An Efficient Geometric Inversion Based Approach to Hole Detection and Restoration in Wireless Sensor Networks

David Cruz Santiago[1] and Habib M. Ammari[2]
[1] College of Computing, Data Science, and Society
University of California, Berkeley
[2] Wireless Sensor and Mobile Autonomous Networks (WiSeMAN) Research Center
School of Engineering
College of Arts and Sciences
Texas A&M International University
davidcruzsantiago@berkeley.edu, Habib.Ammari@tamiu.edu

***Abstract*—Present hole-healing algorithms in wireless sensor networks often result in sub-optimal node placements or incur high computational overhead. In this paper, we develop the Geometric Inversion-Based Hole Restoration Algorithm (GIBHRA), a novel and efficient approach that formulates the hole healing task as the challenge of finding a maximally inscribed circle perfectly tangent to the three boundary nodes defining a coverage void. By applying geometric inversion, GIBHRA effectively calculates the mathematically optimal location for a new restorative node, thereby maximizing its potential coverage within the void. Simulation results demonstrate GIBHRA is decisively superior to benchmark methods for coverage efficiency, node deployment, resource efficiency, and hole healing in areas with coverage holes.**

***Index Terms*—Wireless Sensor Networks, Coverage, k-coverage, Hole Detection, Hole Restoration, Hole Healing**

I. Introduction

Among the various design, efficiency, and protocol challenges in wireless sensor networks, maintaining network coverage stands out as one of the most critical and pivotal problems. A wireless sensor network can be defined as a large network containing a collection of spatially distributed sensor nodes that wirelessly monitor physical or environmental conditions, usually remotely placed [1]. However, the common practice of random node deployment in many real-world applications may result in certain areas in the network remaining uncovered, leading to "coverage holes" or areas left unmonitored. This can severely degrade network performance and result in low effectiveness in achieving coverage. System or hardware failures can also contribute to coverage hole generation as described by Adday et al. [17]. Coverage holes can also occur due to the predictable death of nodes whenever nodes run out of battery or reach power exhaustion as described by Shakhov et al [16]. Although many algorithms have been proposed to detect and heal coverage holes within a network [10, 13], they frequently suffer from high computational overhead or result in sub-optimal placement for new restorative nodes.

To address these limitations, this paper proposes the novel Geometric Inversion-Based Hole Restoration Algorithm (GIBHRA). We formulate the hole restoration problem as a geometric challenge of finding a fourth circle that is mutually tangent to the three sensor discs bounding a coverage void. Our core contribution is therefore a novel and efficient method that leverages the principles of geometric inversion to solve this problem for the general case of non-tangent sensor discs in a network, thereby allowing us to get the ideal node placement location and improve the quality of service within a network.

In this paper, we consider a wireless sensor network consisting of a set of n randomly deployed nodes $S = \{S_1, S_2, \ldots, S_n\}$ in a $2D$ Euclidean plane. Each sensor has a uniform sensing radius R_s, a sensing disc and a respective area it covers within the network. A coverage hole H is defined as a region within the network's monitored area that is not covered by the sensing disc of any node. Such a hole is typically bounded by a subset of boundary nodes, $B \subset S$. The primary objective in this paper to tackle the coverage problem is to deploy a single new sensor node in a coverage void given three non-overlapping and non-intersecting boundary sensor nodes in a WSN. This geometric challenge is based on the classical problem of Apollonius [11]. Our approach consists of first identifying a set of non-overlapping triplets, $T = \{(S_i, S_j, S_k), \ldots\}$, where each triplet satisfies the condition $distance(S_a, S_b) \geq 2R_s$. For each triplet, the subsequent task is to find the center (x_c, y_c) and radius r_c of a fourth circle C_c that is mutually tangent to the sensing discs of S_i, S_j, and S_k. The solution to this problem, (x_c, y_c), provides the optimal placement location for the restorative sensor S_c that guarantees sufficient coverage. The corresponding radius, r_c, provides the largest possible sensing range the new node could have while precisely fitting within the void defined by the three nodes, thereby maximizing the restored area. We define the optimal placement location as the center of the maximally inscribed circle that is mutually tangent to the three boundary sensor discs defining the coverage hole. Through extensive simulation, we demonstrate that GIBHRA is significantly more

978-3-903176-77-5© 2025 IFIP

effective at node deployment, achieving a coverage efficiency of 99.2% in sparse networks as well as more cost effective, requiring up to 48% fewer restorative nodes than benchmark algorithms.

II. Related Work

Over the past years, extensive research efforts have been devoted to studying the notable coverage problem in WSNs, including hole detection and hole restoration [5],[6], and [18].

A. Hole Detection Strategies

Hole detection for k-coverage networks is incredibly important, as detecting holes can help us incorporate hole recovery methods to guarantee coverage. Research on hole detection can be focused on three main categories: topological, geometric, and statistical approaches. Yan et al. [7] uses the topological method of a simplified Rips complex to identify the boundary cycles of coverage holes by detecting non-neighboring edges. These edges are considered boundary edges for the hole detected. Zhang et al. [9] proposes a perimeter-coverage checking approach by identifying intersection points of two perimeters of sensing disks. It then identifies covered and uncovered intersection points, described as crossings. Similarly, the Distributed Coverage Hole Detection protocol is proposed by Sahoo et al. [10], which detects coverage holes by having each node classify the intersection points with its neighbors as either uncovered critical points (CIPs) or covered points. The algorithm then localizes these CIPs spatially and traces them in a clockwise manner to form a bounded or unbounded coverage hole. Khalifa et al. [22] propose an algorithm where, upon a node's failure, its immediate neighbors calculate uncovered intersection points (UIPs) to define the hole's boundary, identify the relevant set of points corresponding to the failure, and then locally broadcast this information to allow all adjacent nodes to map the complete hole perimeter.

B. Hole Healing Strategies

Several papers have explored elegant geometrical approaches to finding optimal hole restoration techniques. Qiu et al. [2] proposes the Distributed Voronoi-based Cooperation scheme, in which Voronoi diagrams utilize both global and local k-order variants to identify uncovered vertices, coverage holes and correspondingly coordinate mobile sensors for healing. Wang et al. [3] propose an interpolating placement scheme to place sensors in an empty field from scratch to prevent holes from ever forming in the initial layout. Wang achieves this by first partitioning the complex field into "single-row regions" and wider "multi-row regions." For single-row regions, sensors are placed along a calculated geometric derived bisector while, in multi-row regions, sensors are deployed in a highly structured, grid-like pattern in either a rectangular or triangular lattice to guarantee complete coverage and connectivity. Khalifa et al. [4] proposes the heuristic hole healing algorithm that repairs coverage holes by adjusting the node's sensing range and relocating it to a new position whenever a node failure is detected. In the situation where a node detects any of its neighboring nodes to fail, it coordinates among its neighboring nodes a node to recover the new uncovered area. This self-healing algorithm heals coverage holes by calculating the target location to be covered and calculating the necessary distance for the selected node to be expanded. Babaie et al. [19] uses triangulation by relying on placing new nodes at heuristic locations derived from the triangle of sensor centers, like its incenter or circumcenter. However, it is not as precise because the placement of the new node is based on geometric properties of the triangle connecting the sensor centers, rather than the actual curvilinear boundaries of the void itself.

III. System Model

Let us consider a wireless sensor network with randomly deployed n sensor nodes in a rectangular grid. It is assumed tangency among nodes is extremely rare, if not impossible, by nature of the randomly deployed process. Furthermore, nodes in the network may have considerable overlap with other nodes to achieve k-coverage. Conversely, there may be sparse nodes that form coverage holes.

This network will be modeled as a set of n static, homogeneous sensor nodes, denoted by $S = \{S_1, S_2, \ldots, S_n\}$. Each node $S_i \in S$ is defined by its cartesian coordinates (x_i, y_i) in a two-dimensional Euclidean plane, for $i = 1, 2, \ldots, n$. For our model, the system model and associated definitions presented are adapted from the distributed coverage hole detection protocol established by Sahoo et al. [10] and the Voronoi diagram hole detection protocol by Qiu et al. [2]:

- Sensing Disc: Each sensor node S_i is assumed to provide complete coverage within a circular region of radius R centered at its coordinates, known as the sensing disc. Any object within this circle centered at its location is said to be detected by the sensing disc. The radius of the sensing disc is referred to as its sensing range, R_s.
- Neighboring Nodes: Two nodes, S_i and S_j, are considered neighbors if their sensing discs intersect. Formally, the set of neighbors for a node S_i, denoted $N(S_i)$, is given by:

$$N(S_i) = \{S_j \in S \mid S_j \neq S_i \text{ and } d(S_i, S_j) < 2R_s\}$$

  where $d(S_i, S_j)$ represents the Euclidean distance between the centers of nodes S_i and S_j.
- Coverage Hole: Any part of the wireless sensor network region not covered by the sensing disc of any sensor.
- Unbounded Hole: A coverage hole is said to be unbounded if its area is not fully enclosed by the sensing discs of the deployed nodes. The area of unbounded holes extends to the perimeter of the network.
- Bounded Hole: A coverage hole is bounded if its area is enclosed by the sensing discs of the deployed nodes.
- Boundary Sensor Node: A node S_i is classified as a boundary sensor if a portion of its sensing disc's circumference is not covered by any of its neighbors in $N(S_i)$.
- Critical Intersection Points: Let there be two nodes, S_i and S_j. A critical intersection point is designated as a point where the sensing boundary of a node S_i intersects with its neighboring node S_j.

IV. Algorithm

In this section, we outline our hole restoration approach as follows. Phase 1 identifies the set of boundary nodes that form a hole in order to avoid redundantly choosing non-boundary sensor nodes. This ensures our algorithm does not waste computational resources on irrelevant, interior nodes. Phase 2, which is our core hole detection method, describes the Delaunay triangulation approach to select three non-intersecting and non-overlapping nodes for hole restoration. Phase 3 describes our novel geometric inversion approach (GIBHRA) and its respective mathematical properties to find a solution for an optimal center for placement in a coverage hole.

A. Algorithm

1) Phase 1: Identify Boundary Sensor Nodes: Phase 1 of the algorithm will filter the entire set of N sensor nodes, S, to identify the subset of m nodes, B, that actively border a coverage hole. This pre-processing step will significantly reduce the search space for more densely deployed wireless sensor networks. All nodes satisfying this boundary condition are subsequently aggregated into the result set, B, to serve as the direct input for the subsequent triplet selection process.

Algorithm 1 Phase 1 - Identify Boundary Nodes

Input: S, the set of all nodes.
Output: B, a list of boundary nodes.

Initialize $B \leftarrow \emptyset$
for all node S_i in the set of all nodes S **do**
    Let $N(S_i)$ be the set of neighboring nodes to S_i.
    if S_i has an arc on its sensing circle not covered by any node in $N(S_i)$ **then**
        Add S_i to the list B.
    end if
end for
return B

2) Phase 2 - Delaunay Triangulation Based Approach: The primary objective of phase 2 is to produce a set of Triplets, $T = \{(S_i, S_j, S_k)\}$ from an input set of N boundary nodes such that all three nodes are non-overlapping, non-intersecting, and are sourced from a pre-identified set of hole-boundary nodes. A triplet is considered valid if its constituent nodes are mutually non-overlapping and non-intersecting, meaning the Euclidean distance between the centers of any two nodes in a triplet is greater than or equal to twice the uniform sensing radius, R_s.

The first fundamental step in solving this problem is identifying three non-intersecting, non-overlapping nodes to serve as a geometric basis for our hole restoration solution. The nodes must be non-intersecting and non-overlapping to ensure a well-defined geometric basis for our hole restoration solution and to ensure at most 8 distinct solutions as described by Gisch et al. [11]. By Gisch et al. [11], when circles intersect or overlap, the problem becomes ill-posed, and the number of valid tangent solutions may become infinite or degenerate. It would also significantly increase the computational complexity in our algorithm and introduce ambiguity in geometric interpretation. Furthermore, in a wireless sensor network, if two nodes intersect or overlap, there is less of a gap or hole in the specific region that needs to be filled. A naive approach to triplet node selection would consist of a brute-force search approach by iterating through all other nodes to find its two closest valid partners, resulting in a $O(m^3)$ time complexity. To improve efficiency, let us consider using a Delaunay triangulation and localization approach adapted from Qiu et al. [2] and Zhang et al. [8], respectively. Delaunay triangulation is utilized by partitioning a set of points in a plane into a mesh of non-overlapping triangles such that the circumcircle of any triangle contains no other points from the set. This step is crucial because it focuses on finding spatially proximate nodes for ideal node selection while inherently pruning distant nodes that would not form a well-defined void. Moreover, localization is utilized in this phase to obtain the precise coordinates (x, y) necessary for Delaunay triangulation to enforce distance-based filtering constraints and ultimately perform our precise geometric inversion required to solve for the restorative node's optimal placement. The algorithm first sets an adaptive distance threshold, T_d, based on the current network's coverage and progressively decreases as the network coverage increases. This adaptive distance threshold ensures our hole detection process is prioritizing major hole areas when the network coverage is low and progressively relaxes the threshold constraint to identify smaller holes as the network becomes denser. It then generates a set of candidate triplets by performing a Delaunay triangulation on the boundary nodes and filters these candidates by ensuring all triangle edge lengths exceed T_d. This edge-length check serves as the primary filter to enforce sufficient separation between nodes in order to satisfy the non-overlapping condition. For each valid triplet, it performs a spatial validation to confirm that the area bounded by the nodes represents a true, uncovered void and the resulting list of confirmed triplets is prioritized by the largest perimeter for sequential restoration. Specifically, this spatial validation ensures that the precise location is not within the sensing range of any existing node in the network.

3) Time Complexity Analysis: Let $m \leq n$ denote the number of hole-boundary nodes identified in Phase 1 and n be the number of nodes in the network. In Phase 2, we construct a Delaunay triangulation on the m boundary nodes using $O(mlogm)$ time complexity. The Delaunay triangulation of m points will produce $O(m)$ candidate triangles. The algorithm then iterates through each of these triangles where it iterates through all n nodes of the original network to see if the circumcenter of each candidate triangle produced by the Delaunay triangulation is covered. This takes $O(n)$ time. Since this $O(n)$ check is performed for all $O(m)$ triangles, the total time complexity is $O(m * n)$.

B. Geometric Inversion Process

Geometric inversion is a powerful problem-solving technique that allows complex problems to be translated to simpler, smaller problems. In particular, the practicality of geometric

Algorithm 2 Adaptive Triplet Selection via Delaunay Triangulation

Input: Node set S, boundary nodes B, coordinates P, sensing radius R_s, coverage ratio ρ_{cov}
Output: Prioritized list of triplets T

function SELECTTRIPLETS($S, B, P, R_s, \rho_{\text{cov}}$)
  if $\rho_{\text{cov}} < 0.60$ **then**
    $\tau_d \leftarrow 1.8 \cdot R_s$
  else if $\rho_{\text{cov}} < 0.85$ **then**
    $\tau_d \leftarrow 1.6 \cdot R_s$
  else
    $\tau_d \leftarrow 1.4 \cdot R_s$
  end if
  $P_B \leftarrow getCoords(B, P)$
  $DT \leftarrow delaunay(P_B)$
  $C \leftarrow \emptyset$
  for each triangle $(v_i, v_j, v_k) \in DT$ **do**
    $(s_i, s_j, s_k) \leftarrow getNodes(v_i, v_j, v_k, B)$
    $d_{ij} \leftarrow dist(s_i, s_j)$
    $d_{jk} \leftarrow dist(s_j, s_k)$
    $d_{ki} \leftarrow dist(s_k, s_i)$
    if $d_{ij} > \tau_d$ **and** $d_{jk} > \tau_d$ **and** $d_{ki} > \tau_d$ **then**
      $(x_c, y_c) \leftarrow findCircleCenter(s_i, s_j, s_k)$
      if $(x_c, y_c) \neq null$ **then**
        if not isCovered$((x_c, y_c), S, R_s)$ **then**
          $p \leftarrow d_{ij} + d_{jk} + d_{ki}$
          Add (s_i, s_j, s_k, p) to C
        end if
      end if
    end if
  end for
  Sort C in descending order of p
  $T \leftarrow extractTriplets(C)$
  return T
end function

inversion lies in the ability to apply an appropriate inversion and reapply the same inversion again in order to translate back the results to the original problem. In this section, we formally define what geometric inversion is, its corresponding properties, and provide our inversion hole restoration approach.

1) Geometric Scenario and the General Problem: The geometric relationship between three nodes to solve for a mutually tangent solution circle can fall into three categories. In case 1, all three nodes are already pairwise tangent to each other. This is also called Soddy circles by Gisch et al. [11]. Solving for a fourth circle inscribed in three pairwise tangent circles is simply done using Descartes' circle theorem as described in [12]. The theorem relates the *curvatures* ($k = 1/r$) of four mutually tangent circles. Given the curvatures k_1, k_2, k_3 of the three known sensor nodes, the curvature k_4 of the restorative inner circle is given by Soddy's formula:

$$k_4 = k_1 + k_2 + k_3 + 2\sqrt{k_1k_2 + k_2k_3 + k_3k_1} \quad (1)$$

This provides a straightforward algebraic solution for a perfectly tangent configuration. However, our model consists of randomly deployed sensor nodes, which makes tangency very rare and therefore impractical to solve. Let us consider case 2 with one pairwise tangent circle and one non-tangent circle. Descartes' Circle Theorem is not useful here because it lacks the geometric structure where the distances between all centers are fixed. Again, due to the nature of randomly deployed sensor nodes, achieving tangency between two nodes is exceptionally rare. Let us now consider case 3 as the most common scenario in a randomly deployed WSN where three non-tangent circles are most common. Finding a solution circle tangent to all three nodes is difficult and non-trivial, as described by [9]. Our proposed method therefore introduces geometric inversion to make the problem easier to solve.

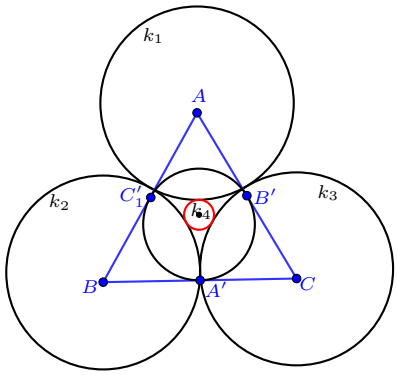


Fig. 1: Three pairwise tangent circles k_1, k_2, k_3 with points A, B, and C as their centers of the primary circles. The restorative circle, known as the inner Soddy circle, k_4, is shown to be mutually tangent to the three larger circles and inscribed within the central, curvilinear gap.

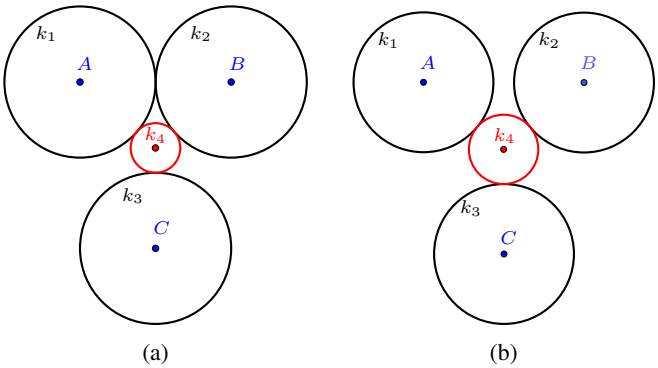


Fig. 2: Illustration of geometric scenarios. (a) A configuration with two tangent circles (k_1, k_2) and one non-tangent circle, k_3. (b) The classic Soddy circle configuration with three mutually tangent circles, where k_4 is the solution Soddy circle.

C. The Geometric Inversion Framework

Geometric inversion is a transformation of the Euclidean plane that maps circles and lines to other circles and lines. The principles of geometric inversion possess properties that are highly suitable for analyzing network topology.

1) Definition of Geometric Inversion: The formal definition of geometric inversion is adapted from [9] and [14]:

Let k be a circle of inversion in the plane with center O and radius r. The geometric inversion with respect to k is a transformation that maps a point P (where $P \neq O$) to its inverse point P'. The definition can be stated in two equivalent ways. Figure 3 provides a visualization.

1) Algebraic Definition: The inverse point P' is the unique point lying on the ray $\overrightarrow{OP}$ that satisfies the given relation: $OP \cdot OP' = r^2$ The points P and P' are called a pair of inverse points with respect to the circle k. By this definition, points inside the circle k are mapped outside, and vice-versa. Points on the circle k are mapped to themselves.

2) Geometric Construction: Equivalently, the inverse point P' can be constructed geometrically. For any point P, its inverse P' is the unique point (other than P itself) through which all circles that pass through P and are perpendicular to the circle of inversion k must also pass. As depicted in Figure 3, the intersection of any two such distinct circles uniquely determines the location of P', which lies on the line passing through O and P.

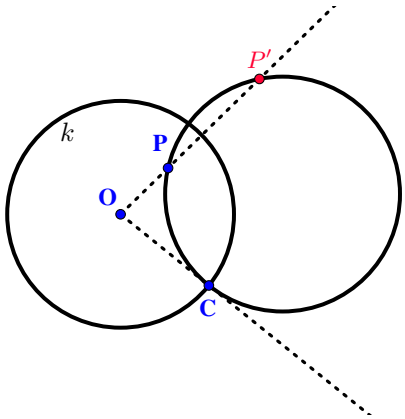


Fig. 3: All circles through point P and perpendicular to k will also pass through a point P'. We mark this P' as the inverse of P. The example circle shown is constructed to pass through P and intersect the circle of inversion, k, at a right angle.

2) Properties of Inversion: Geometric inversion has a powerful set of useful properties. The following properties are adapted from [9] and [14]:

Lemma 1 (Composition of I^2)**.** The geometric inversion $I(O, r)$ is its own inverse. In other words, I *swaps* points in the plane in pairs:

$$X \overset{I}{\leftrightarrow} X_1, \quad Y \overset{I}{\leftrightarrow} Y_1, \quad \text{and} \quad Z \overset{I}{\leftrightarrow} Z, \tag{2}$$

where applying I twice will get us back to where we started. Therefore, for any point P in the plane other than O, applying the inversion twice returns the original point:

$$I(I(P)) = P$$

Proof: Let $P' = I(P)$. By Definition 1, P' is a point on the ray $\overrightarrow{OP}$ such that $OP \cdot OP' = r^2$. Now, let $P'' = I(P')$. By the same definition, P'' must lie on the ray $\overrightarrow{OP'}$ and satisfy $OP' \cdot OP'' = r^2$, since P' is on the ray $\overrightarrow{OP}$, the ray $\overrightarrow{OP'}$ is identical to $\overrightarrow{OP}$. Comparing the two conditions, we have $OP \cdot OP' = OP' \cdot OP''$, which implies $OP = OP''$. As P and P'' are on the same ray from O and at the same distance, they are the same point. Thus, $P'' = P$. ■

Lemma 2 (A circle passing through O)**.** Let k_1 be a circle such that the center of inversion $O \in k_1$. The image of k_1 under inversion is a line m that does not pass through O.

$$I(k_1) = m, \quad \text{with} \quad O \notin m$$

Proof: Let the image of the circle k_1 be the set m, so $I(k_1) = m$. Applying the inversion again gives $I(m) = I(I(k_1))$. By the Composition of I^2 property, $I(I(k_1)) = k_1$. Therefore, the image of the set m is a circle k_1 which passes through the center O. From lemma 1, a set whose image is a circle through the center must be a line not passing through the center. Thus, m is a line with $O \notin m$. ■

Lemma 3 (A circle not passing through O)**.** Let k_2 be a circle such that the center of inversion $O \notin k_2$. The image of k_2 under inversion is another circle k_3 that also does not pass through O.

$$I(k_2) = k_3, \quad \text{with} \quad O \notin k_3$$

Proof: Let the center of inversion O be the origin $(0, 0)$ and the circle of inversion have radius R. An arbitrary circle k_2 not passing through the origin can be described by the general equation:

$$x^2 + y^2 - 2ax - 2by + C = 0, \quad \text{where } C \neq 0$$

The inversion mapping relates a point (x, y) on k_2 to a point (x', y') on its image, k_3, by the transformation:

$$(x, y) = \left(\frac{x' R^2}{x'^2 + y'^2}, \frac{y' R^2}{x'^2 + y'^2} \right)$$

A useful identity derived from this is $x^2 + y^2 = \frac{R^4}{x'^2 + y'^2}$.

Substituting these into the equation for k_2 gives:

$$\frac{R^4}{x'^2 + y'^2} - 2a \left(\frac{x' R^2}{x'^2 + y'^2} \right) - 2b \left(\frac{y' R^2}{x'^2 + y'^2} \right) + C = 0$$

Multiplying by $(x'^2 + y'^2)$ to clear the denominator yields the equation for the image curve k_3:

$$R^4 - 2aR^2 x' - 2bR^2 y' + C(x'^2 + y'^2) = 0$$

Rearranging gives $C(x'^2 + y'^2) - 2aR^2 x' - 2bR^2 y' + R^4 = 0$. Since the coefficients of x'^2 and y'^2 are equal and non-zero (as $C \neq 0$), this is the equation of a circle.

To test if O lies on k_3, we substitute $(x', y') = (0, 0)$ into its equation, which results in $R^4 = 0$. This is a contradiction for any inversion with $R > 0$. Thus, $O \notin k_3$. ■

Lemma 4 (Preservation of Tangency)**.** Let k_a and k_b be two curves (lines or circles) that are tangent to each other at a point P. Under inversion $I(O, r)$, their images, $I(k_a)$ and $I(k_b)$, are also tangent at the image point $P' = I(P)$, provided $P \neq O$. If the point of tangency P is the center of inversion O, the images $I(k_a)$ and $I(k_b)$ are two parallel lines.

Proof: The proof relies on the established property that geometric inversion is a conformal map, meaning it preserves the magnitude of the angle at which any two curves intersect [14]. Let two curves, k_a and k_b, be tangent at a point P. By definition, tangency is the specific case where the angle of intersection between the curves is zero. Since inversion preserves the angle of intersection, the images of the curves, $I(k_a)$ and $I(k_b)$, must also intersect at a zero-degree angle. Therefore, the image curves are tangent at the image point $P' = I(P)$. In the special case where the point of tangency is the center of inversion O, both curves must pass through O. Their images are therefore two distinct lines. Since the original curves intersect only at O, their image lines cannot intersect (otherwise, the inverse of their intersection point would be a second point of intersection on the original curves, which is a contradiction). Thus, the image lines must be parallel. ■

D. Phase 3 - The Inversion-Based Solution Procedure

Following the selection of a triplet (S_i, S_j, S_k), the next phase of the algorithm simplifies the geometric configuration to facilitate a sound solution to the problem. This is achieved through a sequence of two transformations: a temporary radial expansion followed by a geometric inversion.

a) Establishment of a Tangent Configuration: Let the centers of the three non-intersecting and non-overlapping nodes (S_i, S_j, S_k) be denoted by $c_i, c_j,$ and c_k. Let the pairwise Euclidean distances between these centers be calculated as $d_{ij} = \text{distance}(c_i, c_j)$, $d_{ik} = \text{distance}(c_i, c_k)$, and $d_{jk} = \text{distance}(c_j, c_k)$. To identify the pair of nodes that will first achieve tangency under a uniform radial expansion, the minimum of these three distances is determined by:

$$d_{\min} = \min(d_{ij}, d_{ik}, d_{jk})$$

For two circles with an identical radius r_{new} to be tangent, the distance between their centers must be exactly $2r_{new}$. Therefore, the specific radius required to create this initial tangency is found by solving $d_{\min} = 2r_{new}$, which yields:

$$r_{new} = \frac{d_{\min}}{2}$$

This value, r_{new}, represents the uniform radius that would need to be temporarily assigned to the three nodes to force the closest pair into a tangent configuration as seen in case 2. Using the radius r_{new}, three new circles denoted $S'_i, S'_j,$ and S'_k, are defined. By the definition of r_{new}, at least two of these circles are now mutually tangent at a single point. Without loss of generality, let us assume that S'_i and S'_j are the tangent pair. These circles share the same centers as the original triplet but all possess the new, uniform radius r_{new}.

b) Inversion of the Tangent Configuration: The core of the simplification process is a geometric inversion centered at the newly created point of tangency. Let the point of tangency between S'_i and S'_j be the center of inversion, O_{inv} and be the center of a new circle, α. We perform an inversion with respect to a circle, α, of arbitrary radius centered at O_{inv}. We now perform inversion on (S'_i, S'_j, S'_k) with respect to the newly created circle of inversion, α:

- By Lemma 2, since the circles S'_i and S'_j both pass through the center of inversion O_{inv}, their images under inversion, $I(S'_i)$ and $I(S'_j)$, are transformed into two distinct lines, m_i and m_j. As the original circles were tangent only at O_{inv}, their line images are parallel.
- By Lemma 3, the third circle, S'_k, which does not pass through O_{inv}, is transformed into another circle, denoted S''_k.

Figure 4a depicts the geometric configuration before inversion with our newly formed circle, α, as the circle of inversion.

1) Calculation of the Solution Circle in the Inverted Space: The parameters for the solution circle S''_4 with center C''_4 and radius r''_4 are determined as follows:

a) Determining the Radius (r''_4): The radius of the solution circle, S''_4, is determined by the distance between the two parallel lines, m_i and m_j. By definition, any circle tangent to these two lines must have a radius equal to half the perpendicular distance between them. Let this distance be d. The radius of our solution circle is therefore fixed:

$$r''_4 = \frac{d}{2} \tag{3}$$

b) Determining the Center (C''_4): To calculate the coordinates of the center C''_4, we can establish a temporary coordinate system for simplicity. Let the line midway between the parallel lines m_i and m_j be the x-axis. By this construction, the center of our solution circle C''_4 must lie on this axis at some coordinate $(x_4, 0)$, while the center of the known circle C''_k is at a general position (x_k, y_k).

The final condition is the tangency between circles S''_k and S''_4. The distance between their centers must be equal to the sum of their radii:

$$\text{distance}(C''_4, C''_k) = r''_4 + r''_k$$

Substituting the known values and coordinates into this equation gives:

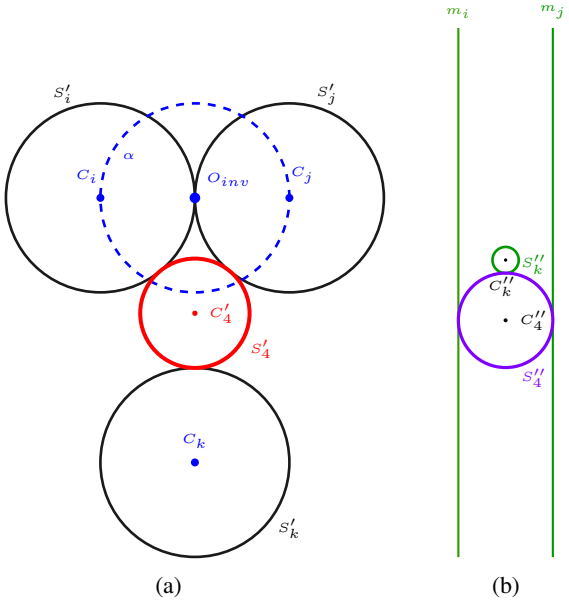
$$\sqrt{(x_4 - x_k)^2 + (0 - y_k)^2} = \frac{d}{2} + r''_k$$

Solving this equation for x_4 yields the two possible x-coordinates for the center of the solution circle:

$$x_4 = x_k \pm \sqrt{\left(\frac{d}{2} + r''_k\right)^2 - y_k^2} \tag{4}$$

These two solutions represent the centers of two possible solution circles where, for the purpose of hole restoration, the inscribed solution is selected. With the center $C''_4 = (x_4, 0)$ and radius r''_4 now fully determined, the final step is to apply the inverse transformation $I(S''_4)$ to obtain the parameters of the restorative circle, S'_4. This process yields the optimal center point, (x_o, y_o), and radius of the one circle that will restore the coverage hole defined by the three selected nodes.

2) Deployment of New Optimal Nodes: Now that we have obtained our optimal center point (x_o, y_o), we deploy new nodes at this center point using the standard uniform radius R_s. This will maximize coverage and provide hole restoration. Figure 5 provides a visualization of the optimal restorative node placement derived from phase 3, showing the new node's center precisely located such that its sensing range is mutually tangent to the three nodes defining the coverage hole.



(a) (b)

Fig. 4: The complete geometric inversion and solution process shown in panels (a) and (b). Panel (a) shows the geometric configuration with triplet $(S'_i, S'j, S'k)$ with a uniform radius r_{new}, our circle of inversion, α, with center O_{inv} and an arbitrary radius. Panel (a) shows our solution circle S'_4 after our inversion process tangent to the three initial circles. Panel (b) on the right shows the inversion in the plane geometric configuration. Circles S_i and S_j are transformed into two parallel lines $(m_i,\ m_j)$. Circle S'_k is transformed into S''_k. Inverting S''_4 back to the original space results in circle S'_4.

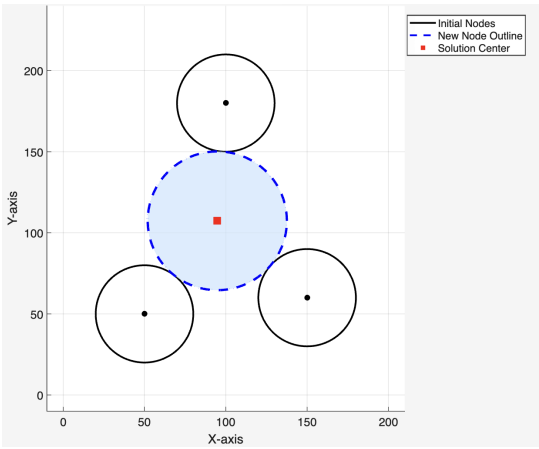


Fig. 5: Using Matlab, the new optimal node location is determined by the center of the solution circle that is mutually tangent to the three sensor nodes. While this figure visualizes the ideal tangent circle, the new node in phase 3 is deployed at its center using the network's uniform sensing radius R_s.

V. Simulation and Results

In this section, we report our simulation results of our novel geometric inversion approach and compare their performance with similar geometric approaches. In particular, our algorithm (GIBHRA) will be compared to the Heal Coverage Holes Algorithm (HCHA) [15]. This approach places the first healing sensor at the in-center of a triangular hole, while subsequent sensors are placed sequentially along the triangle's angle bisectors, with the aim of minimizing overlap. GIBHRA is also compared with the Hole Patching Algorithm (HPA) [20], which selects two adjacent boundary nodes, A and B, and places a new node on the perpendicular bisector of the segment AB. This new node is then positioned at a distance from both A and B that is approximately equal to the network's communication range. Our algorithm is also compared with the Neighbor node Location-based Coverage Hole Recovery algorithm (NLCHR) [21] which places new nodes at the third vertex of an isosceles triangle formed within two boundary nodes. Our model is simulated using MATLAB. To evaluate the performance of our proposed GIBHRA, we conducted extensive simulations in MATLAB on a 1000m x 1000m network area, comparing it against HCHA, HPA and NLCHR across randomly deployed networks with initial densities ranging from 200 to 600 nodes, with all results averaged over 20 independent trials. Each node has a radius, R_s of 50m.

Figure 6 illustrates the quality of each restorative action an algorithm takes. It computes the total new area an algorithm covers throughout its entire iterative healing process and divides it by the total number of new nodes deployed to achieve k-coverage. GIBHRA achieves an average restoration area of approximately 2800 m^2 per node, which is a significant improvement over the next best algorithm, HPA, with an average of approximately 1550 m^2 of average restoration per node deployed. Our algorithm consistently restores a significantly larger area for each deployed node and is maintained across different network densities. Figure 7A evaluates the placement quality via coverage efficiency by computing how much new, previously uncovered hole area it covers. Figure 7 measures the deployment cost of each algorithm by measuring how many new sensor nodes the algorithm needed to add to reach a coverage target of 98%. This allows us to evaluate each algorithm's resource efficiency, where a lower bar signifies a lower cost to achieve the same goal. In Figure 7A, GIBHRA demonstrates efficient hole placement where it consistently restores the most area per node across all tested network densities. As the network approaches full coverage (600 nodes), few holes remain, and the performance difference between the algorithms diminishes. Figure 7B demonstrates superior resource efficiency when using GIBHRA, as it only needs 115 new nodes at 200 nodes and 45 new nodes at 400 nodes compared to the rest of the algorithms which need more new nodes. Figure 8 evaluates the placement quality via Coverage Efficiency, which measures the percentage of a new node's sensing area that productively covers a hole. A higher value signifies a more intelligent placement with less wasteful overlap. GIBHRA demonstrates a significant advantage, achieving 99.2% efficiency in sparse networks, while HPA only reaches 50.8%. This near-optimal result stems from GIBHRA's mathematically rigorous approach of finding the largest inscribed circle that minimizes redundant coverage. As the network density increases to 600 nodes, GIBHRA's efficiency of 27.4% remains approximately three times higher.

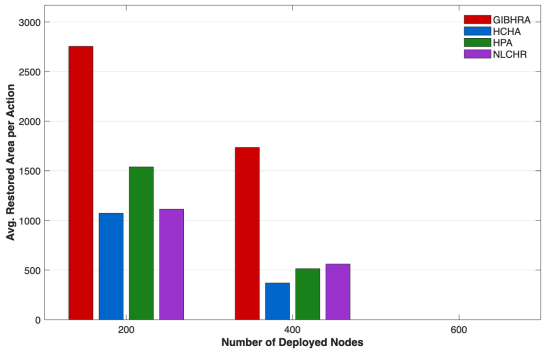
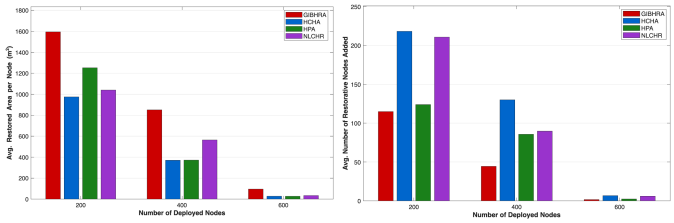


Fig. 6: Comparison of the average newly covered area (m²) per deployed node across varying initial network densities.



(a) (b)

Fig. 7: Comparison of algorithm deployment cost. Figure (a) shows the average restored area per deployed node. Figure (b) shows the number of nodes added to achieve 98% coverage across different initial network densities.

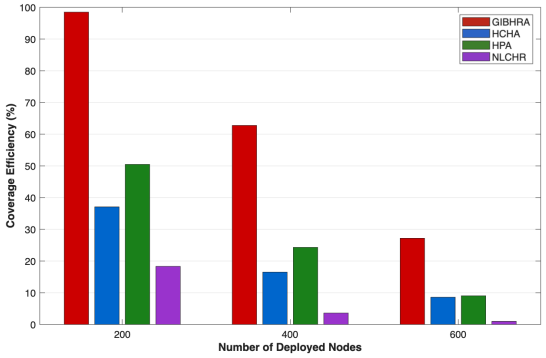


Fig. 8: Comparison of coverage efficiency of newly deployed nodes, measuring the percentage of each node's sensing area that covers uncovered holes across varying network densities

VI. Conclusion

This paper introduced GIBHRA, a novel algorithm that tackles the critical problem of coverage hole restoration in WSNs by applying geometric inversion to find a geometrically optimal solution. Through extensive simulation, we demonstrated that GIBHRA is decisively superior to standard heuristics, achieving a near-perfect 99.2% coverage efficiency while requiring up to 48% fewer restorative nodes than other algorithms. These results validate that a mathematically rigorous approach provides a more effective and resource-efficient solution than simpler geometric heuristics and presents a significant advancement in network coverage optimization. This work demonstrates how inversion can be used to effectively solve complex coverage optimization problems in WSNs and how it has strong potential in the field of computational geometry. Future work can be extended to mobile sensor nodes, heterogeneous sensor nodes, and networks with obstacles.

References

[1] V. Patil and S. Deshpande, "Design of FPGA soft core based WSN node using customization paradigm," *Wireless Pers. Commun.*, vol. 122, pp. 783–805, 2022.
[2] C. Qiu, H. Shen, and K. Chen, "An energy-efficient and distributed cooperation mechanism for k-coverage hole detection and healing in WSNs," in *Proc. IEEE 12th Int. Conf. Mobile Ad Hoc and Sensor Syst.*, Dallas, TX, USA, 2015, pp. 73–81.
[3] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng, "Efficient placement and dispatch of sensors in a wireless sensor network," *IEEE Trans. Mobile Comput.*, vol. 7, no. 2, pp. 262–274, Feb. 2008.
[4] B. Khalifa, A. M. Khedr, and Z. Al Aghbari, "A coverage maintenance algorithm for mobile WSNs with adjustable sensing range," *IEEE Sensors J.*, vol. 20, no. 3, pp. 1582–1591, Feb. 2020.
[5] A. Ghosh and S. K. Das, "Coverage and connectivity issues in wireless sensor networks: A survey," *Pervasive Mobile Comput.*, vol. 4, no. 3, pp. 303–334, 2008.
[6] S. M. Mohamed, H. S. Hamza, and I. A. Saroit, "Coverage in mobile wireless sensor networks (M-WSN): A survey," *Comput. Commun.*, vol. 110, pp. 133–150, 2017.
[7] F. Yan, W. Ma, and F. Shen, "Connectivity based k-coverage hole detection in wireless sensor networks," *Mobile Netw. Appl.*, vol. 25, pp. 783–793, 2020.
[8] C. Zhang, Y. Zhang, and Y. Fang, "Localized algorithms for coverage boundary detection in wireless sensor networks," *Wireless Netw.*, vol. 15, no. 1, pp. 3–20, Jan. 2009.
[9] R. Courant, H. Robbins, and I. Stewart, *What Is Mathematics?: An Elementary Approach to Ideas and Methods*. New York, NY, USA: Oxford Univ. Press, 1996.
[10] P. K. Sahoo, M. J. Chiang, and S. L. Wu, "An efficient distributed coverage hole detection protocol for wireless sensor networks," *Sensors*, vol. 16, no. 3, p. 386, 2016.
[11] D. Gisch and J. Ribando, "Apollonius' problem: A study of solutions and their connections," *Am. J. Undergrad. Res.*, vol. 3, pp. 1–7, 2004.
[12] P. Levrie, "A straightforward proof of Descartes's circle theorem," *Math. Intelligencer*, vol. 41, pp. 24–27, 2019.
[13] J. B. Wilker, "Four proofs of a generalization of the Descartes circle theorem," *Amer. Math. Monthly*, vol. 76, no. 3, pp. 278–282, 1969.
[14] I. E. Leonard, J. E. Lewis, F. Liu, and G. W. Tokarsky, *Classical Geometry*. Hoboken, NJ, USA: Wiley, 2014.
[15] G. Zhang, C. Qi, W. Zhang, J. Ren, and L. Wang, "Estimation and healing of coverage hole in hybrid sensor networks: A simulation approach," *Sustainability*, vol. 9, no. 10, p. 1733, 2017.
[16] V. Shakhov and I. Koo, "Depletion-of-battery attack: Specificity, modelling and analysis," *Sensors*, vol. 18, no. 6, p. 1849, 2018.
[17] G. H. Adday, S. K. Subramaniam, Z. A. Zukarnain, and N. Samian, "Fault tolerance structures in wireless sensor networks (WSNs): Survey, classification, and future directions," *Sensors*, vol. 22, no. 16, p. 6041, 2022.
[18] S. Devi, A. Sangwan, A. Sangwan, M. A. Mohammed, K. Kumar, J. Nedoma, R. Martinek, and P. Zmij, "The use of computational geometry techniques to resolve the issues of coverage and connectivity in wireless sensor networks," *Sensors*, vol. 22, no. 18, p. 7009, 2022, doi: 10.3390/s22187009.
[19] S. Babaie and S. Pirahesh, "Hole detection for increasing coverage in wireless sensor network using triangular structure," *Int. J. Comput. Sci. Issues*, vol. 9, no. 6, pp. 161–165, 2012.
[20] J. Yao, G. Zhang, J. Kanno, and R. Selmic, "Decentralized detection and patching of coverage holes in wireless sensor networks," in *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 7352, 2009, Art. no. 73520B, doi: 10.1117/12.819294.
[21] M. Wu, "An efficient hole recovery method in wireless sensor networks," in *Proc. 24th Int. Conf. Adv. Commun. Technol. (ICACT)*, PyeongChang, Republic of Korea, 2022, pp. 1399–1404, doi: 10.23919/ICACT53585.2022.9728957.
[22] B. Khalifa, Z. Al Aghbari, and A. M. Khedr, "A distributed self-healing coverage hole detection and repair scheme for mobile wireless sensor networks," *Sustain. Comput. Inf. Syst.*, vol. 30, p. 100428, 2021, doi: 10.1016/j.suscom.2020.100428.