations.

One class of boundary detection algorithms [11], [32], for instance, aims to generate a set of continuous 1-manifold boundary, roughly formulated as closed polygonal chains. These algorithms' performance often suffers at a low node density or when very narrow passages exist in the network, as shown in Fig. 1(a)(b). Another class of boundary detection algorithms [12] is based on neighborhood size, using the observation that on average a boundary node has a fewer neighboring nodes than interior nodes. Again, in low density networks, these algorithms often perform poorly, so the generated boundaries are often incomplete, as shown in Fig. 1(d). Besides, for comparison, Fig. 1(c) shows the result of a naive boundary extraction by locally connecting nearby boundary nodes in Fig. 1(d).

Since existing boundary algorithms designed on complete boundaries are not suitable to facilitate extracting a well-connected skeleton graph at a low node density or when very narrow passages exist in the network, as shown in

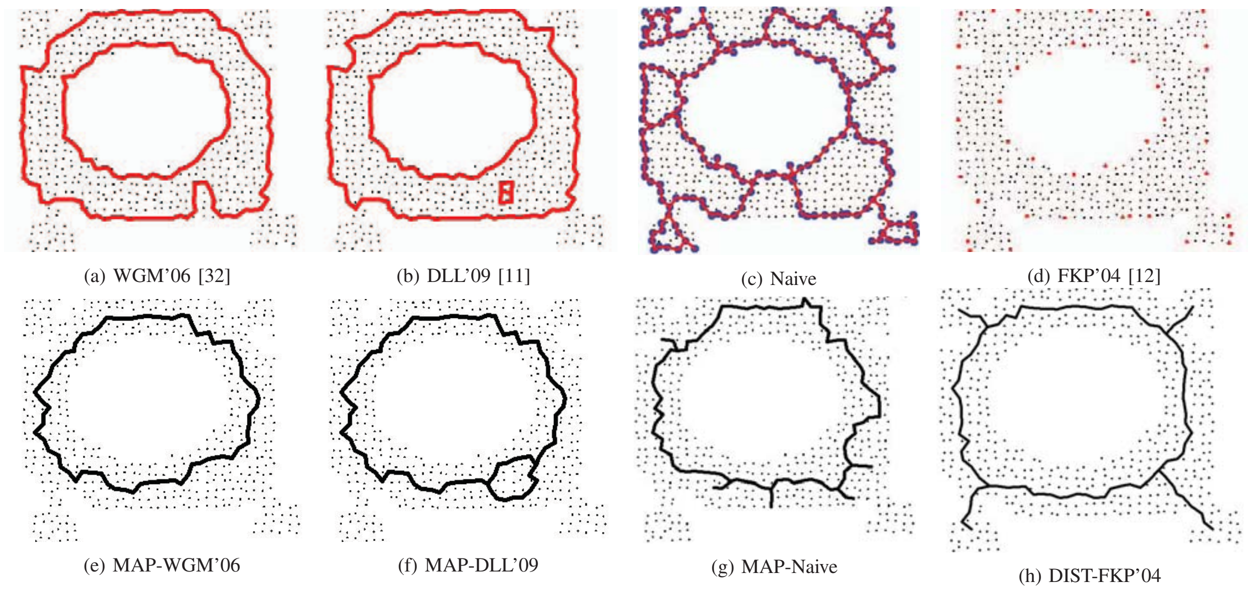
(a) WGM'06 [32] (b) DLL'09 [11] (c) Naive (d) FKP'04 [12]
(e) MAP-WGM'06 (f) MAP-DLL'09 (g) MAP-Naive (h) DIST-FKP'04

Fig. 1. Different ways of boundary recognition and their effects on skeleton extraction. The network has 577 nodes, with average node degree 5.95. Row 1: results of different boundary recognition methods. Row 2: skeleton extraction based on the identified boundaries. The red curves in (a)-(c) represent the boundaries and the boundary nodes in (d) are marked in red. The black curves in (e)-(h) are skeletons. (a) The boundaries identified by [32]; (b) The boundaries identified by [11]; (c) The complete boundaries by locally connecting nearby boundary nodes in (d); (d) The incomplete boundaries identified by [12]; (e) The skeleton extraction using MAP [5] based on the boundary result in (a); (f) The skeleton extraction using MAP [5] based on the boundary result in (b); (g) The skeleton extraction using MAP [5] based on the boundary result in (c); (h) The skeleton extraction using DIST based on the boundary result in (d).

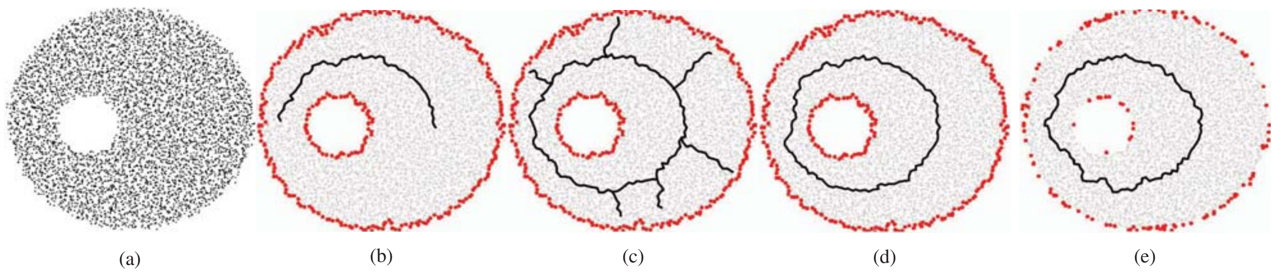
(a) (b) (c) (d) (e)

Fig. 2. Skeleton extraction for an eclipse-shaped network with 5,392 nodes. The boundary nodes are shown in red, and the dark curves represent the skeleton. The average node degree can be adjusted by changing nodal radio range, and is by default 11.9. (a) The original network; (b) The skeleton extracted by CASE [18]; (c) The skeleton extracted by MAP [5]; (d) The skeleton extracted by our algorithm; (e) The skeleton extracted by DIST when the average node degree is only 6.1 and only 50% of the boundary nodes are identified.

Fig. 1(e)(f)(g), in this paper, our approach relies only on incomplete boundaries, which are envisioned as a result of many trivial methods such as [12]. Research is surprisingly lacking for the case.

Our Approach: To address the challenge of incomplete boundaries, we propose a novel approach, named DIST, to skeleton extraction based on *distance transform*, a notion in the computer vision and graphics areas. Different from the existing approaches [18], [5] which extract skeletons directly from a sensor network, the proposed method computes skeletons based on the distance transform of a network. We are aware of a recent piece of work [33] close to ours which uses distance field to find skeleton nodes. However, it requires that all boundary nodes be detected to form boundary cycles. In addition, it is very sensitive to boundary noise: a small bump in a boundary may lead to many falsely identified skeleton nodes.

The contribution of this paper is DIST, a distributed and low cost algorithm for skeleton extraction based on distance transform. We assume that only part of the boundary nodes are known, respecting the fact that full and tight boundaries are very hard to obtain in some situations (e.g., in low density networks). We then compute the shortest path distance of each node to the boundaries, resulting in a distance transform. Accordingly, we identify skeleton nodes in a distributed manner: each node determines whether it is a skeleton node based on its hop distance to the boundaries. Next, the connected components will be formed by limited flooding within these skeleton nodes. In each component, the shortest path with the longest hop distance constructs a skeleton arc. Connecting the skeleton arcs, we obtain a coarse skeleton. Finally we refine the coarse skeleton by trimming short skeleton branches in

TABLE I
LIST OF NOTATIONS

Notation	Description
$\mathcal{D}$	The network $\mathcal{D}$.
$\partial\mathcal{D}$	The boundaries of the network $\mathcal{D}$.
$DT(\mathcal{D})$	The hop count transform of network $\mathcal{D}$.
$d_{\partial\mathcal{D}}p$	The hop count transform of node p.
$N_r(p)$	$r-$hop neighborhood of p, which is the set of nodes which are at most r hops from node p.
$d(p_1, p_2)$	The hop count distance between node p_1 and p_2.
$S(p, q)$	The slope of the line by connecting two nodes p and q.
$AHCT(q)$	The average hop count transform of node q.
$\mathcal{C}(i, j)$	The skeleton cut nodes formed by skeleton arc i and j.
$r(p)$	The root node of p.

the coarse skeleton. In Fig. 2, the result by our algorithm (see Fig. 2(d)) is better than that of MAP [5], [6] (Fig. 2(b)) and CASE [18], [19] (Fig. 2(c)). In particular, even when only 50% of boundary nodes are identified in a sparse network, our algorithm still performs well, as shown in Fig. 2(e).

The rest of the paper is organized as follows. Section II presents the background of distance transform and the motivation of DIST; Section III presents the skeleton extraction algorithm based on distance transform. We evaluate DIST in Section V and introduce related work in Section VI. Section IV discusses the complexity of DIST,and Section VII concludes the paper.

II. Preliminaries

Before introducing our algorithm, we introduce the basic definition of distance transform, which is a foundation of our work. In the computer vision/graphics community, distance transform [3], also known as *distance map* or *distance field* , describes the shortest distance of any given point inside an object to the boundaries of the object. Let $\mathcal{D}$ denote an object, and $\partial\mathcal{D}$ denote the boundaries of $\mathcal{D}$ (A list of notations can be found in Table I). We refer to $d(p, q)$ as the distance between two points p and q. The distance transform of D is defined as

$$DT(\mathcal{D}) = \begin{cases} \min\limits_{q\in\partial\mathcal{D}} d(p, q), p \in \mathcal{D} \\ \qquad 0, p \notin \mathcal{D}, \end{cases} \tag{1}$$

For any point $p \in \mathcal{D}$, $d_{\partial\mathcal{D}}(p) = \min_{q\in\partial\mathcal{D}} d(p, q)$ is referred to as the distance of point p to the boundaries of D. Any distance measure (such as Euclidean distance, Manhattan distance, etc.) can be used to construct the distance transform here. The distance measure is of crucial importance because it directly affects the “medialness” of the derived skeleton [14].

For sensor networks with mere connectivity information, the distance measure is the hop count between two nodes. We also call distance transform of a network as its *hop count transform*, or *hop count map*. Intuitively, the skeleton points are usually located at the *ridge* on the distance transform [14], [22], which has been widely exploited in computer vision community.

In previous work, a node is called a skeleton node if it has equal hop counts to at least two closest boundary nodes [5], [6] or boundary branches [18], [19], [33]. However, this definition is very sensitive to boundary noise: a small bump on the boundary will incur a long skeleton branch. For the case of incomplete boundaries, the result is even worse.

In this paper, for node p, we refer to $\mathcal{N}_r(p)$ as the set of nodes (not including p) which are at most r hops from p. We will show that a skeleton node can be identified locally by comparing its hop count transform with those of its neighbors.

According to Blum’s definition [2], a point p is a skeleton point if the disk centered at point p with radius $d(p)$, denoted by $D(p, d(p))$, is a maximal disk. Note that the centers of the maximal disks can be easily detected by comparing the distance transforms only in a neighborhood [27]. More specifically, if there is a neighbor of p, say point q, such that the disk centered at q with radius $d(q)$ contains $D(p, d(p))$, then $D(p, d(p))$ is not a maximal disk. Thus, to check whether the disk $D(p, d(p))$ is a maximal disk, one only needs to check whether there is a neighbor q, such that $D(q, d(q))$ contains $D(p, d(p))$. $D(q, d(q))$ contains $D(p, d(p))$ if and only if the following condition holds [4]:

$$d(q) \geq d(p) + d(p, q) \tag{2}$$

Note that the radius of the maximal disk of p is actually its distance transform. In fact, the skeleton defined by the maximal disks of D is the set of local maxima of the distance transform [31]. For discrete sensor networks, we can detect skeleton nodes based on hop count transform in a similar way. Formally, node p is a skeleton node if for any node $q \in \mathcal{N}_1(p)$, the following condition is true:

$$d_{\partial\mathcal{D}}(p) > d_{\partial\mathcal{D}}(q) \tag{3}$$

Due to the integer rounding error of hop count distance between nodes, there may be only a few skeleton nodes identified this way. To deal with this, we make a slight modification to Equation 3 and present our definition of skeleton node as follows.

Definition 1: Node p is a *skeleton node* if the hop count transform of p is a local maximum, namely, $d_{\partial\mathcal{D}}(p) \geq \max\{d_{\partial\mathcal{D}}(q)|q \in \mathcal{N}_1(p)\}$.

It is debatable as to what is the most appropriate analog of a continuous skeleton in a discrete sensor network [33]. Our definition of skeleton differs from that given in [18], [5] and we find that the skeleton nodes from this definition lie medially inside the network. That is, they are good approximations to the centers of the maximal disks in the continuous case. On the other hand, the skeleton node determining process based on Definition 1 is still affected by boundary noise. To alleviate such an undesirable effect, we introduce a parameter $r (r > 1)$, that determines how sensitive of the identification process is to the boundary noise, and define *critical skeleton node* as follows.

Definition 2: If $d_{\partial\mathcal{D}}(p) \geq \max\{d_{\partial\mathcal{D}}(q)|q \in \mathcal{N}_r(p)\}$, we call node p a *critical skeleton node*.

Obviously, if node p is a critical skeleton node, it must be a skeleton node; and the converse is not true. In other words, Definition 2 provides a sufficient but not necessary condition to determine a skeleton node in a simple way. In our simulations, $r = 3$ or 4 is found to be a good choice for practical purposes. An undesirable characteristic of the critical skeleton nodes is that they only account for part of the skeleton so in general they are disconnected. Fig. 3(a) gives an example. Theoretically, a skeleton is homotopic to the original shape and should have the same connectivity as the original shape [27]. To recover the complete skeleton, we additionally extract intermediate nodes to connect the critical skeleton nodes (detailed in Section III-D).

III. Skeleton Extraction Algorithm

A. An overview

To determine whether a node is a critical skeleton node, the minimum hop count distance of each node to the boundaries needs to be computed first. To that end, the first step is to discover the boundary nodes. For simplicity we adopt the neighborhood size based algorithm in [12] (though the obtained boundaries may be incomplete). Based on this, the establishment of hop distance transform of each node is straightforward. This helps us to obtain critical skeleton nodes. However, these nodes are in general disconnected. The remaining challenge then is to connect them in a right way.

We first present an outline of our skeleton extraction algorithm, followed by the details of each step.

1) Distance Transform Establishment. Each boundary node floods the network to build a shortest path tree rooted at itself, referred to as a *boundary tree*. Every sensor node is then associated with a single boundary tree whose root has a minimum hop distance to itself (among all boundary nodes). Each node without children marks itself as a leaf node.

2) Critical Skeleton Nodes Identification. Each leaf node p determines whether it is a critical skeleton node by comparing the hop count distance transforms of p and its r-hop neighbors.

3) Coarse Skeleton Establishment. The connected critical skeleton nodes are grouped into a component. All nodes in the same component are assigned a common identifier (e.g., the maximum node ID in the component) via scoped-flooding. Within each component, we connect two farthest skeleton nodes to obtain a skeleton arc, followed by connecting these skeleton arcs properly to generate a coarse skeleton. We note that the connecting process is not straightforward due to the requirement that a skeleton must lie medially inside the network.

4) Refinement. Since the coarse skeleton may contain unwanted skeleton branches with small lengths, the branches need to be removed. In the final phase, we prune those short branches; this gives us a refined skeleton.

B. Distance Transform Establishment

In this subsection, we focus on using boundary nodes to generate a hop count transform $DT(\mathcal{D})$ through local flooding. First, each boundary node p initiates a flooding to build a shortest path tree $T(p)$. The message contains p's ID and a counter that records the hop count the message has traveled. For each interior node q, upon receiving a message from p, if it has not received any message before, q will join $T(p)$, record the parent node who forwarded this message, increase the counter by one and store the counter, and finally forward this message to its neighboring nodes; otherwise, q simply discards this message. By doing so, each node that has the minimum hop count distance to p is associated with $T(p)$. We call $T(p)$ as a boundary tree. This process is conducted repeatedly until every node belongs to a boundary tree.

If the boundary nodes perform their flooding approximately simultaneously, and the flooded message travels at approximately the same speed, each interior node will forward only one message. This will substantially reduce the total number of delivered messages and keep the communication overhead very low (only $O(1)$ per-node message cost).

After this process, each interior node has the knowledge of its nearest boundary node, and the minimum hop count distance (i.e., the counter stored at each node) to the boundaries. That is, a hop count distance transform is established.

Lemma 1: For an interior node q, the hop count transform of q is larger than that of its parent node $P(q)$. That is, $d_{\partial\mathcal{D}}(q) > d_{\partial\mathcal{D}}(P(q))$.

Corollary 1: For an interior node q, if q is not a critical skeleton node, then q's parent node will not be a critical skeleton node.

Proof: Let $P(q)$ be the parent node of q. According to Lemma 1, we have $d_{\partial\mathcal{D}}(q) > d_{\partial\mathcal{D}}(P(q))$. If node q is not a critical skeleton node, then there exists at least one node $s \in \mathcal{N}_r(q)$ such that $d_{\partial\mathcal{D}}(s) > d_{\partial\mathcal{D}}(q)$. Obviously, we have $d_{\partial\mathcal{D}}(s) > d_{\partial\mathcal{D}}(P(q))$. That is to say, $P(q)$ is not a critical skeleton node. ■

Theorem 1: If q is a critical skeleton node, then q must be a leaf node.

Proof: We prove this by contradiction. Since q is a critical skeleton node, q has the largest hop count transform among those of its r-hop neighbors. Namely, $d_{\partial\mathcal{D}}(q) \geq \max\{d_{\partial\mathcal{D}}(s) | s \in \mathcal{N}_r(q)\}$. If q is not a leaf node, then q must have at least one child node. Let node s be a child node of q. According to Lemma 1, s has a larger hop count transform as compared to q, that is, the hop count transform of q is not the largest among those of its r-hop neighbors. That is a contradiction. ■

Theorem 1 provides a necessary but not sufficient condition for critical skeleton node identification. In other words, a critical skeleton node must be a leaf node but not vice versa. As such, one only needs to check whether a leaf node is a critical skeleton node. This can narrow down the scope of checking, thus keeping the total communication overhead low.

C. Critical Skeleton Nodes Identification

In this phase, each leaf node p determines whether itself is a critical skeleton node, by comparing the hop distances of p and its r-hop neighbors $N_r(p)$. The skeleton node identification process works as follows. Each leaf node p of a boundary tree first floods its r-hop neighborhood $N_r(p)$. The flooded message includes p's ID, its hop count transform $d(p)$, and a counter that indicates how many hops the message still needs to travel. The counter is initialized to r. When an intermediate node q receives the flooded message, q checks whether the

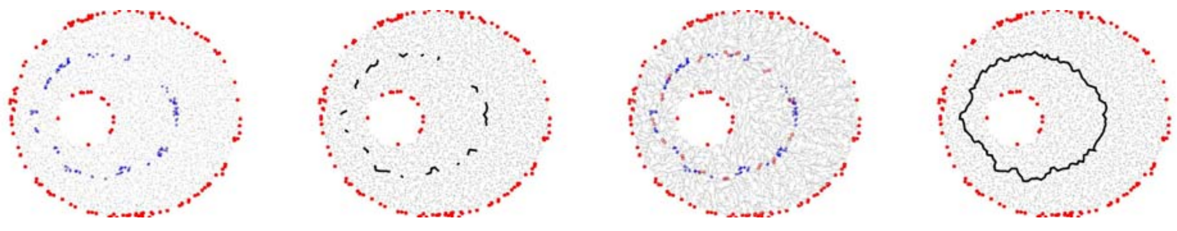
(a) (b) (c) (d)

Fig. 3. Skeleton extraction of the eclipse-shaped network in Fig. 2. (a) Critical skeleton nodes; (b) Skeleton arcs; (c) Skeleton trees and cut pairs. Skeleton nodes are marked in blue and cut pairs are marked in red; (d) Coarse skeleton. The final refined result is shown in Fig. 2 (e).

counter included in the message is larger than 0. If not, q does not forward this message; otherwise, q compares $d_{\partial\mathcal{D}}(q)$ and $d_{\partial\mathcal{D}}(p)$. Only when $d_{\partial\mathcal{D}}(p) > d_{\partial\mathcal{D}}(q)$ and the counter is larger than 0 will q decrease the counter by one and forward this message to its neighbors. For this case, we say p is *reachable* to q. Note that the message will be suppressed by q if $d_{\partial\mathcal{D}}(p) \leq d_{\partial\mathcal{D}}(q)$; and for this case, we call p is *unreachable* to q.

Lemma 2: For two leaf nodes p_1, p_2 that have a distance of $k(\leq r)$ hops, if p_1 is unreachable to p_2, then $d_{\partial\mathcal{D}}(p_1) \leq d_{\partial\mathcal{D}}(p_2)$; otherwise, $d_{\partial\mathcal{D}}(p_1) > d_{\partial\mathcal{D}}(p_2)$.

Theorem 2: A leaf node p is a critical skeleton node if and only if there is no leaf node $s \in \mathcal{N}_r(p)$ reachable to node p.

Proof: we first prove the necessity. If p is a critical skeleton node, then $d_{\partial\mathcal{D}}(p) \geq \max\{d_{\partial\mathcal{D}}(q) | q \in \mathcal{N}_r(p)\}$. Assume that there is a node $s \in \mathcal{N}_r(p)$ reachable to p. This implies that $d_{\partial\mathcal{D}}(s) > d_{\partial\mathcal{D}}(p)$, according to lemma 2. This is a contradiction.

We next prove the sufficiency. If there is no node $s \in \mathcal{N}_r(p)$ reachable to p, that is, for each node $s \in \mathcal{N}_r(p)$, $d_{\partial\mathcal{D}}(s) \leq d_{\partial\mathcal{D}}(p)$. Therefore, $d_{\partial\mathcal{D}}(p) \geq \max\{d_{\partial\mathcal{D}}(q) | q \in \mathcal{N}_r(p)\}$. That is to say, p is a critical skeleton node. ■

An implication of Theorem 2 is that to determine whether a leaf node p is a critical skeleton node, it only needs to check whether there is a node within its r-hop neighbors that is reachable to p. More specifically, in the above process, if a leaf node q receives a flooded message from another leaf node, q is not a critical skeleton node; otherwise, q marks itself as a critical skeleton node. Fig. 3(a) shows the result of this step.

Theorem 3: For two leaf nodes p_1, p_2, if $d(p_1, p_2) = k$ and $d_{\partial\mathcal{D}}(p_1) = d_{\partial\mathcal{D}}(p_2) + k$, then p_2 is unreachable to p_1; and p_1 is reachable to p_2.

Proof: When $k = 1$, $d_{\partial\mathcal{D}}(p_1) = d_{\partial\mathcal{D}}(p_2) + 1$, then the message from p_2 will not be forwarded by p_1. However, the message from p_1 can be forwarded by p_2.

We next prove the case for $k = 2$, where $d_{\partial\mathcal{D}}(p_1) = d_{\partial\mathcal{D}}(p_2) + 2$ and $d(p_1, p_2) = 2$. Then there must exists one node s on a shortest path from p_1 to p_2 who satisfies that $d(p_1, s) = 1, d(s, p_2) = 1$ and $d_{\partial\mathcal{D}}(p_1) > d_{\partial\mathcal{D}}(s) > d_{\partial\mathcal{D}}(p_2)$. When the message from p_2 comes to s, s will not forward the message to its neighbor (including p_1), and thus p_2 is unreachable to p_1. However, when s receives a message from p_1, it will forward this message to its neighbor p_2 since $d_{\partial\mathcal{D}}(p_1) > d_{\partial\mathcal{D}}(s)$, that is, p_1 is reachable to p_2.

We finally prove the case for $k > 2$. since $d(p_1, p_2) = k$, there is one node s such that $d(p_1, s) = k - 2, d(s, p_2) = 2$ and $d_{\partial\mathcal{D}}(p_1) = d_{\partial\mathcal{D}}(s) + k - 2, d_{\partial\mathcal{D}}(s) = d_{\partial\mathcal{D}}(p_2) + 2$. Therefore, the message from p_2 can not be forwarded to s; and consequently, p_2 is unreachable to p_1. If $k - 2 = 2$, p_1 is reachable to s and therefore to p_2 (See the proof above for the case $k = 2$). If $k - 2 > 2$, we can repeat this procedure and it is easy to show that p_1 is reachable to p_2. ■

This theorem guarantees that our algorithm has low communication overhead in the phase of skeleton node identification. This is because for two leaf nodes p_1, p_2, there is at most one leaf node, say, p_1 is reachable to the other leaf node p_2, and the message from p_2 is suppressed.

D. Coarse Skeleton Establishment

So far a set of critical skeleton nodes have been identified. Note that two fundamental properties of a skeleton are: 1) it is medially placed (hence maintains the "medialness"); and 2) it has the simple connectivity as the original shape [27]. That is, by grouping nearby skeleton nodes, a skeleton graph can be considered as a simplified topology of the network [6]. In the continuous domain, the centers of maximal disks are medial and connected. In a discrete network, however, the critical skeleton nodes are generally disconnected. In this section, we identify the intermediate nodes, referred to as *connecting skeleton nodes*, which are used to link two adjacent skeleton arcs. This way a coarse skeleton will be generated.

We first construct a set of connected skeleton components of the critical skeleton nodes, and generate a set of skeleton arcs. This can be done as follows. Each critical skeleton node initiates a controlled flooding with a message containing its node ID and the hop count the message has traveled. When a node p receives a flooded message from a critical skeleton node, say q, there are two cases: 1) if p is a critical skeleton node and q has a larger ID than p, p forwards the message to its neighbors; 2) otherwise, it simply discards the message. By doing so, a set of connected critical skeleton components are formed and the shortest path with the largest length for each component naturally forms a skeleton arc (see Fig.3(b)); moreover each critical skeleton node is assigned a unique ID (e.g.,the maximum node ID in the component).

With these skeleton arcs formed, we next detect intermediate nodes, based on the *slope function* of the distance map, to connect skeleton arcs. In the continuous domain, skeleton arcs follow lines of steepest slope of the Euclidean distance map [29], where the slope of the line xy, $S(x,y)$, is defined as

$$S(x,y) = \frac{d(y) - d(x)}{d(x,y)} \tag{4}$$

where $d(x)$ and $d(y)$ are the Euclidean distance transforms of points x and y, respectively, and $d(x,y)$ is the Euclidean distance between points x and y. When a point x is detected as a skeleton point, the neighbor of x that has a steepest ascending slope is taken as a new skeleton point [10]. The steepest ascent approach can guarantee that the skeleton branches locate medially [14].

In a discrete network, we define the slope, $S(p,q)$, of the line by connecting two nodes p, q as follows

$$S(p,q) = \frac{d_{\partial\mathcal{D}}(q) - d_{\partial\mathcal{D}}(p)}{d(p,q)} \tag{5}$$

If nodes p, q are two neighboring nodes, Equation (5) can be simplified as

$$S(p,q) = d_{\partial\mathcal{D}}(q) - d_{\partial\mathcal{D}}(p) \tag{6}$$

Lemma 3: Let $q\prime$ be a neighbor of a critical skeleton node q. If $S(q\prime, q) \geq max_{s \in N_1(q)} S(s,q)$, then $q\prime$ locates approximately medially, and we call $q\prime$ a connecting skeleton node.

Lemma 4: Let q_1, q_2 be two nodes that have the same hop distance to a critical skeleton node p, namely, $d(p,q_1) = d(p,q_2)$. If $d_{\partial\mathcal{D}}(q_1) \geq d_{\partial\mathcal{D}}(q_2)$, then $S(p,q_1) \geq S(p,q_2)$.

Proof: From Equation (5), we have $S(p,q_1) = \frac{d_{\partial\mathcal{D}}(q_1) - d_{\partial\mathcal{D}}(p)}{d(p,q_1)}, S(p,q_2) = \frac{d_{\partial\mathcal{D}}(q_2) - d_{\partial\mathcal{D}}(p)}{d(p,q_2)}$. Since $d(p,q_1) = d(p,q_2)$ and $d_{\partial\mathcal{D}}(q_1) \geq d_{\partial\mathcal{D}}(q_2)$, $S(p,q_1) \geq S(p,q_2)$ holds. ■

According to Lemma 3 and Lemma 4, we now propose to identify connecting skeleton nodes.

First, we have all critical skeleton nodes synchronize among themselves [13] and start to flood the network at approximately the same time. These skeleton nodes perform a joint flooding with the messages in the form $(ID_i, d_{\partial D}ID_i)$ where ID_i is the ID of the ith transmitting node and $d_{\partial D}ID_i$ is its hop count transform. When a node q receives a flooded message, which is initiated by a critical skeleton node p, and forwarded by $q\prime$, if q has not received a message before, q will join the tree rooted at p, keep record of the parent node $q\prime$ and its hop count transform, append $(q, d_{\partial D}q)$ to the message and forward it to all neighbors, and compute the average hop count transform of q, denoted by $AHCT(q)$, which is the average of the hop count transforms of transmitting nodes (including q); otherwise, q compares the hop count transform of $q\prime$ with that of q's parent node $P(q)$. If $d_{\partial D}q\prime > d_{\partial D}P(q)$, q changes its parent node to $q\prime$, and updates its average hop count transform; otherwise, q just discards the message. This way, a tree rooted at critical skeleton node p, denoted by $T_s(p)$, is constructed in a greedy manner. We call such tree a *skeleton tree*. For each node q, we denote by $r(q)$ its root. Note that two skeleton trees whose root nodes belong to different skeleton arcs may meet. This means that the corresponding two skeleton arcs are adjacent and can be connected. We thus define *cut* nodes, $\mathcal{C}(i,j)$, as the set of nodes such that each of them is rooted at one skeleton tree (skeleton arcs i) and has a neighbor node rooted at another skeleton tree (skeleton arcs j).

With these skeleton trees constructed, we detect a *cut-pair*, based on which we can connect two adjacent skeleton arcs. The definition of a *cut-pair* is as follows.

Definition 3: A *cut-pair* (q_1, q_2) are two nodes such that:
1) q_1 and q_2 are neighboring cut nodes;
2) $r(q_1)$ and $r(q_2)$ belong to different skeleton arcs;
3) q_1 and q_2 have a largest *AHCT* among all cut nodes associated with these two skeleton arcs.

Further, if (q_1, q_2) is a cut-pair, we call q_1 (or q_2) a cut-pair node. All skeleton cut-pairs (and cut-pair nodes) can be detected, see Fig 3(c) for skeleton trees and cut-pairs. The shortest paths from each cut-pair to their roots will form a *connecting path*, which connects two adjacent skeleton arcs and forms one longer skeleton arc. The following theorem shows that the nodes on a connecting path are connecting skeleton nodes, that is, these nodes lie medially.

Theorem 4: Let $(p_1, p_2) \in \mathcal{C}(i,j)$ be a cut-pair and q a node on the connecting path from p_1 to $r(p_1)$, and $d(q, r(p_1)) = k$. For each node s, whose root node is also $r(p_1)$ and $d(s, r(p_1)) = k$, on the path from a cut node in $\mathcal{C}(i,j)$ to $r(p_1)$, we have:
1) $AHCT(q) \geq AHCT(s)$;
2) $S(q, r(p_1)) \geq S(s, r(p_1))$;
3) q is a connecting skeleton node.

Proof: 1) Assume that s is on the path from a cut node $p\prime \in \mathcal{C}(i,j)$ to $r(p_1)$. Obviously, we have $AHCT(p_1) \geq AHCT(p\prime)$. We prove by mathematical induction that the for any such node s, $AHCT(q) \geq AHCT(s)$.

Initial step. We first verify that when $k = 1$, $AHCT(q) \geq AHCT(s)$ holds. Since the process of skeleton tree construction is done in a greedy manner, and thus $d_{\partial\mathcal{D}}(q) \geq d_{\partial\mathcal{D}}(s), AHCT(q) = d_{\partial\mathcal{D}}(q), AHCT(s) = d_{\partial\mathcal{D}}(s)$. As a result, we have $AHCT(q) \geq AHCT(s)$.

Inductive step. We next verify that when $k = l(> 1)$, if $AHCT(q) \geq AHCT(s)$, then, for $k = l + 1$, $AHCT(q) \geq AHCT(s)$ still holds. Let $P(q), P(s)$ denote the parent nodes of q and s respectively, then when $k = l+1$, we have $d(P(q), r(p_1)) = d(P(s), r(p_1)) = l$. Therefore, $AHCT(P(q)) \geq AHCT(P(s))$. Again, since the message is greedily forwarded to construct a skeleton tree, and $P(q)$ is on the path from p_1 to $r(p_1)$, we have $d_{\partial\mathcal{D}}(P(q)) \geq d_{\partial\mathcal{D}}(P(s))$ and $d_{\partial\mathcal{D}}(q) \geq d_{\partial\mathcal{D}}(s)$. Since $AHCT(q) = \frac{AHCT(P(q)) \times l + d_{\partial\mathcal{D}}(q)}{l+1}, AHCT(s) = \frac{AHCT(P(s)) \times l + d_{\partial\mathcal{D}}(s)}{l+1}$, we have $AHCT(q) \geq AHCT(s)$.

2) We next prove that $S(q, r(p_1)) \geq S(s, r(p_1))$. During the proof process above, we have already seen that $d_{\partial\mathcal{D}}(q) \geq d_{\partial\mathcal{D}}(s)$. According to Lemma 4, we have $S(q, r(p_1)) \geq S(s, r(p_1))$.

3) Let $q_1, q_2, \ldots, q_{k-1}$ be the $k-1$ nodes on the path from q to $r(p_1)$, and $s_1, s_2, \ldots, s_{k-1}$ be the $k-1$ nodes on the path from s to $r(p_1)$, where $d(q_i, r(p_1)) = i, d(s_i, r(p_1)) = i (i = 1, 2, \ldots, k-1)$. We have known that $d_{\partial\mathcal{D}}(q_1) \geq d_{\partial\mathcal{D}}(s_1)$ and $S(q_1, r(p_1)) \geq S(s_1, r(p_1))$. Therefore, according to

Lemma 3, q_1 is a connecting skeleton node since q_1 has a largest slope as compared to other neighbors of $r(p_1)$. In a similar way, we can prove that $q_2, q_3, \ldots, q_{k-1}$ and q are connecting skeleton nodes. ■

Based on Theorem 4, each cut node can decide whether it is a cut-pair node according to its average hop count transform; and since each node q on the path from a cut-pair node to the closest skeleton arc has a largest slope, q considers itself as a connecting skeleton node, and informs the transmitting nodes from q to $r(q)$ of their identities as connecting skeleton nodes. These connecting skeleton nodes, together with critical skeleton nodes, form a connected component and thus by connecting themselves, a coarse skeleton is generated, as shown in Fig. 3 (d).

We have so far detected two kinds of skeleton nodes: critical skeleton nodes and connecting skeleton nodes. In the subsequent sections, we call both of them skeleton nodes for short.

E. Coarse Skeleton Refinement

The path between two skeleton nodes on a coarse skeleton may not be the shortest, and there may exist unwanted skeleton branches that contain a few skeleton nodes. Thus we need to refine the coarse skeleton. Specifically, each skeleton node p sets a timer with a random remaining time. When the remaining time reaches 0, p begins to flood within the coarse skeleton and builds a shortest path tree. The message includes the timer and p's node ID. When a skeleton node q receives the message from p, it will compare its timer with that of p; the node with a smaller timer will dominate. Then, a shortest path tree with skeleton branches will be formed. Next we trim this tree based on the length of skeleton branch (that is, the number of skeleton nodes on the skeleton branch). If the length of a branch is less than a certain value (e.g., r), the skeleton branch will be trimmed. Finally, we obtain the refined skeleton; see an example in Fig. 2 (d).

IV. Discussions

A. Complexity analysis

Theorem 5: Our algorithm has a time complexity of $O(\sqrt{N})$ and message complexity of $O(N)$, where N is the number of sensor nodes.

Proof: We first prove the time complexity of our algorithm is $O(\sqrt{N})$. First, to obtain the distance map, each boundary node floods in the network in a distributed way. This process has a time complexity of $O(1)$ in the best case and $O(\sqrt{N})$ in the worst case where some nodes are far away from the boundaries. Second, each node identifies whether it is a skeleton node and this time complexity is $O(1)$. Third, each skeleton node floods within a limited scope to generate skeleton components and skeleton arcs. The time complexity is $O(E_{max})$, where E_{max} is the maximum number of skeleton nodes on skeleton arcs, $O(\sqrt{N})$ at most. Forth, each skeleton component floods in the network to form a skeleton tree, the time complexity of which is usually $O(1)$ and $O(\sqrt{N})$ at most; and connecting two adjacent skeleton arcs to form a coarse skeleton via skeleton cut-pairs takes $O(\sqrt{N})$ time complexity at most. Finally, the refinement of coarse skeleton has a time complexity of at most $O(\sqrt{N})$. In total, the time complexity is $O(\sqrt{N})$.

We next prove the message complexity of our algorithm is $O(N)$. First, the construction of distance map has a message complexity of $O(N)$. This is because boundary nodes perform their flooding at approximately the same time and the flooded message travels at approximately the same speed. Consequently, each interior node forwards only one message. Second, the message complexity of critical skeleton node identification is at most $O(N)$. Let n_{leaf} be the number of leaf nodes of the boundary trees constructed in the phase of critical skeleton node identification. Generally, n_{leaf} is far less than the total number of sensors N. On the other hand, the message from a leaf node with smaller hop count transform will be suppressed, resulting in that on average, each r-hop neighbor of a leaf node will forward the flooding message in the critical skeleton node identification process only once. Third, the generation of skeleton components and skeleton arcs has a message complexity of $O(n_{csn})$, where n_{csn} denotes the number of critical skeleton nodes which is less than N. Forth, to obtain the coarse skeleton, each node will receive one message from a node on skeleton arcs and thus the message complexity is $O(N)$. Finally, the refinement of coarse skeleton has a message complexity of $O(n_{sn})$ where $n_{sn}(< N)$ is the number of skeleton nodes on the coarse skeleton. Overall, the message complexity is $O(N)$. ■

B. The impact of boundary incompleteness on skeleton

The proposed algorithm is robust to boundary noise and boundary incompleteness. Nevertheless, to achieve a good approximation of the true skeleton, the distribution of identified boundary nodes should ideally be even along the boundaries. A big *gap* in the boundaries (resulting from the boundary incompleteness) may incur an unwanted skeleton branch. Besides, the boundary incompleteness also affects the choice of parameter r. When the number of identified boundary nodes is large, a small value of r (e.g., $r = 3$) is enough for producing a good skeleton. However, for the case of fewer boundary nodes, we should choose a relatively larger r (e.g.,$r = 5$) to avoid the bend toward the gap.

V. Performance Evaluation

To evaluate the effectiveness of DIST, we have conducted extensive simulations on various scenarios, comparing with two existing solutions, namely, CASE [18], [19], as well as MAP [5], [6]. In addition, we study how robust the DIST is to node density and boundary incompleteness.

A. Simulation Setup

In the simulation, nodes are randomly deployed with a common communication radio range. We use the neighborhood-based algorithm in [12] to determine boundary nodes. The default parameter for critical skeleton node identification is $r = 3$.

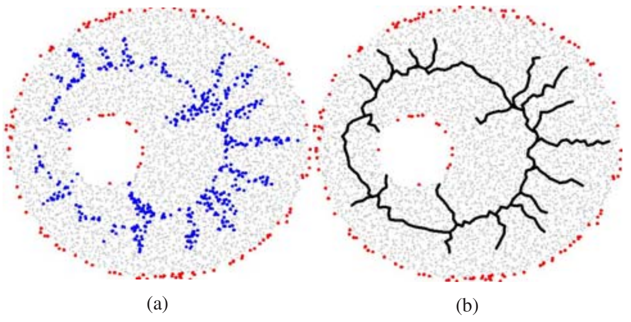
Fig. 4. Skeleton extracted by MAP on the Eclipsed-shaped network in Fig. 2. The average degree is 6.3 and only half of the boundary nodes are detected (marked in red). (a) Skeleton nodes (in blue) equidistant to at least two boundary nodes. In MAP, those unstable medial nodes, whose closest boundary nodes within a small distance (here we set the threshold to be 8 hops), are disregarded; (b) The dark curve represents the skeleton by MAP.

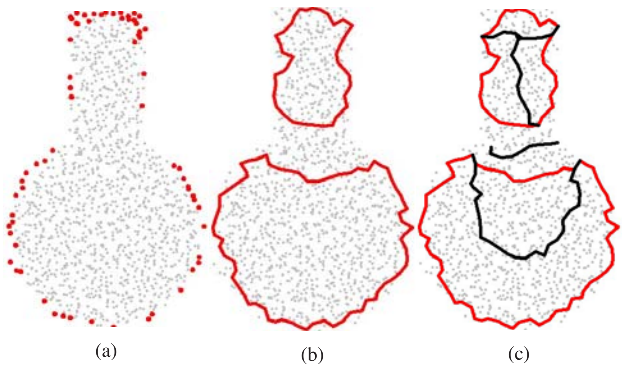
Fig. 5. Boundary extension for a bat-shaped network. The average degree is 6.2 and only 40% of the boundary nodes are detected. (a) The boundary nodes (in red); (b) The dark red curves represent the boundaries by naive extension. (c) Skeleton extracted by CASE.

When evaluating MAP and CASE, we often let them run on complete boundaries, which are a "friendly" case for them. We show even under this case, they often produce unsatisfactory or even wrong results. Fig. 4 and Fig. 5 show the results in the case of incomplete boundaries. High dependence on accurate boundary of these two solutions potentially limit their applicability in practice. As for DIST, we are instead more interested in the more challenging case where the network is sparse and the algorithm is only provided with incomplete boundaries. When the boundaries are complete our algorithm only performs better. It is noted that a node identifies itself if its k-hop neighborhood is less than the given threshold value. One advantage of this algorithm is that it has low message overhead.

We first compare our algorithm with MAP [5], [6] and CASE [18], [19] on three network topologies, namely, bat-shaped network (Fig. 6), Terminal 2 of Paris-Charles De Gaulle airport (Fig. 7) and one-hole network (Fig. 8).

Besides, we vary the average node degree to examine the impact of node density on an S-shaped network (Fig. 9). It is a typical indoor environment without holes. The average node degree varies from 6.54 to 21.91.

B. Simulation Results

Fig. 6 shows the skeletons extracted by MAP (See Fig. 6(b)), CASE (See Fig. 6(c)) and our algorithm (See Fig. 6(d)) on the bat-shaped network. The skeleton extracted by MAP has many unwanted long branches, as shown in Fig.6(b). This is because MAP is sensitive to boundary noise. Specifically, many nodes that have two closest non-adjacent boundary nodes are undesirably identified as skeleton nodes. As for Fig. 6(c), two corner points are identified and the boundary is decomposed into two boundary branches. We can see that the skeleton extracted by CASE is very incomplete – many nodes located medially are missed because their two closest boundary nodes reside on the same boundary branch. Fig.6 (d) is the skeleton extracted by our algorithm when only 40% of the boundary nodes are known. The skeleton by our algorithm is better than those of MAP and CASE.

We next examine the case of the airport terminal network. We see similar results (Fig. 7) as in the bat-shaped network. The skeleton extracted by MAP has many skeleton branches due to boundary noise, as shown in Fig. 7(b). For CASE, due to the improperly chosen parameters, some skeleton nodes are not identified because their two closest boundary nodes are on the same boundary branch. Consequently, the skeleton is only partially extracted by CASE. Fig. 7(d) shows that the our algorithm is able to generate a good skeleton graph even in the case of incomplete boundaries.

We next examine the performance of algorithms on a one-hole network. Different from Fig. 6 and Fig. 7, the network shown in Fig. 8 has a concave hole inside the network. The skeleton extracted by MAP in Fig. 8(b) is satisfactory except that the skeleton has some skeleton branches. In Fig. 8(c), four corner points on the outer boundary are identified and the skeleton nodes are detected accordingly. We can see that the shortest path with the largest length makes the skeleton deviate a little from the "medial" location. The skeleton by our algorithm is comparable to that of MAP, even with only 80% of the boundary nodes known.

Fig. 9 shows the impact of various node densities on our algorithm on the S-shaped network. In Fig. 9 (a), the average degree is only 6.54. In a network with such low density, often only a few (e.g., 20%) of the boundary nodes are identified. One plausible solution to find more boundary nodes is to increase the threshold value. However, that inevitably makes many interior nodes falsely identify themselves as boundary nodes. We can see that there is a big gap in the boundary near the two concave points of the lower-part of the network. Consequently, several nodes on the boundary are incorrectly identified as skeleton nodes; and the final skeleton bends toward (even tangent with) the boundary. Fortunately, the obtained skeleton still captures the main geometric features of the network. In Fig. 9 (b)-(d), the node density is increasing. As a result, more and more boundary nodes are identified, leading to more and more accurate skeleton extraction results.

Overall, MAP suffers from boundary noise and generates many unwanted skeleton branches. CASE is sensitive to the

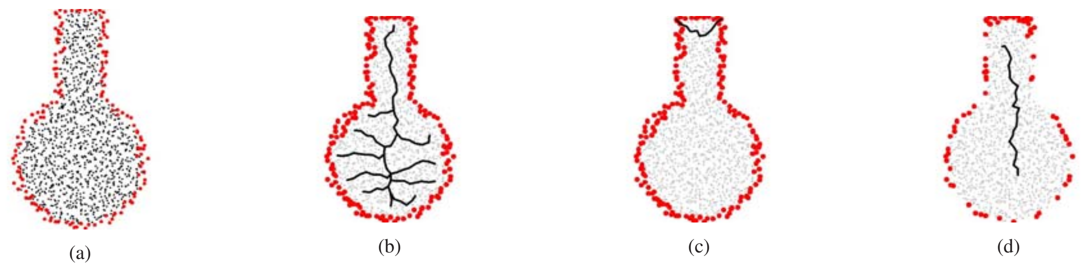
Fig. 6. Skeleton of a bat-shaped network with 1272 nodes and average degree 13.30. (a) Original network; (b) Skeleton extracted by MAP; (c) Skeleton extracted by CASE; (d) Skeleton extracted by DIST using only 40% of the boundary nodes.

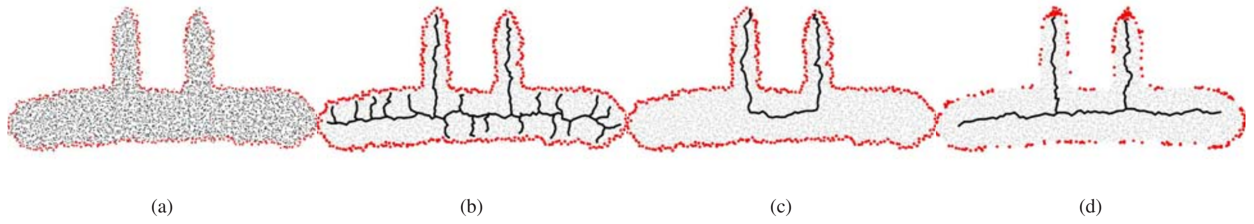
Fig. 7. Skeleton of terminal-shaped network with 5012 nodes, average degree 13.59. (a) Original network; (b) Skeleton extracted by MAP; (c) Skeleton extracted by CASE; (d) Skeleton extracted by DIST using 50% of the boundary nodes.

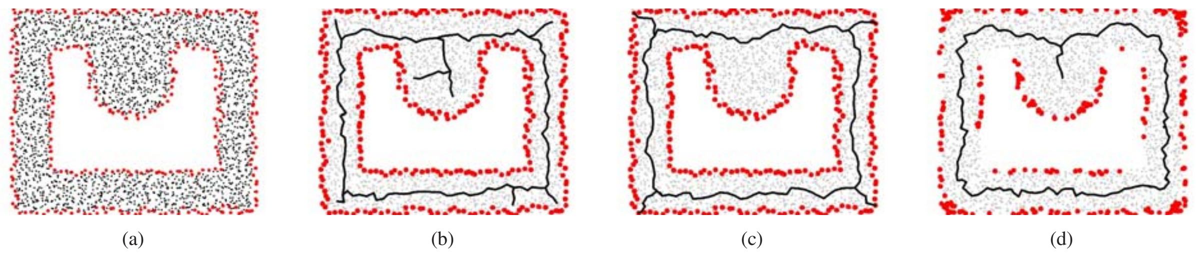
Fig. 8. Skeleton of network with a concave hole with 2777 nodes and average degree 12.99. (a) Original network; (b) Skeleton extracted by MAP; (c) Skeleton extracted by CASE; (d) Skeleton extracted by DIST using 80% of the boundary nodes.

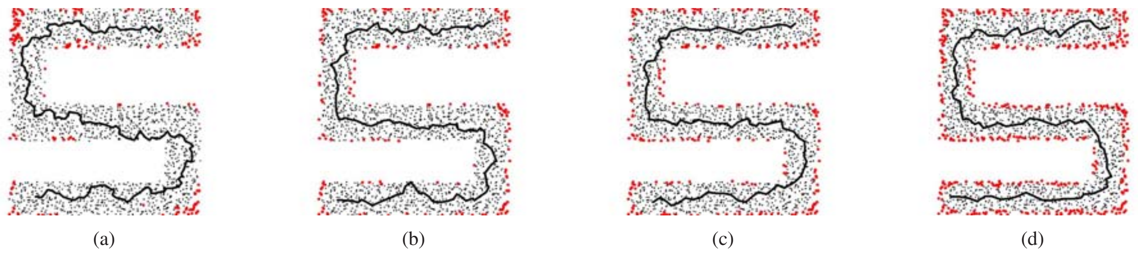
Fig. 9. Skeleton extraction on an S-shaped network with different node degrees. Boundary nodes are marked in red and the dark curve is the skeleton. (a) average degree 6.54; (b) average degree 12.38; (c) average degree 17.41; (d) average degree 21.91.

parameters and if these parameters are not set correctly, the skeleton generated may be unacceptable. Our algorithm is robust to boundary incompleteness and inaccuracies, and can work when the boundary nodes are not fully identified. In addition, our algorithm can work at a very low node density.

VI. Related Work

There are two representative algorithms for skeleton extraction. Gao *et al.* [5], [6] proposed an algorithm to extract skeletons, based on which a routing protocol (MAP) is designed. The experiments show that MAP can improve the routing performance. Jiang *et.al.* [18], [19] proposed a Connectivity-bAsed Skeleton Extraction algorithm (CASE). CASE first identifies some corner points in the boundaries. These corner points then decompose the boundaries into several boundary branches. If an interior node has equal distance to two boundary nodes on different boundary branches, this node will mark itself as a skeleton node. The novelty of CASE is that it can control boundary noise by introducing a corner threshold value. A larger threshold value will cause fewer skeleton nodes to be determined. A common assumption of MAP and CASE is that all boundary nodes are already known, either by manual input or by a boundary recognition scheme. One piece of recent work close to ours is the method in [33] that aims to segment a complex network into a set of nicely shaped parts but it suffers from boundary noise greatly.

VII. Conclusion

We have described DIST, a novel distance transform based skeleton extraction algorithm for wireless sensor networks, using only connectivity information. The algorithm does not require the boundary nodes be complete or accurate, making it more applicable than previous solutions that rely on accurate detection of network boundaries. The algorithm is distributed, robust to boundary noise, and of low complexity. We have compared our algorithm with two existing solutions, CASE and MAP. The results show that our algorithm consistently outperforms those algorithms by generating accurate skeletons from incomplete boundaries.

In the future, we plan to study the sensor's topology-related applications such as segmentation, routing, and localization, which can benefit from the skeleton extraction proposed in this paper. Besides, we will study the skeleton extraction problem in 3D sensor networks, which is more challenging. Finally, we would like to explore the possibility of designing more energy-efficient data processing scheme [15], [16], [17], [8] using skeleton information.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 60803115, Grant 61073147, Grant 61103243, and Grant 61173120; by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the Fundamental Research Funds for the Central Universities under Grant 2011QN014; by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the CHUTIAN Scholar Project of Hubei Province; by the Youth Chenguang Project of Wuhan City under Grant 201050231080; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry). The corresponding author is Hongbo Jiang.

References

[1] H. Blum. Transformation for extracting new descriptors of shape, models for the perception of speech and visual form. *MIT Press*, pages 363–380, 1967.

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

[3] G. Borgefors. Distance transformations in digital images. In *Proc. of Computer Vision, Graphics, and Image Processing*, 1986.

[4] J. W. Brandt and V. R. Algazi. Continuous skeleton computation by voronoi diagram. *CVGIP:Image Understanding*, 55(3):329–338, 1992.

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

[6] J. Bruck, J. Gao, and A. A. Jiang. Map: Medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6):835–853, 2007.

[7] C. Buragohain, D. Agrawal, and S. Suri. Distributed navigation algorithms for sensor networks. In *Proc. of IEEE INFOCOM*, 2006.

[8] J. Cheng, H. Jiang, J. Liu, W. Liu, and C. Wang. On efficient processing of continuous historical top-k queries in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 60(5):2363–2367, 2011.

[9] W. P. Choi, K. M. Lam, and W. C. Siu. Extraction of the euclidean skeleton based on a connectivity criterion. *Pattern Recognition*, 36(3):721–730, 2003.

[10] M. Coupriea, D. Coeurjollyb, and R. Zrourc. Discrete bisector function and euclidean skeleton in 2d and 3d. *Image and Vision Computing*, 25(10):1543–1556, 2007.

[11] D. Dong, Y. Liu, and X. Liao. Fine-grained boundary recognition in wireless ad hoc and sensor networks by topological methods. In *Proc. of ACM MOBIHOC*, 2009.

[12] S. P. Fekete, A. Kroller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *Proc. of the 1st Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[13] S. Ganeriwal, P. Kumar, and M.B.Srivastava. Timing-sync protocol for sensor networks. *SenSys'03:Proc. of the ist international conference on Embedded networked sensor systems*, 2003.

[14] Y. Ge and J. M. Fitzpatrick. On the generation of skeletons from discrete euclidean distance maps. *IEEE Trans. PAMI*, 18(11):1055–1066, 1996.

[15] H. Jiang and S. Jin. Scalable and robust aggregation techniques for extracting statistical information in sensor networks. In *Proc. of IEEE ICDCS*, 2006.

[16] H. Jiang, S. Jin, and C. Wang. Parameter-based data aggregation for statistical information extraction in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 59(8):3992–4001, 2010.

[17] H. Jiang, S. Jin, and C. Wang. Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):1064–1071, 2011.

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

[19] 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. TPDS*, 21(5):710–721, 2010.

[20] H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang. Consel: Connectivity-based segmentation in large-scale 2d/3d sensor networks. In *Proc. of IEEE INFOCOM*, 2012.

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

[22] F. Leymarie and M. Levine. Simulating the grassfire transform using an active contour model. *IEEE Trans. PAMI*, 14(1):56–75, 1992.

[23] M. Li and Y. Liu. Underground structure monitoring with wireless sensor networks. In *Proc. of ACM/IEEE IPSN*, 2007.

[24] M. Li and Y. Liu. Underground coal mine monitoring with wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(2):10–29, 2009.

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

[26] Y. Liu, K. Liu, and M. Li. Passive diagnosis for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 18(4):1132–1144, 2010.

[27] C. Niblack, P. Gibbons, and D. Capson. Generating skeletons and centerlines from the distance transform. *CVGIP:Graphical Models and Image Processing*, 54(5):420 – 437, 1992.

[28] S. Schaefer and C. Yuksel. Example-based skeleton extraction. In *Proc. of Eurographics Symposium on Geometry Processing*, 2007.

[29] H. Talbot and L. Vincent. Euclidean skeletons and conditional bisectors. 1992.

[30] 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. of ACM MOBIHOC*, 2010.

[31] L. Vincent. Efficient computation of various types of skeletons. In *Proc. of SPIE Medical Imaging V*, 1991.

[32] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of ACM MOBICOM*, 2006.

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