Boundary Reconstruction for Wireless Sensor Networks

Sreeram SIVADASAN[1,2], Nagarajan GOVINDAN [1], Amal Dev PARAKKAT[2]
[1]Sathyabama Institute of Science & Technology, Chennai, India
[2]LTCI - Telecom Paris, Institut Polytechnique de Paris, France

Corresponding author: Sreeram Sivadasan (e-mail: sreeramsivadas@gmail.com).

ABSTRACT Wireless Sensor Networks (WSNs) are critical for various applications ranging from environment monitoring to industrial monitoring. The varying and continuously growing interest in this field demands an understanding of the sensor node distribution to ensure robustness and to improve resource utilization for data processing and decision making. In this paper, we focus on reconstructing the boundaries of a wireless sensor network, which also has a lot of applications in IoT and Robotics. As these sensor node locations can be considered as a set of points in the 2D plane, boundary detection of a WSN can be related to classical shape reconstruction problem in Computational Geometry. In this paper, we extend a simple and generic strategy for hole detection to a geometric solution for boundary/shape reconstruction. Furthermore, we introduce a simple and controllable heuristic algorithm to patch the coverage holes identified by our boundary reconstruction algorithm. Not only does this study improve the reliability of WSNs, but it also provides a useful tool for the extensive domain of computational geometry and shape analysis. Our different experiments show that the proposed reconstruction algorithm outperforms the existing state-of-the-art methods, and hole patching gives a simple and controllable solution for mobile node placement.

INDEX TERMS Shape reconstruction, Delaunay Triangulation, Coverage Hole Detection, Dynamic Node Placement, Coverage Optimization

I. INTRODUCTION

WIRELESS sensor networks (WSNs) have become an important domain with applications in different fields, including environmental monitoring, industrial automation, and smart cities. These networks are typically composed of spatially distributed sensors that collaborate with each other to acquire and transmit required data to the user. The performance of a WSN is typically determined by its capacity to cover a designated region. However, the random nature of node deployment, natural/physical barriers present in the designated region, and malfunctioning of sensors often create coverage holes in the network, which further deteriorates the network's functionality.

We define coverage holes as the areas within the designated intended region that do not receive any monitoring from any sensor. Such holes can affect the overall performance and reliability of the network as it lead to missing events and ends up in inaccurate data collection. The consequence of this limitation becomes severe in applications such as disaster management and security surveillance. So, being able to efficiently identify coverage holes and patch them (placing additional mobile nodes to avoid coverage holes) is important. And to find such coverage holes, it is important to know the boundaries defined by the sensor nodes. Current solutions to identify boundaries and patch coverage holes typically rely on geometric, statistical or topological analysis. Although these approaches have shown some success, they often struggle with computational complexity and require global network information. Different from existing solutions, this work primarily focuses on the precision and controllability aspects of boundary detection and patching problems, respectively.

The location of the sensor nodes can be considered as a set of 2D points where the problem of finding their boundary is analogous to a solution of the shape reconstruction problem. From the perspective of computational geometry, given a set of planar points, the shape reconstruction problem asks for a shape that best approximates the points [1]. Although most algorithms in this direction focus on reconstructing exterior boundaries alone, there are a few works that are capable of reconstructing interior (holes) and exterior boundaries. However, almost all the algorithms in this direction are created and evaluated from a human perception point of view with applications in Computer Graphics and related fields. Unlike this line of work (but still creates comparable or better re-

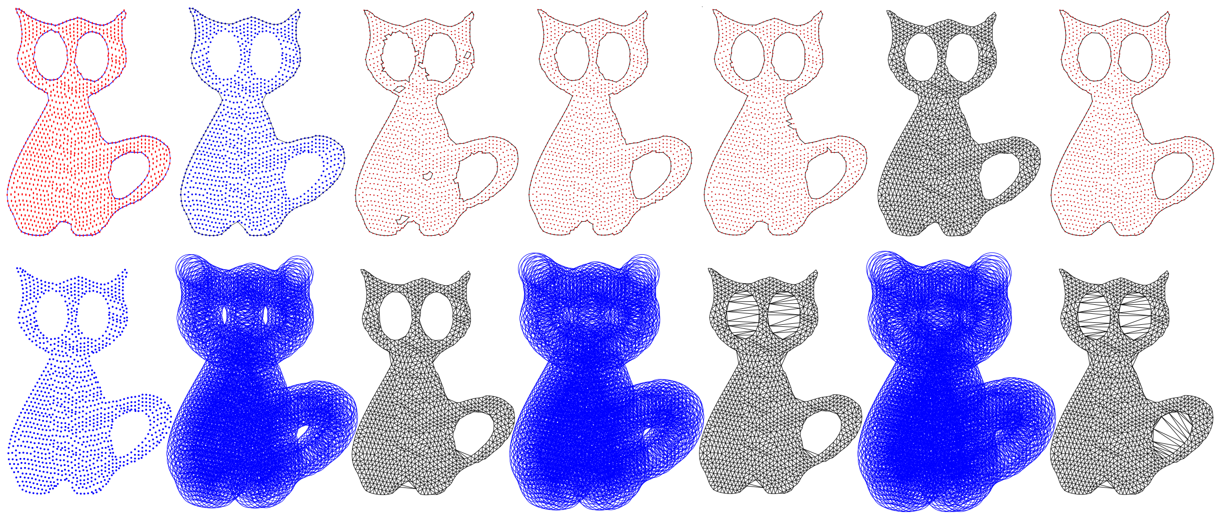


FIGURE 1. Top row: Results of various shape reconstruction algorithms. Left to right: α-Shape, Simple shape, EC-Shape, CT-Shape, Discern, Our result, our result after extracting the boundary edges. Bottom row: The input point set and the effect of varying the parameter corresponding to sensor radius.

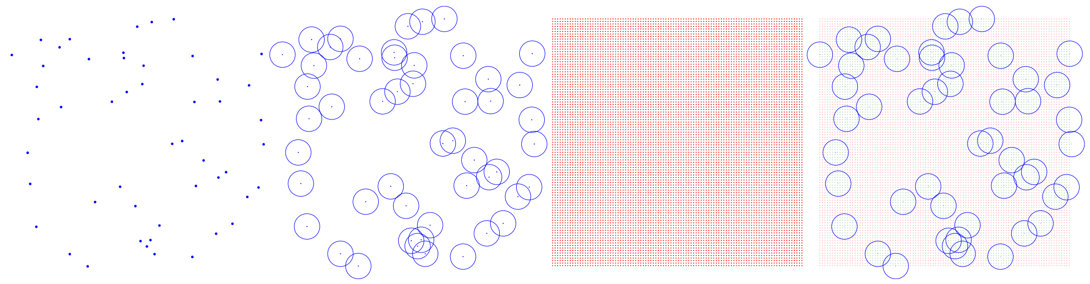


FIGURE 2. Hole detection strategy. Left to right: Input points, coverage areas drawn as circles around each point, grid points, and the covered and uncovered points in green and red colors.

sults), the proposed method introduces a powerful and generic algorithm for shape reconstruction (both exterior and interior boundaries) with a focus on WSNs. In other words, unlike the literature, the proposed method takes the sensing radius into account when identifying the shape and hole. Having such a reconstruction algorithm is not only valuable for the shape modeling community but also helps analyze the node distribution and understand the underlying topology in WSNs. Figure 1 compares the output of our algorithm with state-of-the-art methods in shape reconstruction.

In summary, our main technical contribution is a simple hole detection strategy, which is extended to create a powerful and more generic solution to the boundary reconstruction problem. In addition, we introduce a simple and controllable geometric solution to patch coverage holes.

II. RELATED WORK

In this section, we summarize the important works relevant to the proposed method in two categories: in the context of WSN and in the context of shape reconstruction. Although we introduce a simple preliminary solution for hole patching, that is not our primary objective, and therefore, we direct interested readers to a recent survey by Singh et al. [2] to learn more about the literature on hole patching.

Boundary reconstruction in WSN: Identifying boundaries and coverage holes and using them for efficient routing is an important problem in WSN [3]. The early works in this direction used geometric [4] and topological [5] constraints to capture boundaries - a detailed survey on these techniques can be seen in Chen et al. [6]. Though simple, these methods often struggle with exact boundary extraction and non-uniform sensor distribution. These methods are later improved by using common geometric structures like Voronoi diagram [7], minimum spanning tree (MST) [8], and Delaunay triangulation [9]. Though the usage of Delaunay structure (along with Voronoi and MST) has been used a lot in the literature, our algorithm shows superior performance in detecting the boundaries - which also makes it an alternative starting point for algorithms like TELPAC [10]. A few other works in this direction worth mentioning includes Blind Swarm-based Approach [11], Two Anchor (TA) & Cyclic Segment Sequence (CSS) Algorithm [12], Hop-based Approach [13], Clustering-based Approach [14] and Distributed Sector Cover Scanning (DSCS) & Directional Walk (DW) [15]. Unfortunately, these algorithms require additional information or have constraints on possible boundaries. Recently, researchers have been focusing on extending such techniques for specific applications [16]. Recent detailed surveys and comparisons can be seen in

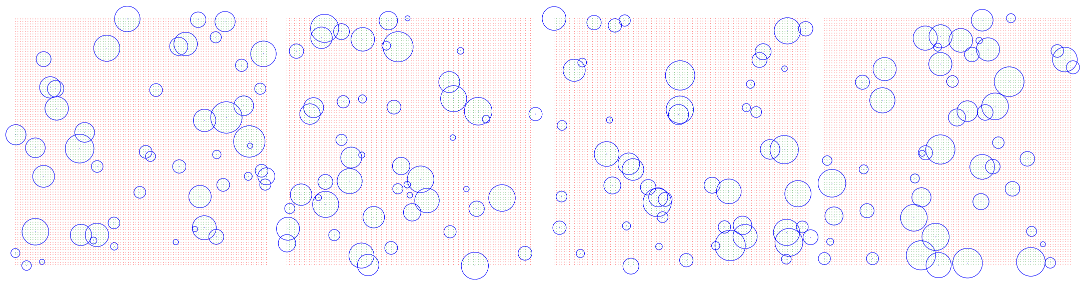


FIGURE 3. Results generated by our hole detection strategy for each node having a different random sensing range.

Kundu et al. [17] and Sharma et al. [18] [19].

Shape reconstruction: From a geometry perspective, shape reconstruction algorithms can be classified into two: curve reconstruction and shape reconstruction from dot patterns. We limit our literature to shape reconstruction from dot patterns as sensor nodes in WSN can be considered as a dot pattern. Interested readers can read the recent survey [20] to know more about the literature on curve reconstruction. One of the initial works in the direction of shape reconstruction from dot patterns is by Edelsbrunner et al. [21], called α-shape. As the main motivation of α-shape is the generalization of the convex hull, it was further refined by Melkemi et al. [22] to introduce A-shape. Following these works, a sculpting strategy was introduced in χ-shape [23] (and a non-Delaunay variant called *simple*-shape [24]) and further improved by Peethambaran et al. [25]. As sculpting strategies typically need a separate condition to detect holes, and to improve the sculpting criteria, an empty circle approach was later introduced by Methirumangalath et al. [1], [26]. Another work in this direction, CT-shape [27], uses the local Delaunay properties to decide whether an edge should be retained or not, which was then extended to create a more generic approach called *Discern* [28]. A very recent work in this direction called *Prudent Carving* [29] uses intelligent ways to prune and extract the required details starting from a convex hull. A detailed explanation and comparisons of works in this direction can be seen in recent surveys [30], [31]. Unlike all these algorithms, our method varies in different ways. The main difference is the improved quality of the results. Second, our method uses a single strategy for inner and outer boundary detection. Third, the proposed method gives more control to the user and is not highly sensitive to parameter tuning (making it easy to tune). Also, this parameter is readily available in a WSN as it corresponds to the sensing radius.

III. HOLE DETECTION & RECONSTRUCTION

In this section, we first describe our simple strategy for hole detection. Given a set of sensor node locations and its sensing radius, our objective is to identify the regions in the area of interest that cannot be covered by any of the sensors. We then extend this strategy to introduce a simple yet powerful solution for boundary/shape reconstruction.

Let us first start with a few preliminaries in WSN:

- Sensor node: The basic building unit of a WSN, a device that collects and transmits the required data. A node N_i is represented by its location (x_i, y_i).
- Sensing radius: A sending radius r_i of a node N_i is the radius within which N_i can detect events or monitor the surroundings.
- Coverage area of a node: The coverage area of a sensor node N_i with sensing radius r_i is defined as a disk D_i with radius r_i centered at N_i. In simple words: $D_i = \{(x, y) \in R^2 \mid (x - x_i)^2 + (y - y_i)^2 \leq r_i^2\}$
- Coverage area of a network: The coverage area of a network is the union of the coverage areas of all sensor nodes, defined as $C = \bigcup_{i=1}^{N} D_i$

Given a monitoring region (area of interest, typically defined as a rectangle) M and the coverage area of a network C, a coverage hole H is defined as $H = M \backslash C$.

As shown in Figure 2, our strategy starts with discretizing the monitoring region using a grid. The grid points are then individually checked to know whether they lie inside the coverage area of any of the sensor nodes. For simplicity and efficiency, we implement it a little differently. Assuming that the sensing radius is constant, we first compute the Delaunay triangulation of the input point set, defined as:

- Delaunay Triangulation - A Delaunay triangulation for a set of points P in a plane is a triangulation $DT(P)$ such that the circumcircle of any triangle in $DT(P)$ is empty of any other point in P.

To check whether a point on the grid gp_i is covered by a sensor n_i with radius r, we insert each g_i into the triangulation and compute the distance to its connected vertices. As nearest neighbors are, by definition, neighbors in Delaunay triangulation, we can tag g_i as covered if $||gp_i - n_i|| \leq r$ for some neighbor n_i. After marking the covered nodes for all the sensor nodes, we extract and return the uncovered nodes as points outside the expected boundary. The g_i is then removed from the Delaunay triangulation. Unfortunately, this efficient method would not work if the nodes have varying sensing radii. In such cases, we employ an alternative technique where we begin at the grid point nearest to each sensor node's location. We then iteratively expand outwards, covering all neighboring points within the node's sensing radius.

As the identified uncovered nodes are discrete points, we then create a graph $G = (V, E)$ such that the vertices are the grid points and edges connecting neighboring (horizontally, vertically and diagonally) vertices. The region covered by

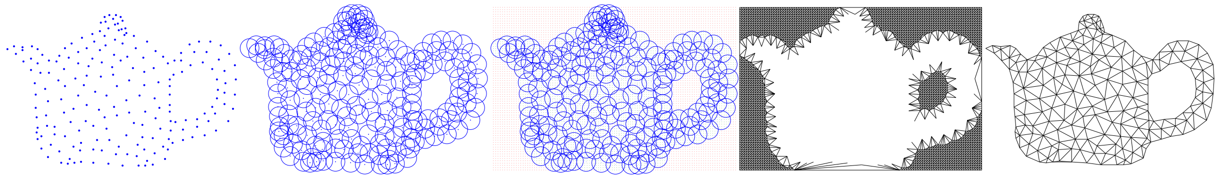
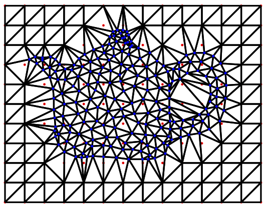


FIGURE 4. Overview of our algorithm. Left to right: Input point set, circles with sensor radius around each point, covered (green) and uncovered (red) points in the grid, edges in the *Remove* set, Our result.

this graph can be considered as part of the hole regions. This strategy can directly be applied for nodes with varying sensing ranges, as shown in Figure 3 (which is common in real-life WSNs). Although this simple strategy seems trivial, we build our boundary/shape reconstruction and hole patching algorithms on it.

Next, we extend this strategy to the boundary/shape reconstruction. The overall algorithm is shown in Algorithm 1. The process starts with creating a grid of resolution δ. In our implementation, we fixed the thickness of the grid as $\frac{\text{AABB width}}{100}$ if the width of the axis-aligned bounding box (AABB) of the point set is greater than its height; otherwise, $\frac{\text{AABB height}}{100}$ (in the remaining section, we use the parameter δ to represent a factor by which the AABB width/height is divided, i.e., $\frac{\text{AABB width/height}}{\delta}$). In the next step, all the grid points that are covered by any of the sensor nodes are marked as covered. Subsequently, we compute the Delaunay triangulation DT_1 of the union of uncovered grid points and input points P.



As shown in the inset, the desired boundary edges are a subset of this Delaunay triangulation DT_1. Our main observation is that, thanks to the properties of Delaunay triangulation, with appropriate sampling and sensing radius, no two grid points will have an edge between them that crosses a desired boundary edge. Based on this observation, we first identify all the edges of DT_1 that are not between two points in P and store them in a set *Remove*. The Delaunay triangulation of P alone, DT_2, is then computed and pruned further to reconstruct the final shape by removing all the edges in DT_2 that intersect with edges in *Remove*. Please note that this idea is inspired by the Crust algorithm [32], where Voronoi vertices are taken instead of the uncovered grid points; however, these Voronoi vertices are meaningful only when the input is a boundary sampled curve.

Algorithm 1: Boundary Reconstruction Algorithm

```
Data: Input points $P$, Sensor radius $r$
Create a square grid $G$ of resolution $\delta$ around $P$
$Covered = \phi$, $Remove = \phi$, $Shape = \phi$
for each input point $p_i$ do
    for each $g_j \in G \mid ||p_i - g_j|| \leq r$ do
        $Covered = Covered \bigcup g_j$
$Uncovered = G - Covered$
$DT_1$ = Delaunay triangulation of $Uncovered \bigcup P$
for each edge $e_{ij} \in DT_1$ do
    Extract the endpoints $p_i$ and $p_j$ of $e_{ij}$
    if $p_i \notin P$ or $p_j \notin P$ then
        $Remove = Remove \bigcup e_{ij}$
$DT_2$ = Delaunay triangulation of $P$
for each edge $e_{ij} \in DT_2$ do
    if $e_{ij}$ do not intersect with $e_{ab}, \forall e_{ab} \in Remove$ then
        $Shape = Shape \bigcup e_{ij}$
Return $Shape$
```

The overall steps in the algorithm are shown in Figure 4. As can be seen, the resultant *Shape* captures the outer and inner boundaries represented by the point set. While deleting edges from DT_2, we can remember all the retained triangles whose one neighbor is removed and use their interface edge to extract the boundary edges alone. Figure 5 shows various challenging results generated using our method. As can be seen, our simple method could capture complex shapes, including multiple components, outer boundaries, and multiple and complex holes (inputs taken from existing methods).

The resulting shape depends on the input parameter r corresponding to the sensing radius, and as can be seen in Figure 6, r has a huge effect on the resulting shape. Though this relation is important for the application in WSN, it might not be important for visual applications as humans might not

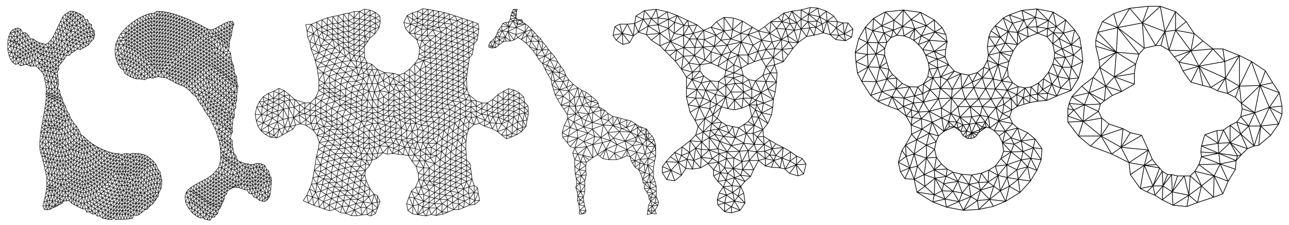


FIGURE 5. Various challenging shapes reconstructed using our methods (the input point sets are taken from existing papers).

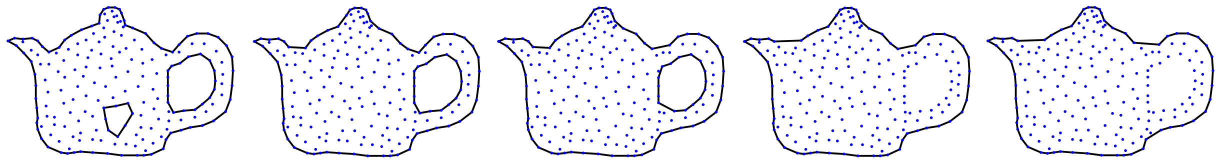


FIGURE 6. Effect of varying the radius parameter corresponding to the sensing radius. Left to right: $r = 10, 15, 20, 25, 30$.

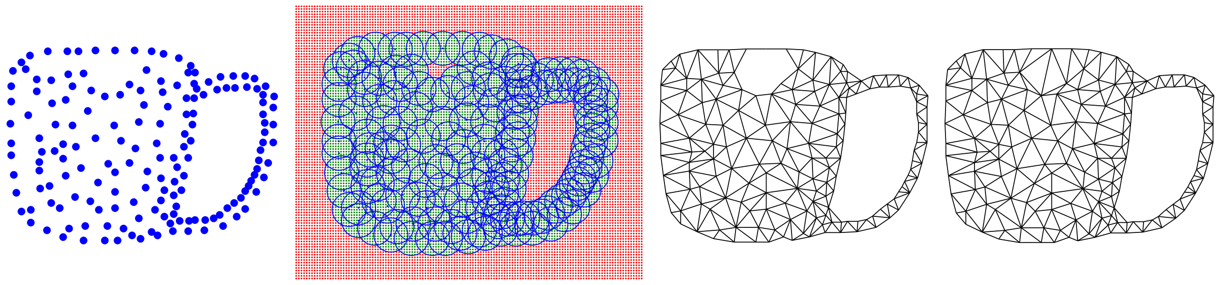


FIGURE 7. Left to right: Input points, covered and uncovered grid points, the result of our method, result of our method with a constraint on the number of points in a region.

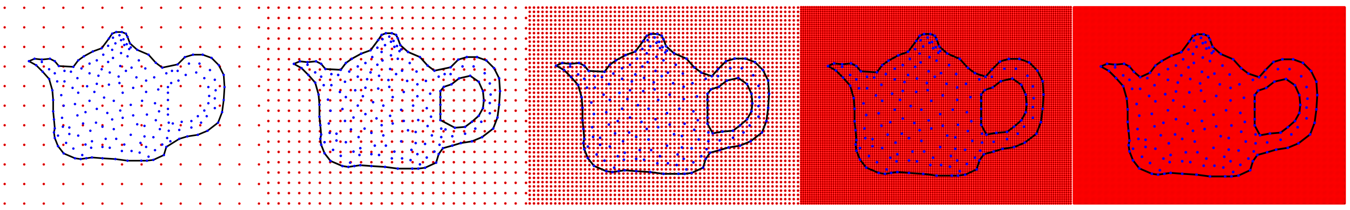


FIGURE 8. Varying grid resolutions. Left to right: $\delta = 10$, $\delta = 20$, $\delta = 50$, $\delta = 100$, $\delta = 200$, taking 0.054s, 0.131s, 1.125s, 11s, 150s respectively.

perceive it as a hole. Figure 7 shows such an example where the sparse and random sampling creates an unwanted hole. It is important to note that for applications where such holes are unnecessary, we can easily remove them based on the number of uncovered points within a connected component, as illustrated. As shown in Figure 8, though we fixed δ as 100, the resolution of the grid has an impact on the reconstructed shape and the overall computation time.

IV. HOLE PATCHING

In applications like WSN, rectifying the holes detected from the boundaries is as important as identifying them. The typical solution for so-called hole patching is to place mobile nodes in the hole regions to make sure that these regions are covered. A simple solution for such a method would be to densely distribute mobile nodes within the hole regions, which, unfortunately, is not a cost-effective approach. So, our objective is to intelligently place mobile nodes in such a way that the coverage holes are rectified with a minimal number of sensors. As explained in [33], finding optimal positions to place these mobile nodes is an NP-hard problem. So, we introduce a heuristic algorithm to do this mobile node placement. As will be explained later, our main objective is to introduce a priority to characterize the efficiency of placing mobile nodes at a particular position and to have more control over the mobile node placement strategy.

As shown in Algorithm 2, our method starts with the pruned graph structure our hole identification process returns - a graph connecting neighbor uncovered grid points. Our first step is to analyze the connected components of this pruned graph. Here, a connected component is defined as follows:

- Connected Component: A connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by a path.

This can be seen as a way to cluster the hole regions. We then individually process the connected components. Initially, a minimum-oriented bounding box (MOBB) is computed using the $Minimum\_Oriented\_Bounding\_Box(C)$ function. We used the Rotating Calipers Algorithm [34] to compute this MOBB. In simple words, the MOBB is the bounding box with the smallest area that encloses a given set of points. We then pack squares with diagonal length r (the largest square that could fit inside the coverage area of the mobile node) into the MOBB using the $Decompose\_to\_Square()$ function. As it may not always be possible to perfectly fit squares inside an MOBB, we fit as many squares as possible inside the MOBB and then distribute the remaining space evenly between them to create overlapping squares. To achieve this, we first rotate the MOBB to align it with the coordinate axes and overlay it with a grid of squares, each with a diagonal length of r. All squares intersecting with the MOBB are selected and adjusted to fit completely within its boundaries. Finally, the squares and the MOBB are transformed back to their original orientation. Please note that the resulting squares align with

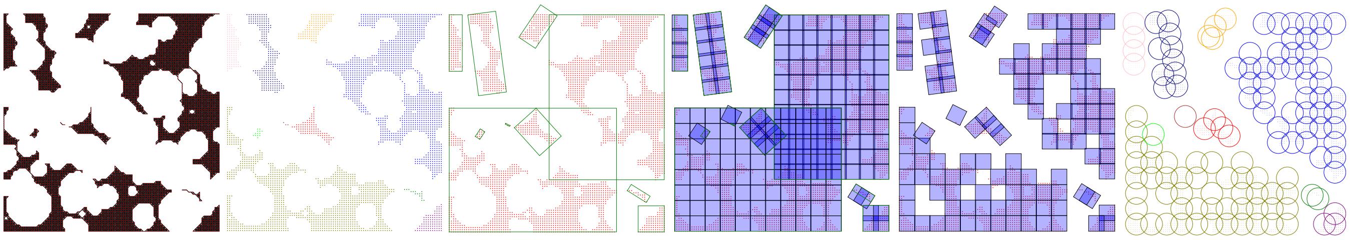


FIGURE 9. Left to right: Pruned graph, Clusters after connected component analysis, MOBB of each cluster, MOBB decomposed to squares, Squares identified by our algorithm to place mobile nodes, Final result.

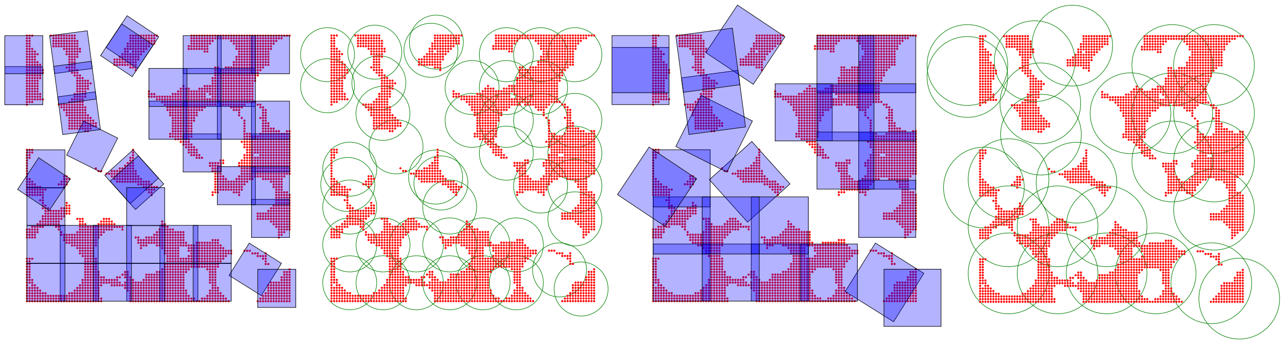


FIGURE 10. Squares identified by our algorithm along with our final result for different mobile node sensing radius.

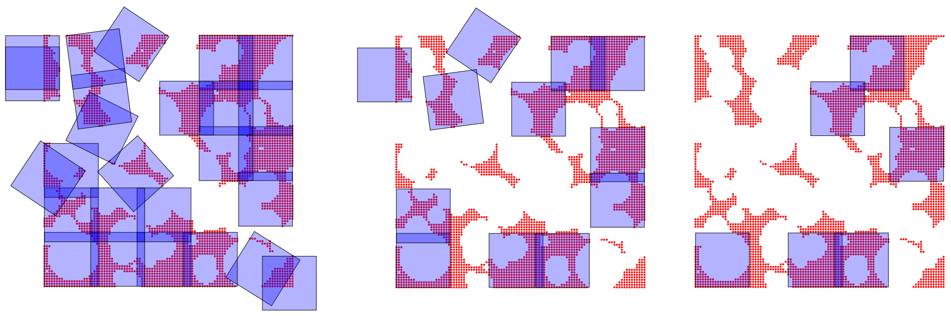


FIGURE 11. Effect of limiting the number of mobile nodes based on the amount of grid points it covers.

the MOBB orientation.

After the square fitting, we assign a priority to each of these squares based on the number of uncovered grid points that lie inside the circle enclosing the square. Based on this priority, the squares are inserted into a priority queue (implemented using a heap data structure). The squares are then taken in order from this priority queue, and mobile nodes are placed in its center. Since placing a mobile node will cover a set of previously uncovered grid points, we update the priority of each square by considering only the uncovered points after placing this new mobile node.

The overall pipeline for hole patching is shown in Figure 9. As shown in Figure 10, this simple hole patching algorithm can effectively cover various complex coverage holes and can be modified depending on the sensing radius of mobile nodes. Thanks to clusters and our priority-based mobile node placement strategy, the user can have increased control over the node placement (as will be explained in Section V).

It is worth noting that we can also limit the placement of

Algorithm 2: Hole Patching Algorithm

```
Data: Pruned graph $G$ connecting neighbor uncovered
      grid points, Mobile node radius $r$
Do a Connected Component Analysis on $G$
$PQ = \phi$
for each connected component $C$ do
    $MOBB = Minimum\_Oriented\_Bounding\_Box(C)$
    $S = Decompose\_to\_Square(MOBB, r)$
    for each square $s \in S$ do
        Compute priority $pri$ for $s$
        $PQ.Enqueue(s, pri)$
while $PQ \neq \phi$ do
    $s = PQ.Dequeue()$
    Place a mobile node in the center of $s$
    Recompute the priorities and update the PQ
```

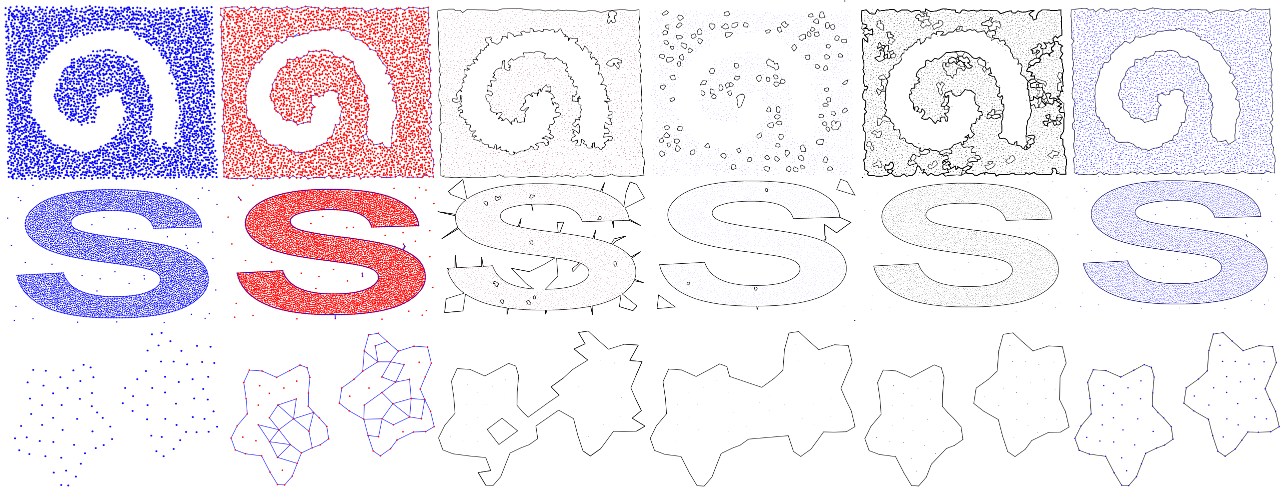


FIGURE 12. Comparison with various State-of-the-art algorithms. Top to bottom: Complex shape with random point distribution, point set with outliers, and multiple sparse components. Left to right: Input points, Results of α-shape [21], EC-shape [1], [26], CT-shape [27], Discern [28], and our results.

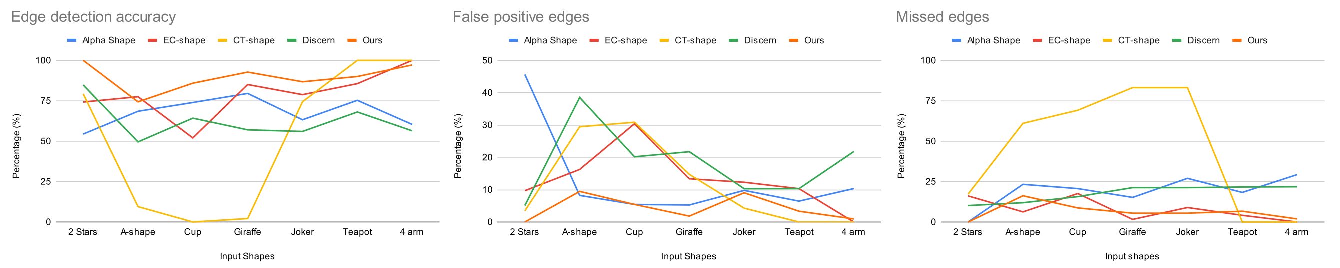


FIGURE 13. Accuracy evaluation with various State-of-the-art algorithms for boundary/shape reconstruction. Left to right: edge detection accuracy (higher better), percentage of false positive edges (lower better), and percentage of missed edges (lower better).

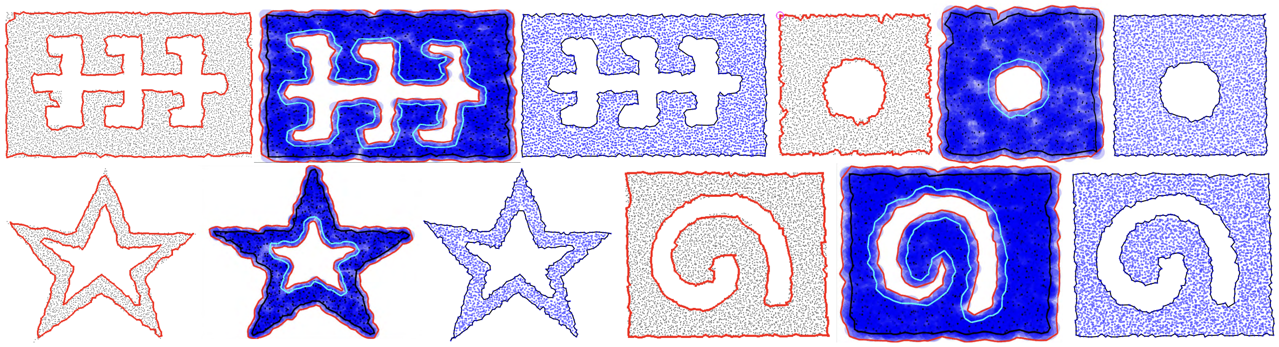


FIGURE 14. Similar complex hole pattern reconstructed by a state-of-the-art method [35] and a recent method [2] along with our results.

mobile sensor nodes by considering the number of previously uncovered grid points they can cover. Figure 11 shows a sample case where, from left to right, it covers at least one point, 100 points, and 200 points. As can be seen, this strategy will help us better place the available mobile nodes in a reasonable way.

Mobile nodes with varying radii: Our algorithm can be easily extended to handle mobile nodes with varying radii. Thanks to our generic framework, we can sort all sensors in descending order of their sensing radii and, in each iteration, decompose the MOBB based on the radius and place appropriate sensor nodes. We can iteratively do this until all the sensors are placed.

V. RESULTS & DISCUSSION

Implementation details: We implemented our algorithm in C++ using the CGAL library. All the experiments were conducted on an Apple M2 Max 32GB Mac studio, and the results were written to an SVG file for better visualization. Our boundary detection algorithm took around 6 seconds for a 366x298 grid, with 0.35 seconds to run our algorithm and the remaining to check the size of the hole for clustering.

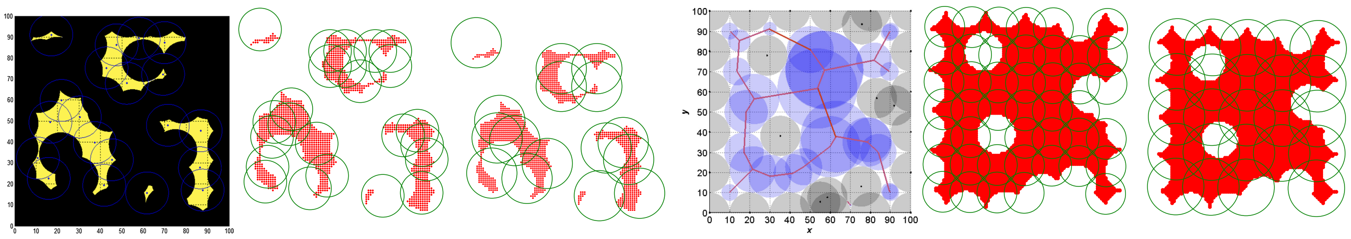


FIGURE 15. Sample visual comparison of our hole patching with 3HA [36] and Tree-based approach [8].

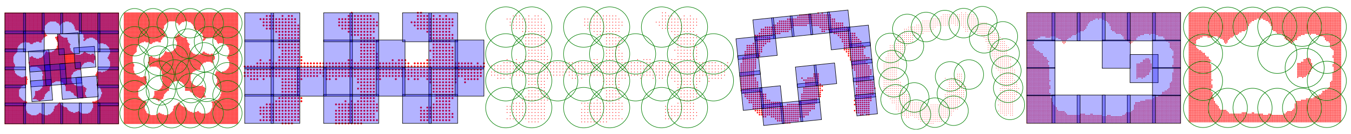


FIGURE 16. Results of our hole patching on various shapes.

Though the clustering part could be improved using a spatial data structure and parallel processing, we keep it for future work. Also, the grid resolution has an effect on the run time and the result, and by reducing it to 128x94, the runtime is reduced to 0.7 seconds, with 0.1 seconds for running our algorithm and 0.6 seconds for clustering the hole region. Similarly, the algorithm for hole patching is influenced by both grid resolution and sensing radius. To illustrate, in the experimental setup shown in Figure 10, where 50 sensor nodes with varying radii were deployed on a 100x100 grid, the process took 0.4 seconds. Upon increasing the sensing radius, the runtime decreased to 0.12 seconds.

Additional functionalities: As explained earlier, the size of the grid and/or the sensing radius are used to tune our boundary detection algorithm. Also, to avoid undesired small holes, the result could be pruned based on the size of the cluster after the connected component analysis. Similarly, not only does our simple hole patching heuristics give a nice placement of mobile nodes, but it also has a few advantages which are not limited to:

Handling arbitrary hole geometry: As in many existing hole patching algorithms, we do not place any restrictions on the hole geometry.

Limiting the number of mobile nodes: As mobile nodes are usually costly [37], we can have a limit on the number of available mobile nodes. Thanks to our algorithm, priority-based deployment helps us place these available mobile nodes in the best possible locations.

Associating importance: Again, as the priority gives the amount of uncovered grid points, we can associate importance to each of these potential positions. It is possible to define when we have to consider a position important (for example, if the placement of a new node covers at least k number of uncovered grid points).

Please note that, compared to literature, our method offers several key advantages. First, unlike [36], no assumption is made on the node distribution or hole topology, making our method more flexible. Different from methods like [38], [39], our method is scalable and easily adaptable to different topologies. Although the precision of hole patching depends on the grid resolution that we used for hole detection, we can ensure full coverage by reducing the square size to account for this imprecision, making our method differ from others that do not guarantee full coverage [40]. Finally, our approach has comparatively lower computational complexity and greater flexibility in handling various real-life scenarios compared to methods discussed in [41]. A few more results of our patching on complicated holes are shown in Figure 16.

Complexity: Let the number of grid points and input points be $|G|$ and $|P|$, respectively, and define $k_1 = |G|.|P|$ and $k_2 = |G|+|P|$. In the current naive implementation for the reconstruction of the boundary, the overall complexity consists of several steps: it takes $O(k_1)$ to find the uncovered points, $O(k_2 \log k_2)$ to compute the Delaunay triangulation DT_1, and $O(k_2)$ to compute the removal set. Furthermore, it takes $O(|P| \log |P|)$ to compute DT_2, and finally $O(|P|.k_2)$ to compute the final reconstructed shape. The total overall complexity is thus $O(k_1{+}k_2 \log k_2{+}k_2{+}|P| \log |P|{+}|P|.k_2)$, which simplifies to $O(|G|.|P| + (|G| + |P|)\log(|G| + |P|) + |P| \log |P| + |P|.(|G|{+}|P|))$. Similarly, the complexity of the hole patching algorithm is dominated by the priority computation for each square. This requires $O(|G|)$ connected components, each containing $O\left(\frac{wh}{r^2}\right)$ squares, where w and h are the width and height of the MOBB, and r is the radius of the mobile node. Consequently, a naive implementation of the priority computation results in an overall complexity of $O\left(|G|^2 \cdot \frac{wh}{r^2}\right)$.

Comparison: As our primary focus is on hole detection, we qualitatively and quantitatively compare our results with various state-of-the-art shape reconstruction techniques in Figures 12 and 13, respectively. To summarize:

α-shape: Similar to ours, α-shape could generate a visually good result for random point distributions and hole shape (the spiral hole). However, the results start deteriorating once outlier points are introduced. Also, though it could capture multiple components, if the points are sparse

and non-uniformly distributed, α-shape captures unnecessary details. On average, the α-shape resulted in an edge detection accuracy of approximately 68%, a false positive rate of 13% and 19% of missed edges.

EC-shape: As shown, it could not distinguish outlier points and multiple components. Also, it identified many pseudo holes in completely random distribution. The edge detection accuracy, false positive edges and missed edges, on average, are 79%, 13% and 7%.

CT-shape: It had difficulty in identifying boundaries from a random distribution but could withstand outliers up to a level. Once the multiple components are sparse and close to each other, they cannot be reconstructed by CT-shape. The average accuracies of results generated by the CT-shape are 52%, 11% and 44%, respectively.

Discern: It could give good results for outliers and multiple components (though a few points are missed in our sparse multiple-component example). But, similar to EC-shape, it ended up in identifying many pseudo holes inside a randomly distributed point set. The average accuracies are 62%, 18%, and 17%, respectively.

Our result: Expect from a wrong edge in the outlier case, our algorithm gave better results in the case of sparse multiple components and random distribution. Accuracy-wise, our method outperformed other algorithms by giving average accuracies of 89%, 4% and 2%.

Please note that from an accuracy perspective, we aim to increase edge detection accuracy (higher is better) while decreasing false positive edges and missed edges (lower is better). In addition, we present a visual comparison of our method with two hole detection works from a WSN perspective. As demonstrated in Figure 14, our results either outperform or are comparable to other techniques.

Figure 15 shows another visual comparison of the inputs that we recreated from 3HA and [8]. Our method gave a good placement of the mobile nodes close to 3HA. And [8] assumes a varying fixed sensing radius for mobile nodes (which is not practical) and also results in many uncovered regions.

Limitations and Future Work: Our main focus was on the aspect of boundary/shape reconstruction, which, as demonstrated in an extensive comparison, surpasses state-of-the-art methods. However, our proposed method has some limitations. First, the reconstruction algorithm relies on r, which, in the context of WSN, can be predetermined from the sensing radii of the nodes. While we used a fixed value of δ for all our experiments, the user may need to manually tune its value to get the desired result. Second, although our simple hole patching strategy gives comparable mobile node placements, starting with an MOBB might not always be optimal, especially in dynamic WSN environments.

In future, it will be interesting to introduce an optimization on the mobile node positions that starts from the initial grid position and allows local movement of the mobile node positions to improve the overall coverage, as the current results are preliminary. Also, exploring various alternative priority assignment strategies for placing the mobile nodes will be interesting. Though we believe it would be possible to assume a sampling model (for example, r-sampling as in [25]), we keep it for the future as our current focus was more on the practicality of our method. It is also important to note that the reconstruction accuracy is highly dependent on the grid resolution. Therefore, it is crucial to relate the sampling model to the local features of the hole and then to the grid resolution, ensuring that the grid resolution is appropriately chosen to capture these details. Another natural future work would be to extend the reconstruction algorithm to 3D.

VI. CONCLUSION

In this paper, we extend a simple strategy for reconstructing boundaries in the wireless sensor network. The algorithm is easy to implement, requires only a single parameter (which is indeed predefined for sensor nodes), and is experimentally shown to give good results. As shown in the extensive comparison, the proposed boundary reconstruction outperforms all existing shape reconstruction methods, especially in challenging cases like random distribution, sparse sampling, and multiple components. In contrast to other strategies widely used in WSN for boundary reconstruction, our method does not make any assumptions about the type/shape of the hole or node distribution and has less computational overhead. We believe this method will benefit researchers working on computational geometry, WSNs and related fields, as it will facilitate further analysis of the region, hole, and node distribution in WSNs. Additionally, though not the primary objective, we have also presented a simple and easy-to-implement heuristic for patching coverage holes.

REFERENCES

[1] S. Methirumangalath, A. D. Parakkat, and R. Muthuganapathy, "A unified approach towards reconstruction of a planar point set," *Computers & Graphics*, vol. 51, pp. 90–97, 2015.
[2] P. Singh and Y.-C. Chen, "Sensing coverage hole identification and coverage hole healing methods for wireless sensor networks," *Wireless Networks*, vol. 26, no. 3, pp. 2223–2239, 2020.
[3] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing holes in sensor networks," *Mobile networks and Applications*, vol. 11, pp. 187–200, 2006.
[4] S. P. Fekete, M. Kaufmann, A. Kröller, and K. Lehmann, "A new approach for boundary recognition in geometric sensor networks," *arXiv*, 2005.
[5] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Proceedings of the 2005 joint workshop on Foundations of mobile computing*, 2005, pp. 44–53.
[6] D. Chen and P. K. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 1, pp. 50–67, 2007.
[7] C. Qiu, H. Shen, and K. Chen, "An energy-efficient and distributed cooperation mechanism for k-coverage hole detection and healing in wsns," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, 2017.
[8] W. Li and Y. Wu, "Tree-based coverage hole detection and healing method in wireless sensor networks," *Computer networks*, vol. 103, 2016.
[9] W. Li and W. Zhang, "Coverage hole and boundary nodes detection in wireless sensor networks," *Journal of network and computer applications*, vol. 48, pp. 35–43, 2015.
[10] P. Le Nguyen, K. Nguyen, H. Vu, and Y. Ji, "Telpac: A time and energy efficient protocol for locating and patching coverage holes in wsns," *Journal of Network and Computer Applications*, vol. 147, p. 102439, 2019.
[11] V. De Silva, R. Ghrist, and A. Muhammad, "Blind swarms for coverage in 2-d." in *Robotics: Science and Systems*, 2005, pp. 335–342.
[12] Y. Bejerano, "Coverage verification without location information," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 631–643, 2011.

[13] I. M. Khan, N. Jabeur, and S. Zeadally, "Hop-based approach for holes and boundary detection in wireless sensor networks," *IET Wireless Sensor Systems*, vol. 2, no. 4, pp. 328–337, 2012.
[14] F. Wang and H. Hu, "Coverage hole detection method of wireless sensor network based on clustering algorithm," *Measurement*, vol. 179, 2021.
[15] L.-H. Zhao, W. Liu, H. Lei, R. Zhang, and Q. Tan, "Detecting boundary nodes and coverage holes in wireless sensor networks," *Mobile information systems*, vol. 2016, no. 1, p. 8310296, 2016.
[16] X. Wei, H. Guo, X. Wang, X. Wang, and M. Qiu, "Reliable data collection techniques in underwater wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 404–431, 2021.
[17] S. Kundu and N. Das, "A study on boundary detection in wireless sensor networks," *Innovations in Systems and Software Engineering*, 2023.
[18] P. Sharma, R. P. Singh, M. A. Mohammed, R. Shah, and J. Nedoma, "A survey on holes problem in wireless underground sensor networks," *IEEE Access*, vol. 10, pp. 7852–7880, 2021.
[19] P. Sharma and R. P. Singh, "Coverage hole identification & healing in wireless underground sensor networks," *Measurement: Sensors*, 2022.
[20] S. Ohrhallinger, J. Peethambaran, A. D. Parakkat, T. K. Dey, and R. Muthuganapathy, "2d points curve reconstruction survey and benchmark," in *Computer Graphics Forum*, vol. 40, no. 2, 2021, pp. 611–632.
[21] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on information theory*, 1983.
[22] M. Melkemi and M. Djebali, "Computing the shape of a planar points set," *Pattern Recognition*, vol. 33, no. 9, pp. 1423–1436, 2000.
[23] M. Duckham, L. Kulik, M. Worboys, and A. Galton, "Efficient generation of simple polygons for characterizing the shape of a set of points in the plane," *Pattern recognition*, vol. 41, no. 10, pp. 3224–3236, 2008.
[24] A. Gheibi, M. Davoodi, A. Javad, F. Panahi, M. M. Aghdam, M. Asgaripour, and A. Mohades, "Polygonal shape reconstruction in the plane," *IET computer vision*, vol. 5, no. 2, pp. 97–106, 2011.
[25] J. Peethambaran and R. Muthuganapathy, "A non-parametric approach to shape reconstruction from planar point sets through delaunay filtering," *Computer-Aided Design*, vol. 62, pp. 164–175, 2015.
[26] S. Methirumangalath, S. S. Kannan, A. D. Parakkat, and R. Muthuganapathy, "Hole detection in a planar point set: An empty disk approach," *Computers & Graphics*, vol. 66, pp. 124–134, 2017.
[27] S. B. Thayyil, A. D. Parakkat, and R. Muthuganapathy, "An input-independent single pass algorithm for reconstruction from dot patterns and boundary samples," *Computer Aided Geometric Design*, 2020.
[28] S. B. Thayyil, J. Peethambaran, and R. Muthuganapathy, "A sampling type discernment approach towards reconstruction of a point set in r2," *Computer Aided Geometric Design*, vol. 84, p. 101953, 2021.
[29] F. Sheikhi, B. Zeraatkar, F. Amereh, S. S. Firouzabadi, and E. R. Ghalehjoughi, "Prudent carving: a progressively refining algorithm for shape reconstruction from dot patterns," *The Journal of Supercomputing*, 2024.
[30] J. Peethambaran, S. Ohrhallinger, and A. D. Parakkat, "Shape characterization of point sets in 2d." EUROGRAPHICS, 2022.
[31] F. Sheikhi, B. Zeraatkar, and S. Hanaie, "Dot to dot, simple or sophisticated: a survey on shape reconstruction algorithms," *Acta Informatica*, vol. 60, no. 4, pp. 335–359, 2023.
[32] N. Amenta, M. Bern, and D. Eppstein, "The crust and the β-skeleton: Combinatorial curve reconstruction," *Graphical models and image processing*, vol. 60, no. 2, pp. 125–135, 1998.
[33] X. Li and D. K. Hunter, "Distributed coordinate-free hole recovery," in *ICC Workshops-2008 IEEE International Conference on Communications Workshops*. IEEE, 2008, pp. 189–194.
[34] G. T. Toussaint, "Solving geometric problems with the rotating calipers," in *Proc. IEEE Melecon*, vol. 83, no. 83, 1983, p. A10.
[35] Y. Wang, J. Gao, and J. S. Mitchell, "Boundary recognition in sensor networks by topological methods," in *Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006.
[36] A. Khelil, R. Beghdad, and A. Khelloufi, "3ha: hybrid hole healing algorithm in a wireless sensor networks," *Wireless Personal Communications*, vol. 112, pp. 587–605, 2020.
[37] P. Mathur, R. H. Nielsen, N. R. Prasad, and R. Prasad, "Cost benefit analysis of utilising mobile nodes in wireless sensor networks," *Wireless Personal Communications*, vol. 83, pp. 2333–2346, 2015.
[38] P. Sharma and R. P. Singh, "Coverage hole identification & healing in wireless underground sensor networks," *Measurement: Sensors*, 2022.
[39] S. Kumari, P. K. Mishra, A. K. Sangaiah, and V. Anand, "Priority based k-coverage hole restoration and m-connectivity using whale optimization scheme for underwater wireless sensor networks," *International Journal of Intelligent Networks*, vol. 4, pp. 240–252, 2023.
[40] X. Liu, "Sensor deployment of wireless sensor networks based on ant colony optimization with three classes of ant transitions," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1604–1607, 2012.
[41] R. Chowdhuri and M. K. D. Barma, "Enhancing network reliability: Exploring effective strategies for coverage-hole analysis and patching in wireless sensor networks," *Wireless Personal Communications*, 2024.

SREERAM SIVADASAN (Member, IEEE) received the B.E. and M.Tech degrees in Computer Science and engineering from the Visvesvaraya Technological University, Karnataka, India in 2004 and 2007. He is pursuing PhD in Computer Science and Engineering from Sathyabama Institute of Science & Technology, Chennai, India and is a visiting researcher at LTCI - Telecom Paris, Institut Polytechnique de Paris, France. His research area is Computational Geometric Algorithms for WSN in Edge-computing environment.

NAGARAJAN GOVINDAN (Senior Member, IEEE) received the B.E. degree in electrical and electronics engineering from MS University, in 2000, the M.E. degree in applied electronics from Anna University, in 2005, and the M.E. and Ph.D. degrees in computer science and engineering from Sathyabama University, in 2007 and 2015, respectively. He is currently a Faculty Member of the Department of Computer Science and Engineering, School of Computing, Sathyabama Institute of Science and Technology, Chennai, India. He has published more than 70 research articles in peer-reviewed journals, such as IEEE conference, ACM, Springer-Verlag, Inderscience, and Elsevier. He also has contributed 15 book chapters thus far for various technology books. He has authored and edited three books thus far and is focusing on some of the emerging technologies, such as the IoT, edge/fog computing, artificial intelligence (AI), data science, blockchain, digital twin, and 5G. His current research interests include computer vision, the IoT, 5G, edge/fog computing, artificial intelligence, machine learning, and wireless sensor networks.

AMAL DEV PARAKKAT is a tenured Assistant Professor of Computer Graphics at LTCI-Telecom Paris, Institut Polytechnique de Paris, France. A graduate of the University of Calicut and Mahatma Gandhi University in India, he earned his PhD from the Indian Institute of Technology Madras, India, in 2019. After completing his PhD, he worked as a postdoctoral researcher at Ecole Polytechnique in France, as an Assistant Professor at the IIT Guwahati, and finally as a Research Associate at TU Delft in the Netherlands. His research focus is on practical algorithms for digital content creation tasks, with the main focus on Sketch-based interfaces. He also works on fundamental problems in digital geometry processing, including reconstruction and meshing.