Sensor Deployment by a Robot in an Unknown Orthogonal Region: Achieving Full Coverage

Eduard Mesa-Barrameda †, Nicola Santoro †, Wei Shi §, Najmeh Taleb †

†*School of Computer Science*
Carleton University, Canada
§*Faculty of Business and Information Technology*
University of Ontario Institute of Technology, Canada
E-mail: emesabarrameda@gmail.com, santoro@scs.carleton.ca, wei.shi@uoit.ca, najmehtaleb@hotmail.com

***Abstract*—When deploying a wireless sensor network in an unknown environment, commonly referred to as Region of Interest (ROI), the main goal is for the entire region to be covered by the sensing ranges of the deployed sensors. While this goal of *full coverage* is easily achieved in presence of human intervention, it becomes problematic if the region is dangerous or inaccessible to human. An approach recently proposed to solve the problem is to use a *robot* to deploy the sensors; the main advantages respect to the alternative of employing *mobile sensors* are the reduced costs (due to manufacture and maintenance cost of common static sensors vs. mobile ones) and the reduced complexity of the coordination and control algorithms. Indeed several solution algorithms to achieve deployment of sensors by a robot in an unknown region have been proposed in the literature. Unfortunately, even when restricted to orthogonal regions (e.g., city maps, building plans, etc), all the existing algorithms fail to achieve full coverage of the ROI. Specifically, following the existing protocols, the robot would leave uncovered areas near either the boundaries or *critical areas* (e.g. areas that are linked to the rest of the region by a narrow corridor).**

In this paper we present an algorithm that overcomes these problems and guarantees that the deployment of the sensors by the robot achieves *full coverage* in any simply connected orthogonal ROI, whose topology is unknown to the robot.

The proposed algorithm has minimal requirements: it does not need GPS but only local orientation by the robot; the communication range of a deployed sensor is limited to its deployed neighbours, and the robot has a similar range; the total number of sensors used is minimal. Also minimal are the robot's memory requirements, the total amount of robots movements and of communication between robot and sensors.

I. Introduction

A. Framework and Problem

When deploying a *wireless sensor network* (WSN) in a *region of interest* (ROI), one of the most important goals is to ensure that the sensors are able to *cover* the region. However, in environments that are possibly hazardous or inaccessible to humans, deployment typically is performed by random scattering of static sensors over the region (e.g., from the air); coverage can only be achieved probabilistically and by employing a massive amount of sensors, way larger than what really needed in a precision deployment. In spite of the large cost in terms of number of sensors, with this approach, coverage of the ROI cannot be guaranteed.

To overcome this problem, there are two basic approaches. The first is to endow the sensors with locomotion capabilities so that they can scatter in the region autonomously so to achieve a desired degree of coverage. Indeed, the study of self-deployment of *mobile sensors* is a promising research area (e.g., see [1]–[10]). However, mobile sensors are clearly more expensive than static ones, and their mobility is in any case rather limited and energy-consuming. Furthermore, their coordination to cover an unknown environment efficiently is a complex task, and a very costly one in terms of number of mobile sensors, the amount of time used to finish the task, and the number of messages being exchanged between the mobile sensors, etc (e.g., see [11]).

The other approach to solve this problem is to use *mobile robots* to deploy standard sensors: a robot capable of accessing areas of the ROI inaccessible to humans, can carefully and precisely deploy the sensors in the region so to (ideally) achieve maximum coverage with minimal number of sensors [11]–[19]. In this paper we are interested in solving the coverage problem using this approach with a single robot.

There are few algorithms in the literature that provide sensor deployment by a mobile robot [20]. They operate by superimposing a (triangular or hexagon or square) grid on the ROI, typically assumed to be an orthogonal region; the robot deploys the sensors on the vertices of the grid.

However, the existing protocols do not fully solve the problem. First of all, in some of them, the robot is only capable of deploying sensors in elementary regions: rectangles, squares or circles (e.g., [21], [22]. In less elementary orthogonal regions, the existing algorithms may allow the sensors to cover a possibly large part of the region, but they fail to achieve *full coverage*: after the sensors' deployment by the mobile robot, some areas of the ROI will remain not covered by any sensor. This is due to two primary problems during the deployment. In some cases the robot gets stuck in a *dead end* due to pre-deployed sensors or boundaries, ending the deployment while some parts of the environment are still uncovered (e.g., [11], [13], [14], [19]). In other cases, the robot fails to leave sensors to cover some areas near the *boundaries*, especially at the *corners* (e.g. [14], [18]).

In addition to the above problems, *all* the existing algorithms fail to cover a region that contains *critical areas*, that is an area linked by a narrow corridor (see Figure 3). In fact, any such area will render the grid graph employed by the

978-1-4799-7615-7/14/$31.00 ©2014 IEEE

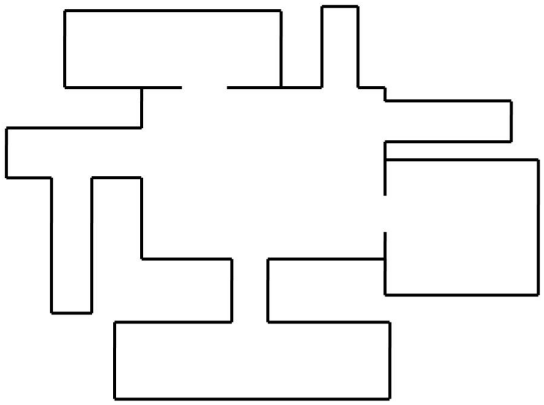

Fig. 1. An arbitrary Othogonal Region of Interest with Problematic Areas.

algorithm *disconnected* for some initial locations of the robot; thus, (possibly large) part of the region will not covered.

The research goal is to develop a protocol that overcomes all these problems and guarantees that the deployment of the sensors by the robot achieves full coverage in any orthogonal ROI, whose topology is unknown to the robot.

B. Main Contributions

In this paper we present an algorithm that allows a robot to deploy sensors so to ensure *full coverage* of a arbitrary simply connected (i.e., without holes) orthogonal region, whose topology is a priori unknown.

The protocol $FullCoverageDeployment$ allows the robot to detect not only all possible problems with boundaries and corners, but also (and more importantly) all the critical areas, and to handle all these problematic regions correctly and efficiently. In fact, the deployed sensor network not only achieves full coverage of the region, but it does so with a *minimal* number of sensors. In fact, the removal of any sensor deployed by the robot would create a sensing hole in the region.

The requirements for our solution are minimal.

The robot has neither a map of the environment nor a GPS. It is solely capable of local orientation; that is it can distinguish the four directions North, West, South, East (e.g., as provided by a compass). It has a limited sensing range within which it can detect the region's boundaries and the already deployed sensors. The memory requirements of the robot are very limited; in fact during its operation it does not construct a map of the region, and it does not remember the details of the previous deployment. Indeed it can operate with only $O(\log n)$ bits of memory, where n is the total number of deployed sensors.

The communication range of a deployed sensor is limited to its deployed neighbours, and the robot has a similar range. The total amount of communication between the robot and the sensors is $O(n)$ messages.

The total amount of robot's movements (each covering the distance between two neighbouring sensors) and thus of energy spent in movement is $O(n)$, which is optimal.

C. Related Work

Few works have studied sensor deployment in Wireless Sensors and Robot Networks (WSRN). LRV (Least Recently Visited) is a deployment and coverage maintenance algorithm [23], [24]. In this algorithm the robot communicates with sensors, and they give a direction to the robot for sensor's placement. LRV algorithm requires many unnecessary movements to deploy sensors in order to achieve full coverage. Moreover, these extra movements lead to an extremely large number of messages sent from the robots [17], [18]. Algorithm OFRD (Obstacle-Free Robot Deployment) in [13] and ORRD (Obstacle-Resistant Robot Deployment) in [11] introduce an SLD (Snake-Like Deployment) approach to solve the problem. In these algorithms with a serpentine movement, the robot deploys sensors in the environment. However, SLD algorithms do not guarantee full coverage because the robot gets stuck in dead ends due to obstacles or early deployed sensors [17], [25]. Moreover, it is not clear under what conditions the algorithms terminate [25]. OFPE (Obstacle-Free and Power-Efficient deployment) algorithm presented in [14] uses a single robot to explore the environment in a spiral parttern while deploying sensors. This algorithm does not guarantee full coverage due to previously described dead end problem [17], [18]. More over, the sensing hole problem near the boundaries of ROI is not addressed in this algorithm either. Shiu et.al. [19] introduce an algorithm using a single robot to deploy sensors in a concave region. The algorithm presented in [17] addresses the FOCUSED coverage (F-coverage) problem. In F-coverage, sensors surround a Point Of Interest (POI) and maximize coverage radius. In this algorithm, a robot deploys sensors on a hexagon grid layer by layer. In [15], [16] a flying robot (helicopter) deploys sensors on a desired network topology like star, grid or any random topology. BTD (Back-Tracking Deployment) algorithm introduced in [18] presents a dead end recovery policy to provide a back tracking method for situations that the robot is stuck due to early deployed sensors or obstacles. In this algorithm, a number of robots which are equipped with GPS are scattered randomly in an unknown bounded Region Of Interest (ROI). They are preloaded with static sensors and are able to detect obstacles and boundaries. This algorithm cannot achieve full coverage near the boundaries nor in any critical area. In the next section, we describe the model, its related assumptions and define the critical areas in an arbitrary orthogonal region and the grid that will be used to deploy sensors by a robot starting from an arbitrary point in this ROI.

II. Model and Definitions

A. Robot and Sensors

Let $\mathcal{R}$ be a simply connected orthogonal region (i.e., without holes), and let P be the orthogonal polygon of its boundaries The problem we consider and solve is that of a robot r, with no knowiege $\mathcal{R}$, deploying static sensors in $\mathcal{R}$ so that the

entire region is covered by the sensing ranges of the deployed sensors.

The robot r is a computational entity capable of moving in $\mathcal{R}$ and endowed with limited sensing capabilities, called *visibility*, communication capabilities.

The visibility radius $\rho(r)$ of the robot is at least twice the sensing radius $\rho(s)$ of the sensors ; i.e., $\rho(r) \geq 2\rho(s)$. Within its radius, r can see deployed sensors as well as boundaries of the ROI, but it might not be able to see across the region's boundary. Its communication radius $C(r)$ is at least as powerful as the communication radius $C(s)$ of a sensor; i.e., $C(r) \geq C(s)$. The robot can communicate only with the sensors deployed within its communication range, and this communication is not impeded by boundaries. The robot has available a local coordinate system, providing a consistent notion of the four directions North, South, East, West (e.g., as provided by a compass); it does *not* need a global localization mechanism (e.g., as provided by GPS). The robot r enters the region $\mathcal{R}$ from an arbitrary point, called *entry point*. The robot carries the static sensors that it can deploy at any point within $\mathcal{R}$; as in [18], it is assumed that the number of sensors carried by R is unbounded, i.e. sufficient to cover the entire region.

Each sensor s has limited sensing and communication capabilities. Let $\rho(s)$ denote the sensing range of a sensor; once deployed in $\mathcal{R}$, the sensing ability of the sensor may be compromised by any boundary in this sensing range. In other words, a sensor might not be able to sense the region behind the boundary of $\mathcal{R}$. Let $C(s)$ denote its communication range; we assume $C(s) \geq 2\rho(s)$; within this range, communication is not impeded by boundaries. In addition to communication with the robot, when required by the algorithm, deployed sensors periodically send a Hello message to neighbouring sensors.

The local memory of both the robot and the sensors are limited. In our protocol $O(logn)$ bits suffice.

B. Grid and Problematic Areas

Given a simply connected orthogonal region $\mathcal{S}$ and a point $u \in \mathcal{S}$, let $\mathcal{G}(\mathcal{S}, u)$ be an orthogonal grid, composed of square cells of length $\sqrt{2}\rho(s)$, logically superimposed on $\mathcal{S}$ according to the local coordinate system of the robot r, with u being location $(0, 0)$. Let $G(\mathcal{S}, u)$ be the sub-grid composed only of the vertices and edges of $\mathcal{G}(\mathcal{S}, v)$ that are inside $\mathcal{S}$. In the following, unless otherwise specified, with the term *grid* we will refer to G.

For grid vertex (x, y), we denote by $(x, y+1), (x+1, y+1), (x, y-1), (x-1, y-1)$ its neighbouring vertices (if they exist) in the directions North, East, South and West, respectively, as perceived by the robot.

With respect to the grid $\mathcal{G}(\mathcal{S}, v)$, there are three types of areas that present problems for a complete coverage of $\mathcal{S}$: *boundary holes*, *corner holes*, and *critical areas*; they indeed cause the existing protocols to fail and are formally defined as follows.

Definition 2.1: Boundary Hole.
Let d be the distance from a vertex v on the grid $\mathcal{G}(\mathcal{S}, v)$ to the boundary of $\mathcal{S}$ in direction $dir \in \{North, South, East, West\}$; if $\sqrt{2}/2\rho(s) < d' < \sqrt{2}\rho(s)$, we say that there exists a *boundary hole* in the area between v' and the boundary of $\mathcal{S}$ in direction dir.

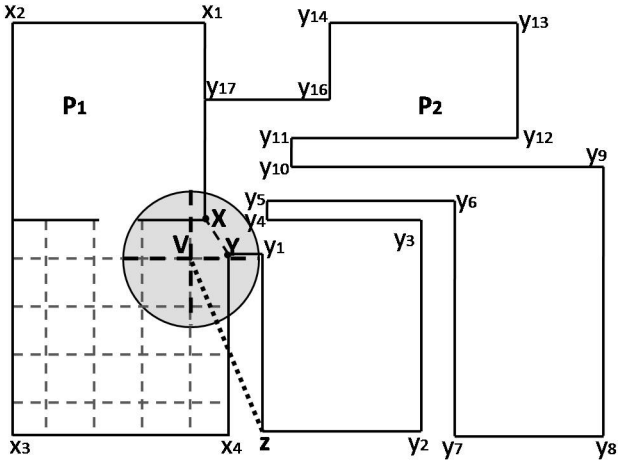

Fig. 2. Polygon P_2 contains critical area since line segment $\overline{zv}$ is located outside of its boundaries.

Definition 2.2: Corner Hole.
Let d' be the distance from a vertex v on the grid $\mathcal{G}(\mathcal{S}, v)$ to the boundary of $\mathcal{S}$ in direction $dir' \in \{NorthEast, SouthEast, NorthWest, SouthWest\}$.
If $\rho(s) < d' < 2\rho(s)$, we say that there exists a *corner hole* in the area between v' and the boundary of $\mathcal{S}$ in direction dir'.

Definition 2.3: Critical Area.
Let P denote the polygon that represents the boundary of $\mathcal{S}$. Let v be a vertex of the grid $\mathcal{G}(\mathcal{S}, v)$ and C_i be one of the four quadrants of circle C centered at v. Consider two points x and y on P determining a segment $\overline{xy}$ visible from v and entirely contained inside of P and C_i. This segment defines two polygons P_1^{xy} and P_2^{xy}. The first one starts from x and includes every vertex of P between x and y in counter-clockwise order, while the second one is obtained in the same way but starting from y. We say that polygon P_j^{xy}, $j \in \{1, 2\}$, is a *critical area* if and only if there is a vertex z on P_j^{xy} that satisfies any of the following conditions:

1) Segment $\overline{zv}$ is not totally inside P (provided that $\overline{zv}$ does not intersect P).
2) The length of $\overline{zv}$ is greater than $2\rho(s)$.

Notice that all the above definitions are constructive, in the sense that each provides an algorithmic detection mechanism.

III. Deployment Algorithm

A. Strategy and Structure

The naive strategy employed in the other algorithms, is to have the robot r traversing the edges of the grid (e.g., in a depth-first fashion), and deploying a sensor on any still empty vertex of the grid. Notice that, since the robot does not know $\mathcal{R}$ nor the grid, this strategy is naive but not necessarily trivial to implement; the dead end problem can however be easily avoided [18]. In any case, as mentioned in the introduction,

such a naive strategy fails to achieve full coverage in presence boundary holes, corner holes, and critical areas.

The proposed protocol, called *Full Coverage Deployment* (FCD), is built on top of the naive strategy with necessary rule modifications that, as we will show later, are sufficient to guarantee full coverage as well as minimality of the solution.

The robot will start exploring the ROI by traversing the grid, occasionally moving out of it to deal with boundary and corner holes. What is unique in this algorithm is the fact that it will also detect and handle the presence of critical areas. To deal with these areas, the algorithm uses not a single grid but rather a multiplicity of them, as explained later; however, at any time it is only using one. Moving on an edge of the current grid will be called a *regular movement*. During the deployment, the robot determines the presence of any problematic area. The action taken by the robot will depend on the nature of the found problems (critical area, boundary hole or corner hole); notice that more than one problem can be determined at the same time. Each of these situations is dealt by the robot sequentially and usually requires backtracking.

In general, information has to be stored by the robot to be processed later (e.g., which detected problem has to be dealt next, the path to be followed to backtrack, the location to move next, etc). In our algorithm, some of the needed information is distributed among the deployed sensors A deployed sensor keeps (up to) seven pieces of information: *coordinates*, *sequence number*, *state*, *colour*, *back pointer*, *status*, and ID of *supportive* sensor. The information given by the robot when deploying a sensor are the *coordinates* of the sensor with respect to the grid, as well as the number of already deployed sensors, called *sequence number*; notice that both coordinates and sequence number are unique, so can be used as the sensors ID. The robot also assigns to the sensor a *state*: *regular*, *boundary*, or *entrance*: a regular sensor (or R-sensor) is deployed on a grid vertex, a boundary sensor (or B-sensor) is deployed to overcome a boundary hole or a corner hole, and an entrance sensor (or E-sensor) is deployed at the entrance of a critical area. Additionally, a E-sensor has a *colour* and a *back pointer*; the colour is white if the sensor has an empty neighbouring grid location, black otherwise; the back pointer points to the location of the last white deployed sensor; both pieces of information are initially provided by the robot, and updated by the sensors. Moreover, each E-sensor has a *status*, covered or uncovered; it is covered if the critical area at whose entrance it is located has been fully deployed, and uncovered otherwise; it also keeps the ID of the sensor, called *supportive sensor*, from where its critical area has been detected; both pieces of information are provided (and updated in the case of status) by the robot.

Each sensor periodically sends a Hello message communicating its information to the sensors within its communication range; in this way sensors update their informations (in particular, back pointer and status). As a rule, E-sensors only consider the messages sent from their supportive sensor, to update their back pointer.

Before proceeding with the description of the the algorithm, let us describe an important thing on how critical areas are dealt with. Once a critical area is detected by the robot from a location u, an E-sensor will be deployed at the (center of the) corridor entrance, say u'; u' will act as a "virtual edge", called *entrance edge*, logically separating from $\mathcal{R}$ the region $\mathcal{R}'$ composed of the corridor and the region on the other side of it. The sub-region $\mathcal{R}'$ will be dealt with as if dealing with a separate ROI with u' as the starting point.

The protocol starts with the robot at the starting point calculating the grid and deploying a R-sensor. At the appropriate location u (initially, the starting point; subsequently always the location of R-sensor or E-sensor), the sensor performs the following steps in order.

Step 1- Detect critical areas and deploy E-sensors

In this step, the robot partitions the local area to detect any critical areas in its visibility radius and it computes the locations where E-sensors must be deployed (see Section III-C for details), creating a E-sensors deployment list. Using this list, the robot goes around and deploys E-sensors; for all these E-sensors, the sensor at u is their supportive one. After being deployed, an E-sensor sends a Hello message containing its information to its supportive and neighbouring sensors. At the end of this step, the robot returns to u and informs the sensor of the number of the (uncovered) critical areas around it.

Step 2- Detect boundary problems and deploy B-sensors

The robot creates a coverage map of its visibility area. To create this virtual map, the robots removes from consideration all the detected critical areas and considers their entrance edges as part of the boundaries. It puts on the map all the sensors already deployed in the visibility area, including the E-sensors it deployed in Step 1, as well as the R-sensor at u. Using the coverage map, the robot detects boundary and corner holes and computes the location of where B sensors should be deployed (as explained in Section III-B), creating a B-sensor deployment list. It then puts on the map a B-sensor on each of the computed locations. When the map is completed, the robot removes from the map (and the list) any redundant B-sensor. Using the list, the robot goes around to actually deploys B-sensors. At the end of this step, r goes back to u and discards map and lists from its memory.

Step 3- Cover critical areas

If critical areas were determined in Step 1, the robot deals with each of them sequentially. To deal with a critical area, the robot moves from (the supporting sensor at) u to the location u' of the E-sensor deployed (in Step 1) at the entrance of the corridor. The robot will enter the critical area and execute the algorithm to cover $\mathcal{R}'$ using u' as the starting point. Notice that the grid $G(\mathcal{R}', u')$ that the robot computes and uses for this area is different from the grid used before entering the corridor (see Figure 3). Once it has finished with its deployment in $\mathcal{R}'$, the robot will return to u, and act as if $\mathcal{R}'$ did not exist and the entrance to the corridor is closed by a boundary.

Step 4- Progress and Backtrack

If there are no (more) critical areas to be dealt with in the visibility area, the robot proceeds with its exploration of the uncovered area. It checks if there are neighbouring

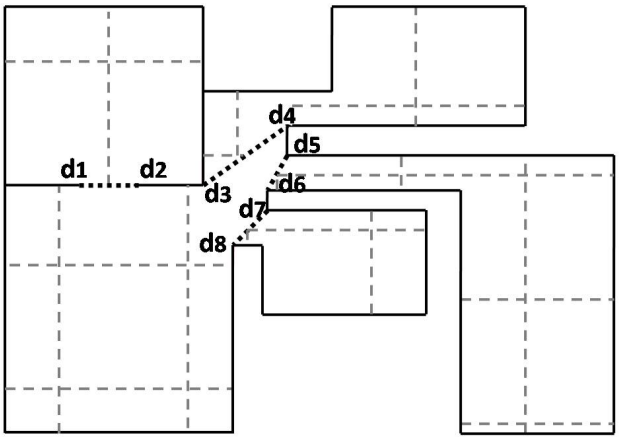

Fig. 3. Multiple grids.

grid locations still uncovered. If so, it chooses one (breaking symmetry using the order West, East, North and South on the directions), moves there, and starts the execution of these steps. Otherwise, it backtracks. For this purpose, it uses the back pointer of the sensor at u as the destination; it sends a message to all adjacent sensors asking for their back pointers. Then, it moves to the neighbouring sensor with the lowest ID whose back pointer location is the same as the robot's destination; this is the short cut method for back tracking developed in [18].

In the following two subsections we explain in more details how boundary and corner holes and critical area are detected and handled.

B. Boundary Handling Rules

To compute the coverage map efficiently, instead of the circle defined by the sensing range, we use the *effective coverage square* of a sensor [11], i.e., the square with length $\sqrt{2}\rho(s)$; notice that there are neither gaps nor overlaps between effective coverage areas of the sensors. Any redundancy created when calculating deployment locations of B-sensors using this approach will be detected on the map, and removed before the actual deployment takes place.

The procedure is rather simple:

1) For each of the main directions (East, West, North, and South) detect if a boundary hole exists, using Definition II-B: $\sqrt{2}/2\rho(s) < d < \sqrt{2}\rho(s)$ where d is the distance of u from the boundary (if any in the visibility range) in the given direction.
2) For each of the corner directions (Northeast, Southeast, Southwest, or Northwest) detect if a corner hole exists, using Definition II-B: $\rho(s) < d < 2\rho(s)$ where d is the distance of u from the boundary (if any in the visibility range) in the given direction.
3) Whenever a boundary or corner hole is detected, add its location to the B-sensors deployment list.

It is possible that the B-sensors list created by the corner holes is redundant, i.e., a corner hole would be actually covered by a combination of already deployed sensors and some B-sensor locations in the list. This is not a problem because, once the B-sensors are virtually deployed on the map, any redundancy will be detected by the robot and removed from the map before the actual deployment takes place.

Algorithm $FullCoverageDeployment(FCD)$
Main-Direction (North, East, South, West)
Corner-Direction (Northeast, Southeast, Southwest, Northwest)
Wake up
Locate first R-sensor
while *there are uncovered areas* **do**
  Decomposition ()
  if *critical area(s) detected* **then**
    Compute and store the location of all E-sensors within the visibility circle in the E-sensors list.
    Deploy E-sensors in the middle of detected entrance edges and mark the E-sensor as uncovered.
    Create-Coverage-Map ()
  end
  else
    Boundary-Handling-Rule ()
  end
  if *there is uncovered critical area* **then**
    Cover-critical-area()
  end
  if *Dead-End* **then**
    Back-Track (). **if** *there is uncovered critical area* **then**
      Cover-critical-area()
    end
    else
      Do Regular Movement.
    end
  end
  Regular Movement.
end

C. Detecting and Handling Critical Areas

It is possible that within the visibility radius of the robot there are several critical areas. To properly detect and identify each of them, in addition to the detection mechanism provided by Definition 2.3 we employ a set of partitioning rules to segment the area.

The overall procedure is as follows. Consider the visibility circle of robot r shown in Figure 4(a) as the highlighted circle.

1) Divide the visibility circle of robot r into four quadrants.
2) Create, for each quadrant, a *visibility polygon* V; in our case, this is a simple polygon containing all points of the quadrant visible from r's position. Figure 4(b) shows the visibility polygon ($p_0p_1p_2...p_{11}$) for quadrant 1 which contains multiple critical areas.
3) Decompose the visibility polygon V, using the idea

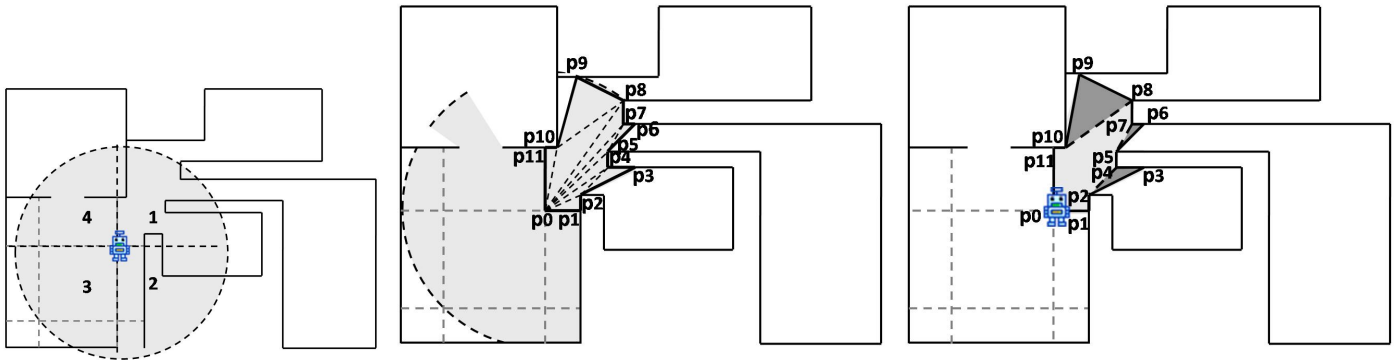

Fig. 4. (a) Visibility circle of the robot and dividing it to four quadrants 1, 2, 3, and 4. (b) Created visibility polygon ($p_0p_1p_2...p_{11}$) for quadrant 1 which contains multiple critical areas. (c) The polygon ($p_0p_1p_2p_4p_5p_7p_8p_{10}p_{11}$) denotes the main polygon and line segments $\overline{p_2p_4}$, $\overline{p_5p_7}$, and $\overline{p_8p_{10}}$ are Entrance edges.

presented in [26] adapted to our particular problem. Specifically,

a) Triangulate V by connecting any vertex p_i of V to the robot position (vertex p_0) unless there is a vertex p_j laying on the segment $\overline{p_0p_i}$. If there are still some pieces of V with more than 3 vertices, keep triangulating them. Figure 4(b) also shows the triangulation process of the visibility polygon.

b) Merge all the triangles adjacent to p_0 forming a polygon $\hat{V}$ (called main polygon), which divides V in several disjoint polygons.

4) Report any edge of $\hat{V}$ that is not an edge of V and where both its extremes are laying on the boundary of the ROI as critical areas. These reported edges are called *entrance edges*. Figure 4(c) illustrates the main polygon and the entrance edges of the detected critical areas.

5) Compute the middle of each entrance edge and add it as the location of an E-sensor in the E-sensor list.

IV. Correctness of Algorithm Full Coverage Deployment (FCD)

As illustrated in the previous section, algorithm *FullCoverageDeployment* partitions the ROI into disjoint pieces; and defines a new grid for each sub-ROI to deploy sensors applying the same algorithm. In the following section first we prove that algorithm *FullCoverageDeployment* totally covers a sub-ROI. Then we will show that algorithm *FullCoverageDeployment* covers the entire ROI by showing that all the other sub-ROIs are covered as well.

First of all let see in more details partitioning rules, since it will help us to proof the correctness of the algorithm.

Partitioning rules in detail: Let $v = p_0,p_1,...,p_k$ the vertices in P_v ordered in counterclockwise where P_v is the visibility polygon created at current vertex v (note that P_v is the created visibility polygon for one of the quadrants of the visibility circle). For any point p_j, if the segment $\overline{vp_j}$ is a diagonal in P_v then the segment is added to the triangulation. If there is a point p_i such that the segment $\overline{vp_i}$ is not a diagonal in P_v, then since every vertex in P_v is visible from v, either $\overline{p_ip_{i+1}}$ or $\overline{p_{i-1}p_i}$ is collinear with v. If no such cases happen, the polygon is completely triangulated and every triangle contains v, so that no edge is reported as entrance to a critical area. Before we continue the triangulation process, note that as a rule it is not possible for two consecutive edges in P_v to be collinear with v. This is because, if that case happens, the common vertex would be removed from P_v. Furthermore, if we consider three alternate edges (their extremes are consecutive vertices of P_v), since the ROI is orthogonal, it is not possible that all of them become collinear with v. Thus let us consider what happens if two alternate edges $\overline{p_{i-1}p_i}$ and $\overline{p_{i+1}p_{i+2}}$ are collinear with v.

Let $\hat{P}$ be a set such that for any $p_i \in \hat{P}$, $p_i \in P$ and the segment $\overline{vp_i}$ is not a diagonal of P_v (either $\overline{p_{i-1}p_i}$ or $\overline{p_ip_{i+1}}$ is an edge in P_v collinear with v (Figure 5(a))). Since the ROI is orthogonal the only possible case is that $p_i, p_{i+1} \in \hat{P}$ and $p_{i-1}, p_{i+2} \notin \hat{P}$. In this case, the convex quadrilateral $\overline{p_{i-1}p_ip_{i+1}p_{i+2}}$ is cut from the rest by the edge $\overline{p_{i-1}p_{i+2}}$, and this edge is reported as an entrance to a critical area. If there is an isolated collinear edge (none of the two prior and next edges are collinear (Figure 5(b))), then we take it's extreme p_i belonging to $\hat{P}$. The edge $\overline{p_{i-1}p_{i+1}}$ separates the triangle $\Delta p_{i-1}p_ip_{i+1}$ from the rest, so this edge is reported as an entrance to the critical area providing that both p_{i-1} and p_{i+1} belong to the boundary of the ROI P.

Lemma 4.1: Partitioning rules only detect critical areas.

Proof: According to the partitioning rules, any reported edge $\overline{xy}$ is completely visible from v and totaly contained in P, where v is the current vertex from where the critical area is being reported, and P denotes orthogonal polygon determined by the boundary of the ROI. Furthermore, both x and y lie on P. To prove the lemma we need to show that polygon P_j^{xy} separated by $\overline{xy}$ which does not contain v is a critical area. According to the definition of critical area, P_j^{xy} is a critical area if we can find a vertex z such that the segment $\overline{zv}$ is not totally contained inside P, or z is at distance greater than $2\rho(s)$. However, if an edge $\overline{xy}$ is reported by decomposition-procedure then, at least one of the points x or y (let say x) is the extreme of an edge $\overline{xx'}$ in the visibility polygon P_v which

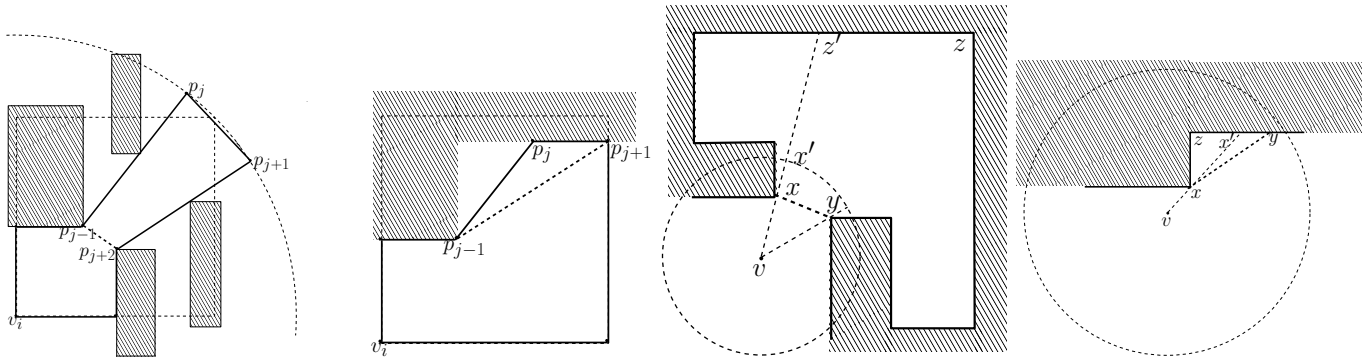

Fig. 5. (a) The quadrilateral $\overline{p_{i-1}p_ip_{i+1}p_{i+2}}$ is cut from ROI, and edge $\overline{p_{i-1}p_{i+2}}$ is reported as an entrance edge. (b) The triangle $\Delta p_{i-1}p_ip_{i+1}$ is cut from ROI, and edge $\overline{p_{i-1}p_{i+1}}$ is reported as an entrance edge. (c) Line segment $\overline{xx'}$ collinear with v, and x' does not belong to P. (d) Line segment $\overline{xx'}$ collinear with v, and x' belongs to P.

is collinear with v (Figure 5(c)). Since the ROI is orthogonal, $\overline{xx'}$ cannot belong to P. Moreover, as it was mentioned x belongs to P. Hence, only two cases could happen. Either x' belong to P or not.

First let us assume x' does not belong to P, then it is at distance $2\rho(s)$ (Figure 5(c)). In this case we could extend the ray starting from v and passing trough x' until it touches P at a point z' in P_j^{xy} farther than $2\rho(s)$. If z' is one of the vertex of P_j^{xy} then z' would be z. But if z' is not a vertex of P_j^{xy}, we take the furthest vertex among the extremes of the edge in P_j^{xy} containing z'. So, we found a vertex (z) in P_j^{xy} at distance greater than $2\rho(s)$ which means P_j^{xy} is a critical area so that the reported edge $\overline{xy}$ is an entrance edge.

Now, let us assume that x' belongs to P (Figure 5(d)). In this case, since $\overline{xx'}$ does not belong to P, we can say that segment $\overline{xx'}$ defines a partition of the polygon P_j^{xy}. We show this partition by $P_j^{xx'}$ which does not contain neither the point y nor v. Note that in this polygon ($P_j^{xx'}$), only the vertices x and x' are visible from v. Thus, if we take a vertex z in $P_j^{xx'}$ distinct from x and x' the segment $\overline{vz}$ will not be totaly contained in P. Hence, since $P_j^{xx'}$ is a partition of P_j^{xy}, P_j^{xy} is a critical area so that the reported edge $\overline{xy}$ is an entrance edge in this case as well. ■

Lemma 4.2: Algorithm *FullCoverageDeployment* detects every critical area.

Proof: In order to prove the lemma, let us assume by contradiction that, after termination, there is a critical area P_j^{xy} from the point of view of a vertex v that is not detected. Let us assume without lost of generality that P_j^{xy} is minimum (there are no other segment $\overline{x'y'}$, visible from v, defining a critical area $P_j^{x'y'}$, such that $P_j^{x'y'}$ is contained in P_j^{xy}). Note that, in this case, detecting a minimum critical area means detecting the entrance edge of it, or visiting it completely from one vertex of the grid. Since P_j^{xy} is minimum, two cases can happen wether $\overline{xy}$ is collinear with v or not.

First we consider the case when $\overline{xy}$ is not collinear with v. In this case according to the partitioning rules $\overline{xy}$ is reported as entrance edge which is a contradiction. This fact is illustrated in Figures 6 (a) and (b).

Now we consider the case that the line segment $\overline{xy}$ is collinear with v. Since P_j^{xy} is minimum, just the edge $\overline{xy} \in P_j^{xy}$ is visible from v, and no other point inside P_j^{xy} could be seen from v. So, the line segment $\overline{xy}$ which is collinear with v is part of an edge in the visibility polygon P_v. Consider the edge $\overline{XY}$ of P_v containing $\overline{xy}$, such that X is between v and Y. Note that X could be equal to x and Y to y (Figure 6 (c)). According to the partitioning rules, there is a vertex O in P_v such that the edge $\overline{XO}$ is reported as entrance edge to P_j^{xy} provided that O belongs to P. Thus, if O belong to P the edge would be reported and we have a contradiction.

Thus, let us assume that O does not belong to P. In this case there is a grid vertex v' between v and y such that the segment $\overline{xy}$ is also completely visible from v'. Otherwise, X would be connected with O on P. In this case P_j^{xy} might be completely visible from v' which is a contradiction (Figure 7(a)). So, let us consider the case that P_j^{xy} is not completely visible from v' (Figure 7(b)). If the segment $\overline{xy}$ does not define minimal critical area from v', then there is a segment $\overline{x'y'}$ such that from the point of view of v' the critical area $P_j^{x'y'}$ defined by this segment is minimal (Figure 7(c)). According to the partitioning rules an entrance edge will be reported for $P_j^{x'y'}$. Note that in this case some part of P_j^{xy} is completely visible from v', and for the other part which is not visible ($P_j^{x'y'}$) an entrance edge ($\overline{x'y}$) is reported which is a contradiction. Hence, every critical area is reported by algorithm *FullCoverageDeployment*.

■

Lemma 4.3: After removing all the detected critical areas from a sub-ROI, algorithm *FullCoverageDeployment* fully covers the remaining area.

Proof: Algorithm *FullCoverageDeployment* applies partitioning rules (Decomposition-Procedure) to detect critical areas. After detecting each entrance edge it will drop an E-sensor in the middle of it, enters the critical area and performs recursively the same algorithm to deploy sensors inside it. Let P' be the polygon obtained after removing every reported critical area from P. Note that we consider the entrance edges as edges of P'. Let G' be the same grid as G extended over the entire sub-ROI such that every square in G' contains at least

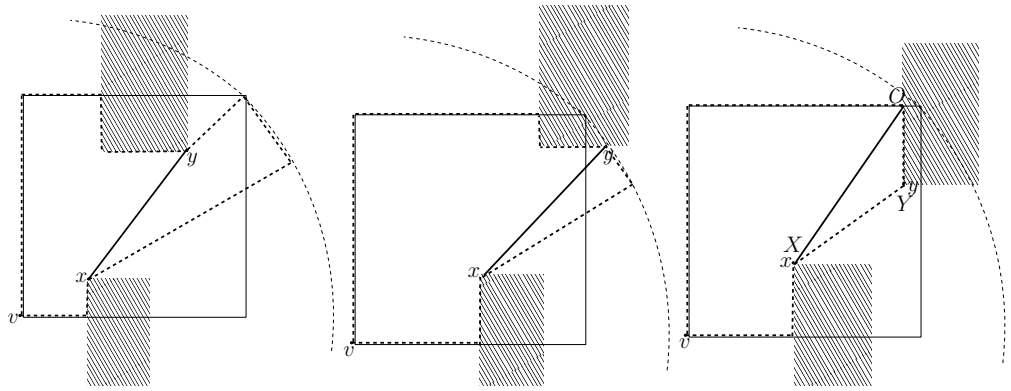

Fig. 6. (a) and (b) Line segment $\overline{xy}$, which is not collinear with v, is reported as the entrance edge of the critical area P_j^{xy}. (c) $\overline{XO}$ is reported as an entrance edge of P_j^{xy}.

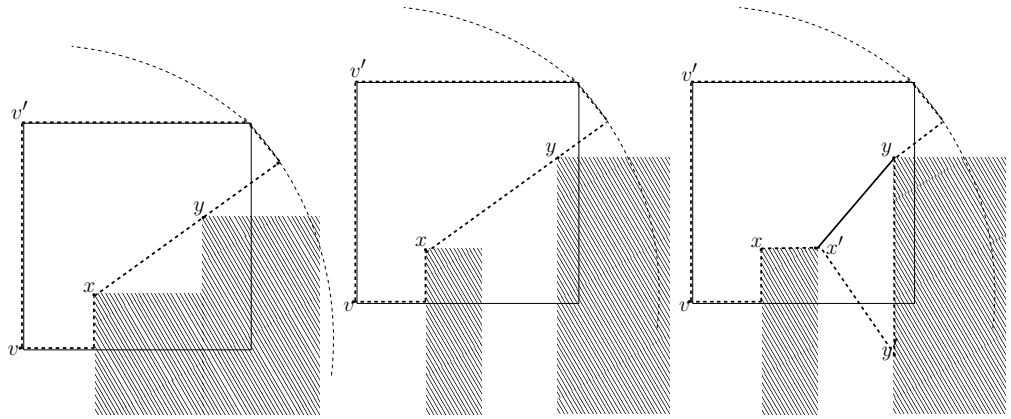

Fig. 7. (a) P_j^{xy} is completely visible from v'. (b) P_j^{xy} is the minimum critical area from v. (c) $\overline{x'y}$ is reported as an entrance edge of $P_j^{x'y'}$

one vertex of G. We first assume that the cell S is intersected by P. By contradiction, we assume that there is a point q that is still uncovered after termination of the algorithm. In this case q is located in one of the squares S_i where v_i is not accessible. By not accessible we mean that no R-sensor will be deployed on v_i because of the boundary, or v_i contains a R-sensor but S_i is disconnected by the boundary such that q and v_i are located in two disconnected areas. Note that, if this is not the case, q would be covered by the sensor located on v_i. Thus, we can assume that the region of S_i containing v_i is disconnected from the region of S_i containing q. Furthermore, since we only need to consider the region containing q, we can assume that there are only two disconnected regions of S_i.

We know then that S_i is disconnected in two different pieces; thus, the boundary must to intersect its border at least two times, otherwise it will remain connected. In this case the area containing q intersects at least one of the line segments joining v_i to the other three vertices of S, let say $\overline{v_i v_j}$. However, since this intersection occurs inside S_i, the robot will see the boundary along this segment at distance between $\frac{\sqrt{2}}{2}\rho(s)$ and $\sqrt{2}\rho(s)$ if v_i and v_j are adjacent or between distances $\rho(s)$ and $2\rho(s)$ if they are opposed. But, according to the boundary handing rules, it will consider to place a B-sensor on the segment $\overline{v_i v_j}$ next to the boundary (coverage map). So, there will be either a B-sensor or an E-sensor inside the region of S_i containing q covering it which is a contradiction. ■

Theorem 4.4: Algorithm $FullCoverageDeployment$ completely covers any complex ROI.

Proof: In order to prove the theorem let us first define a logically sub-division of the ROI forming a tree as it follows. The root of this tree will be the entire ROI containing the starting point. From this point the robot defines a grid, and according to lemma 4.2 it will detect all the critical areas for this grid. Critical areas are introduced by the entrance edges which divide the ROI into disconnected sub-ROIs. These sub-ROIs will be the children of the root. The starting point of each sub-ROI will be the middle point of the entrance edge. Applying the same idea for each sub-ROI we define the new nodes until we reach the leaves which are the sub-ROIs containing no critical areas from the grid defined at their starting point. Now we need to show that following the rules of the algorithm the robot traverses the entire tree and covers the entire ROI.

According to the algorithm, upon entering the ROI ($T[Root]$) and will move on the grid to cover the ROI. We know by lemma 4.2 that all the critical areas (the root's children) will be detected, and according to the algorithm each

Sub-Procedures $of\ Algorithm\ FCD$
Decomposition ()
Triangulate the visibility polygon.
Merge created triangles

Create-Coverage-Map ()
Remove all the detected critical areas and consider their entrance edges
as part of the boundary.
Drop all main and corner neighboring R-sensors, and E-sensors.
Considering the coverage square of all deployed sensors,
Check-Boundary-Handling-Rule ().

Cover-critical-area()
Ask the current sensor the location of next uncovered E-sensor.
Move to the location of E-sensor and inform it to change its status to
covered.
Applies algorithm B for the region inside the detected critical area.

Check Boundary-Handling-Rule ()
for *8 cardinal Directions starting from North* **do**
  if *in Main-Direction* $\sqrt{2}/2\rho(s) < d < \sqrt{2}\rho(s)$ **then**
    push the location of B-sensor in B-sensors List.
  end
  if *in Corner-Direction* $\rho(s) < d < 2\rho(s)$ ***and*** *no one of associate sensors drop a B-sensor in associate square* **then**
    push the location of B-sensor in B-sensors List (if the robot cannot see the associate sensors, it assumes they do not drop a B-sensor in associate square).
  end
end
while *B-sensors List is not empty* **do**
  Drop a B-sensor.
end
Return to last deployed R-sensor.

Back-Track ()
if *back-pointer of the current sensor is NULL* **then**
  Terminate algorithm.
end
else
  Back track to the back-pointer of the current sensor.
end

time a critical area is found the robot gets into it. When the robot enters into a child critical area X, it moves to its start point and places an E-sensor. The back pointer of this sensor will not be Null if and only if the tree $T[Root - X]$ (the sub tree containing the root but not X) is not totally covered. At this moment the robot is at the starting point of a disconnected sub-ROI. So, we have a smaller instance of the same problem in which the ROI is defined by the sub-Tree rooted at X. According to the algorithm, the robot does the same until it reaches a leaf of the tree ($T[Root]$). Since the leaf does not have any critical area, according to lemma 4.3 it will be eventually totally covered. At that moment the robot follows the back pointers to the starting point which directs the robot to the leaf's parent Y. Then the robot will continue moving along the grid defined by the starting point of Y until it finds a new critical area which means there is an unvisited child of Y, or the entire sub-ROI is covered. If there is still an unvisited child, the robot will move into it, otherwise it will back track to Y's parent. Therefore the robot will never leave a node until all its children and itself are covered. It implies the robot traverses the tree like a Depth-first search (DFS).

Now let us assume that the robot is at the moment in which it is getting into the last unvisited node which is a leaf. Note that, at that moment, any node which is not on the path from the Root to this leaf is totally covered. According to lemma 4.3 the robot will eventually completely cover the leaf and back tracks if the back pointer of the starting point is not Null. Then it will continue acting the same way (covering the node and back tracking) until it gets a node which after completely covering it the back pointer of its starting point is Null. At this moment two cases are possible either the node is the Root or not. Whatever the case every single node will be covered which means that the entire ROI is covered. ■

A. Complexity Analysis

Theorem 4.5: Algorithm FCD guarantees to achieve full coverage of an arbitrary orthogonal ROI using a single robot deploying minimal number of static sensors with $O(n)$ moves and messages and $O(\log n)$ memory.

V. Conclusions and Future Work

In this paper we have presented an algorithm, *Full Coverage Deployment*, for sensor deployment by a robot in an arbitrary simply connected orthogonal region, unknown to the robot. This algorithm is the first to achieve *full coverage*. It does so by identifying and characterizing the condition which makes the previous protocol fail: the presence of what we call *critical areas*. We provide a detection mechanism, and integrate it into a robot guidance protocol that allows it to achieve the desired task without any sensing hole. The protocol is very efficient. The number of deployed sensors is minimal. The robot requires minimal orientation capabilities, limited communication and sensing range (similar to those of a sensor), and very limited memory: $O(\log n)$ bits, where n is the number of deployed sensor. Furthermore, the total number of moves it

performs is optimal, and the total amount of communication by the robot is $O(n)$ messages.

There are several important direction for future research. Following are some:

- Immediate is the extension to have more than one robot; indeed, using a team of robots may speed up deploying process, though communication and synchronization of the team could be complex. [23], [27]
- In our algorithm, the number of robots is minimal, but not necessarily minimum. An interesting quest is to find a correct protocol that achieves full coverage with fewer sensors. One possible avenue could be to use a triangular or hexagonal grid instead of the square one used here.
- In this algorithm the robot and sensors use the same amount of memory: $O(\log n)$. The number of transmitted messages and robot's movement is linear in the number of deployed sensors. The number of sensors deployed in order to achieve the full coverage remains linear.
- Our protocol correctly covers an orthogonal ROI which is simply connected, i.e. it does not contains holes. It is clearly important to achieve coverage also in orthogonal regions that are connected but not simply. This particular problem is very challenging because the sensing holes created by obstacles may confuse the robot when detecting critical areas.

ACKNOWLEDGMENTS

This research is sponsored in part by the National Science and Engineering Research Concil (NSERC) of Canada under Grant No. 371977-2009 RGPIN.

REFERENCES

[1] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[2] M. Garetto, M. Gribaudo, C. Chiasserini, and E. Leonardi, "A distributed sensor relocation scheme for environmental control," *Proc. 4th IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, pp. 1–10, 2007.

[3] A. Howard, M. Mataric, and G. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[4] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, "Focused coverage by mobile sensor networks," *Proc. 6th IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems (MASS)*, pp. 466–475, 2009.

[5] ——, "Strictly localized sensor self-deployment for optimal focused coverage," *IEEE Trans. on Mobile Computing,*, vol. 10, no. 11, pp. 1520 – 1533, 2011.

[6] H. Mousavi, A. Nayyeri, N. Yazdani, and C. Lucas, "Energy conserving movement-assisted deployment of ad hoc sensor networks," *IEEE Communications Letters*, vol. 10, no. 4, pp. 269–271, 2006.

[7] S. Poduri, S. Pattern, B. Krishnamachari, and G. Sukhatme, "Using local geometry for tunable topology control in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 2, pp. 218–230, 2009.

[8] R. Ramadan, H. El-Rewini, and K. Abdelghany, "Optimal and approximate approaches for deployment of heterogeneous sensing devices," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no. 1, 2007, 14 pages.

[9] G. Wang, G. Cao, and T. L. Porta, "Movement assisted sensor deployment," *Proc. 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, pp. 2469–2479, 2004.

[10] S. Yang, M. Li, and J. Wu, "Scan-based movement-assisted sensor deployment methods in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1108–1121, 2007.

[11] C. Chang, C. Chang, Y. Chen, and H. Chang, "Obstacle-resistant deployment algorithms for wireless sensor networks," *IEEE Tran. On Vehicular Technology*, vol. 58, no. 6, pp. 2925–2941, 2009.

[12] M. Batalin and G. S. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 661–675, 2007.

[13] C. Chang, H. Chang, C. Hsieh, and C. Chang, "OFRD: Obstacle-free robot deployment algorithms for wireless sensor networks," *Proc. IEEE Wireless Communications and Networking Conf. (WCNC)*, pp. 4371–4376, 2007.

[14] C. Chang, J. Sheu, Y. Chen, and S. Chang, "An obstacle-free and power-efficient deployment algorithm for wireless sensor networks," *IEEE Tran. On Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 4, pp. 795–806, 2009.

[15] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Deployment and connectivity repair of a sensor network with a flying robot," *Springer Tracts in Advanced Robotics*, vol. 21, no. 2006, pp. 333–343, 2006.

[16] P. Corke, S. Hrabar, R. Petersony, D. Rus, S. Saripalli, and G. Sukhatme, "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3602 – 3608, 2004.

[17] R. Falcon, X. Li, and A. Nayak, "Carrier-based coverage augmentation in wireless sensor and robot network," *Proc. 7th IEEE Int. Workshop on Wireless Ad hoc and Sensor Networks (WWASN)*, 2010.

[18] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic, "Back-tracking based sensor deployment by a robot team," *Proc. 7th IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pp. 1–9, 2010.

[19] L. Shiu, "The robot deployment scheme for wireless sensor networks in the concave region," *Proc. IEEE Int. Conf. on Networking, Sensing and Control (ICNCS)*, pp. 581–586, 2009.

[20] M. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, vol. 26, no. 2, pp. 181–196, 2004. [Online]. Available: http://robotics.usc.edu/publications/369/

[21] J. Chen, S. Li, and Y. Sun, "Novel deployment schemes for mobile sensor networks," *Sensors*, vol. 7, no. 11, pp. 2907–2919, 2007.

[22] T. T. Lai, W. Chen, P. H. K. Li, and H. Chu, "Triopusnet: automating wireless sensor network deployment and replacement in pipeline monitoring," *Proc. 11th international conference on Information Processing in Sensor Networks*, pp. 61–72, 2012.

[23] M. Batalin and G. S. Sukhatme, "Sensor coverage using mobile robots and stationary nodes," *Proc. Int. Society for Optics and Photonics Engineering (SPIE)*, vol. 4868, pp. 269–276, 2002.

[24] ——, "Multi-robot dynamic coverage of a planar bounded environment," *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[25] X. Li, A. Nayak, D. Simplot-Ryl, and I. Stojmenovic, "Sensor placement in sensor and actuator networks," in *Handbook of Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. John Wiley, 2010.

[26] D. Avis and G. Toussaint, "An efficient algorithm for decomposing a polygon into star-shaped polygons," *Pattern Recognition*, vol. 13, no. 6, pp. 395–398, 1981.

[27] X. Li, G. Fletcher, A. Nayak, and I. Stojmenovic, "Placing sensors for area coverage in a complex environment by a team of robots," in *ACM Transactions on Sensor Networks*, 2014, p. To appear.