Distance Transform-Based Skeleton Extraction and Its Applications in Sensor Networks

Wenping Liu, *Student Member*, *IEEE*, Hongbo Jiang, *Member*, *IEEE*, Xiang Bai, *Member*, *IEEE*, Guang Tan, *Member*, *IEEE*, Chonggang Wang, *Senior Member*, *IEEE*, Wenyu Liu, *Member*, *IEEE*, and Kechao Cai

Abstract—We study the problem of skeleton extraction for large-scale sensor networks with reliance purely on connectivity information. Existing efforts in this line highly depend on the boundary detection algorithms, which are used to extract accurate boundary nodes. One challenge is that in practical this could limit the applicability of the boundary detection algorithms. For instance, in low node density networks where boundary detection algorithms do not work well, the extracted boundary nodes are often incomplete. This paper brings a new view to skeleton extraction from a distance transform perspective, bridging the distance transform of the network and the incomplete boundaries. As such, we propose a distributed and scalable algorithm for skeleton extraction, called DIST, based on *DIS*tance*T*ransform, while incurring low communication overhead. The proposed algorithm does not require that the boundaries are complete or accurate, which makes the proposed algorithm more practical in applications. First, we compute the distance transform of the network. Specifically, the distance (hop count) of each node to the boundaries of a sensor network is estimated. The node map consisting of the distance values is considered as the distance transform (the distance map). The distance map is then used to identify skeleton nodes. Next, skeleton arcs are generated by controlled flooding within the identified skeleton nodes, thereby connecting these skeleton arcs, to extract a coarse skeleton. Finally, we refine the coarse skeleton by building shortest path trees followed by a prune phase. The obtained skeleton is robust to boundary noise or shape variations. Besides, we present two specific applications that benefit from the extracted skeleton: identifying complete boundaries and shape segmentation. First, with the extracted skeleton using DIST, we propose to identify more boundary nodes to form a meaningful boundary curve. Second, the utilization of the derived skeleton to segment the network into approximately convex pieces has been shown to be effective.

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

1 Introduction

The distribution of the sensors and the overall network topology are imperatively needed for a variety of applications, such as data routing, localization, and path planning, in wireless sensor networks. Often the geographical locations and the node deployments may vary greatly, resulting in that the topology of the sensor networks is affected by such factors as obstacles, deployment randomness, and so on. One primitive representing the network topology is *skeleton* (also known as *medial axis*).

The methodology of skeleton is not new: It has been extensively studied in computer vision [6] and computer graphics [18] community for years where the skeleton is an important descriptor that contains both the topological and geometrical properties of the object. In wireless sensor networks, the skeleton information of the sensor network can greatly improve the performance of routing, location service, segmentation, and navigation algorithms. Despite a wide range of work on this in the past, skeleton extraction continues to be a challenge, given the connectivity information alone for sensor networks. Furthermore, there are harsh requirements on system performance in terms of message/time complexity in sensor networks.

Although the designs of the existing connectivity-based algorithms [4], [5], [13], [14] for skeleton extraction in sensor networks have demonstrated great ingenuity, and their effectiveness is shown through extensive simulations, the main drawback of them is the high dependency on the boundary detection algorithms, which are used to extract accurate boundary nodes. As a result, this makes existing solutions possibly inapplicable in practical. In addition, in existing works, a skeleton node is such that it has equal hop counts to at least two closest boundary nodes [4], [5] or boundary branches [13], [14], [22]. This definition, however, suffers from boundary noise. That is, a small bump on the boundary will incur a long skeleton branch; for the case of incomplete boundaries, the result is even worse.

We tackle the problem of skeleton extraction in wireless sensor networks where a part of boundary nodes can be easily identified (e.g., by neighborhood-based algorithm [9]) and exploited to build distance transform, based on

- *W. Liu is with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China, and the Hubei University of Economics, Wuhan 430205, China. E-mail: wenpingliu2009@gmail.com.*
- *H. Jiang, X. Bai, W. Liu, and K. Cai are with the Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China 430074. E-mail: {hongbojiang2004, xiang.bai, caikechao}@gmail.com, liuwy@mail.hust.edu.cn.*
- *G. Tan is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, 1068 Xueyuan Avenue, Shenzhen University Town, Shenzhen 518055, China, and the Huazhong University of Science and Technology. E-mail: guangtan@gmail.com.*
- *C. Wang is with InterDigital Communications, South Wing, Melville, NY 11747. E-mail: cgwang@ieee.org.*

Manuscript received 19 Feb. 2012; revised 7 Oct. 2012; accepted 8 Oct. 2012; published online 19 Oct. 2012.
Recommended for acceptance by X.-Y. Li.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2012-02-0118.
Digital Object Identifier no. 10.1109/TPDS.2012.300.

TABLE 1
List of Notations

Notation	Description
$\mathcal{D}$	The network.
$\partial\mathcal{D}$	The boundary of the network.
$DT(\mathcal{D})$	The hop count transform of the network.
$d_{\partial\mathcal{D}}(p)$	The hop count transform of node p.
$N_r(p)$	$r-hop$ neighborhood of p, which is the set of nodes that are at most r hops from node p.
$d_E(x,y)$	The Euclidean distance between points x and y.
$d(p,q)$	The hop count distance between nodes p and q.
$S(p,q)$	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 formed by skeleton arc i and j.
$r(p)$	The root node of p.
$D(p, d_{\partial\mathcal{D}}(p))$	The disk centered at point p with radius $d_{\partial\mathcal{D}}(p)$.

connectivity information only. Our strive is to 1) find the skeleton nodes and 2) connect them in a proper way. To address the above-mentioned problems stemmed from the skeleton definition, this paper brings a new view to skeleton extraction from a *distance transform* perspective, bridging the distance transform of the network and the incomplete boundaries. Apart from the existing approaches [4], [13] which extract skeletons directly from a sensor network, the proposed method computes skeletons based on the distance transform of a sensor network. The identified skeleton nodes are lay medially inside the network, that is, they are good approximations to the centers of the maximal disks in the continuous case.

In this paper, we propose DIST, a distributed and scalable algorithm for skeleton extraction based on DIStance Transform, while incurring low communication overhead. In computer vision community, distance transform [2] (or *distance map*, *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 $\partial\mathcal{D}$ the boundaries of $\mathcal{D}$ (A list of notations can be found in Table 1). We refer to $d_E(x,y)$ as the distance between two points x and y. The distance transform of $\mathcal{D}$ is thus defined as

$$DT(\mathcal{D}) = \begin{cases} \min\limits_{y\in\partial\mathcal{D}} d_E(x,y), & x \in \mathcal{D} \\ 0, & x \notin \mathcal{D}. \end{cases} \quad (1)$$

For any point $x \in \mathcal{D}$, $d_{\partial\mathcal{D}}(x) = \min_{y\in\partial\mathcal{D}} d_E(x,y)$ is referred to as the distance of point x to the boundaries of $\mathcal{D}$. Any distance measure (such as euclidean distance, Manhattan distance, etc.) can be used to construct the distance transform here. In DIST, first we assume a part of the boundary nodes are identified by a neighborhood-based boundary detection algorithm. We then compute the hop count distance of each node to the boundaries, resulting in a hop count transform. Subsequently, we identify skeleton nodes in a distributed manner: Each node determines whether it is a skeleton node based on its hop count 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 largest length forms a skeleton arc. Connecting the skeleton arcs, we obtain a coarse skeleton. Finally, we refine the coarse skeleton by trimming skeleton branches in the coarse skeleton. As the example shown in Fig. 1, the result by our proposed method (see Fig. 1d) is better than that of MAP [4], [5] (see Fig. 1b) and CASE [13], [14] (see Fig. 1c). More desirably, even when only 50 percent of the boundary nodes are identified in a sparse network, our proposed algorithm still performs well, shown in Fig. 1e.

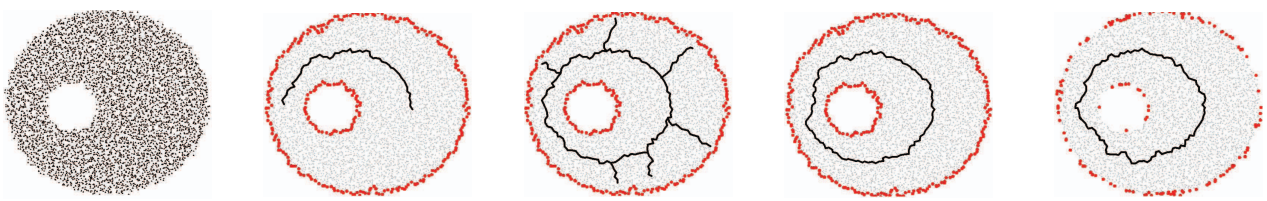
The second contribution is the demonstration of the effectiveness of DIST on two specific applications that benefit from the extracted skeleton: identifying complete boundaries and shape segmentation. With the extracted skeleton using DIST, we propose to identify more boundary nodes in turn, and to connect these boundary nodes to form a meaningful boundary curve. This "complete boundaries" can be useful for many sensor applications that require complete boundaries. Besides, we study how to utilize the derived skeleton to segment the network into approximately convex pieces such that algorithms and protocols that assume a regularly shaped field can be applied in each convex piece.

The rest of this paper is organized as follows: Section 2 presents the background of distance transform and the motivation of our work, and Section 3 presents the details of DIST. We evaluate DIST in Section 4. We introduce two specific applications of DIST in Section 5. Finally, Section 6 concludes the paper.

2 Preliminaries

For sensor networks with mere connectivity information, the distance measure is the hop count between nodes. In the sense of distance measure, we also call the distance transform of a sensor network as *hop count transform* (or *hop count map*).

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



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

Fig. 1. Skeleton extraction of an eclipse-shaped network with 5,392 nodes. In (b)-(e), the dark-colored nodes (marked in red) are boundary nodes, and the curves denote the skeletons. The default average degree is 11.9. (a) The original map. (b) CASE [13]. (c) MAP [4]. (d) DIST. (e) We reduce the radio range such that the average degree is only 6.1 and 50 percent of the boundary nodes are identified where the skeleton is extracted by DIST.

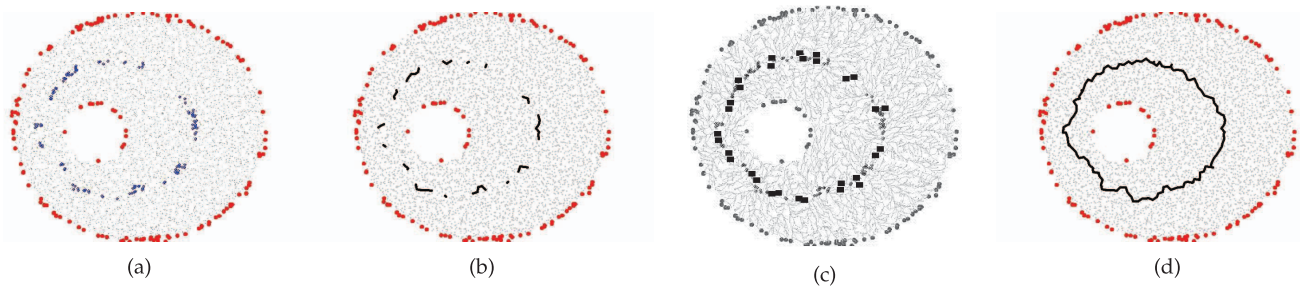


Fig. 2. Skeleton extraction. (a) Critical skeleton nodes. (b) Skeleton arcs. (c) Skeleton cut pairs are shown as rectangles. (d) Coarse skeleton. The final refined result is shown in Fig. 1d.

According to the Blum's definition [1], a point x is a skeleton point if it is the center of a maximal disk. Note that the centers of the maximal disks can be easily detected by comparing the distance transforms only in a neighborhood. More specifically, if there is a neighbor of x, say, point y, satisfying that the disk centered at y with radius $d_{\partial\mathcal{D}}(y)$ contains $D(x, d_{\partial\mathcal{D}}(x))$, then $D(x, d_{\partial\mathcal{D}}(x))$ is not a maximal disk. As such, to check whether the disk $D(x, d_{\partial\mathcal{D}}(x))$ is a maximal disk, one only needs to determine whether there is a neighbor y, such that $D(y, d_{\partial\mathcal{D}}(y))$ contains $D(x, d_{\partial\mathcal{D}}(x))$. $D(y, d_{\partial\mathcal{D}}(y))$ contains $D(x, d_{\partial\mathcal{D}}(x))$ if and only if the following condition holds [3]: $d_{\partial\mathcal{D}}(y) \geq d_{\partial\mathcal{D}}(x) + d_E(x, y)$.

Note that the radius of the maximal disk of point x is actually its distance transform. As a matter of fact, the skeleton by the maximal disks of $\mathcal{D}$ is the set of points whose distance transforms are locally maximal [21]. For discrete wireless sensor networks, we can detect skeleton nodes based on hop count transform in a similar way. Specifically, node p is a skeleton node if for any node $q \in \mathcal{N}_1(p)$, the following condition is true:

$$d_{\partial\mathcal{D}}(p) > d_{\partial\mathcal{D}}(q). \tag{2}$$

Due to the integer rounding error of hop count distance between nodes, there might be only a few skeleton nodes identified by above way. To deal with this, we make a slight modification to (2) 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 locally maximal, namely, $d_{\partial\mathcal{D}}(p) \geq \max\{d_{\partial\mathcal{D}}(q) \mid q \in \mathcal{N}_1(p)\}$.*

It is still to be debated for what is the most appropriate analog of the continuous skeleton in a discrete wireless sensor network [22]. Our definition of skeleton node differs from that given in [4], [13], 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 downside, the skeleton node identification process based on Definition 1 still suffers from boundary noise due to the nature of discrete wireless sensor network. To alleviate such problems, we introduce a parameter $r(\geq 1)$, which determines how sensitive of the identification process is to the boundary noise, and accordingly define *critical skeleton node* as follows:

Definition 2. *If $d_{\partial\mathcal{D}}(p) \geq \max\{d_{\partial\mathcal{D}}(q) \mid q \in \mathcal{N}_r(p)\}$, 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 that allows to identify a skeleton node in a simple way. In our simulations, $r = 3$ or 4 is found to be a good choice for practical purpose. One undesirable characteristic of the critical skeleton nodes is that they only account for a few skeleton nodes such that in general they are disconnected. Fig. 2a shows an example. Theoretically, a skeleton is homotopic to the original object and should have the same simple connectivity as the original object [17]. To achieve this purpose, we additionally extract intermediate nodes to connect the critical skeleton nodes (detailed in Section 3.4).

3 Skeleton Extraction Algorithm

3.1 An Overview

As we mentioned in Section 2, 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 beforehand. To that end, the first step is to discover the boundary nodes. As mentioned in Section 1, we use the neighborhood-based algorithm in [9] to detect the boundary nodes, even if the obtained boundaries could be incomplete. Based on these identified boundary nodes, the establishment of hop count distance transform of each node is trivial. Subsequently, we identify critical skeleton nodes based on the obtained hop count distance transform. However, the critical skeleton nodes are in general disconnected. The challenge, after the phase of critical skeleton node identification, is to connect these critical skeleton nodes in a right way.

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

1. *Distance map establishment*: Each identified boundary node floods inward the network to build a shortest path tree (referred to as boundary tree) rooted at itself. Every sensor node is then associated with a boundary tree and keeps track of the minimum hop count distance from the boundaries, as well as the nearest boundary node (i.e., the root). 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 in a distributed manner.

3. *Coarse skeleton establishment*: The connected critical skeleton nodes are grouped into a component. All critical skeleton nodes in the same component are assigned a common identifier (e.g., the maximum node ID in the component) via scoped flooding initiated by every critical skeleton node. Within each component, we link two farthest skeleton nodes to obtain a skeleton arc, followed by connecting these skeleton arcs properly to generate a coarse skeleton. We emphasize 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 might contain unwanted skeleton branches with small lengths (defined by the number of nodes in the skeleton branch), the coarse skeleton needs to be pruned. In the final phase, we trim the skeleton branches with small lengths, and the refined skeleton is obtained.

3.2 Distance Map Establishment

In this section, we mainly focus on how to use boundary nodes to generate a hop count transform $DT(\mathcal{D})$ through local flooding. This can be done in a distributed manner as follows: First, each boundary node p initiates a flooding message to build a shortest path tree $T(p)$. The flooded message contains the ID of p and the number of hops that the message has traveled. For each interior node q, upon receiving a message from p, if q has not received any message before, q will join the tree $T(p)$, record the parent node that 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, a shortest path tree $T(p)$ is constructed; and each node that has the minimum hop count distance to p is associated with the shortest path tree $T(p)$. We call such tree $T(p)$ as a boundary tree. This process is conducted repeatedly until every node belongs to a boundary tree.

As emphasized in [22], if the identified boundary nodes perform their flooding approximately simultaneously, and the flooded messages travel at approximately the same speed, each interior node will forward only one message. This will reduce the total number of delivered messages substantially 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 (i.e., the counter stored at each node) to the boundaries, and a hop count map is established.

Lemma 1. *For an interior node q, its hop count transform is larger than that of its parent node $P(q)$.*

Corollary 2. *For an interior node q, if q is not a critical skeleton node, then $P(q)$ will not be a critical skeleton node.*

Theorem 3. *If q is a critical skeleton node, then q must be a leaf node.*

Theorem 3 provides a sufficient but not necessary 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 identify whether a leaf node is a critical skeleton node. This can narrow down the scope of critical skeleton identification process, thereby keeping the total communication overhead low. In next section, we will show how a leaf node identifies itself in a distributed fashion.

3.3 Critical Skeleton Nodes Identification

After the establishment of the hop count transform, we next conduct the so-called *skeleton node identification process*. Each leaf node p determines whether itself is a critical skeleton node, by comparing the hop count 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 constructed in Section 3.2 first performs a controlled-flooding operation within its r-hop neighbors $N_r(p)$. The flooded message includes the ID of p, its hop count transform $d_{\partial\mathcal{D}}(p)$, 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 it is an r-hop neighbor of p. 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 node 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 4. *For two leaf nodes p_1, p_2 which have a separation 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, $d_{\partial\mathcal{D}}(p_1) > \partial\mathcal{D}(p_2)$.*

Theorem 5. *A leaf node p is a critical skeleton node if and only if there is no leaf node $s \in \mathcal{N}_r(p)$ such that node s is reachable to node p.*

One implication of Theorem 5 is that to identify whether a leaf node p is a critical skeleton node; it only needs to check whether there is a node within its r-hop neighborhood that is reachable to p. More specifically, in the above-mentioned skeleton node identification 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. 2a shows the result of this step.

3.4 Coarse Skeleton Establishment

So far, a set of critical skeleton nodes have already been identified. Note that two fundamental properties of a skeleton are: 1) It is medially placed (therefore, maintains the "medialness"); and 2) it has the simple connectivity as the original shape [17]. In continuous case, the centers of maximal disks are medial and connected. With discrete wireless sensor network, however, the critical skeleton nodes lie medially but unfortunately, they are generally disconnected. In this section, we propose to identify intermediate nodes (referred to as *connecting skeleton nodes*), which lie medially and can be used to link two adjacent skeleton arcs. Accordingly, a coarse skeleton will be generated.

We first construct a set of connected skeleton components of the critical skeleton nodes identified in Section 3.3 and generate a set of skeleton arcs. This can be done as follows: Each critical skeleton node issues a controlled

flooding with a message containing its node ID and the number of hops that 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 the node ID of q is larger than that of p, p forwards the flooding message to its neighbors; or, 2) otherwise, it simply discards this 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. 2b); and each critical skeleton node is assigned a unique identifier (e.g., the maximum node ID in the component). Without ambiguity, we call only the node on skeleton arcs critical skeleton node.

With these skeleton arcs formed, we now detect intermediate nodes, based on the *slope function* of the distance map, to connect skeleton arcs. In continuous case, skeleton arcs follow lines of steepest slope of the euclidean distance map [19], where the slope of the line xy, $S(x, y)$, is defined as

$$S(x,y) = \frac{d_{\partial\mathcal{D}}(y) - d_{\partial\mathcal{D}}(x)}{d_E(x,y)}. \tag{3}$$

When a point x is detected as a skeleton point, the neighbor of x which has a steepest ascending slope is identified as a new skeleton point [7]. The steepest ascent approach can guarantee that the skeleton branches locate medially [11].

In discrete wireless sensor networks, we define the slope, $S(p, q)$, of the line by connecting two nodes p, q as follows:

$$S(p,q) = \frac{d_{\partial\mathcal{D}}(q) - d_{\partial\mathcal{D}}(p)}{d(p,q)}. \tag{4}$$

If nodes p, q are two neighboring nodes, (4) can be simplified as

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

Lemma 6. *Let q' be a neighbor of a critical skeleton node q. If $S(q', q) \geq max_{s\in N_1(q)} S(s, q)$, then q' locates approximately medially, and we call q' as a connecting skeleton node.*

Lemma 7. *Let q_1, q_2 be two nodes that have the same hop count 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 $S(p, q_1) \geq S(p, q_2)$.*

According to Lemmas 6 and 7, we now propose to identify connecting skeleton nodes. First, we have all critical skeleton nodes synchronize among themselves [10] and start to flood the network at approximately the same time. These skeleton nodes perform a flood with the messages of 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 issued by a critical skeleton node, say p, and forwarded by q', if q has not received a message before, q will join the tree rooted at p, keep record of the parent node q' 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' with that of $q's$ parent node $P(q)$. If $d_{\partial D}(q') > d_{\partial D}(P(q))$, q changes its parent node as q' and updates its average hop count transform; otherwise, q only 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 as 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, which shows that the corresponding two skeleton arcs are adjacent and can be connected. We thus define *cut* nodes, $\mathcal{C}(i, j)$, as the nodes where two skeleton trees, whose roots belong to different skeleton arcs i and j, respectively, meet.

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* in our paper is given 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. As such, all skeleton cut-pairs (and cut-pair nodes) can be detected, see Fig. 2c 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 8. *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.*

Based on Theorem 8, each cut node can claim 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 the largest slope, q claims 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. 2d.

We have so far detected two kinds of skeleton nodes: critical skeleton nodes and connecting skeleton nodes. In the next section, we call both of them, without ambiguity, skeleton nodes for short.

3.5 Coarse Skeleton Refinement

Undesirably, the path between two skeleton nodes of the coarse skeleton may be not the shortest path, and there may exist unwanted skeleton branches, i.e., with a small number of skeleton nodes. As such, we need to refine the coarse skeleton by limited flooding within skeleton nodes on the coarse skeleton. More specifically, each skeleton node p sets

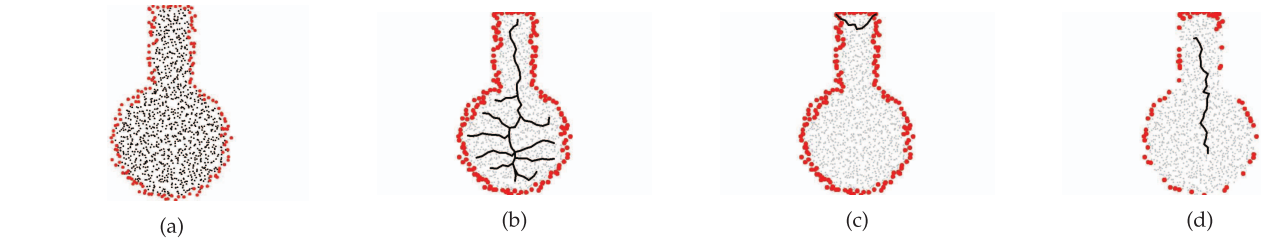


Fig. 3. Skeleton of a bat-shaped network with 1,272 nodes, average degree 13.30. (a) Original network. (b) Skeleton extracted by MAP. (c) Skeleton extracted by CASE. (d) Skeleton extracted by DIST using only 40 percent of the boundary nodes.

a timer with a random remaining time. When the remaining time of p reaches 0, node p begins to flood within the coarse skeleton and builds a shortest path tree. The message includes the timer and the node ID of p. When a skeleton node q receives the message from p, it will compare its timer with the timer of p, and the node with a smaller timer will dominate. As such, 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). Specifically, if the length of a branch is less than a certain value (we can simply set this value as r, which is used in Section 3.3 for critical skeleton node identification), the skeleton branch will be trimmed. Finally, we obtain the refined skeleton, as shown in Fig. 1d.

3.6 Discussion

3.6.1 Limitations of DIST

As a skeleton extraction scheme that uses connectivity information only, DIST relies on the assumption that network nodes are relatively evenly distributed, for example, a perturbed grid or uniform random one, as well as the internode distance has a much smaller variance than the global network size. This is required for a reasonable accuracy when the distance is estimated by hop count. When the network is too small, or the node density is nonuniform, the accuracy of skeleton extraction will be affected seriously.

3.6.2 Effect of System Parameters

The parameter r is used to determine how sensitive of the identification process is to the boundary noise, and accordingly, we define critical skeleton node based on it. For instance, an area locally can be consider an exact nonstraight boundary area for a small value of r. In contrast, for a large r, this area could be considered to be boundary noise and the generated skeleton will be derived not from it to avoid this kind of noise. Otherwise, many nodes that have two closest boundary nodes that are not nearby are undesirably identified as skeleton nodes. That is, r is a predefined system parameter, reflecting the tradeoff between boundary accuracy and boundary noise from the user perspective. The choice of a r value then depends on the applications.

4 Performance Evaluation

To evaluate the effectiveness of DIST, we have conducted extensive simulations on various scenarios, comparing with two existing solutions, namely, CASE [13], [14], as well as MAP [4], [5]. In addition, we study the influence of node density on DIST and show how robust DIST is to node density and boundary incompleteness, and the robustness to communication radio model is also examined. In this section, we first present our simulation setup, followed by the simulation results.

4.1 Simulation Setup

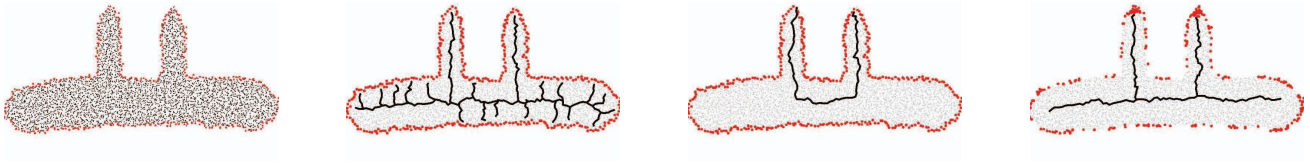
In our simulation settings, we first generate a figure with a given shape, and then we read this figure file to derive a set of pixels with coordinates (x, y), which accordingly determines a set of points in the given $2D$ plane. By adding some perturbations to these points, we thus generate a network scenarios with the given shape, where the coordinate of a point represents the location of a sensor node. Further, we assume all sensor nodes have the same communication radio range; and the communication between nodes follows a unit-disk graph model by default. We use the neighborhood-based algorithm in [9] to identify boundary nodes, and the parameter for critical skeleton node identification is $r = 3$ unless otherwise specified.

For fair comparison, we evaluate CASE and MAP algorithms based on complete boundaries while evaluating DIST based on incomplete boundaries by reducing the radio range, thereby causing sparse networks (low node density).

4.2 Simulation Results

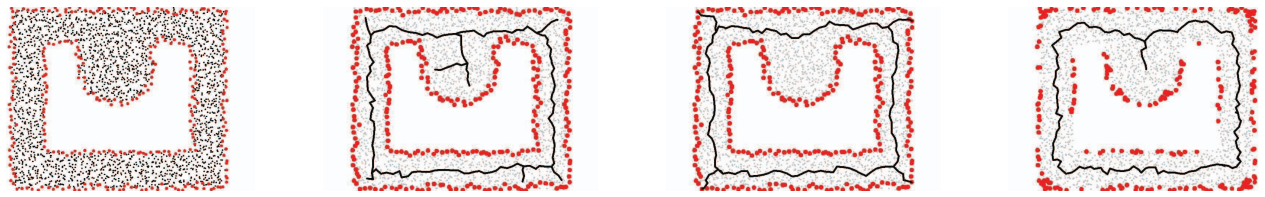
Fig. 3 shows the skeletons extracted by MAP (see Fig. 3b), CASE (see Fig. 3c), and DIST (see Fig. 3d) on a bat-shaped network. The skeleton extracted by MAP has many unwanted long branches, as shown in Fig. 3b. This is because MAP suffers from boundary noise. Specifically, many nodes, which have two closest boundary nodes that are not nearby, are undesirably identified as skeleton nodes. As for Fig. 3c, two corner points are identified by which the boundary is decomposed into two boundary branches. We can see that the skeleton extracted by CASE is quite a few, and many nodes really located medially are not extracted because their two closest boundary nodes are on the same boundary branch. Fig. 3d shows the skeleton extracted by DIST when only 40 percent of boundary nodes are obtained. We can clearly see that DIST controlled boundary noise efficiently, and the extracted skeleton has no unwanted skeleton branches. Obviously, the skeleton by DIST is better than (at least comparable to) that of MAP and CASE.

We next examine the performance of algorithms on airport terminal network. We see similar results as in the bat-shaped network, as shown in Fig. 4. The skeleton extracted by MAP has many skeleton branches due to boundary noise, as shown in Fig. 4b. For CASE, due to the



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

Fig. 4. Skeleton of terminal-shaped network with 5,012 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 percent of the boundary nodes.



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

Fig. 5. Skeleton of network with a concave interior hole, 2,777 nodes, average degree 12.99. (a) Original network. (b) Skeleton extracted by MAP. (c) Skeleton extracted by CASE. (d) Skeleton extracted by DIST using 80 percent of the boundary nodes.

improperly chosen parameters, some skeleton nodes are not identified because their two closest boundary nodes are on the same boundary branch. Consequently, only partial skeleton is extracted by CASE. One might increase the threshold value of corner node; however, this could incur that many boundary nodes are identified as corner nodes. As a result, a skeleton with many unwanted skeleton branches will be generated. Fig. 4d shows that DIST is able to generate a good skeleton graph even in the case of incomplete boundaries.

We next study the performance of algorithms on one-hole network. Different from Figs. 3 and 4, the network shown in Fig. 5 has a concave hole (possibly caused by obstacles or nodes failures, etc.) inside the network. The skeleton extracted by MAP in Fig. 5b is good except that the skeleton has some skeleton branches. In Fig. 5c, four corner points on outer boundary are identified, and the skeleton nodes are detected accordingly. Since there are only five boundary branches, we can see that the shortest path with the largest length make the skeleton a little deviation from the "medial" location. The skeleton by DIST is comparable to that of MAP, even with only 80 percent of the boundary nodes. This is because DIST regards the nodes with locally maximal hop count transform as skeleton nodes and thus controls boundary noise efficiently. As such, for networks with incomplete boundary information, the derived skeleton is still desirable.

To further understand the message complexity, we compute message costs of DIST on examined scenarios, in terms of the total number of transmitted messages of the algorithm, as shown in Table 2. We can see that DIST always causes considerable message cost, and it scales well to the network size.

Overall, MAP suffers from boundary noise and incurs many skeleton branches. CASE is sensitive to the parameters and if these parameters are not set correctly, the skeleton could be unacceptable. DIST is robust to boundary incompleteness and inaccuracy, and it can work when the boundary nodes are not fully identified, with a slightly bend toward the gap formed by two adjacent boundary nodes that are not nearby. In addition, DIST can work even when the node density is very low.

5 Applications of Skeleton Information

Many applications can benefit from the extracted skeleton. In this section, due to space limit, we only present two specific applications, namely, identifying complete boundaries and shape segmentation.

5.1 Skeleton-Based Routing

To the best of our knowledge, the most appealing application using skeleton so far is to achieve load balancing for routing [4]. Roughly speaking, each sensor node is given a name related to its position with respect to the skeleton. The routing decision is then derived based on the names of the source and destination nodes and guarantees delivery with reasonable and natural routes. More technical details can be found in [4].

To that end, we first conduct simulations to measure the routing performance with DIST, based on the network in Fig. 6. We randomly select 12,000 source and destination pairs for routing. Fig. 7a shows the results of load distribution on sensor nodes. We can see that, compared with other skeleton results, DIST leads to more uniform loads (i.e., the number of transmitted packages) on nodes. The reason is that DIST results in a well-connected skeleton graph well capturing the geometry of the networks, performing routing in parallel with the reference paths on the skeleton. In contrast, other algorithms often lead to undesirable skeleton results, resulting in unbalanced loads

TABLE 2
Message Costs under Different Scenarios

Topology	Network size	CASE	MAP	DIST
Eclipse(see Fig. 1)	5,392	15,613	12,285	16,786
Bat(see Fig. 3)	1,272	2,931	3,038	4,868
Terminal(see Fig. 4)	5,012	16,736	12,225	16,302
One-hole(see Fig. 5)	2,777	9,701	6,153	9,611

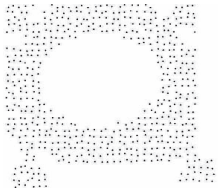
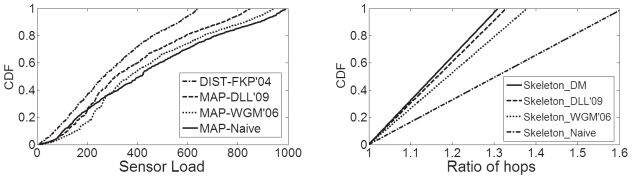


Fig. 6. A network with 577 nodes and avg.deg 5.95.



(a) CDF of routing load (b) CDF of routing path stretch

Fig. 7. Routing performance using different skeleton results.

on the different sides of the skeleton. Especially, some skeleton could be missed, causing that packets could follow the boundary heavily as mentioned in [4]. Fig. 7b shows the path stretch of routing, defined as the ratio of the routing path length to the shortest path hop count between the nodes, using different skeleton results. DIST outperforms other algorithms because it achieves smaller stretches. The reason is that a well-connected skeleton graph is prone to being represented by straight lines locally. That can lead to a short routing path since the routing is performed in parallel with the reference paths on the skeleton.

5.2 Shape Segmentation

A great number of existing protocols achieve much better performance in regular field than in irregular one [22], and thus, the shape segmentation has attracted a lot of attention from wireless sensor network community (e.g., [8], [16], [20], [22]). We assume that the sensor field is a polygonal environment, where the field boundaries (inner boundaries and outer boundary) are all simple polygons. A polygon is said to be convex if for each vertex, its inward (directed toward the sensing field) angle is no greater than π. If there is a vertex whose inward angle is greater than π, we call such vertex as a concave (or dull) point. In this section, we propose to decompose a sensor network with complex shape into convex pieces based on the obtained skeleton.

Obviously, to identify whether a polygon is convex, one only needs to detect whether there is a concave point. If there is no concave point, the polygon (the network) itself is convex; otherwise, we can draw a line, referred to as a segment line (e.g., bisectors), from each concave point until the line hits an edge of the polygon (namely, the boundary of the network). By doing so, we can decompose the network into convex pieces, of which each segment line is a boundary curve. The key problem here is to identify concave points, which are located on the boundaries of the sensing field. Generally, for the case of incomplete boundaries, this is rather difficult since some concave points may not be identified. In most cases, however, a concave point may *bend* the skeleton and form a *corner* skeleton node whose closest boundary nodes includes the concave point. That is, a boundary node identifies itself a concave node if there is a *corner* skeleton node corresponding to this boundary node. As such, it is possible for us to decompose the network into convex pieces based on the skeleton. Generally, for a skeleton node p, there are four possible cases:

1. p has no nearest concave boundary nodes. For this case, we cannot identify a concave node by using the information of p.
2. p has only one nearest concave boundary node. In this case, we can detect a concave node by checking whether the corresponding skeleton node is a *corner* skeleton node.
3. p has two concave nearest boundary nodes. For this case, we need a special treatment.
4. p has three or more nearest boundary nodes, and at least one of them is a concave node. In this case, we call p a joint node that has at least three neighboring skeleton nodes.

Overall, above-mentioned skeleton nodes of cases 2-4 are useful in our shape segmentation algorithm, and we call these nodes *feature nodes*. With these feature nodes, we can segment the skeleton into a set of branches, based on which the network is decomposed into convex pieces.

Now, we present our method for detecting feature nodes. We first detect corner skeleton nodes and joint skeleton nodes, and the identification of *hidden* feature node is deferred for later. Note that joint skeleton node can be easily identified by simply counting the number of its neighboring skeleton nodes after the pruning process of coarse skeleton in Section 3.4. We mainly address how to identify corner skeleton nodes. To that end, we first give our definition of the curvature of a skeleton node, similar with that in [13], in wireless sensor networks as follows:

Definition 4. *For a skeleton node p, let $d_k(p)$ denote the maximum hop count distance between two skeleton nodes that are k-hop neighbors of p. The k-hop curvature of skeleton node p, $c_k(p)$, is defined as*

$$c_k(p) = \frac{d_k(p)}{2 \times k}. \tag{6}$$

Intuitively, if $c_k(p)$ is close to 1, skeleton node p will be more likely an ordinary skeleton node, that is, p locates at the middle of the skeleton, and if $c_k(p) < 1$, skeleton node p may locate at a corner of the skeleton. With this definition, we thus define a *corner* skeleton node as follows:

Definition 5. *A skeleton node p is a* corner *skeleton node if $c_k(p) < \delta$, where $\delta \in (0, 1)$ is a predefined parameter.*

As such, for a given δ, each corner skeleton node can identify itself by computing its k-hop curvature. The corner skeleton nodes and joint skeleton nodes of Fig. 5 are shown in Fig. 8a.

With the corner skeleton nodes and joint skeleton nodes identified, we can segment the skeleton into a set of branches. See Fig. 8b. Meanwhile, we assign a unique branch ID (e.g., the maximum node ID in each branch) to each skeleton node in the branch. After that, each skeleton node floods in the network, and every node claims its

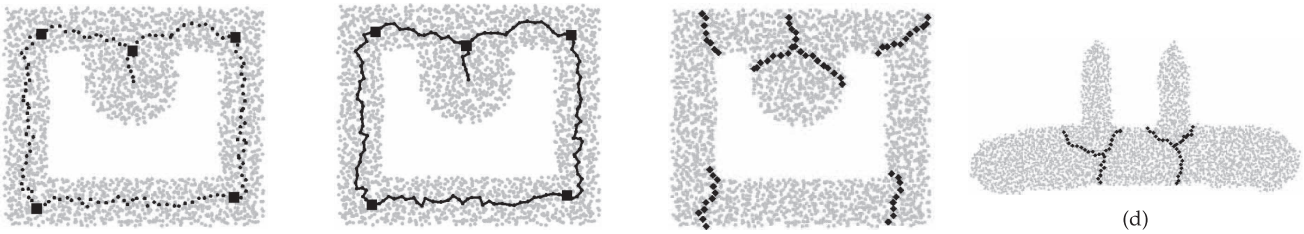


Fig. 8. Skeleton-based segmentation on various networks. The dark nodes in (c) and (d) decompose the networks into nice pieces. (a) Feature nodes of Fig. 5 are shown as large rectangles. Four skeleton nodes located at the corners of the skeleton are identified as *corner* skeleton nodes, and the skeleton node in the top-middle part is a joint skeleton node. (b) Skeleton branches are derived from the identified feature nodes. (c) Segmentation result of Fig. 5. Message cost 12,686. (d) Segmentation result of Fig. 4. Message cost 18,622.

branch ID as the branch ID of its closest skeleton node. Doing so, each node q is assigned a branch ID and keeps record of its closest skeleton node p, and the hop count distance between q and p. Those nodes with the same branch ID naturally form a connected component, that is, the network is segmented into several pieces, as shown in Figs. 8c and 8d.

It is noted that these pieces may be not convex, and hidden skeleton nodes have not been identified yet. Next, we address how to identify hidden skeleton nodes based on the segmentation result, and segment the network into convex pieces. A hidden feature node segments a skeleton branch into two subbranches: One has the same branch ID as before, and the other subbranch chooses the maximum node ID of this subbranch as its new branch ID. For each skeleton node on the subbranch with a new branch ID, it informs all nodes, whose closest skeleton nodes belong to this new subbranch, of the new branch ID. After that, the segmentation of the network is finished, and each piece is approximately convex.

Theorem 9. *After the segmentation, the network has been decomposed into approximately convex pieces.*

Proof. Note that for each concave node q on the boundaries, there is a feature skeleton node p that has already been identified. We only prove the second case, and the other cases can be proved similarly. Since concave node p is one nearest boundary node of q, that is, the ball centered at q will be tangent to the boundary at p, then the line $\overline{pq}$ is vertical to the tangent line at p. According to DIST, the line $\overline{pq}$ (together with its extended line) also decomposes the shape into two parts, of which $\overline{pq}$ is a boundary. It can be easily proved that the two angles at p for these two parts are both less than π; therefore, these two pieces are both convex. □

It is noted that the segmentation algorithm may oversegment the network when there is a complex topology of the extracted skeleton. Due to the discrete nature, one possible case here is that there might be many skeleton nodes identifying themselves as feature skeleton nodes (including joint skeleton nodes, corner skeleton nodes, and hidden feature skeleton nodes). To avoid the network being oversegmented, together with the consideration that a skeleton should be junction detective (such that different logical parts can be separated successfully from the skeleton), a special care is taken here.

6 Conclusion

We have proposed a novel distance transform-based skeleton extraction algorithm in wireless sensor networks, using only connectivity information. The proposed algorithm does not require that the boundary nodes are complete or accurate. We have overcome a series of difficulties in critical skeleton node identification, generating skeleton arcs and coarse skeleton, and refining the coarse skeleton. We emphasize that our proposed algorithm is distributed, robust to boundary noise, applicable for networks with low node density, and of low complexity. We compare DIST with two existing solutions, CASE and MAP. Our experience shows that DIST can achieve a consistent better approximation of the skeleton, while using only a small subset of boundary nodes. We finally introduce two specific applications, namely, identifying complete boundaries and shape segmentation, which can benefit from the extracted skeleton.

We plan to study the skeleton extraction problem in 3D sensor networks, which is more challenging. Besides, we would like to explore the possibility of designing a more energy-efficient data processing scheme [12] using skeleton information.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 61073147, 61103243, 61173120, 61202460, 71101047, and 61271226; 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 Grants 2011QN014 and 2012QN078; by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the Fok Ying Tung Education Foundation under Grant 132036; 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 in [15]. Hongbo Jiang is the corresponding author of this paper.

References

[1] H. Blum, "Biological Shape and Visual Science (Part I)," *Theoretical Biology*, vol. 38, no. 2, pp. 205-287, 1973.

[2] G. Borgefors, "Distance Transformations in Digital Images," *Computer Vision, Graphics, and Image Processing*, vol. 34, pp. 344-371, 1986.

[3] J.W. Brandt and V.R. Algazi, "Continuous Skeleton Computation by Voronoi Diagram," *CVGIP: Image Understanding,* vol. 55, no. 3, pp. 329-338, 1992.

[4] J. Bruck, J. Gao, and A.A. Jiang, "MAP: Medial Axis Based Geometric Routing in Sensor Networks," *Proc. ACM MobiCom,* 2005.

[5] J. Bruck, J. Gao, and A.A. Jiang, "MAP: Medial Axis Based Geometric Routing in Sensor Networks," *Wireless Networks,* vol. 13, no. 6, pp. 835-853, 2007.

[6] W. Choi, K. Lam, and W. Siu, "Extraction of the Euclidean Skeleton Based on a Connectivity Criterion," *Pattern Recognition,* vol. 36, no. 3, pp. 721-729, 2003.

[7] M. Coupriea, D. Coeurjollyb, and R. Zrourc, "Discrete Bisector Function and Euclidean Skeleton in 2D and 3D," *Image and Vision Computing,* vol. 25, no. 10, pp. 1543-1556, 2007.

[8] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang, "Glider: Gradient Landmark-Based Distributed Routing for Sensor Networks," *Proc. IEEE INFOCOM,* 2005.

[9] S.P. Fekete, A. Kroller, D. Pfisterer, S. Fischer, and C. Buschmann, "Neighborhood-Based Topology Recognition in Sensor Networks," *Proc. First Int'l Workshop Algorithmic Aspects of Wireless Sensor Networks,* 2004.

[10] S. Ganeriwal, P. Kumar, and M.B. Srivastava, "Timing-Sync Protocol for Sensor Networks," *Proc. Int'l Conf. Embedded Networked Sensor Systems,* 2003.

[11] Y. Ge and J.M. Fitzpatrick, "On the Generation of Skeletons from Discrete Euclidean Distance Maps," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 11, pp. 1055-1066, Nov. 1996.

[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 and Distributed Systems,* vol. 22, no. 6, pp. 1064-1071, June 2011.

[13] H. Jiang, W. Liu, D. Wang, C. Tian, X. Bai, X. Liu, and W. Liu, "CASE: Connectivity-Based Skeleton Extraction in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2009.

[14] 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 and Distributed Systems,* vol. 21, no. 5, pp. 710-721, May 2010.

[15] 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," *Proc. IEEE 32nd Int'l Conf. Distributed Computing Systems,* 2012.

[16] W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang, "Approximate Convex Decomposition Based Localization in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* 2012.

[17] C. Niblack, P. Gibbons, and D. Capson, "Generating Skeletons and Centerlines from the Distance Transform," *CVGIP: Graphical Models and Image Processing,* vol. 54, no. 5, pp. 420-437, 1992.

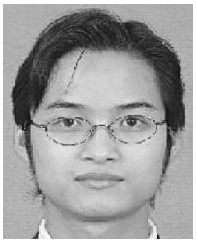
[18] S. Schaefer and C. Yuksel, "Example-Based Skeleton Extraction," *Proc. Eurographics Symp. Geometry Processing,* 2007.

[19] H. Talbot and L. Vincent, "Euclidean Skeletons and Conditional Bisectors," *Proc. SPIE Visual Comm. and Image Processing (VCIP),* vol. 1818, pp. 862-876, 1992.

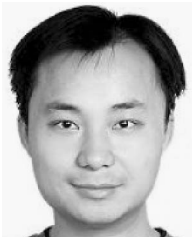
[20] G. Tan, M. Bertier, and A.M. Kermarrec, "Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing," *Proc. IEEE INFOCOM,* 2009.

[21] L. Vincent, "Efficient Computation of Various Types of Skeletons," *Proc. SPIE Medical Imaging V,* vol. 1445, pp. 297-311, 1991.

[22] X. Zhu, R. Sarkar, and J. Gao, "Shape Segmentation and Applications in Sensor Networks," *Proc. IEEE INFOCOM,* 2007.



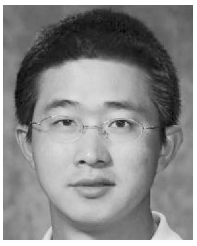
Wenping Liu received the BS and MS degrees from the Huazhong University of Science and Technology, China. After that, he joined the faculty of the Hubei University of Economics. He is currently working toward the PhD degree in the Department of Electronics and Information Engineering, Huazhong University of Science and Technology. His research interests include statistical modeling and wireless sensor networks. He is a student member of the IEEE.



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

Xiang Bai received the BS degree in electronics and information engineering and the MS degree in electronics and information engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2003 and in 2005, respectively. From January 2006 to May 2007, he worked in the Department of Computer Science and Information, Temple University. He is currently working toward the PhD degree at HUST and has recently joined the University of California, Los Angeles, as a joint PhD student. His research interests include computer graphics, computer vision, and pattern recognition. He is a member of the IEEE.

Guang Tan received the BS degree from the Chongqing University of Posts and Telecommunications, China, in 1999, the MS degree from the Huazhong University of Science and Technology, China, in 2002, and the PhD degree in computer science from the University of Warwick, United Kingdom, in 2007. He is currently an associate researcher at the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China, where he works in the area of distributed systems and networks. From 2007 to 2010, he was a postdoctoral researcher at INRIA-Rennes, France. He is a member of the IEEE.



Chonggang Wang received the PhD degree in computer science from the Beijing University of Posts and Telecommunications. He has conducted research with the NEC Laboratories America, AT&T Labs Research, University of Arkansas, and Hong Kong University of Science and Technology. His research interests include future Internet, machine-to-machine communications, and cognitive and wireless networks. He is a senior member of the IEEE.

Wenyu Liu received the BS degree in computer science from Tsinghua University, Beijing, China, in 1986, and the diploma and doctoral degrees, both in electronics and information engineering, from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1991 and 2001, respectively. He is currently a professor and an associate chairman in the Department of Electronics and Information Engineering, HUST. His current research areas include computer graphics, multimedia information processing, and computer vision. He is a member of the IEEE.

Kechao Cai received the BS degree from HUST in 2010, where he is currently working toward the MS degree. His current research area is wireless sensor networks.