HybridExploration: a Distributed Approach to Terrain Exploration using Mobile and Fixed Sensor Nodes

Ettore Ferranti, Niki Trigoni and Mark Levene

***Abstract*— When an emergency occurs within a building, it may be initially safer to send autonomous mobile nodes, instead of human responders, to explore the area and identify hazards and victims. Exploring all the area in the minimum amount of time and reporting back interesting findings to the human personnel outside the building is an essential part of rescue operations. Our assumptions are that the area map is unknown, there is no existing network infrastructure, long-range wireless communication is unreliable and nodes are not location-aware. We take into account these limitations, and propose a novel algorithm, HybridExploration, that makes use of both mobile nodes (robots, called *agents*) and stationary nodes (inexpensive smart devices, called *tags*). As agents enter the emergency area, they sprinkle tags within the space to label the environment with *states*. By reading and updating the state of the local tags, agents are able to coordinate indirectly with each other, without relying on direct agent-to-agent communication. In addition, tags wirelessly exchange local information with nearby tags to further assist agents in their exploration task. Our simulation results show that the proposed algorithm, which exploits both tag-to-tag and agent-to-tag communication, outperforms previous algorithms that rely only on agent-to-tag communication.**

I. Introduction

When an emergency occurs within a building, it is crucial for the first responders to acquire as much information as possible on the ongoing situation, in order to identify and contain hazards and coordinate the rescue of victims. Initially, the area is off-limits and hazardous for anyone not wearing respiratory equipment, garments or barrier materials to protect themselves from exposure to biological, chemical, and radioactive hazards. This kind of suit can be very heavy and bulky, consequently limiting the first responders' movements, and reducing their sensing capacity (touch, vision, and hearing). A group of autonomous mobile nodes, referred to as *agents*, should therefore be deployed in the area to acquire all the information that could assist the tasks of the first responders.

Effort sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-08-1-3022. The U.S. Government is authorized to reproduce and distribute reprints for Government purpose notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

E. Ferranti and N. Trigoni are with the Computing Laboratory, University of Oxford, Oxford, UK Ettore.Ferranti | Niki.Trigoni@comlab.ox.ac.uk

M. Levene is with the School of Computer Science and Information Systems, Birkbeck College, University of London, UK Mark@dcs.bbk.ac.uk

The rescue operations however may be obstructed by a number of limitations, e.g. the possible lack of a terrain map, the failure of previously established networks, and the short-range and often unreliable wireless indoor communication. In addition, it might be difficult to use GPS positioning inside a building, so an agent cannot rely on knowledge of its exact location within the terrain.

In this paper, we take into account these limitations, and assume that agents can rely only on local information that is sensed in their vicinity (which other agents have left behind them as a trace), before making the next exploration step. We propose an approach to area exploration in which a swarm of *agents* enter the emergency area and dynamically deploy a network of stationary sensor nodes, referred to as *tags*, in order to label the environment. Agents do not communicate directly with each other; instead, they coordinate indirectly by leaving traces of information on the tags that they deploy on the space. In addition, the set of deployed tags form a multi-hop wireless network, to further assist the agents in their exploration task.

The key contribution of our proposed exploration algorithm, named HybridExploration, is that it combines two modes of communication: between agents and tags (agent-to-tag) and multi-hop communication within the stationary network of tags (tag-to-tag). It is fully distributed and does not require centralized control of the agents to determine their next move. It only exploits short-range communication between agents and tags, and among tags, and does not use unreliable long-range communication among agents, or agents and the human responders.

The rest of the paper is organized as follows. Section II presents the model, objectives and assumptions of our work, and Section III describes the new HybridExploration algorithm. Section IV presents a thorough experimental analysis of the proposed algorithm and the three competing approaches. An overview of related work is provided in Section V, followed by conclusions and directions for future work in Section VI.

II. Model

We consider the task of exploring a hazardous terrain using a group of autonomous *agents*. We assume a very simple model of the area, in which the environment is divided into a grid of square cells, whose size depends on both sensing coverage and communication range of the agent. In particular, when an agent is at the centre of a cell, it must be able to cover the entire area with a sensor attached to it to scan for victims or hazards. Therefore, the size x of a

cell must be determined by the range of the sensor (r_{sense}) and by the communication range of the agent (r_{comm}). In particular, if we refer to Figure 1, it must be that $x \leq \frac{2r_{sense}}{\sqrt{2}}$. Furthermore, since the agent must be able to communicate with wireless nodes, referred to as *tags*, in everyone of the 8 cells around it, it must be that $x \leq \frac{2r_{comm}}{3\sqrt{2}}$.

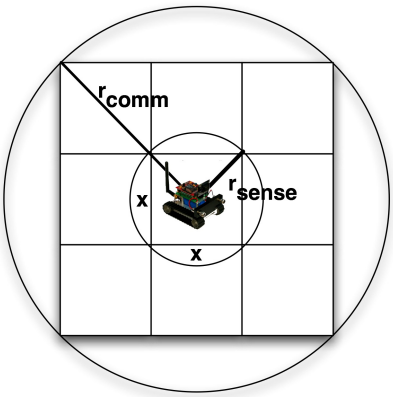

Fig. 1. Example of an agent equipped with a sensor. The sensor must cover the entire area of the cell, and the agent must be able to communicate with the 8 cells around it.

A cell can be in one of the following states:

- ***Wall***: The cell cannot be traversed by an agent because it is blocked by an obstacle.
- ***Unexplored***: No agent has been in the cell yet, and therefore no tag has been deployed there yet.
- ***Explored***: The cell has been traversed at least once, but the agents might need to go through it again in order to reach other *unexplored* cells.
- ***Visited***: The agents have already explored the cell, and they do not need to go through it again to reach other cells.

In the remainder of this section, we provide a high level overview on the agent movement and tag deployment, and discuss the agent-to-tag and tag-to-tag communication modes.

Agent movement and deployment of tags: Agents are initially deployed in one of the boundary cells and, in each step, they are able to move from the current cell to one of the four adjacent cells in the North, East, South or West directions. As they move to an *unexplored* cell, they deploy a miniature device (e.g. mote or RFID), referred to as *tag*, capable of storing small amounts of information about the state of the local cell.

Agent-to-tag communication: In emergency situations, long-range wireless communication may be intermittent and unreliable, so we assume that agents are able to communicate only by reading and updating the tags installed in the local and 8 neighbouring cells. We thus only consider distributed exploration algorithms, in which agents make independent decisions about how to navigate through the terrain based on local state.

Tag-to-tag communication: In the proposed algorithm, HybridExploration, we introduce *virtual agents*, i.e. active messages that cause the execution of a small piece of code on the tag that receives them. Virtual agents alter the state of the local tag and are further disseminated to neighboring tags within communication range. The underlying assumption of our model is that a tag in a cell is able to communicate wirelessly with the tags in the four adjacent cells.

A. Objectives

We are now going to introduce two objectives, which we will use to assess the performance of the algorithm we are going to examine in Section III and IV.

1) **Exploration Objective**: all cells in the area are traversed by an agent at least once. This means that no cell is left in the *unexplored* state. When this objective is achieved, cells can be in any of the *explored*, *visited* or *wall* states.
2) **Termination Objective**: all cells in the area are either *walls* or *visited*. No cell is left in the *unexplored* or *explored* state.

By definition, the *Exploration Objective* is always achieved earlier (or at the same time as) than the *Termination Objective*. Both objectives should be achieved in the minimum amount of time, because in an emergency scenario as the one we are considering, speed is essential. The faster the *Exploration Objective* is achieved, the faster victims and hazards are identified. The quicker the *Termination Objective* is achieved, the earlier human responders can enter the area with the certainty that there are no hidden hazards. The efficiency of an algorithm can be measured by how fast it is able to achieve both the *Exploration* and *Termination Objectives*. The goal of the present work is therefore to devise an efficient algorithm that achieves both objectives in a rapid manner.

III. The HybridExploration algorithm

The HybridExploration algorithm gracefully combines two parallel protocols, one followed by physical agents (robots, or simply agents) and one followed by virtual agents. A physical agent takes significantly longer to move from one cell to another (physical robot motion) than a virtual agent (message propagation). Hence, the time of completing the exploration task is measured as the minimum number of physical steps required by physical agents to explore the entire area.

In Section III-A, we describe in detail the protocol that runs on the physical agents. This protocol enables them to cover the entire area of interest eventually, but it has two weaknesses. First, physical agents are often inefficient in exploring the area, as they cover the same cells multiple times, instead of focusing their efforts on *unexplored* parts of the space. Second, although physical agents eventually manage to visit every cell at least once, they are not aware when this happens, i.e. they have no indication of when the exploration task terminates. These two problems motivate the introduction of the *Virtual Agent Protocol* in Section III-B.

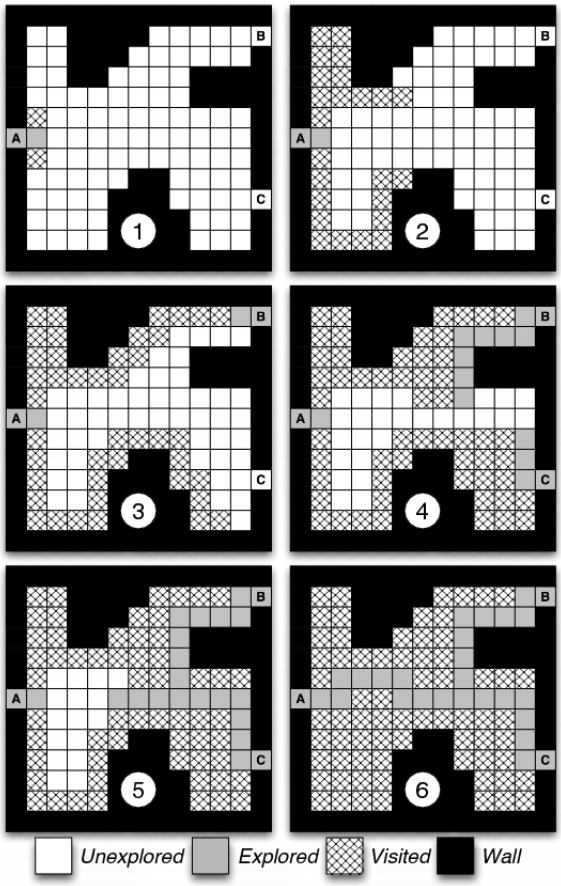

Fig. 2. Two physical agents running the Physical Agent Protocol are used to explore a room.

The two protocols work in synchrony and together constitute our proposed HybridExploration algorithm.

A. Physical Agent Protocol

The main idea behind the algorithm followed by the physical agents is that of thickening the existing walls by progressively marking the cells that surround them as *visited*. Referring to Figure 2, cells A, B and C represent doors. The agents enter the room from door A and gradually thicken the walls by marking cells adjacent to walls as *visited* (Stages 1, 2 and 3). Agents stop thickening walls if they are at risk of disconnecting two *unexplored* parts of the network. For example, in Stage 4, the physical agents create two corridors of *explored* cells in order not to disconnect *unexplored* cells in the inner part of the room from *unexplored* cells in other rooms (beyond doors B and C). When the exploration of the current room is finished (Stage 6), the cells A, B and C are connected by corridors of *explored* cells. These corridors will allow agents to traverse the room through doors A, B and C to access other *unexplored* rooms in the building. In the description of the algorithm, we refer to *wall* and *visited* cells as inaccessible cells, and to *unexplored* or *explored* cells as accessible cells. The algorithm aims to progressively thicken the blocks of inaccessible cells, whilst always keeping accessible cells connected. The latter can be achieved by maintaining corridors of *explored* cells that connect all *unexplored* parts of the network.

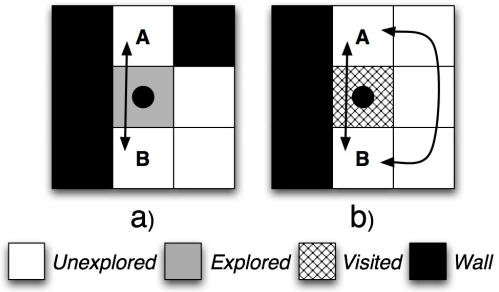

Fig. 3. Marking rules: the agent must decide to mark the current cell as *explored* or *visited*. In the first example (a) the current cell cannot be marked as *visited* because it is the only available passage between adjacent cells A and B. In the second example (b) the current cell is marked as *visited* because there is an alternative passage between A and B on the right.

The Physical Agent Protocol consists of two discrete steps. In the *marking step*, the agent marks the current cell choosing between the *explored* and *visited* states. In the *navigation step*, the agent decides which cell to go to next.

Marking step: Every time an agent is in an *unexplored* cell (with no tags on it), it deploys a tag and updates the state of the cell, choosing between the *explored* and *visited* states. The current cell is marked as *visited* if it does not block the path between two accessible cells around it. Otherwise it is marked as *explored*. Figure 3 provides two examples of the marking step: one where the current cell is marked as *explored* (map a), and one where it is marked as *visited* (map b). In the first example, the only path between the two *unexplored* cells A and B traverses the cell in which the agent (black spot) is at the moment, thus the cell cannot be marked as *visited*. In the second example, there is an alternative path on the right end side of the map, thus the current cell can be marked as *visited* without closing the way between A and B. Note that such alternative paths are easy to compute locally, because they are strictly confined to the 8-cell perimeter of the current cell.

Navigation step: In this step, the agents take a decision about which cell to access next. Priority is always given to the *unexplored* cells which are adjacent to the current one (Figure 4a). If the *unexplored* cells are more than one, they can be chosen at random (Figure 4c). If the agent is equipped with a laser sensor or a sonar, and thus capable of detecting if a neighbouring cell is surrounded by other black (obstacles) cells, the cell which is most likely to be marked as *visited* is chosen, i.e. the one with the most black cells around it (Figure 4b). If there are no *unexplored* cells, the *explored* cell which has been visited the least amount of times in the past is selected (Figure 4d). To do that, the agents simply need to increase a counter on the tag of a cell each time they traverse it, and then read all the counters of the adjacent cells and choose the minimum. In this way, an agent which traverses the same cell twice can take different decisions about the next move instead of always going toward the same direction, thus

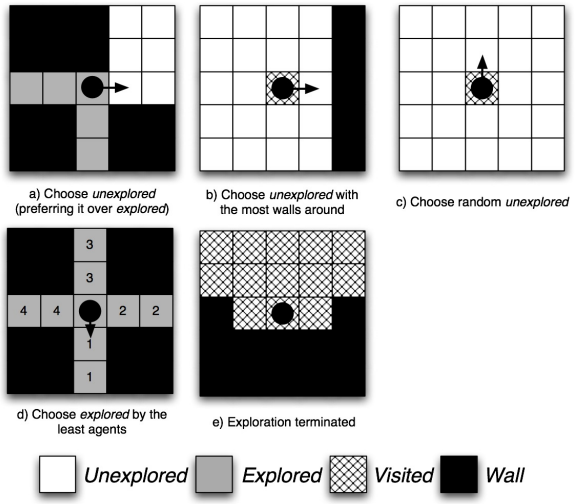

Fig. 4. Different situations in which the agent applies the navigation rules to decide which one of the adjacent cells it will go to during the next move.

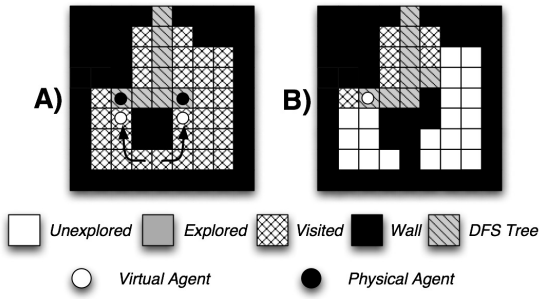

Fig. 5. Exploration rules for virtual agents. (A) Since the virtual agents are just messages, they move infinitely faster than the physical ones. To avoid virtual agents marking as *visited* a branch of the DFS tree in which a physical agent is exploring, the virtual agents always stop when they need to mark as *visited* a cell which is occupied by a physical one. As a result, a virtual agent could follow a physical one and mark the cells immediately behind it, literally "pushing" the physical agent toward *unexplored* areas. (B) When a virtual agent needs to mark as *visited* a cell with at least an *unexplored* adjacent neighbour, it stops until the unknown area is *explored* by a physical agent. In this way, *unexplored* areas will never be disconnected from the *explored* corridors.

avoiding being trapped in a loop. Finally, when the agent is surrounded by inaccessible (*wall* or *visited*) cells, it stops its exploration task (Figure 4e).

B. Virtual Agent Protocol

To speed up the exploration process and eventually achieve the *Termination Objective*, we introduce the concept of virtual agents. These are active messages propagated from cell to cell via the corresponding wireless tags. Virtual agents can only move to cells that are already deployed with tags; they cause changes in the current cell's state and make informed decisions about which cell to traverse next. Like physical agents, virtual agents move from one cell to an adjacent cell in the North, East, South or West directions, and they cannot traverse *visited* or *wall* cells. Unlike physical agents, they cannot traverse *unexplored* cells since there are no tags deployed there. Hence, they consider *explored* cells as their own territory, and build Depth-First-Search (DFS) trees along the corridors of *explored* cells that the physical agents leave behind. The goal of the virtual agents is to remove cyclic paths (loops) of *explored* cells.

The protocol that they run is a variant of the DFS algorithm. The main idea is that virtual agents extend the DFS tree with new *explored* cells in their way downwards, and mark these cells as *visited* when they traverse them in the opposite upward direction. If we consider two agents running the protocol, for example, they will move together and include each *explored* cell in their way into the DFS tree. At the first intersection, they continue to extend the DFS tree, but the one extends the left branch and the other the right branch. When the two virtual agents meet, they cannot extend the DFS tree any further; hence, they traverse the branches upwards marking cells in the way as *visited*.

The example above proved how the Virtual Agent Protocol tries to balance its resources across different branches of the DFS tree (one virtual agent followed the left branch and the other the right branch). The mechanism used for distributing virtual agents to different parts of the network is as follows: each *explored* cell has a counter that measures how many virtual agents have traversed it coming from the same parent cell. Note that a cell can be accessed by a virtual agent from at most one parent cell. As virtual agents traverse the DFS tree downwards, they increment the counters associated with each traversed cell. At intersections, they choose to move to the *explored* cells traversed by the minimum number of virtual agents, so as to assign the same number of agents to each branch of the DFS tree.

We now discuss two rules that virtual agents should follow to work in harmony with physical agents.

Rule 1: a virtual agent cannot mark a cell as *visited* if that cell is occupied by a physical agent, and it always has to wait until the physical agent is gone before continuing marking. An example of this behaviour is depicted in Figure 5(A), where virtual agents are following physical agents while they avoid "overtaking" them and subsequently blocking them in a branch with *visited* cells.

Rule 2: a virtual agent cannot mark a cell as *visited* if at least one of the adjacent cells is *unexplored*. An example of this case is provided in Figure 5(B), where the virtual agent stops any activity until the adjacent cell is *explored* by a physical agent.

To summarize, the Virtual Agent Protocol is capable of quickly removing cyclic paths of *explored* cells, formed by the Physical Agent Protocol. The role of virtual agents is to assist physical agents, by following them closely and cleaning up unwanted *explored* states in some of the cells, in order to help physical agents achieve the *Exploration* and *Termination Objectives* faster. The rules followed by virtual agents ensure that they always work in harmony with physical agents, i.e. they never delay or trap physical agents before the *Termination Objective* is achieved. For a detailed description of both Physical and Virtual Agent Protocols, the interested reader can refer to the pseudocode provided in [1].

IV. Simulation Results

We developed a simulation tool to test the performance of the proposed HybridExploration algorithm and competing approaches (i.e. Ants [2], MDFS [3], and Brick&Mortar [3]). All of the tested approaches use an Agent-to-Tag communication paradigm, and assume an environment divided in a grid of square cells. In particular, agents running Ants leave traces on the cells they visit and then always choose to move to the least visited cell. MDFS instead is a distributed version of the DFS algorithm, in which agents use tags in the cells to coordinate and explore a single DFS tree. Brick&Mortar finally lets the agents mark the already visited cells, so that they do not need to be traversed in the future, while leaving corridors connecting all the unexplored areas of the map, until every single cell has been traversed. All of the above uses simple tags that can only be updated by agents (e.g. passive RFID), while we believe that by using HybridExploration, with a Tag-to-Tag communication paradigm, we might improve the overall efficiency of the exploration.

The Ants algorithm performance appears in the graphs reporting only the exploration times, because this approach is not capable of achieving the *Termination Objective*. This happens because the Ants agents are not able to identify when the exploration terminates, therefore all cells remain permanently marked as *explored* and none of them is subsequently marked as *visited*. We are able to study the impact of i) the number of obstacles, ii) the terrain size, and iii) the number of rooms on the performance of the three algorithms. In each experiment, we vary the values of one parameter, and assign default values to the remaining ones. The default values are: a map of 2500 (50x50) cells with 30 obstacles and 36 (6x6) rooms, which is explored by 20 agents. The agents are deployed from the top left cell of the area. We consider two performance metrics: i) the *exploration time* (Figure 6), i.e. the number of steps required to achieve the *Exploration Objective*, and ii) the *visiting time* (Figure 7), i.e. the number of steps required to achieve the *Termination Objective*. We chose the number of steps to evaluate our performance metrics because it represents the major source of energy consumption for the robot during the exploration process.

We also consider two different area types: i) *Office*: area inspired by a real building plan, with corridors, offices, and an open-space area. ii) *Series*: a long corridor which traverses several rooms. The latter scenario is inspired by a mine or another underground map in which each room has a door entering from the previous one and another door leading to the next. Our objective is to investigate how the performance of the proposed and competing algorithms varies depending on the spatial layout of rooms and doors in a building.

Effect of obstacles: Let us first study the impact of obstacles on the performance of HybridExploration and competing algorithms, as shown in Figures 6A/B and 7A/B. In both scenarios, the introduction of obstacles does not seem to slow down MDFS towards achieving both *Exploration* and *Termination Objectives*. In contrast, Brick&Mortar, which has a complex and time-consuming mechanism for resolving loops around obstacles [3], suffers from having to resolve an increasing number of obstacles. Furthermore, as one would expect, the plots denoting the exploration time of Brick&Mortar and MDFS cross over towards the middle of the x-axis, since Brick&Mortar is faster with few obstacles, but becomes inefficient with many obstacles.

Our proposed algorithm, HybridExploration, has lower exploration and visiting times than the others in both scenarios, and for varying numbers of obstacles. The reason is that it combines the ability of Brick&Mortar to quickly mark cells as *visited* when there are no obstacles, with the ability of MDFS to resolve loops, when there are many obstacles. It handles the presence of loops very well thanks to the virtual agents which are able to close them very quickly after they are created, while the physical agents are free to explore the remaining part of the area as fast as possible. The comparative benefits of HybridExploration are more pronounced in the Series scenario (Figures 6B and 7B), where it is up to 50% faster than Brick&Mortar and MDFS.

Brick&Mortar is affected by obstacles, while MDFS can cope with them much more easily. HybridExploration, thanks to the virtual agent protocol, can maintain good performance even if the number of obstacles is increased.

Effect of area size: The next question that we address is whether the algorithms scale gracefully as we increase the number of cells in the area. A comparison of the four algorithms in both scenarios is depicted in Figures 6C/D and 7C/D. As one would expect, as we increase the area size, the exploration and termination times increase for all algorithms in all scenarios. Let us take a closer look at the behavior of MDFS and Brick&Mortar. In terms of exploration time, small areas favour MDFS over Brick&Mortar, whereas in large areas Brick&Mortar performs better than the MDFS. The two algorithms meet towards the middle of the x-axis in all three graphs. The reason is that MDFS typically traverses each cell more than once, whereas Brick&Mortar only once unless it has to resolve loops. The visiting times of the two algorithms is very close to the respective exploration times.

Our HybridExploration algorithm, which combines the strengths of MDFS and Brick&Mortar, outperforms both of them (and Ants) in every scenario, and scales gracefully with the area size. In the Series scenario (Figures 6D and 7D), where the loops around obstacles are longer than in the other scenario, the ability of HybridExploration to close loops using virtual instead of physical agents gives it a significant advantage over Brick&Mortar and MDFS.

HybridExploration scales gracefully with the number of cells, thanks to the positive effect of the Physical agent protocol.

Effect of rooms: Figures 6E/F and 7E/F show the effect of varying the number of rooms in the area while maintaining the same number of cells, and thus the same area size. It is peculiar how Brick&Mortar performs poorly in terms of visiting time, when there are no rooms at all (open-space area). This can be explained by observing that the loops are

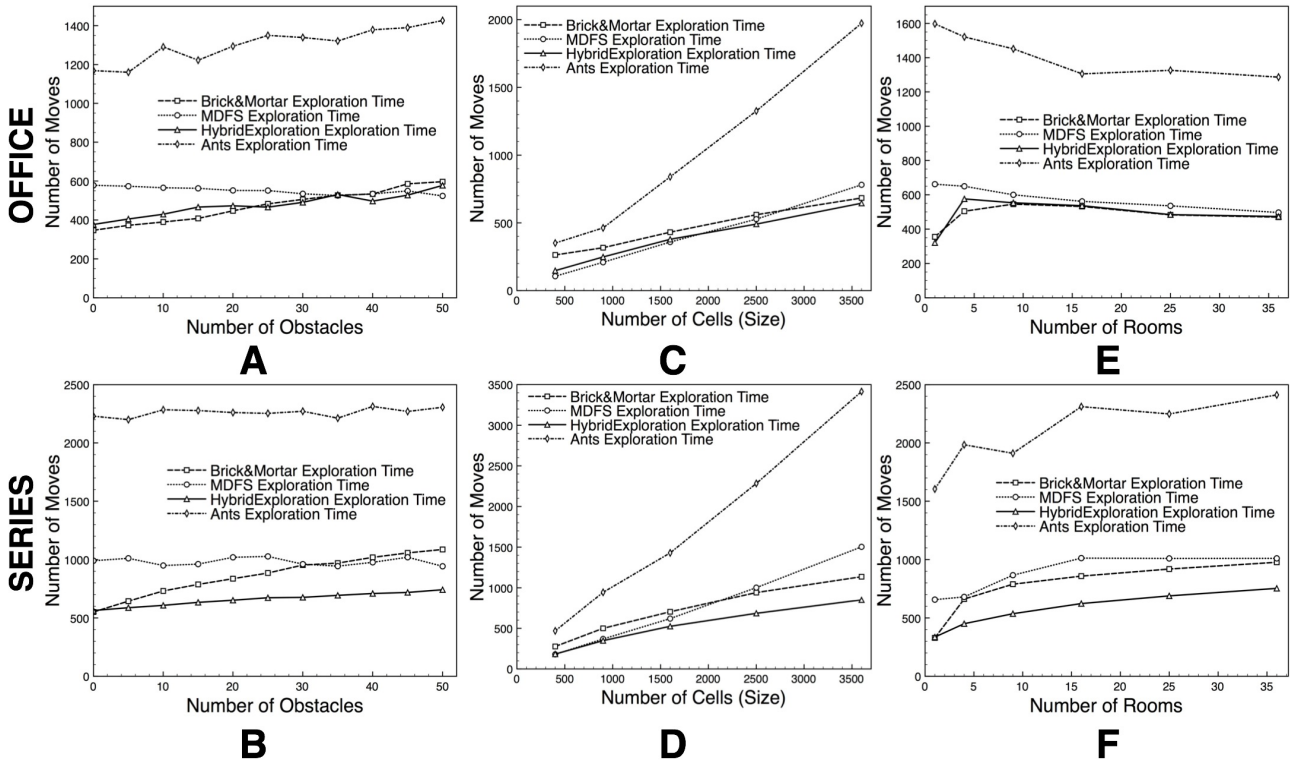

Fig. 6. Effect on exploration time, when changing number of obstacles, size of the area or number of rooms.

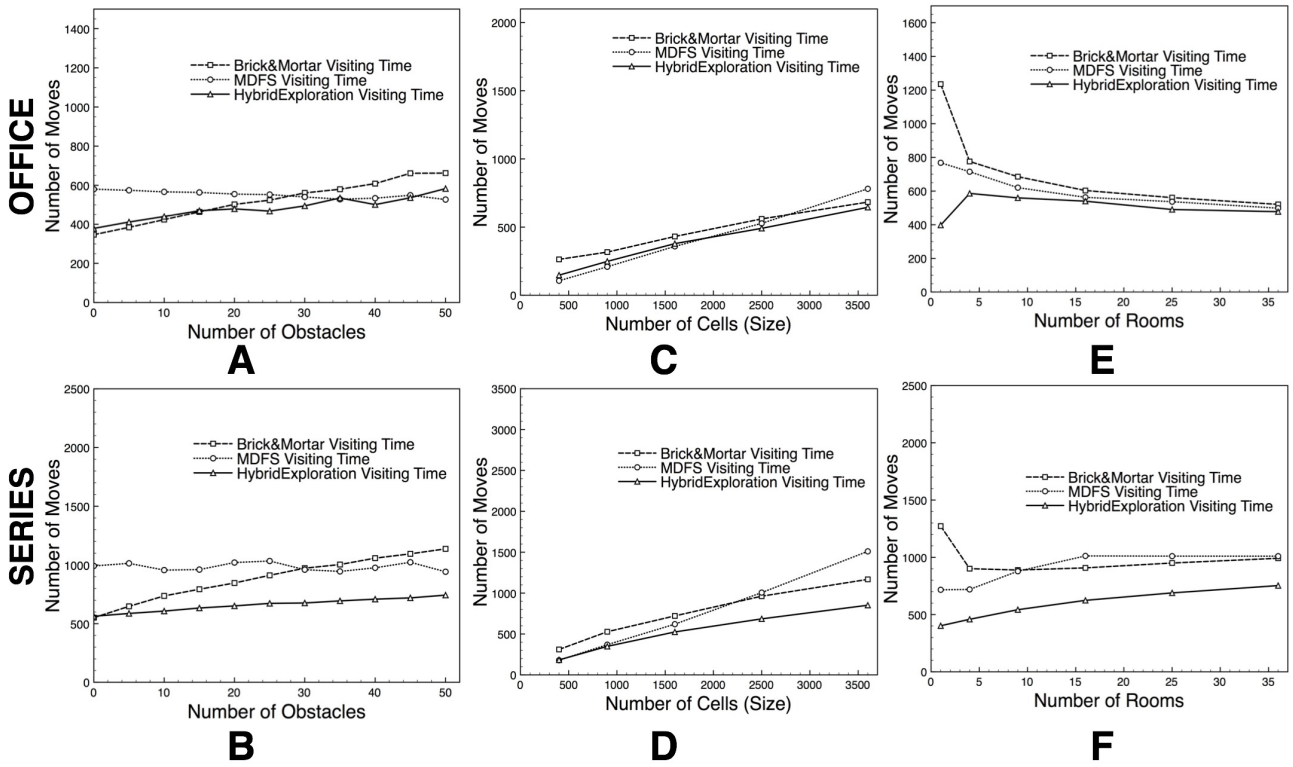

Fig. 7. Effect on visiting time, when changing number of obstacles, size of the area or number of rooms.

not bound within a room but the agents can build loops which are as large as the whole area, and therefore the time needed to close them is much longer than in cases where a loop cannot be larger than the size of a room.

The behavior of MDFS is also interestingly different depending on the scenario. In the *Office* scenario (Figures 6E and 7E), the performance of MDFS both in terms of exploring and visiting times hardly depends on the number of rooms. We observe a small decrease in the exploration and visiting times of MDFS, simply because some of the cells that were previously *unexplored* are now *wall* cells forming the frames of rooms. These *wall* cells do not require to be traversed and marked as *explored* or *visited*. The behavior of MDFS in the *Series* scenario (Figures 6F and 7F) is rather unexpected. MDFS is slowed down by the presence of rooms although with more rooms there are fewer cells to cover. We think that this is because one of the strengths of the MDFS algorithm is that the agents can build trees of *explored* cells, with several branches spanning different rooms in the scenario, and process them at the same time. This parallelisation though is not possible in the *Series* scenario, where the rooms form a chain and need to be explored one after the other, without the possibility to explore more than one at the same time. The agents running MDFS therefore need firstly to mark the cells in the room as *explored*, then traverse them again to mark them as *visited* and finally move on to the next room, without efficiently using all the agents in a parallel way during the exploration. The Brick&Mortar and HybridExploration algorithms on the contrary can directly mark every cell as *visited*, so that, although they cannot explore more rooms at the same time in parallel, the overall exploration and visiting times are much less than the MDFS ones.

The HybridExploration algorithm outperforms the other two algorithms in both scenarios; again, the Series *scenario presents the most interesting case, where the benefits of HybridExploration are more pronounced.*

V. Related Work

Choset [4] provides a survey of coverage algorithms and distinguishes them into *off-line* and *on-line*. In the former, the agents are previously provided with a map of the area to explore, while in the latter, also called *sensor-based*, no assumption is made concerning the availability of an environmental map for the agents. Our approach differs from related work in that we are investigating the subclass of *on-line* algorithms that rely on communication through the instrumented environment.

Since the *off-line* algorithms previously know the map of the environment, in theory they could build a set of optimal paths to cover the area using all the available agents in the minimum amount of time (optimal solution). In practice, this is not possible because this problem is NP-complete, therefore heuristics are needed to find a feasible solution in polynomial time. Such a heuristic is proposed in [5], where the authors prove that the original problem is NP-complete, and propose a polynomial algorithm, which runs in the worst case eight times slower than the optimal solution. Agmon et al. [6] propose a faster tree construction algorithm, while Hazon et al. [7], [8] focus on the robustness of the solution, so that even if only one robot remains in operation, it will be able to carry and complete the exploration task.

To the best of our knowledge, little work [9], [2], [10], [3] has investigated the problem of *on-line* area exploration by letting agents coordinate indirectly by tagging the environment, subsequently reading and updating the state of the deployed tags. Moreover, existing algorithms are not able to autonomously decide when the exploration is terminated. Recognising when the exploration terminates is of primary importance in an emergency scenario such as the one we are tackling: if the robots have to report back to the first responders the situation inside the building, they absolutely need to know when to stop exploring, and to do that they need to be sure whether all the area has been explored or not.

In our previous work, we proposed two algorithms, MDFS and Brick&Mortar, to solve the terminating issue in a distributed fashion by marking cells already visited by the agents, but these algorithms do not exploit tag-to-tag communication, i.e. they do not use the multi-hop communication capabilities of the deployed sensor network. Furthermore, agents running MDFS are poorly coordinated, and this leads toward long exploration times, while agents running Brick&Mortar use a complex loop resolution mechanism that significantly delays the task of area exploration, especially in topologies with many obstacles (e.g. desks in the middle of an open space).

The feasibility of the approach is supported by Hähnel et al. [11], who proved how a robot can use RFID tags already placed on an area to localize itself and navigate through the rooms, and recently by Kleiner et al.[10], who presented a robot which is able to autonomously drop RFID tags on the environment and implement an existing *on-line* exploration algorithm [12].

One of the first *on-line* methods that uses a cellular decomposition of the environment is the ants-inspired algorithm presented in [9] and [2]. The area is divided into a grid of square cells on which the exploring agents leave traces of their passage, similarly to real ants leaving pheromone. Agents (or ants) tend to move to the least visited cells, i.e. the cells with the least amount of pheromone, and they increase the number of visits (pheromone) as they hop onto a cell. A similar approach to the Ants algorithm uses a sensor network infrastructure to provide agents with information about the visited areas and direct them to the least recently visited direction [13].

Batalin and Sukhatme have done plenty of work [13], [14], [15], [16] using radio beacons to guide the navigation of robots and assist them in the coverage of an unknown terrain. In particular, their robots are able to detect the beacons (which are pre-deployed into the environment), choose one of them and move toward it, always guided by their radios. Furthermore, in [17], the same authors suggest that the beacons could be dynamically deployed by a robot (an approach

similar to our previous work [3]), which could then use the same Least Recently Visited (LRV) navigation approach. We believe that the relatively simple algorithm they use could be further improved to reach better performance in exploring the area as fast as possible.

A different approach (centralized) has been presented in [18]. In this work, the authors use a Boustrophedon [19] technique to let the robots cover the area. Yamauchi presents a frontier-based exploration algorithm [20], where the agents explore the environment, represented by a regular grid of cells, keeping in their memory a map of the area and always directing themselves "to the boundary between open space and uncharted territory". The algorithm also makes the same strong assumptions that we found in [18] namely perfect localization, communication and mapping, and uses a centralized communication approach, because the map is stored in a common server.

VI. Conclusion and Future Work

In this work, we proposed a new algorithm, HybridExploration, for the multi-agent exploration of unknown terrains. Our algorithm is based on an architecture consisting of both mobile nodes (robots, called *agents*), and stationary nodes (inexpensive smart devices, called *tags*). As the former enter the emergency area, they drop tags into the environment to label it with a *state*. By reading and updating the state of the local tags, agents are able to coordinate indirectly with each other. In addition, agents are further assisted during the exploration task by tags able to wirelessly exchange local information. We compared our novel approach against three existing algorithms, Ants, Brick&Mortar, and MDFS. Our simulation results show that our HybridExploration algorithm is significantly faster than the three competing ones in a variety of scenarios.

In the future, we would like to explore how to use tags as a network infrastructure, to communicate interesting events back to human responders. Another challenge is to cope with different cell distributions, without assuming that the tags are deployed in a grid network.

References

[1] "Hybridexploration algorithm extended version." web.comlab.ox.ac.uk/oucl/work/ettore.ferranti/HybridExploration.pdf.

[2] J. Svennebring and S. Koenig, "Building Terrain-Covering Ant Robots: A Feasibility Study," *Autonomous Robots*, vol. 16, pp. 313–332, May 2004.

[3] E. Ferranti, N. Trigoni, and M. Levene, "Brick&Mortar: An On-Line Multi-Agent Exploration Algorithm," in *ICRA07: Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp. 761–767, IEEE Press, April 2007.

[4] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[5] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-Robot Forest Coverage," in *IROS05: Proceedings of the 2005 IEEE International Conference of Intelligent Robots and Systems*, pp. 3852–3857, IEEE Press, August 2005.

[6] N. Agmon, N. Hazon, and G. A. Kaminka, "Constructing Spanning Trees for Efficient Multi-Robot Coverage, booktitle = ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation," pp. 1698–1703, IEEE Press, May 2006.

[7] N. Hazon and G. A. Kaminka, "Redundancy, Efficiency, and Robustness in Multi-Robot Coverage," in *ICRA05: Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pp. 735–741, IEEE Press, April 2005.

[8] N. Hazon, F. Mieli, and G. A. Kaminka, "Towards Robust On-line Multi-Robot Coverage," in *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 1710–1715, IEEE Press, May 2006.

[9] S. Koenig and Y. Liu, "Terrain Coverage with Ant Robots: a Simulation Study," in *AGENTS01: Proceedings of the 2001 ACM International Conference on Autonomous Agents*, pp. 600–607, ACM Press, May 2001.

[10] A. Kleiner, J. Prediger, and B. Nebel, "RFID Technology-based Exploration and SLAM for Search And Rescue," in *IROS06: Proceedings of the 20062006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4054–4059, IEEE Press, October 2006.

[11] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and Localization with RFID Technology," in *ICRA04: Proceedings of 2004 IEEE International Conference on Robotics and Automation*, pp. 1015–1020, IEEE Press, April 2004.

[12] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp. 376–378, June 2005.

[13] M. A. Batalin and G. S. Sukhatme, "The Analysis of an Efficient Algorithm for Robot Coverage and Exploration based on Sensor Network Deployment," in *ICRA05: Proceedings of 2005 IEEE International Conference on Robotics and Automation*, pp. 3478–3485, IEEE Press, April 2005.

[14] M. A. Batalin and G. S. Sukhatme, "Efcient Exploration Without Localization ," in *ICRA03: Proceedings of 2003 IEEE International Conference on Robotics and Automation*, pp. 2714–2719, IEEE Press, May 2003.

[15] M. Batalin, G. Sukhatme, and M. Hattig, "Mobile Robot Navigation using a Sensor Network," in *ICRA04: Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 636–642, IEEE Press, April 2004.

[16] M. A. Batalin and G. S. Sukhatme, "Coverage, Exploration and Deployment by a Mobile Robot and Communication Network," in *Proceedings of the International Workshop on Information Processing in Sensor Networks*, pp. 376–391, ACM, April 2003.

[17] M. A. Batalin and G. S. Sukhatme, "The design and Analysis of an Efficient Local Algorithm for Coverage and Exploration," *IEEE Transactions on Robotics*, vol. 23, pp. 661–675, August 2007.

[18] C. S. Kong, N. A. Peng, and I. Rekleitis, "Distributed Coverage with Multi-Robot System," in *ICRA06: Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 2423–2429, IEEE Press, May 2006.

[19] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *FSN97: Proceedings of 1997 International Conference on Field and Service Robotics*, December 1997.

[20] B. Yamauchi, "Frontier-Based Exploration using Multiple Robots," in *AGENTS98: Proceedings of the 1998 ACM International Conference on Autonomous Agents*, pp. 47–53, ACM Press, May 1998.