Collaborative Event Detection Using Mobile and Stationary Nodes in Sensor Networks

Theofanis P. Lambrou and Christos G. Panayiotou
Department of Electrical and Computer Engineering
University of Cyprus
Nicosia, Cyprus.
Email: {faniseng,christosp}@ucy.ac.cy

Abstract—Monitoring a large area with stationary sensor networks requires a very large number of nodes which with current technology implies a prohibitive cost. The motivation of this work is to develop an architecture where a set of mobile sensors will collaborate with the stationary sensors in order to reliably detect and locate an event. The main idea of this collaborative architecture is that the mobile sensors should sample the areas that are least covered (monitored) by the stationary sensors. Furthermore, when stationary sensors have a “suspicions” that an event may have occurred, they report it to a mobile sensor that can move closer to the suspected area and can confirm whether the event has occurred or not. An important component of the proposed architecture is that the mobile nodes autonomously decide their path based only on local information (their own beliefs and measurements as well as information collected from the stationary sensors in their communication range). We believe that this approach is appropriate in the context of wireless sensor networks since it is not feasible to have an accurate global view of the state of the environment.

Index Terms—Event detection, mixed sensor networks, path-planning, coverage holes.

I. INTRODUCTION

Recent progress in two seemingly disparate research areas namely, distributed robotics and low power embedded systems has led to the creation of mobile sensor networks [1]. Autonomous node mobility brings with it its own challenges, but also alleviates some of the traditional problems associated with static sensor networks. It is envisaged that in the near future, very large scale networks consisting of both mobile and static nodes will be deployed for applications ranging from environmental monitoring to military applications [2].

In this paper we consider the problem of monitoring a large area using wireless sensor networks (WSNs) in order to detect and locate an event. In this context, we assume that the event emits a signal that is propagated in the environment. The sensors capture attenuated and noisy measurements of the signal and the objective is to reliably detect the presence of the event and estimate its position. By reliably we mean that we would like to minimize the probability of miss event (an event that remains undetected) subject to a constraint on the probability of false alarms (the sensors report an event due to noise). Note that in many applications false alarms are as bad (if not worse) than missed events. In addition to the incurred cost for sending response personnel to the area of the event, frequent false alarms may lead the users to ignore all alarms and as a result even detected events will go unnoticed.

To achieve reliable detection in a large area, it is necessary to deploy a huge number of sensors which with the current technology implies a prohibitive cost [3]. For example consider a lake to be monitored for events (i.e. monitor water turbidity). If the lake has an area of $20km \times 20km$, and we assume that each sensor has a reliable sensing range (detection range) $r_d=10m$, then the number of sensor nodes needed to monitor the entire lake is of the order of $10^6$ which with today’s technology implies a prohibitive cost.

Given that it is infeasible to reliably cover the entire area with stationary nodes, in this paper we investigate an alternative way of monitoring the area using several stationary and some mobile sensor nodes that collaborate in order to detect an event as fast as possible. The main idea is that the mobile nodes will collaborate with the stationary nodes (and with each other) in order to sample areas that are least covered by the stationary nodes. In the context of WSNs, sensor nodes are fairly inexpensive and unreliable devices, thus it is not feasible to have an accurate state of each sensor node in the field (some nodes may have failed or been carried away by wind). As a result one cannot have all necessary information to centrally solve a path planning problem and predetermine the path that each sensor node should follow in order to sample the areas least covered. In our approach, mobile nodes navigate through the sensor field autonomously using only local information (i.e., the mobile node’s beliefs and measurements as well as information collected from the nodes, stationary or mobile, that are in its communication range).

The main contribution of this paper is that it proposes a distributed architecture where stationary and mobile sensors collaborate in order to reliably detect and locate an event. In the context of WSNs, several approaches exists for identifying the point where a mobile node should go in order to improve the coverage (for details see Section VII). All these approaches solve a static problem and to the best of our knowledge, none of them considers the path that the mobile node should follow in order to get to its destination. The path planning algorithms considered in this paper are motivated by the approach presented in [4] where two or more agents are moving in an area cooperatively searching for targets of interest and avoiding obstacles or threats.
The paper is organized as follows. Section II describes the model that has been adopted and the underlying assumptions. Section III presents an event detection architecture for mixed sensor networks. Section IV analyzes three algorithms for detecting coverage holes. Section V presents the dynamic path selection algorithm for mobile nodes and Section VI presents several simulation results using various sensor fields with randomly deployed sensor nodes. Section VII reviews related work in the area of coverage for both stationary and mobile sensor networks. The paper concludes with Section VIII where we also present plans for future work.

II. MODEL DESCRIPTION

We consider a set \( S \) with \( N = |S| \) stationary sensor nodes that are randomly placed in a rectangular field \( \mathcal{R}_x \times \mathcal{R}_y \) at positions \( x_i = (x_i, y_i), \ i = s_1, \cdots, s_N \). In addition, we assume that a set \( M \) of \( M = |M| \) mobile sensor nodes are available and their position after the \( k \)-th time step is \( x_i(k) = (x_i(k), y_i(k)), \ i = m_1, \cdots, m_M, \ k = 0, 1, \cdots \).

We assume that all sensors know their locations through a combination of GPS and localization algorithms.

An event source \( \mathcal{E} \), if present, is located at position \( x^e = (x^e, y^e) \) which is randomly placed inside the field. We assume that this event source emits a constant signal \( V \) in the surrounding environment. As we move away from the source, we assume that the measured signal is inversely proportional to the distance from the source raised to some power \( \alpha \in \mathcal{B} \subset \mathbb{R} \) which depends on the environment. As a result, the \( t \)-th measurement of sensor \( i \in \{N \cup M\} \) is given by

\[
    z_{i,t} = \min\left\{ V_{\text{sat}}, \frac{V}{r_i^\alpha} \right\} + w_{i,t} \tag{1}
\]

where \( V_{\text{sat}} \) is the maximum measurement which can be recorded by a sensor, \( r_i \) is the radial distance of sensor \( i \) from the source,

\[
    r_i = \sqrt{(x_i - x^e)^2 + (y_i - y^e)^2}, \tag{2}
\]

and \( w_{i,t} \) is additive Gaussian noise with zero mean and variance \( \sigma^2 \).

A sensor node reports that it has reliably detected the event if the average measurement it receives is greater than the detection threshold \( r_d \). This threshold is determined in a way such that the probability of false alarm is less than a given constraint \( p_{fa} \). This calculation can be done as in [3] but for the purposes of this paper, it is assumed that this threshold is given. This threshold defines a disc around the sensor with radius \( r_d \). If the event occurs within this disc, then it is assumed that it is reliably detected. For the purposes of this paper we assume that an event is detected by the network if at least one sensor (stationary or mobile) detects the event, however, other fusion rules are also possible, e.g., an event is detected if at least \( u \) sensors detect the event. Similarly, we assume that we are given a “suspicion” threshold \( \tau_s < \tau_d \) such that if the average measurement of the sensor \( i \), \( \tau_s \leq \bar{z}_i \leq \tau_d \), then the sensor does not report a detection, however, it may report that it “suspects” that there may be an event around its area. Note that \( \tau_s \) defines a disc around the sensor with radius \( r_s \leq r_d \), thus a node may report the suspicion if the event exists in the “donut” that is formed by the suspicion disc when the detection disc is removed. The event suspicion may be used in different ways. It can be reported to the sink which may fuse the information from several sensors or it can be given to a mobile node which will collaborate with the stationary sensors in order to move closer to the suspected event area to confirm the existence or not of the event. In this paper, the suspicion will be used as in the later example.

As far as the mobile nodes are concerned, each mobile node has a grid map of the field as shown in Fig. 1. The entire field area \( \mathcal{A} = \mathcal{R}_x \times \mathcal{R}_y \) is divided by congruent rectangles to make up a grid. Each cell in the grid can be addressed by index \((i,j)\) in two dimensions and each vertex has coordinates \((i \times dx, j \times dy)\) in 2D (in sensor field area) for some real numbers \(dx\) and \(dy\) representing the grid spacing or cell dimensions. For simplicity we set \(dx = dy = d\ell\) which means that the cells are square. The dimensions of the grid are \(X \times Y\) where \(X = [\mathcal{R}_x/d\ell]\) and \(Y = [\mathcal{R}_y/d\ell]\). Similarly, the detection range of each node \(r = [r_d/d\ell]\). Moreover, we use \( i = [x_i/d\ell]\) and \( j = [y_i/d\ell] \) in order to transform sensor coordinates \(x_i\) into indexes of the Grid. \([z]\) indicates the smallest integer greater or equal to \(z\). Also, for any cell \((i,j)\) we define a neighborhood as the set of all cells that are at a distance \(r\) from cell \((i,j)\), i.e., for all \(1 \leq i \leq X\), \(1 \leq j \leq Y\)

\[
    \mathcal{N}_r(i,j) = \{ p,q : (p-i)^2 + (q-j)^2 \leq r^2 \} \tag{3}
\]

where \(1 \leq p \leq X\), \(1 \leq q \leq Y\). In the memory of the mobile mode, the grid is represented by a matrix \(G\) where each entry \(g(i,j)\) in \(G\) represents the probability of detecting the event if the event has occurred in the area that corresponds to the \((i,j)\)-th square of the map. For simplicity, initially, every element of the matrix \(g(i,j) = 1\) for all cells that correspond to areas in the detection range of the stationary sensors and \(g(i,j) = 0\) otherwise.

As pointed out earlier, it is unlikely that the mobile will have an accurate picture of the state of all stationary sensors.
in the field. The main idea of our approach is to update the map as the mobile node moves around in the field. Thus if the mobile encounters a node not on the map, or if it discovers that a node on the map is no longer functioning, it updates the corresponding entries in the matrix $G$ appropriately. Furthermore, as it moves around, it also samples the environment and thus it increases the values of $g(i, j)$ that corresponds to sampled areas.

Note that in this paper we assume a static environment where a source (if present) is turned on at the beginning of the simulation time and stays on for the entire duration. Thus, simple updating rules are adequate, however, note that for a more dynamic environments where sources turn on and off dynamically, other updating rules are also investigated (e.g., at every time step, $g(i, j, k)$ is multiplied by an appropriate discounting factor). Another simplifying assumption we make in this paper is that all mobiles are “sharing” the same map, however, we point out that we also investigate the behavior of the proposed architecture when each mobile has its own map and study algorithms for merging these maps together when the mobiles come in a communication range.

Next, we present a generic collaborative architecture that can be used so that the mobile sensors can collaborate with the stationary ones.

### III. AN EVENT DETECTION ARCHITECTURE FOR MIXED SENSOR NETWORKS

Based on the assumptions of this paper, if an event occurs in the detection area of a healthy stationary sensor node, then it is immediately detected. However, when the tolerated probability of false alarm is small, then the detection area of each node is also small and thus for applications of monitoring large areas, it is likely that many areas are poorly monitored. The objective of the proposed architecture is to facilitate the collaboration between the mobile and stationary nodes in order to identify these poorly covered areas so that the mobiles can search them. Furthermore, the architecture allows for the collaboration between stationary and mobile nodes so that the latter can search areas for which some stationary nodes have reported “suspicion” in order to confirm the existence or absence of an event source.

In this context, each mobile can be in one of two modes, searching or navigating. Once a mobile node arrives at a large poorly covered area (coverage hole) it switches in the searching mode where it searches the entire hole area for an event. Such search algorithms have been developed in robotics and in autonomous unmanned vehicles (AUVs) fields, e.g., the standard (exhaustive) search, random search, and cooperative search approaches [4]. In all other cases, the mobile is in a navigating mode moving towards a target $x^*_i(k) = (x^*_1(k), y^*_1(k))$, $i = m_1, \ldots, m_M$. Note that the navigating mode is based on a receding horizon approach and thus, the target may change at every step $k$ as new information becomes available. Also, when in the navigating mode, a mobile can determine its target in either of two ways.

- If a node in the mobile’s communication range reports an event “suspicion”, then the target is the suspected area. In this case the target becomes the estimated location of the event which can be computed by the mobile using its own measurements as well as the measurements of all nodes in its communication range. For the purposes of this paper we use a least squares estimate [5], however, we point out that other algorithms could also be used.
- Otherwise, the target of the mobile node becomes the center of the biggest coverage hole (not assigned to other mobiles’). In Section IV we propose an efficient coverage hole detection algorithm that can be used by mobile nodes.

Finally, when in the navigating mode, the mobile determines its next step in such a way as to move towards the target, but also avoiding any areas that are covered by another mobile or any stationary node. In the next section we investigate three algorithms for determining the center of a coverage hole and in the following section we present the specific path finding algorithm employed by each mobile node when in the navigating mode.

### IV. COVERAGE HOLE DETECTION ALGORITHMS

The coverage of a WSN represents the quality of surveillance that the network can provide, for example, how well a region of interest is monitored by sensors, and how effectively a WSN can detect events. In this section we present two coverage hole detection algorithms and compare their complexity with the Grid Scan Algorithm [6]. Using a coverage hole detection algorithm a mobile node (or the sink)$^1$ can estimate the coordinates $x^*_i = (x^*_i, y^*_i), i \leq 1, \ldots, M$ of the $M$ biggest coverage holes centers (which become the target coordinates of the $M$ mobiles). Since this algorithm may run frequently (as new information regarding the state of the field becomes available) it is required that it is computationally efficient.

We define coverage $C$ as the probability of detecting an event which can occur uniformly in the sensor field. Using a grid similar to the grid map defined in section II, $C$ is given by

$$C = \frac{1}{X \times Y} \times \sum_{1 \leq i \leq X \atop 1 \leq j \leq Y} g(i, j). \quad (4)$$

For the purposes of this paper, we have already assumed that

$$g(i, j) = \begin{cases} 1 & \text{if } c(i, j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $0 \leq c(i, j) \leq N$ is the number of sensors (stationary or mobile) that cover the area of cell $(i, j)$. The aim of this section is to determine where the $M$ mobiles should be placed in order to maximize coverage (i.e., maximize $(4)$). This algorithm can be run by an individual sensor to determine its target destination or by a cluster head to determine the destinations of all mobiles in its range.

$^1$In this paper, since it is assumed that all mobile nodes share the same grid, it is easy to assign different coverage holes (targets) to different mobiles. In a distributed environment, various algorithms are investigated such as having a cluster head decide the assignment.
A. Grid Scan Algorithm

The Grid Scan algorithm [6] estimates the coverage holes as described below: For each available mobile node Grid Scan scheme finds \( c(i, j) \) for each cell \((i, j)\). Then, for each cell \((i, j)\), it finds the number of non-covered cells (i.e., \( c(p, q) = 0 \)), in the sensing range of radius \( r \). In other words, it counts the cells with \( c(p, q) = 0 \) in \( N_r(i, j) \). We denote this by \( h(i, j) \). The center of the hole is the center of the cell \( Z = (i, j) \) which has the maximum number of neighboring non-covered cells, i.e., \( Z \) is the cell \((i, j)\) where

\[
Z = (i^*, j^*) = \arg \max_{1 \leq i \leq X, 1 \leq j \leq Y} \{h(i, j)\}. \tag{6}
\]

The details of the algorithm are listed in Fig. 2. Although the Grid Scan scheme estimates the coverage holes accurately, its computational complexity is fairly high for the context of WSNs. Another weakness of this algorithm appears in the case where more than one cells have the same maximum value. In that case the algorithm selects the first cell by default and as a result most estimated holes fall very near to each other.

The running time of Grid Scan, if \( X = Y = L \), is

\[
T_{GS}(M, L, N, r) = M \times \left( \left( \left( \left( N + \left( M - 1 \right) / 2 \right) \times L^2 \right) + \left( L^2 \times 2r \times 2r \right) \right) + L^2 \right). \tag{7}
\]

Given that \( N \gg M \) the computation complexity is

\[
O(M \times L^2 \times (N + r^2)) \tag{8}
\]

Grid Scan Algorithm

1: for each mobile sensor \( m \in M \)
2: \hspace{1em} for each sensor \( n \in S \)
3: \hspace{2em} for each cell \((i, j)\) \( \in (X, Y)\)
4: \hspace{3em} if \((i, j) \in N_r(x_m, y_m)\)
5: \hspace{4em} \( c(i, j) = c(i, j) + 1 \)
6: \hspace{3em} end
7: \hspace{2em} end
8: end
9: for each cell \((i, j)\) \( \in (X, Y)\)
10: \hspace{1em} for each \((p, q) \in N_r(i, j)\)
11: \hspace{2em} if \( c(p, q) = 0 \)
12: \hspace{3em} \( h(i, j) = h(i, j) + 1 \)
13: \hspace{2em} end
14: \hspace{1em} end
15: end
16: \( (i^*, j^*) = \arg \max \{h(i, j)\} \)
17: \( x_m = i^*, \ y_m = j^* \)
18: \( S = S \cup \{m\} \)
19: end

Fig. 2. Pseudo code for Grid Scan Algorithm

B. One Scan Algorithm

The One Scan algorithm is a simple heuristic that improves the computational efficiency of the Grid Scan algorithm. One Scan scheme finds the \( c(i, j) \) value for each cell in the Grid and for each cell \((i, j)\) it finds the number of the neighboring non covered cells \( c(p, q) = 0 \) in the sensing range of radius \( r \), i.e \( h(i, j) \) values. This computation take place only once. The new detected hole is the center of the cell \( Z = (i, j) \) which has the maximum number of neighboring non covered cells, i.e \( Z \) is the cell \((i, j)\) given by (6). Subsequently, we set \( h(i, j) = 0 \) for all cells that are in a neighborhood with radius \( 2r \) from the detected hole center. Finally, we continue finding the next \( Z \) using equation (6) until we determine all required coverage holes. The details of the algorithm are listed in Fig. 3. The running time of One Scan if \( X = Y = L \) is

\[
T_{OS}(M, L, N, r) = \left( N \times L^2 \right) + \left( L^2 \times (2r)^2 \right) + M \times ((4r)^2 + L^2) \tag{9}
\]

and since \( N \gg M \), the computational complexity will be

\[
O(L^2 \times (N + M + r^2)) \approx O(L^2 \times (N + r^2)) \tag{10}
\]

One Scan Algorithm

1: for each sensor \( n \in S \)
2: \hspace{1em} for each cell \((i, j)\) \( \in (X, Y)\)
3: \hspace{2em} if \((i, j) \in N_r(x_n, y_n)\)
4: \hspace{3em} \( c(i, j) = c(i, j) + 1 \)
5: \hspace{2em} end
6: \hspace{1em} end
7: end
8: for each cell \((i, j)\) \( \in (X, Y)\)
9: \hspace{1em} for each \((p, q) \in N_r(i, j)\)
10: \hspace{2em} if \( c(p, q) = 0 \)
11: \hspace{3em} \( h(i, j) = h(i, j) + 1 \)
12: \hspace{2em} end
13: \hspace{1em} end
14: \hspace{1em} end
15: \hspace{1em} end
16: \( (i^*, j^*) = \arg \max \{h(i, j)\} \)
17: \( x_m = i^*, \ y_m = j^* \)
18: \( c(p, q) = 0 \)
19: end

Fig. 3. Pseudo code for One Scan Algorithm

C. Zoom Algorithm

Using the principle of divide and conquer we propose the Zoom algorithm which is very efficient in computation complexity, time and memory. The idea is to divide the Grid in four equal segments, and choose the segment with the maximum number of empty cells i.e the segment with the maximum number of cells with \( c(i, j) = 0 \). Then, this procedure is repeated until either the segment size is equal to a single cell or until all segments have the same number of empty cells. In the first case the hole center position will be the center of the cell. In the second case, the hole center position

\footnote{In more general one can choose the segment with the least coverage as defined by (4).}
will be the lower right corner of the upper left segment (the center of the segment during the previous iteration). Fig. 4 illustrates the idea of zooming for hole detection. The details of the algorithm are listed in Fig. 5. To evaluate the complexity of the algorithm, we need to determine the number of times that the Grid will be divided. In the worst case, i.e., when the algorithm will stop with a single cell the Grid will be divided at most $\kappa$ times, such that

$$\frac{L}{2^\kappa} \geq 1 \Rightarrow \kappa \leq \lg L$$ 

(11)

where $\kappa$ is the number of iterations (height of the generated tree (see Fig. 4), with $\kappa = \lfloor \lg L \rfloor \in \mathbb{Z}^+$, and $\lg L$ is the binary logarithm ($\log_2 L$). Note that again we assumed that $L = X = Y$. As a result, the running time function of Zoom algorithm in the worst case will be

$$T_Z(M, L, N, r) = (2r)^2 \times (M + N) + 4M \times \lg L \times \left(1 + \sum_{i=1}^{\lfloor \lg L \rfloor} \left(\frac{L}{2^i}\right)^2\right)$$

(12)

and since $N \gg M$ and $L \gg r$, its complexity is

$$O(L^2 \times \lg L \times M + r^2 \times (N + M)) \approx O(L^2 \times \lg L \times M).$$

(13)

**Zoom Algorithm**

1: for each sensor $n \in S$
2:   for each cell $(i, j) \in N_r(x_n, y_n)$
3:     $c(i, j) = c(i, j) + 1$
4: end
5: end
6: for each mobile $m \in M$
7:   for each zooming step $z_x$, $x = 1, \ldots, \kappa$
8:     for each segment $q_s$, $s = 1, \ldots, 4 \in \mathbb{Z}_x$
9:       $a(q_s) = a(q_s) + 1$
10:      if $a(q_s) = 0$
11:         $a(q_s) = a(q_s) + 1$
12:     end
13:   end
14: end
15: if $a(q_1) = a(q_2) = a(q_3) = a(q_4)$
16:   $x_m = \max\{i : (i, j) \in Q_1\}$
17:   $y_m = \min\{j : (i, j) \in Q_1\}$
18:   break
19: end
20: $(q^*_s) = \arg \max\{a(q_s)\}$
21: $x_m = \min\{i : (i, j) \in Q^*_s\}$
22: $y_m = \min\{j : (i, j) \in Q^*_s\}$
23: end
24: place mobile sensor at $(x_m, y_m)$
25: for each cell $(p, q) \in N_r(x_m, y_m)$
26:   $c(p, q) = c(p, q) + 1$
27: end
28: end

**Fig. 5.** Pseudo code for the Zoom Algorithm

---

**V. DISTRIBUTED PATH PLANNING ALGORITHM**

In this section we present the distributed path planning algorithm utilized by each mobile sensor in order to navigate towards its target. The requirements for the navigation algorithm are shown below but first we note that in order to ease the notation, we dropped the index for each mobile, i.e., $x(k)$ refers to the position of the $i$-th $i \in M$ mobile sensor.

1) Guide each mobile sensor node in the sensor field from its initial position $x(0)$ to its target position (e.g., the center of the coverage hole) $x^i$.

2) Collaborate with stationary and other mobile nodes in order to reduce the probability of miss event or improve the field coverage by sampling areas not covered by other sensors.

3) Collaborate with other sensors in order to investigate areas where it is “suspected” that a source is present.

The algorithm presented in this section is motivated by [4] and is based on a receding-horizon approach. In this family of algorithms, the controller evaluates the one step cost based on local information (e.g., information available in its communication range), and approximates the cost to go for several more steps based on the controller’s perception of the field. Then, it selects the next position $x(k+1)$ that optimizes an overall cost to be defined, and repeats.
Suppose, that during the $k$th step, the mobile node is at position $\mathbf{x}(k)$ and it is heading to a direction $\theta$. The next possible points are the $\nu$ points $(\mathbf{y}_1, \ldots, \mathbf{y}_\nu)$ that are uniformly distributed on the arc that is $\rho$ meters away from $\mathbf{x}(k)$ and are within an angle $\theta - \phi$ and $\theta + \phi$ as shown in Fig. 6. Note that the parameters $\rho$ and $\phi$ can be used to also model the maneuverability constraints of the mobile platform. At the $k$th position, the mobile node evaluates a cost function $J_i(\mathbf{y}_i)$ for all candidate locations $(\mathbf{y}_1, \ldots, \mathbf{y}_\nu)$ and moves to the location $\mathbf{x}(k + 1) = \mathbf{y}_i$, where $i^*$ is the index that minimizes $J_i(\mathbf{y}_i)$,

$$ J_{i^*}(\mathbf{y}_{i^*}) = \min_{1 \leq i \leq \nu} \{ J_i(\mathbf{y}_i) \} . $$

The cost function $J_i(\cdot)$ is in the form

$$ J_i(\mathbf{y}) = \sum_{j \in \{1, \ldots, m\}} w_j J^*_i(\mathbf{y}) $$

where the functions $J^*_i(\cdot)$ are defined to achieve certain objectives as defined next and $w_1, w_c$ and $w_m$ are positive weights such that $w_1 + w_c + w_m = 1$ and are selected such that a desirable mobility performance is achieved (for example, if it is desired that a mobile quickly moves to its target destination, then $w_1$ is made large).

The cost $J^*_i(\mathbf{y})$ is a function that pulls the mobile towards the target location and is a function of the distance between mobile node and target position. This function should take a smaller value as the mobile moves towards the target destination thus for the purposes of this paper it is given by

$$ J^*_i(\mathbf{y}) = \frac{1}{\sqrt{A}} \| \mathbf{y}(i) - \mathbf{x}_i \| $$

where $A$ is the sensor field area.

The cost $J^c_i(\mathbf{y})$ is a function that pushes the mobile away from covered areas (either by stationary or mobile sensors). This function should take a larger value if the candidate position is adequately covered by other sensors and a small value otherwise. Thus, for this paper, the following cost is used

$$ J^c_i(\mathbf{y}) = \frac{1}{\pi r^2_d} \sum_{(i,j) \in N_{r_d}(\mathbf{y})} g(i,j) $$

where $N_{r_d}$ is given by (3) and recall that $r_d$ is the detection range of the sensor.

Finally, to facilitate the collaboration between mobiles, we use the cost function $J^m_i(\mathbf{y})$ which penalizes each candidate position $\mathbf{y}_i$ that is close to other mobiles that are heading towards (or returning from) the same direction as the mobile tries to determine its next position. Specifically, when determining its next position, the mobile defines the set $\Lambda$ that includes all other mobiles that are in its communication range and satisfy the following two conditions. 1) The mobiles that do not follow behind and 2) the mobiles that have a heading direction $\delta$ such that $|\theta - \delta| \leq \varphi$ (the two mobiles are heading towards the same direction) or $|\theta - \delta| \geq 180^\circ - \varphi$ (the two mobiles are heading towards opposite directions), where $\varphi$ is the maximum allowed difference in heading angle. (see Fig. 7). For this paper the collaboration function is given by

$$ J^m_i(\mathbf{y}) = \sum_{\lambda \in \Lambda} \beta \exp \left( \frac{-r_{i,\lambda}}{2} \right) $$

where $\beta$ is a positive design constant and $r_{i,\lambda}$ is the distance between the candidate position $\mathbf{y}_i$ and the mobile $\lambda$.

**VI. Simulation Results**

In this section we present some simulation results where we first compare the performance of the three hole detection algorithms presented earlier and also show two representative scenarios with the movement of a set of two mobile nodes.

**A. Coverage Hole Detection Algorithms**

In the first scenario, there are 300 randomly deployed stationary sensors in an area of $500m \times 500m$. The coverage area of each sensor is a disc with radius $r_d = 20m$ and we set $d_l = 1m$. For this scenario we investigate the required computation time and the coverage improvement of the three algorithms by assuming that each mobile node is simply placed in the center of the detected hole. As seen in Fig. 8 the Grid Scan algorithm can achieve slightly better results in terms of coverage than the One Scan and Zoom algorithms since it can more accurately detect the center of a hole (Grid Scan and One Scan achieve almost the same coverage). On the other hand, Fig. 9 indicates that the computational time requirements of the zoom algorithm are negligible (zoom bars are too short to be seen) compared to the requirements of the One Scan and Grid Scan algorithms. For all experiments we used MATLAB.
R2006b on an Intel Pentium 4, 3.6 GHz CPU machine. The actual time taken for each experiment is shown in Table I. We emphasize that the efficiency of the Zoom algorithm allows it to easily run on a mobile node so that it can dynamically detect new holes (due to node failures that the mobile node discovers in its path).

In a second scenario we compare the computational requirements of the three algorithms for a larger field $2km \times 2km$ with 1000 stationary sensors. We set $dl = 1$ and the detection range for each sensor $r_d = 30m$. We assume that we try to determine the 10 biggest holes in the field. Table II shows the relative computational times, both from the simulation results as well as the derived running time functions (theoretical results column). From these results, we see that the zoom algorithm is 3 to 4 orders of magnitude faster than the other algorithms. Note that the discrepancy between the experimental and theoretical results of the zoom algorithm is because the theoretical results assume the worst case scenario. Finally, note that even though the Grid Scan achieves a slightly better coverage (see Fig. 8), for fields with low to moderate density, the detected holes fall very close to each other. On the other hand, the Zoom algorithm identifies holes that are more uniformly distributed in the field as shown in Fig. 10.

### Table I: Average Computation Time Over 10 Fields

<table>
<thead>
<tr>
<th># Mobile</th>
<th>One Scan</th>
<th>Zoom</th>
<th>Grid Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24.8768 sec</td>
<td>0.0556 sec</td>
<td>24.8588 sec</td>
</tr>
<tr>
<td>2</td>
<td>24.8822 sec</td>
<td>0.0776 sec</td>
<td>51.6161 sec</td>
</tr>
<tr>
<td>3</td>
<td>24.8875 sec</td>
<td>0.1001 sec</td>
<td>78.3004 sec</td>
</tr>
<tr>
<td>4</td>
<td>24.8928 sec</td>
<td>0.1238 sec</td>
<td>105.0711 sec</td>
</tr>
<tr>
<td>5</td>
<td>24.8982 sec</td>
<td>0.1454 sec</td>
<td>131.8238 sec</td>
</tr>
</tbody>
</table>

### Table II: Relative Computational times

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Experimental results</th>
<th>Theoretical results</th>
</tr>
</thead>
<tbody>
<tr>
<td>GridScan $T_{GS}/T_{GS}$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>OneScan $T_{OS}/T_{GS}$</td>
<td>0.09</td>
<td>0.08 $3.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>Zoom $T_{Z}/T_{GS}$</td>
<td>$0.5 \times 10^{-3}$</td>
<td>$3.2 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

### B. Distributed Path Planning Algorithms

In the first simulation experiment we use a team of two mobile nodes to illustrate the behavior of the proposed path planning algorithm. We assume a field with 100 randomly deployed stationary sensors in a $300m \times 300m$ area. The detection radius of all sensors is $r_d = 10m$ and it is indicated by the dotted circles. In this simulation there is no event source. Thus the two mobile sensors navigate collaboratively through the field, sampling points that are not adequately covered by the stationary sensors, moving towards their target locations, which are computed using the zoom algorithm. For this scenario the following parameters have been used: $\rho = 5m$, $\phi = 30^\circ$, $\nu = 10$ and $r_e = 2.5 \cdot r_d$. 

![Fig. 8. Average Coverage over 10 fields](image1)

![Fig. 9. Average Computation Time over 10 fields](image2)

![Fig. 10. Experimental execution time for 10 coverage holes detection and the corresponding target coordinates for mobile sensors](image3)
where \( r_c \) is the communication range of the mobile nodes. Moreover we set \( d\ell = 1, \varphi = \phi/2, \beta = 10 \).

In Fig. 11 we show the paths that the mobiles follow for two sets of weights \( w_t, w_c, \text{and} \ w_m \). In the first set we have \( w_t = 1, w_c = 0, \text{and} \ w_m = 0 \), i.e., the objective is to send the mobiles to the targets as soon as possible. The path followed is show with red color in Fig 11 where one can see that the mobiles move in straight lines towards their targets and there is no collaboration between the sensors (both mobiles cover similar areas parts, which are also covered by the stationary sensors).

Fig. 11 also show the paths of the two mobiles when the weights are set to \( w_t = 0.1, w_c = 0.4, \text{and} \ w_m = 0.5 \) (black paths). As seen from the paths followed by the two mobile sensors, there is collaboration between mobile and stationary sensors in the sense that the mobiles have found two different paths that are least covered by the stationary sensors. Also notice how the two mobiles repelled each other due to \( J^m \) at the beginning of their motion.

In the last simulation scenario we show the paths followed by two mobile nodes when a static source exists (the source is turned on at the beginning of the simulation time and stays on for the entire simulation). Again we assume 100 randomly deployed sensors in a \( 300m \times 300m \) field and we set the following parameters: \( V_{sat} = 500, V = 3000, \alpha = 2 \) and \( \sigma = 0.5 \). The detection threshold of all sensors is \( \tau_d = 30 \) (thus \( r_d = 10m \)), and the suspicion threshold is \( \tau_s = 5 (r_s = 24.5m) \) also the communication range is \( r_c = 4 \times r_d = 40m \). For path planning parameters we use \( \rho = 5m, \phi = 30^\circ, \nu = 10, \varphi = \phi/2, \beta = 10 \) and set the grid map \( d\ell = 1 \). We also set \( w_t = 0.3, w_c = 0.3, \text{and} \ w_m = 0.4 \). Fig 12 shows the position of the event source. Note that there is no stationary sensor in a radius \( r_d \) around the event, thus this event would remain undetected. However, there are three sensors within \( r_s \) from the source and these three sensors will report the "suspicion" to a mobile note that passes through their area. Initially, both mobiles are in a navigating mode as described in Section III and they move towards their target. Once the mobile node at the right (see Fig 12) gets the suspicion message by a static node in its communication range, then it switches its target to the estimated location of the event (see Fig.12 for the switch point). The estimated location of the event is computed by the mobile using its own measurements as well as the measurements of all nodes in its communication range. This estimation is updated in each step as the mobile node can obtain more accurate measurements as it moves towards the event source area. Finally, when the mobile detects the event (i.e. gets a measurement above the detection threshold \( \tau_d \)) it switches its target back to the initial target and continues its trip.

VII. RELATED WORK

The work presented in this paper is partially related with two research fields, the area coverage in WSNs and path planning in the fields of mobile robotics and UAVs. Although many researchers in the WSNs area have studied the coverage problem, to the best of our knowledge, this is the first time that a general architecture is proposed that combines the coverage problem with distributed path planning algorithms so that the mobile nodes can navigate towards poorly covered areas. The benefit of this approach is that events that would have remained undetected can now be detected.

Next, we present a brief overview of papers that address the coverage problem in the context of WSNs. For a more thorough survey of the coverage problem the reader is referred to [7] and [8]. Also [9] presents a survey of the holes problem (coverage, routing, jamming, sink holes, etc) in WSNs.

In [6] authors proposed the Grid Scan algorithm to find the maximum blind region in order to deploy additional static sensors. The proposed scheme is a multi-step scheme where each step is a greedy exploration process over all potential redeployment points. Only one potential point with
maximum number of neighboring non-covered grids in the sensing range is chosen as the point at which the newly added sensor node will be deployed. As shown in the this paper, the proposed Zoom algorithm is computationally significantly more efficient.

Next, we present several other approaches that have been proposed in order to determine the coverage holes where mobile nodes can be deployed. All these approaches do not consider the path that the mobile should follow in order to reach its destination. In [10] authors used Voronoi diagrams to discover the existence of coverage holes. A node needs to know the location of its neighbors to construct its Voronoi diagram. The diagram partitions the whole space into Voronoi polygons. Each polygon has a single node with the property that every point in the polygon is closer to this node than any other node. A sensor node compares its sensing disk with the area of its Voronoi polygon to estimate any local coverage hole. Three distributed self-deployment algorithms have been proposed to calculate new optimal positions to which mobile sensors should move to increase coverage: Vector based (VEC), Voronoi based (VOR) and Minimax algorithm.

The same authors in [11] describe a bidding protocol, for mixed sensor networks that use both static and mobile sensors to achieve a cost balance. Their algorithm considers a random initial deployment, where static sensors detect their local coverage holes based on Voronoi diagrams. The mobile sensors calculate coverage holes formed at their current position if they decide to leave their current position. The static sensors bid mobile sensors based on the size of their detected coverage hole. A mobile sensor compares the bids and decides to move if the highest bid received has a coverage hole size greater than the new hole generated in its original location due to its movement. The bids are broadcasted locally up to two hops and the static sensors are able to direct neighboring eligible mobile sensors to a point close to the farthest vertex of their Voronoi polygon. However, the local broadcast may prevent the bid messages reaching mobile sensors if they are located farther than two hops. In this case the authors propose a mixed architecture for the coverage problem.

In [12] authors address the problem of enhancing coverage in a mixed sensor network. They present a method to deterministically estimate the exact amount of coverage holes under random deployment using Voronoi diagrams and use the static nodes to estimate the number of additional mobile nodes needed to be deployed and relocated to the holes locations to maximize coverage. The static nodes also find out the optimal positions of those mobile nodes based on certain heuristics. In our case we use a small number of mobile nodes that move collaboratively using path planning algorithms in order to enhance the event detection probability of the stationary sensor network.

Sensor relocation has been studied in [13], which focuses on finding the target locations of the mobile sensors based on their current locations and the locations of the sensed events. In [14] a polynomial-time algorithm is presented in terms of the number of sensors to determine whether every point in the service area of sensor networks is covered by at least $k$ sensors, where $k$ is a predefined value. With this algorithm, WSNs work well in situations that require stronger coverage and impose more stringent fault-tolerant capability. In [15], the authors provide a polynomial-time, greedy, iterative algorithm to determine the best placement of one sensor at a time in a grid based scenario, such that each grid is covered with a minimum confidence level. They model the obstacles as static objects and assume that a complete knowledge of the terrain is available. As already mentioned, none of the aforementioned approaches considers the actual path that each mobile should follow.

Next, we present some path planning algorithms that have been proposed and are relevant to our work. A good overview of motion planning in robotics is given in [16]. As already mentioned, the path planning algorithms presented in this paper have been motivated by the approach in [4] where an approach for cooperative search by a team of distributed agents is presented. In that approach two or more agents move in a geographic environment, cooperatively searching for targets of interest and avoiding obstacles or threats. Authors in [17] use the concept of Voronoi diagrams and triangulation to provide polynomial-time worst case and best case algorithms for determining maximal breach path and maximal support path, respectively, in a sensing field. On similar lines, in [18], the authors use the concepts of minimal and maximal exposure paths to find out how well an object moving on an arbitrary path can be observed by the sensor network over a period of time. The algorithm in [18] uses certain graph theoretic abstractions and compute minimal exposure path using Dijkstra’s single source shortest path algorithm or Floyd-Warshals all pair shortest path algorithm. These approaches are centralized and solve static problem instances in the sense that they do not allow changes in the field once the paths have been computed.

In [19], the authors have focused on the coverage capabilities that result from the continuous random movement of the sensors. However, in this paper, we develop distributed path planning algorithms for cooperative movement of the mobile sensors.

Finally, we present some approaches that address the coverage problem in mobile sensor networks (all sensors are mobile). In [20] authors have looked at the problem of how mobile sensors move collaboratively in order to search a region and also incorporate communication costs into the coverage control problem.

The coverage concept with regard to the many-robot systems was introduced by Gage [21], who defined three types of coverage: blanket coverage, barrier coverage, and sweep coverage. Potential field techniques for robot motion planning were first described by Khatib [22] and have since been widely used in the mobile robotics community for tasks such as local navigation and obstacle avoidance. A robot moving according to the potential will never hit obstacles, but it may get stuck in local minima.
Assuming that all sensors have motion capabilities, several approaches have been developed to address the coverage problem using the concept of potential fields [23], [24], and virtual forces in [25]. In [23], the authors propose a deployment strategy using mobile autonomous robots that maximize the area coverage with the constraint that each of the nodes has at least \( k \) neighbors. The sensing field is modeled using attractive and repulsive forces exerted on each node by all other nodes.

The network stabilizes when equilibrium is reached, i.e., the net force on each node becomes zero. This approach is computationally expensive because as the network size grows or new nodes join, all nodes need to reconfigure themselves to satisfy the equilibrium criteria.

In a similar fashion, assuming that all sensors have motion capabilities, the authors of [24] proposed a potential field-based algorithm in which nodes are treated as virtual particles subjected to virtual force. Virtual forces repel the nodes from each other and from obstacles, and ensure that the initial configuration of nodes quickly spreads out to maximize coverage area. In [25], the authors presented another virtual-force-based sensor movement strategy to enhance network coverage after an initial random placement of sensors. A cluster head computes the new locations of all the sensors after the initial deployment that would maximize coverage and then nodes reposition themselves to the designated locations. After the execution of the algorithm and once the effective sensor positions are identified, a one-time movement is carried out to redeploy the sensors.

VIII. CONCLUSION & FUTURE WORK

In this paper we propose a collaborate event detection architecture for WSNs consisting of a large number of stationary nodes and a few mobile nodes. The benefit of this architecture is that the mobile nodes collaborate with the stationary nodes so that they sample the areas least covered by the stationary nodes. In this way, events that would have remained undetected can now be detected.

In the proposed architecture a number of issues are still open and are being investigated. First, when does the mobile switches targets (from the center of the hole to the suspected area). Note that if the suspicion threshold is set very low, there is a danger that most sensors will report a suspicion and as a result the mobile will waste its time responding to false alarms. Second, investigate more dynamic environments where sources appear and disappear dynamically and stationary sensors also fail or being redeployed randomly. In this case, \( G \) should be multiplied by appropriate discounting rates so that it reflects that uncertainty. Finally, we also need algorithms for updating the maps of the mobiles in a totally distributed fashion as well as algorithms for fusing the maps of two mobile nodes together (when they come within communication range).

ACKNOWLEDGMENT

This work is partly supported by the Cyprus Research Promotion Foundation under contract ENIΩX/0505/30.

REFERENCES