Visibility-based multi-agent deployment in orthogonal environments

Anurag Ganguli Jorge Cortés Francesco Bullo

***Abstract*— This paper addresses the problem of deploying a group of robotic agents equipped with omnidirectional vision in a simply connected orthogonal environment to achieve complete visibility. The agents are point masses with discrete-time first-order dynamics with no prior knowledge of the environment. Each agent can sense distances to the environment boundary and to other agents within line-of-sight. Communication is possible only between collocated agents. The agents operate asynchronously. The paper also addresses the problem of complete visibility deployment under the additional constraint that the visibility graph of the final agent locations is connected. We provide distributed algorithms that are guaranteed to solve the above problems if a sufficient number of agents are available. Remarkably, this number is identical to the number assuming complete prior knowledge of the environment. A final contribution of the paper is the characterization of the robustness properties of the algorithms to agent failures in the case of deployment with connectivity constraints.**

I. Introduction

Recently, much research has focused on the use of unmanned robots for the purpose of surveillance and search. This paper provides algorithms to deploy robotic agents with limited capabilities to monitor an unknown environment. The environment is assumed to be simply connected, i.e., without holes, and orthogonal, i.e., polygonal with sides either parallel or perpendicular to one another. Orthogonal environments are interesting because they can be used to model indoor and urban environments. The agents are modeled as point masses with first-order dynamics. The agents are all identical except with distinct identifiers (UID). No assumption is made about the UIDs except that they are distinct. The agents are assumed to operate asynchronously and to have limited communication and sensing capabilities: they can communicate only with collocated agents and they can sense the distance to the environment boundary or to any other agent within line of sight. It is practical to assume limitations on the communication bandwidth: we assume that agents can communicate only their UIDs to other agents. The first objective is to deploy the agents starting from a single location so that all points of the environment are visible to at least one agent. We present a distributed algorithm to solve the above problem requiring no more than $\lfloor n/4 \rfloor$, agents where n is the number of vertices in the environment. A second objective is to deploy the agents in such a way so that the visibility graph of the final configuration of the agents is connected. We also present a distributed algorithm to solve the problem requiring no more than $(n-2)/2$ agents.

Deployment of robotic sensors have been studied in centralized and decentralized contexts, centralized referring to the fact that the environment is known a priori and decentralized otherwise. In the former setting, this problem becomes the classical Art Gallery Problem in the computational geometry literature, which aims to find both the minimum number of "guards" required and the locations of these guards to achieve complete visibility of a given polygonal environment. This problem was first analyzed by Chvátal, see [1], in the famous Art Gallery Theorem stating that $\lfloor n/3 \rfloor$ guards are sufficient and sometimes necessary to guard any simply connected polygon with n vertices. Kahn, Klawe and Kleitman [2] proved that in simply connected orthogonal environments, $\lfloor n/4 \rfloor$ guards are sufficient and sometimes necessary. In [3], Pinciu gives a constructive algorithm to prove that $n/2 - 2$ connected set of guards are always sufficient and occasionally necessary in a simply connected orthogonal environment.

Relevant works in the decentralized setting include [4], where an incremental heuristic for deployment is proposed, and [5] where distributed algorithms for coverage control based on Voronoi partitions are designed. Coordinated deployment of multiple heterogeneous robots has also been studied in [6]. Deployment locations are user-specified after an initial map of the unknown environment has been built.

Another related body of work is that of robotic exploration of unknown environments since a strategy to solve the deployment problem might be to first explore and then solve the centralized problem. The most relevant literature to the current problem include topological exploration of graph-like environments by single and multiple robots [7], [8], [9], [10]. In these problems, it is either assumed that agents can synchronize their motions to fuse their data, or read and write to the nodes of the graph. These assumptions are stronger compared to the assumptions in the present treatment. Synchronizing motions and fusing data are additional complications, especially in the presence of limited communication bandwidth. Also, writing to nodes in a graph is not possible in the case of exploration of unknown environments. Finally, the problem of deployment is very different from the problem of exploration. Assuming that exploration is possible without accumulating errors, in the absence of a central processor, the robots would have to allocate tasks amongst themselves. Our strategy, on the other hand, is a simple one-step strategy for deployment, without the need for synchronization, achieving the worst-case optimal bounds in terms of number of robots required, and under limited communication constraints.

Anurag Ganguli is with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, and with the Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106, USA, aganguli@uiuc.edu

Jorge Cortés is with the Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA 95064, USA, jcortes@ucsc.edu

Francesco Bullo is with the Department of Mechanical Engineering, University of California, Santa Barbara, CA 93106, USA, bullo@engineering.ucsb.edu

Due to space limitations, the proofs of all results in this paper are omitted, and will be presented elsewhere.

II. Preliminaries and notation

We begin by introducing some basic notation. Let $\mathbb{R}$ represent the set of real numbers. Given two points $x, y \in \mathbb{R}^2$, let $[x, y]$ represent the *closed* segment between x and y. Given a finite set X, let $|X|$ represent the cardinality of the set. Let P to refer to tuples of elements in $\mathbb{R}^2$ of the form $(p_1, \ldots, p_N)$. With a slight abuse of notation, we shall use P interchangeably with a point set of the form $\{p_1, \ldots, p_N\}$.

An *orthogonal* environment, Q, is a polygonal environment whose boundary is composed of segments that are parallel or perpendicular to each other. An environment Q is *simply connected* if it does not contain any holes. Let $\mathrm{Ve}(Q)$ and $\mathrm{Ver}(Q)$ be the list of vertices and reflex vertices. Recall that a reflex vertex is one with interior angle strictly greater than π radians. A vertex that is not reflex is a convex vertex.

We now describe some useful notions of visibility. A point $q \in Q$ is *visible from* $p \in Q$ if $[p, q] \subset Q$. The *visibility set* $\mathcal{V}(p) \subset Q$ from a point $p \in Q$ is the set of points in Q visible from p. A star-shaped subset of Q is a set $\mathcal{S}$ such that there exists $p \in \mathcal{S}$ with the property that $\mathcal{S} \subset \mathcal{V}(p)$. The set of all such points $\mathcal{S}$ is referred to as the *kernel* of $\mathcal{S}$ denoted by $\ker(\mathcal{S})$. We now define the following:

Definition 2.1: (i) A *diagonal* of Q is a segment with end points in $\mathrm{Ve}(Q)$ but otherwise belonging to the interior of Q.

(ii) A partition of a compact set X is a collection of compact, simply connected sets $\{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$ with disjoint interiors and with $\cup_{i=1}^{N} \mathcal{P}_i = X$.

(iii) The *visibility graph* $\mathcal{G}_{\mathrm{v},Q}(p_1, \ldots, p_n)$ of a set of points $\{p_1, \ldots, p_n\}$ in Q is a graph with the node set equal to $\{p_1, \ldots, p_n\}$ and with (p_i, p_j) being an edge if and only if $[p_i, p_j] \subseteq Q$ and vice versa.

Next we describe the capabilities of an agent followed by the problem description.

III. Agent model and problem description

We consider a group of robotic agents modeled as point masses, moving in a simply connected orthogonal environment, Q. Each agent has a unique identifier (UID), say i. Let p_i refer to the position of agent i. Each agent is equipped with an omnidirectional line-of-sight sensor capable of measuring the distance to any object visible to it (another agent or the environment boundary). Thus, the agent can sense everything within its star-shaped visibility set $\mathcal{V}(p_i)$. Each agent can also communicate with any other agent in close proximity to itself; for simplicity, we assume that communication is possible only with collocated agents. The agents have on-board processors whose clocks are *not synchronized*. The agents *do not* possess a Global Positioning System (GPS). Each agent has access to some memory $\mathcal{M}_i$. We use $\mathcal{M}_i$ to denote all the necessary information that cannot be obtained by i via local sensing and communication.

We now describe some specifics about the agents' operations. An agent i can broadcast its UID to all collocated agents. Such a broadcast is denoted by `BROADCAST`(i). It can also receive broadcasts from other agents. We also assume that there is an arbitrary time delay between a broadcast and the corresponding reception which is upper bounded by $\delta > 0$. Every agent i repeatedly performs the following sequence of actions beginning at a time instant, say T_l^i:

(i) send repeated `BROADCAST`(i) every δ seconds, until it starts moving;

(ii) `LISTEN` for at least 2δ seconds before processing the information;

(iii) `PROCESS` the received information. Also continue to `LISTEN` during this interval;

(iv) `MOVE` to a desired point.

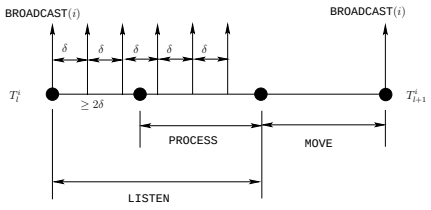

Fig. 1. Sequence of actions for agent i beginning at time T_l^i. Instantaneous `BROADCAST`(i) events are represented by vertical pulses. The `MOVE` interval might be empty if the agent does not move. The subsequent instant T_{l+1}^i is the time when the agent stops performing the `MOVE` action and it is not predetermined.

At any time t in the `MOVE` interval, agent i, moves according to the following discrete-time control system:

$$p_i(t + \Delta t) = p_i(t) + u_i,$$

where $\|u_i\| \leq 1$. The control action, u_i, depends on time, on the memory contents, $\mathcal{M}_i$ at that time, and on the information obtained from communication and sensing. This model is similar in spirit to the *partially asynchronous network model* described in [11].

Given this agent model, our first goal is to design a provably correct discrete-time algorithm to deploy agents on locations such that each point of the environment is visible to at least one agent. This is the *visibility-based deployment problem*. Our second goal is to design a provably correct discrete-time algorithm to solve the visibility-based deployment problem under the additional constraint that the final configuration of agents is connected. This is the *connected visibility-based deployment problem*.

IV. Incremental partition algorithm

Here we describe a procedure to incrementally partition an orthogonal environment Q into star-shaped sets. Given Q and $s \in \mathrm{Ve}(Q)$ such that an adjacent vertex t is convex, the `Incremental Partition Algorithm` computes a finite ordered set of star-shaped polygonal sets, $\mathcal{P}_{\text{vc-}Q}(s)$. The algorithm also returns a finite ordered set of points $\mathcal{N}_{\text{vc-}Q}(s)$ with $|\mathcal{N}_{\text{vc-}Q}(s)| = |\mathcal{P}_{\text{vc-}Q}(s)|$ and with $\mathcal{P}_{\text{vc-}Q}(s)_i$ visible from $\mathcal{N}_{\text{vc-}Q}(s)_i$, where $\mathcal{P}_{\text{vc-}Q}(s)_i$ and $\mathcal{N}_{\text{vc-}Q}(s)_i$ are, respectively, the ith elements of $\mathcal{P}_{\text{vc-}Q}(s)$ and $\mathcal{N}_{\text{vc-}Q}(s)$

We now begin with the description of the `Incremental Partition Algorithm`. It consists of two components: (i) the `Removable-quadrilateral` computation (described in the Appendix); and (ii) the

`Star-set computation` algorithm described as follows. The variable $\mathcal{P}_{\text{star-shaped}}$ in the algorithm contains a polygon. An execution of the `Star-set computation` algorithm is also illustrated in Figure 2.

Algorithm: `Star-set computation`

Input: Orthogonal simply-connected polygon Q, tilted edge e such that at least one vertex of e is convex, and a specified vertex v of e

Initialization: $\mathcal{P}_{\text{star-shaped}} := \emptyset$, $X := Q$, $[a, b] := e$

Compute removable quadrilateral C of X with tilted edge e using `Removable-quadrilateral`.

Set $\mathcal{P}_{\text{star-shaped}} = \mathcal{P}_{\text{star-shaped}} \cup C$.

`If` any edge, e' of C containing v is a diagonal of X, `then`

Repeat all the above steps with X being the environment on the opposite side of e' as $\mathcal{P}_{\text{star-shaped}}$ and $e = e'$.

Return: $\mathcal{P}_{\text{star-shaped}}$, v

Beginning with Q as the polygon, $[s, t]$ as the initial

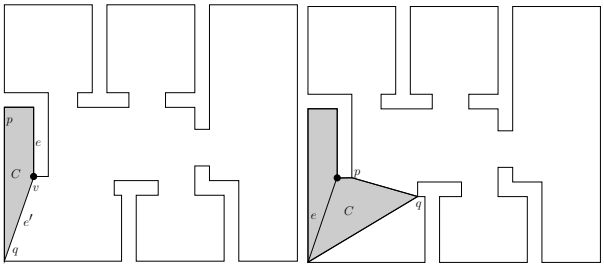

Fig. 2. The `Star-set computation` algorithm. On the left, the shaded region represents the quadrilateral C computed in the first step of the computation routine. The edge e' is a diagonal of Q; thus the first step is repeated again with X as the unshaded portion of the environment and e' as the new tilted edge. The result is shown in the figure on the right. The union of the shaded regions represents $\mathcal{P}_{\text{star-shaped}}$ as computed by the `Star-set computation` algorithm.

tilted edge and $v = s$, the `Star-set computation` algorithm is executed for every diagonal generated until there are no diagonals. However, for every diagonal, the choice of which vertex serves as v is important. This is done as follows. Given a tilted edge e containing vertex v, the set $\mathcal{P}_{\text{star-shaped}}$ is constructed according to the `Star-set computation` algorithm. The vertices of $\mathcal{P}_{\text{star-shaped}}$ are numbered in a counter-clockwise fashion with v as the first vertex. For any edge of $\mathcal{P}_{\text{star-shaped}}$ that is a diagonal of Q, the vertex that is odd numbered is chosen as the new vertex v. The `Removable quadrilateral` together with the `Star-set computation` algorithm and the above rules for choosing the vertex v constitute the `Incremental Partition Algorithm`.

Let $\mathcal{P}_{\text{vc-}Q}(s)$ be the resulting collection of $\mathcal{P}_{\text{star-shaped}}$ sets and let $\mathcal{N}_{\text{vc-}Q}(s)$ be the corresponding v vertices. We refer to $\mathcal{P}_{\text{vc-}Q}(s)$ as the *visually-connected vertex-induced partition* of Q starting from s; see Figure 3 (left). The following lemma characterizes the properties of $\mathcal{P}_{\text{vc-}Q}(s)$ and $\mathcal{N}_{\text{vc-}Q}(s)$.

Lemma 4.1: Given any simply connected orthogonal region Q and $s \in \text{Ve}(Q)$ such that there exists an adjacent vertex that is convex, the following statements are true:

(i) $\mathcal{P}_{\text{vc-}Q}(s)_i \in \mathcal{P}_{\text{vc-}Q}(s)$ is star-shaped with $\mathcal{N}_{\text{vc-}Q}(s)_i \in \ker(\mathcal{P}_{\text{vc-}Q}(s)_i)$ for all i;

(ii) $|\mathcal{P}_{\text{vc-}Q}(s)| = |\mathcal{N}_{\text{vc-}Q}(s)| \leq \frac{n-2}{2}$ where $n = |\text{Ve}(Q)|$;

(iii) the visibility graph $\mathcal{G}_{\text{v},Q}(\mathcal{N}_{\text{vc-}Q}(s))$ has a single connected component;

(iv) if $\mathcal{P}_{\text{vc-}Q}(s)_i$ and $\mathcal{P}_{\text{vc-}Q}(s)_j$ share a diagonal, then $\mathcal{N}_{\text{vc-}Q}(s)_i, \mathcal{N}_{\text{vc-}Q}(s)_j$ are mutually visible.

We now define the *visually-connected vertex-induced tree*.

Definition 4.2: Given a simply connected orthogonal environment Q, and $s \in \text{Ve}(Q)$ with an adjacent convex vertex, the visually-connected vertex-induced tree $\mathcal{G}_{\text{vc-}Q}(s)$ is the graph with node set $\mathcal{N}_{\text{vc-}Q}(s)$ and with $(\mathcal{N}_{\text{vc-}Q}(s)_i, \mathcal{N}_{\text{vc-}Q}(s)_j)$ being an edge iff $\mathcal{P}_{\text{vc-}Q}(s)_i \cap \mathcal{P}_{\text{vc-}Q}(s)_j$ is a diagonal of Q.

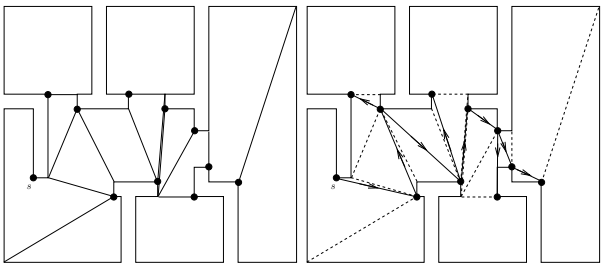

Fig. 3. The figure on the left is the visually-connected vertex-induced partition of the orthogonal polygon Q induced by s. The set of points represented by the black discs is $\mathcal{N}_{\text{vc-}Q}(s)$. The diagonals inside Q are the boundaries of the sets comprising the partition. The figure on the right is an illustration of the visually-connected vertex-induced tree for the same partition rooted at the node s. The black discs represent the nodes. The thick directed lines are the edges of the tree, and the arrows indicate the direction away from the root.

The following is a consequence of Lemma 4.1.

Corollary 4.3: Given any simply connected orthogonal environment Q and $s \in \text{Ve}(Q)$ such that there exists an adjacent vertex that is convex, the following hold true:

(i) the graph $\mathcal{G}_{\text{vc-}Q}(s)$ is a rooted tree[1] with s as the root;

(ii) the number of nodes of $\mathcal{G}_{\text{vc-}Q}(s)$ is less than or equal to $\frac{n-2}{2}$;

(iii) if $(\mathcal{N}_{\text{vc-}Q}(s)_i, \mathcal{N}_{\text{vc-}Q}(s)_j)$ is an edge of $\mathcal{G}_{\text{vc-}Q}(s)$, then $\mathcal{N}_{\text{vc-}Q}(s)_i$ and $\mathcal{N}_{\text{vc-}Q}(s)_j$ are mutually visible.

As a result of our discussion, we have converted the visibility-based deployment problem to the problem of deploying agents on *every node* of $\mathcal{G}_{\text{vc-}Q}(s)$ under the assumption that all agents are initially located at the same node. Next, we design an algorithm that identifies a smaller subset of nodes of $\mathcal{N}_{\text{vc-}Q}(s)$ from which the entire polygon is visible and deploys agents on this smaller set.

V. Visibility-based deployment

Here, we start by identifying in Section V-A a subset $\mathcal{N}$ of $\mathcal{N}_{\text{vc-}Q}(s)$ such that $\cup_{i=1}^{|\mathcal{N}|} \mathcal{V}(\mathcal{N}_i) = Q$ and $|\mathcal{N}| \leq \lfloor \frac{n}{4} \rfloor$. The visibility-based deployment problem further reduces then to deploying agents on the set of points $\mathcal{N}$. The algorithm design that achieves this is presented in Sections V-B to V-D.

[1]A connected graph with no simple cycles is a tree. A tree is called a rooted tree if there exists a node that has been specified as a root and all edges have a natural direction, either towards or away from the root. In this paper, we assume that the edges are directed away from the root. Given any node x of a rooted tree, let y be a node such that there exists a directed edge from y to x. Then y is the parent of x and correspondingly x is the child of y. The root has no parent. A predecessor of a node x is any other node from which a directed path exists to x

A. Desirable agent locations

Let us first notice that, for all $i \in \{1, \ldots, |\mathcal{P}_{\text{vc-}Q}(s)|\}$, the set $\mathcal{P}_{\text{vc-}Q}(s)_i$ is a union of convex quadrilaterals. Let q_i denote the number of convex quadrilaterals in $\mathcal{P}_{\text{vc-}Q}(s)_i$. The algorithm to compute the set $\mathcal{N}$ is as follows.

Algorithm: $\mathcal{N}$-`computation-algorithm`

Input: The graph $\mathcal{G}_{\text{vc-}Q}(s)$ with simply connected orthogonal polygon Q, and $s \in \text{Ve}(Q)$ with an adjacent convex vertex.

Initialization: $\mathcal{N} = \emptyset$, $P_{\text{mark}} = \emptyset$

`While` $P_{\text{mark}} \neq \mathcal{N}_{\text{vc-}Q}(s)$ `do`

Take any node, say $\mathcal{N}_{\text{vc-}Q}(s)_i$, of $\mathcal{G}_{\text{vc-}Q}(s)$ which has no children or whose children all belong to P_{mark}.

Let c_{occ} denote the number of children of $\mathcal{N}_{\text{vc-}Q}(s)_i$ belonging to $\mathcal{N}$.

`If` $q_i - c_{\text{occ}} = 0$ `then`

$P_{\text{mark}} = P_{\text{mark}} \cup \{\mathcal{N}_{\text{vc-}Q}(s)_i\}$.

`else`

$P_{\text{mark}} = P_{\text{mark}} \cup \{\mathcal{N}_{\text{vc-}Q}(s)_i\}$; $\mathcal{N} = \mathcal{N} \cup \{\mathcal{N}_{\text{vc-}Q}(s)_i\}$.

Return: $\mathcal{N}$

The following proposition enumerates the properties of $\mathcal{N}$.

Proposition 5.1: Given a simply connected orthogonal environment Q and $s \in \text{Ve}(Q)$ such that an adjacent vertex is convex, let $\mathcal{N}$ be computed according to the $\mathcal{N}$-`computation-algorithm`. Then the following statements are true:

(i) $\cup_{i=1}^{|\mathcal{N}|} \mathcal{V}(\mathcal{N}_i) = Q$; and

(ii) $|\mathcal{N}| \leq \lfloor \frac{n}{4} \rfloor$,

where $\mathcal{N}_i$ is the ith element of $\mathcal{N}$, and $n = |\text{Ve}(Q)|$.

We now design a distributed algorithm to deploy agents on the set of nodes given by $\mathcal{N}$. Note that $\mathcal{N}$ is a subset of $\mathcal{N}_{\text{vc-}Q}(s)$. Therefore, to move between the nodes of the set $\mathcal{N}$, we design local navigation algorithms to move between the nodes in $\mathcal{N}_{\text{vc-}Q}(s)$.

B. Local node-to-node navigation algorithms

In a rooted tree, every neighbor of a node is either a child or a parent. Also, in the visually-connected vertex-induced tree, neighboring nodes are mutually visible; see Corollary 4.3(iii). Therefore, moving between adjacent nodes consists of moving along a straight line from one point to another, possible due to the first order dynamics of the agents described in Section III. This constitutes the `Move-to-Parent Algorithm` and `Move-to-Child Algorithm`. It is easy to see that navigation is straightforward if sufficient information is available to the agents. In other words, at a node the agent must have information about the location of its parent and children. Additionally, it must also be able to decide if the node belongs to the set $\mathcal{N}$. This is the subject of the following subsection.

C. Distributed information processing

For an agent is to execute the `Move-to-Parent Algorithm`, it needs to know where the parent is located. To compute the locations of the children, an agent at a node must be able to compute its star-shaped set from that node. For that, it must know the diagonal(s) (A node can possibly be located at the intersection of two diagonals; see the child of s in Figure 3 (right)) that will separate its star-shaped set from the parent set. This geographic information is gathered and managed by the agents via the following state transition laws. At this time, we make full use of the computation and sensing abilities of the agents mentioned in Section II.

(i) The memory content $\mathcal{M}$ of each agent has four components: $(p_{\text{parent}}, p_{\text{last}}, g_1, g_2)$, where p_{parent} is an ordered list of points in $\mathbb{R}^2$, p_{last} is a point in $\mathbb{R}^2$, g_1 and g_2 are ordered list of elements belonging to $\mathbb{R}^2 \times \mathbb{R}^2$. For any agent i, at time $t = 0$, $\mathcal{M}_i(0) = \{(p_i(0)), p_i(0), (((p_i(0), p_i(0))), ((p_i(0), p_i(0)))\}$.

During run time, $\mathcal{M}$ is updated to acquire and maintain the following meaning: p_{parent} is the list of *locations of the predecessor nodes* to the agent's current position, p_{last} is the location of the *last node* visited by the agent, and g_1 and g_2 are lists of *locations of diagonals that separate the predecessor sets*, all measured relative to the agent's current position. This is accomplished as follows:

(ii) After an agent moves from a node $\mathcal{N}_{\text{vc-}Q}(s)_i$ to a child node $\mathcal{N}_{\text{vc-}Q}(s)_j$ located on diagonals described by vertices v_1', v_1'' and v_2', v_2'' via `Move-to-Child Algorithm`, its memory $\mathcal{M}$ is updated as follows: $\mathcal{N}_{\text{vc-}Q}(s)_i - \mathcal{N}_{\text{vc-}Q}(s)_j$ is added to the beginning of the list p_{parent}, $p_{\text{last}} := \mathcal{N}_{\text{vc-}Q}(s)_i - \mathcal{N}_{\text{vc-}Q}(s)_j$, and $(v_1' - \mathcal{N}_{\text{vc-}Q}(s)_j, v_1'' - \mathcal{N}_{\text{vc-}Q}(s)_j)$ and $(v_2' - \mathcal{N}_{\text{vc-}Q}(s)_j, v_2'' - \mathcal{N}_{\text{vc-}Q}(s)_j)$ are added to the beginning of the lists g_1 and g_2 respectively.

(iii) After an agent moves from a node $\mathcal{N}_{\text{vc-}Q}(s)_j$ to the parent node $\mathcal{N}_{\text{vc-}Q}(s)_i$ via `Move-to-Parent Algorithm`, its memory $\mathcal{M}$ is updated as follows: the first elements of p_{parent}, g_1 and g_2 are deleted and $p_{\text{last}} := \mathcal{N}_{\text{vc-}Q}(s)_j - \mathcal{N}_{\text{vc-}Q}(s)_i$.

D. Global exploration and deployment algorithm

In the previous sections, we have designed local node-to-node navigation algorithms and also specified how the memory must be managed to execute them. As a final step, to ensure that two agents do not occupy the same node, we utilize the communication capabilities of the agents. Agents collocated at the same node exchange their UIDs and take the appropriate decision.

At this time, we have all the elements to design a global exploration algorithm that leads the agents to deploy on the nodes $\mathcal{N}$. The algorithm is as follows.

Algorithm: `Depth-first Deployment`

Input: Simply connected orthogonal polygon Q, and $s \in \text{Ve}(Q)$ with an adjacent convex vertex, N agents located at s

`For` every `PROCESS` interval for agent j located at node $\mathcal{N}_{\text{vc-}Q}(s)_i$ of $\mathcal{G}_{\text{vc-}Q}(s)$ `do`

Compute the locations of the children and order them based on j.

`If` p_{last} is last child or no child exists `then`

`If` maximum UID received is greater than j

`Move-to-Parent Algorithm` towards p_{parent_1}.

`else`

Let c_{occ} be the number of child nodes occupied by other agents.

`If` $q_i - c_{\text{occ}} = 0$

`Move-to-Parent Algorithm` towards p_{parent_1}.

```
    else
      Stay at current node.
  else
    Move-to-Child Algorithm towards next child in the ordering.
```

E. Convergence analysis

We now present the main result of this section.

Theorem 5.2: Given a simply connected orthogonal polygon Q, let $p_1(0) = \cdots = p_N(0) = s$, represent the initial positions of an asynchronous network of N agents as described in Section III. Let s be a vertex of Q with an adjacent convex vertex. Let the behavior of the agents be governed by the `Depth-first Deployment` algorithm. Then the following are true:

(i) there exists a finite time t^* after which there is at least one agent on $\min\{|\mathcal{N}|, N\}$ nodes of the set $\mathcal{N}$;
(ii) if $N \geq \lfloor \frac{n}{4} \rfloor$, then the visibility-based deployment problem is solved in finite time.

Remark 5.3: The assumptions on the initial point s can be removed easily. For example, starting from *any* single location in Q, the first stage of deployment could be moving towards the nearest vertex and then following a wall until a vertex satisfying the assumptions in Theorem 5.2 is reached. The `Depth-first Deployment` algorithm can then be executed starting from this new vertex.

VI. Connected visibility-based deployment

In the previous section, we designed an algorithm for deployment of agents in a simply connected orthogonal environment to achieve complete coverage. However, such a deployment does not guarantee that the visibility graph of the final configuration of the agents is connected. In many cases, this is desirable when after deployment, the sensed data from all the agents is to be gathered at a single node via line-of-sight communication. We also see later in this section that connectivity via line-of-sight enables the agents to sense the failure of other agents and take the necessary repair action.

Therefore, in this section we design an incremental partition and deployment algorithm for orthogonal environments with the property that if $P = (p_1, \ldots, p_N)$ is the final position of the agents then $\cup_{i=1}^{N} \mathcal{V}(p_i) = Q$ and $\mathcal{G}_{\text{v},Q}(P)$ has one connected component. We also characterize the robustness of the algorithm to agent failures. To solve this problem, it suffices to deploy agents on *every* node of the visually-connected vertex-induced tree, $\mathcal{G}_{\text{vc-}Q}(s)$. This follows from Lemma 4.1(iii) and the definition of $\mathcal{G}_{\text{vc-}Q}(s)$. The local navigation algorithms have already been discussed in Section V-B. In the following, we discuss the information processing and global deployment aspects.

A. Distributed information processing

As will be clear in the next section, the algorithm to solve the connected visibility based deployment problem requires agents to navigate only from the parent to the children. Because of this simplification, it suffices for the memory to be given by $\mathcal{M}_i = (p_{\text{parent}}, g_1, g_2)$, where p_{parent} is a point in $\mathbb{R}^2$, and g_1 and g_2 are elements belonging to $\mathbb{R}^2 \times \mathbb{R}^2$. For any agent i, at time $t = 0$, $\mathcal{M}_i(0) = \{p_i(0), (p_i(0), p_i(0)), (p_i(0), p_i(0))\}$. The difference from the non-connected deployment lies in the fact that p_{parent}, g_1 and g_2 are just single elements instead of lists and p_{last} is absent. After an agent moves from a node $\mathcal{N}_{\text{vc-}Q}(s)_i$ to a child node $\mathcal{N}_{\text{vc-}Q}(s)_j$ located on diagonals described by vertices v_1', v_1'' and v_2', v_2'' via `Move-to-Child Algorithm`, its memory $\mathcal{M}$ is updated as follows: $p_{\text{parent}} := \mathcal{N}_{\text{vc-}Q}(s)_i - \mathcal{N}_{\text{vc-}Q}(s)_j$, and $g_1 := (v_1' - \mathcal{N}_{\text{vc-}Q}(s)_j, v_1'' - \mathcal{N}_{\text{vc-}Q}(s)_j)$ and $g_2 := (v_2' - \mathcal{N}_{\text{vc-}Q}(s)_j, v_2'' - \mathcal{N}_{\text{vc-}Q}(s)_j)$.

B. Global exploration and deployment algorithm

As before, to prevent two or more agents from occupying the same node, we adopt the method of comparing UIDs. The deployment algorithm is described as follows.

Algorithm: `Connected Depth-first Deployment`
Input: Simply connected orthogonal polygon Q, and $s \in \text{Ve}(Q)$ with an adjacent convex vertex, N agents located at s.
Find the number of UIDs received during the `LISTEN` action greater than its own UID, say m.
Find the number of child nodes, c.
Find the number of agents that it can sense on the child nodes and on the paths between the present node and the child nodes, say n_{path}.
`If` $c > n_{\text{path}} + m$
  Order the children according to some scheme common for all agents.
  $d := c - n_{\text{path}} - m$
  `Move-to-Child Algorithm` towards the dth child among those that are unoccupied and do not have another agent on the path towards them
`else`
  Stay at current node

C. Convergence analysis and robustness to failures

In this section, we analyze the convergence properties of the `Connected Depth-first Deployment` algorithm. We also characterize the robustness properties of the algorithm to agent failures. However, we begin by defining what we understand by a failure.

Definition 6.1: An agent i is said to have failed at time t_f, if for all $t \geq t_f$, it cannot be sensed by any other agent and cannot communicate with any other agent.

Theorem 6.2: Given a simply connected orthogonal polygon Q, let $p_1(0) = \ldots = p_N(0) = s$, represent the initial positions of an asynchronous network of N agents as described in Section III. Let s be a reflex vertex of Q with an adjacent convex vertex. Let the behavior of the agents be governed by the `Connected Depth-first Deployment` algorithm. Assume N_f agents fail in finite time. Then the following are true:

(i) there exists a finite time t^* after which there is at least one agent on $\min\{|\mathcal{N}_{\text{vc-}Q}(s)|, N - N_f\}$ nodes of $\mathcal{G}_{\text{vc-}Q}(s)$.
(ii) if $N - N_f \geq \frac{n-2}{2}$, then the connected visibility-based deployment problem is solved in finite time.

Remark 6.3: If no agents are assumed to fail, then $N_f = 0$. Thus the visibility-based deployment problem with connectivity constraint is solved in finite time in $N \geq \frac{n-2}{2}$.

VII. Conclusions

We have presented distributed asynchronous algorithms for agents equipped with line-of-sight sensing and communication capabilities in simply connected orthogonal polygons. Provably correct algorithms are designed to solve the deployment problem, both with and without constraints on the connectivity of the final agent configuration. For both problems, the number of agents sufficient to complete the task is the same as the number if the environment was known a priori. When connectivity constraints are imposed, the proposed algorithm is robust to individual agent failures.

VIII. Acknowledgment

This material is based upon work supported in part by AFOSR Award F49620-02-1-0325, by NSF Award CMS-0626457, and by NSF CAREER Award ECS-0546871.

References

[1] V. Chvátal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory. Series B*, vol. 18, pp. 39–41, 1975.

[2] J. Kahn, M. Klawe, and D. Kleitman, "Traditional galleries require fewer watchmen," *SIAM Journal on Algebraic and Discrete Methods*, vol. 4, no. 2, pp. 194–206, 1983.

[3] V. Pinciu, "Connected guards in orthogonal art galleries," in *Computational Science and Its Applications (ICCSA)* (V. Kumar, M. L. Gavrilova, C. J. K. Tan, and P. L'Ecuyer, eds.), vol. 2669 of *Lecture Notes in Computer Science*, pp. 886–893, Springer Verlag, 2003.

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

[5] J. Cortés, S. Martínez, 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.

[6] R. Simmons, D. Apfelbaum, D. Fox, R. Goldman, K. Haigh, D. Musliner, M. Pelican, and S. Thrun, "Coordinated deployment of multiple heterogenous robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, (Takamatsu, Japan), pp. 2254–2260, 2000.

[7] I. Rekleitis and V. Dujmović, "Efficient topological exploration," in *IEEE Int. Conf. on Robotics and Automation*, (Detroit, MI), pp. 676–681, May 1999.

[8] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Topological exploration with multiple robots," in *International Symposium on Robotics and Applications*, (Anchorage, Alaska), May 1998.

[9] M. Dynia, J. Kutylowski, F. Meyer auf der Heide, and C. Schindelhauer, "Smart robot teams exploring sparse trees," in *International Symposium of Matematical Foundations of Computer Science*, (Stará Lesná, Slovakia), Aug. 2006.

[10] P. Fraigniaud, L. Gąsieniec, D. R. Kowalski, and A. Pelc, "Collective tree exploration," in *LATIN 2004* (M. Farach-Colton, ed.), vol. 2976 of *Lecture Notes in Computer Science*, pp. 141–151, Springer Verlag, 2004.

[11] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.

[12] A. Lubiw, "Decomposing polygonal regions into convex quadrilaterals," in *First Annual Symposium on Computational Geometry*, (Baltimore, MA), pp. 97–106, 1985.

Appendix

A *1-rectangular region* is defined as follows, see [12].

Definition 1.1: A 1-rectangular region is a polygonal region without holes and with a distinguished edge e called the tilted edge such that: (i) there are an even number of edges; (ii) the edges except possibly e are alternately (around the polygon) horizontal and vertical; (iii) all interior angles are less than or equal to $270°$; (iv) the nose of the tilted edge contains no vertices.

The *nose* of the tilted edge is the triangular region inside the 1-rectangular region with one horizontal side, one vertical side, and the tilted edge as hypotenuse. The nose is closed along the hypotenuse, open along the other two sides, and excludes the three corners; see Figure 4 (a). The nose of a horizontal or vertical side is empty. An orthogonal polygon without holes is 1-rectangular since any edge is a tilted edge. Given a tilted edge $[a, b]$ of a 1-rectangular region such that b is a convex vertex, we may assume one of the two cases shown in Figure 4 (b). The other cases can be obtained by reflection and/or rotations of these two. We now describe an algorithm to compute a convex quadrilateral with one edge as $[a, b]$ [12]. This quadrilateral is termed *removable* since its deletion from the 1-rectangular region leaves behind polygon(s) that are in turn 1-rectangular. Let (x_a, y_a), (x_b, y_b) and (x_p, y_p) represent the points a, b and p, where p is defined in the algorithm. H_g is the half-plane defined by a constraint g.

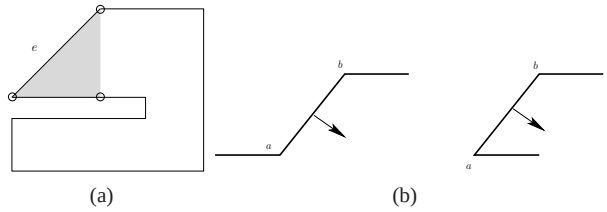

Fig. 4. Left: A 1-rectangular region; tilted edge e and nose (shaded region). Right: Illustration of the possible cases when $[a, b]$ is a tilted edge of 1-rectangular region and b is a convex vertex. Arrows point towards the polygon interior.

Algorithm: `Removable-quadrilateral`

Input: 1-rectangular region Q, tilted edge $[a, b]$ such that either a or b is a convex vertex

$\mathcal{R}_1 := (H_{x \geq x_b} \cap H_{y \leq y_b} \cap H_{y > y_a}) \setminus \{(x_b, y_a), b\}$.

Let $p \in \mathrm{Ve}(Q) \cap \mathcal{R}_1$ be of minimum x then maximum y.

$\mathcal{R}_2 := (H_{x > x_a} \cap H_{x \leq x_p} \cap H_{y \leq \frac{y_p - y_a}{x_p - x_a}(x - x_p) + y_p}) \setminus \{a, p\}$.

Let $q \in \mathrm{Ve}(Q) \cap \mathcal{R}_2$ be of maximum y and then maximum x.

`If` q does not exist `then`

  Let f be the horizontal edge vertically below p.

  Let y^f be the y coordinate of f.

  $\mathcal{R}_3 := (H_{x \geq x_p} \cap H_{y < y_b} \cap H_{y \geq y^f}) \setminus \{p, (x_p, y^f)\}$.

  Let $q \in \mathrm{Ve}(Q) \cap \mathcal{R}_3$ be of minimum x and then minimum y.

Return: Quadrilateral $abpq$

See Figure 5 for a graphical illustration of $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$. We refer the reader to [12] for a proof on the existence of p and q and hence the existence of $abpq$.

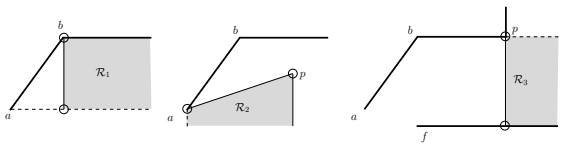

Fig. 5. Illustration of regions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$.