Demonstration of Multi-Robot Search and Secure

Fotios Katsilieris, Magnus Lindhé, Dimos V. Dimarogonas, Petter Ögren and Karl H. Johansson

Abstract—**We consider the search and secure problem, where intruders are to be detected in a bounded area without allowing them to escape. The problem is tackled by representing the area to be searched as a traversability graph, which is reduced to a tree graph by placing stationary robots to remove loops. The search of the remaining tree is performed using two strategies that represent different trade-offs between the needed search time and the number of robots. Proof of correctness is provided for these two strategies. The proposed algorithm was implemented and demonstrated as part of an outfield experiment involving a team of Rotundus spherical robots.**

I. Introduction

Patrolling industrial plants, storage areas or military camps is tedious at best and can potentially be very dangerous. Furthermore, if coordination is required so the area is guaranteed to be free of intruders after the search is completed, this can prove difficult for humans. This suggests using robots for assistance. The robots can be equipped with sensors specially tailored to the circumstances and data can be fed to a human operator, monitoring the system from a safe location.

Using robots to search an area with guaranteed intruder detection will here be referred to as the *search and secure problem*. Informally we can define it as: given a map of a bounded area with polygonal obstacles, determine trajectories for robots equipped with omnidirectional sensors so that all intruders will be sensed within finite time. This requires coordination to prevent intruders from sneaking back into an area that has already been searched. We allow for any number of intruders and only assume that their movement is continuous. This paper does not consider detection of the intruders from the sensor information, instead an intruder is considered detected if there is a line of sight between a sensor and the intruder.

This paper presents the demonstration of a physical multi-robot system that solves the search and secure problem. The robots are equipped with cameras that wirelessly feed video to an operator. The system uses two different strategies to solve the search and secure problem, representing different trade-offs between the time used and the number of robots. The paper also formally proves the correctness of the strategies and presents simulations that illustrate the performance on larger problems than used in the demonstration.

F. Katsilieris, Magnus Lindhé and Karl H. Johansson are with the ACCESS Linnaeus Centre, School of Electrical Engineering, Royal Institute of Technology, Sweden. E-mail: fotio@kth.se, lindhe / kallej@ee.kth.se.

Dimos V. Dimarogonas is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, USA. E-mail: ddimar@mit.edu

Petter Ögren is with the Totalförsvarets forskningsinstitut (FOI), Stockholm, Sweden. E-mail: petter.ogren@foi.se

The work presented here was part of the AURES (Autonomous UGV-Systems for REconnaissance and Surveillance) project, funded by the Swedish Material Administration through the TAIS (Technologies of Autonomous and Intelligent Systems) programme.

The paper is organized as follows: In Section II we discuss related research. In Section III we present some preliminaries and the procedure for generating a search graph from the map. Section IV covers the two strategies we use, as well as proofs of correctness. In Section V we give some simulation results and Section VI describes the demonstration of the system. Finally, we present conclusions and possible directions for future work in Section VII.

II. Related Research

The search and secure problem is related to the art gallery problem, where one seeks positions for static robots or cameras to fully cover an area. An overview of results is given by Shermer [1]. More recently, Ganguli *et al.* [2] have addressed it in a robotic deployment context. In contrast to the art gallery problem, we do not require instant detection of intruders, but only within a finite time. Another related problem is that of exploration, where there is no escaping intruder. Instead, the goal is to visit all parts of an environment in an efficient manner. For an example, see Brass *et al.* [3].

Searching and securing falls under the broad category of pursuit-evasion problems, which have been studied under different assumptions on the sensing capabilities of the searcher(s) and the geometry of the environment. Suzuki *et al.* [4] studied simple polygons and searchers with one or more search rays, while Gerkey *et al.* [5] introduced a searcher with a nonzero field of view. Isler *et al.* [6] triangulated the search space, abstracted it into a graph and also suggested using stationary robots to reduce more complex environments to simple polygons. We employ this framework, but instead of the randomized search strategy of Isler *et al.* or Kolling and Carpin [7], we propose a deterministic approach with guarantees of capture in finite time (which is given along with the robot trajectories at the end of the execution of the algorithm).

The discrete problem of searching a graph was was initially addressed by Parsons [8], who, for each k, characterized the class of trees that require at least k searchers. For a general graph, Megiddo *et al.* [9] have shown that finding the minimal number of searchers needed is NP-hard. Recently, Kolling and Carpin [10], [11] have reported results on searching indoor environments by abstracting them as graphs and reducing the graphs to trees. They use rooms and doorways as vertices and edges, which gives a weighted graph where several searchers may be required to block an edge or search a vertex. We have instead chosen to base the graph on the triangulation, which suits outdoor environments better and also permits some simplifications of the graph, as shown later. Further, we present two search strategies that offer a trade-off between the number of searchers and the search time.

The final contribution of this paper is to report on the implementation and demonstration of our search strategies in a physical multi-robot system.

III. Preliminaries

This section defines some graph theoretic preliminaries and the method used to abstract a map into a traversability graph. Finally, we show how blocking robots can be placed to reduce the remaining graph to a tree.

A. Some Notions from Graph Theory

A *graph* $G = (V, E)$ is a set of *vertices* $v \in V$ and *edges* $e \in E$. Two vertices v and u are considered *neighbors* if there exists an edge $e = (u, v)$ that connects them. In an *undirected* graph, $e = (u, v) \in E \Rightarrow e = (v, u) \in E$.

A *path* in a graph is an ordered set of vertices $\{v_1, v_2, \ldots, v_n\}$ such that $\{v_i, v_{i+1}\} \in E, i = 1, \ldots, n-1$. A graph is *connected* if there exists a path between any two vertices.

A *tree* T is a connected graph where no loops exist. We consider a tree to be undirected, but we designate one of the vertices as the *root*, which gives each edge a natural orientation towards or away from the root. Neighbors in the direction away from the root are called *children* and all vertices except the root have one *parent*, which is the neighbor towards the root. A *leaf* of a tree is a vertex that has no children.

B. Constructing the Traversability Graph

Here we describe how to construct an undirected graph called the traversability graph from the map and what its properties are. First the obstacle-free area of the map is triangulated, as illustrated in Fig. 1. Then the triangles are merged into larger convex regions, shown in Fig. 2. The merging is done in a greedy fashion and a triangle is merged with its neighbors if the merging results in a convex polygon.

This set of regions forms the vertices of the traversability graph. The set of edges consists of all pairs of regions that share a side. This means that a robot or intruder can only move between regions that are neighbors in the traversability graph. Further, because the regions are convex, a robot inside a region will always have a line of sight to any intruder that passes the same region. In contrast to [10], there are no weights associated with edges or vertices and the edges do not represent any physical space. Therefore, only the vertices can contain intruders.

C. Reducing the Graph to a Tree

The traversability graph may have loops that allow an intruder to escape the searching robots. To remove the these, we place stationary robots called *blockers*. When a blocker is placed in a region, the corresponding vertex and all its edges are removed from the traversability graph. By placing enough blockers, we can thus reduce the traversability graph to a traversability tree. In Fig. 3, this could correspond to placing blockers in vertices 6 and 7. The region where the robots start or enter the area is used as the root, denoted $\bar{v}$.

To identify which edges need to be removed, we assign a unit weight to all edges and compute the minimum spanning tree. All edges in the original graph that are not part of the spanning tree are flagged for removal. Then we place the blockers in a greedy manner, by placing each blocker in a vertex that removes as many unwanted edges as possible. As explained below, we can make this edge removal more efficient by taking advantage of the geometry of the original problem.

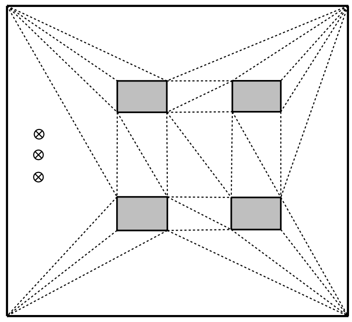

Fig. 1. A bounded region with four obstacles (gray) and three robots (circles). The obstacle-free area is divided into triangles.

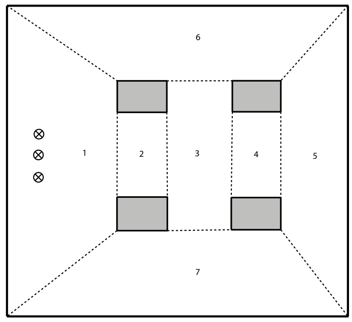

Fig. 2. The triangles are merged into larger convex regions. A robot inside a convex region detects all intruders in the same region.

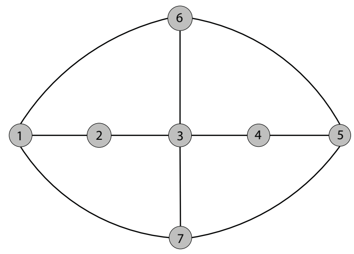

Fig. 3. The traversability graph for the example scenario. The vertices of the graph represent the convex regions and the edges show which regions are neighbors, so a robot or intruder can pass between them.

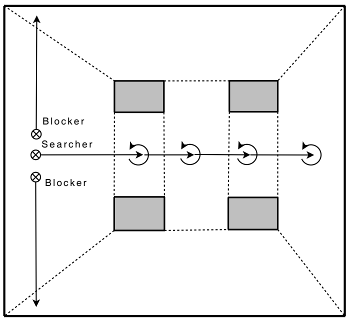

Fig. 4. The robots move along the generated trajectories in order to search and secure the area. First the blockers are moved to their positions, then the searcher is deployed to search the remaining regions and secure the area.

D. Covering Multiple Regions

Because of the convexity of the regions, a region can also be covered from any point on its border. We use this to simplify the problem in two ways: First, we strive to place blockers at corners where two or more regions intersect, which means that all of those regions are covered and can be removed from the graph. Second, it allows a simple pruning operation on the tree graph described above. When searching a leaf, the searcher can stand on the border between the leaf and its parent and thus ensure that no intruder enters the parent region. So to get the problem on the same form as in [8], we remove all leaves before computing the searcher trajectories. This extra movement to the border of each leave is then added to the trajectories before execution. Except in the trivial case of the tree being a single vertex, this pruning can significantly reduce the number of searchers needed. The pruned traversability tree is denoted $\bar{T}$.

E. Executing the Search

When executing the search, we first send the blockers to their positions, then the searchers start. For each robot, we first find the shortest path in the traversability graph. It is then translated to a series of waypoints, using the centroid of each convex region.

To make the path collision-free, we then add a waypoint between each pair of centroids, namely the midpoint of the side that connects the two regions. This ensures that the robots move inside the convex regions at all times. Each waypoint trajectory is then checked against inter-robot collisions, and if a risk is found, one of the robots is delayed some at the waypoint preceding the collision point. To account for the robot size, we add a safety margin around all obstacles when constructing the map.

IV. Searching a Tree Graph

Here we present two strategies for searching the pruned traversability tree and we prove that they work. The depth first strategy uses fewer searchers, but takes longer time than the simultaneous strategy.

The searchers always start from $\bar{v}$ and, as described above, they can only move between neighboring vertices. The intruders are assumed to start from anywhere and move infinitely fast, but also only between neighboring vertices. This leads to the following definitions:

Definition 1 (Contamination): A vertex v is *contaminated* if, after it was last visited by a searcher, there has existed a searcher-free path to it from a contaminated vertex. Originally, all vertices are contaminated.

Definition 2 (Securing a graph): A graph is *secured* if it contains no contaminated vertices.

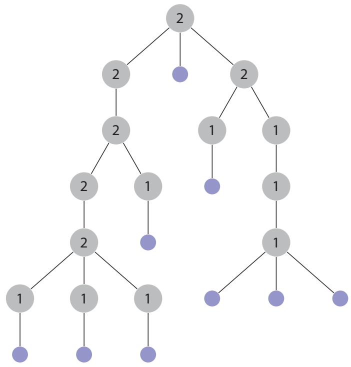

Fig. 5. An example of depth first searching. The leaves of the traversability tree are pruned as a first step (small circles). Then the label $R(v)$ of each remaining vertex describes the number of robots used to search the subtree with root v.

A. Depth First Strategy

The depth first strategy recursively computes a label $R(v)$ for every vertex. We will later show that it represents the number of searchers needed to search the subtree whose root is v, if allow the risk of contaminating the parent of v. We assume that a vertex v has n children, $\{v_1, v_2, \ldots, v_n\}$, without loss of generality ordered so that $R(v_1) \geq R(v_2) \geq ... \geq R(v_n)$. To simplify the notation, we let T_j be the tree with root v_j. The labels are assigned as

$$R(v) = \begin{cases} 1 & \text{if } n = 0 \ (v \text{ is a leaf}) \\ R(v_1) & \text{if } R(v_1) > R(v_2) \\ R(v_1) + 1 & \text{otherwise.} \end{cases} \qquad (1)$$

Now the depth first search strategy can be described as recursively applying the following rule, starting from the root $\bar{v}$ of the pruned traversability tree:

Definition 3 (Depth First Rule): To search a tree with root v, send $R(v_1)$ searchers to search T_1. When they return, send $R(v_2)$ searchers to search T_2 and so on until T_n has been searched. Then return to the parent node. If v has no children, return directly to the parent node.

Fig. 5 shows an example traversability tree with labels $R(v)$. As described in Section III-D, the leaves are pruned in a preprocessing step. To show that the depth first strategy secures the tree in finite time, we first need the following lemma.

Lemma 1 (Recursive Searching): Assume that the subtrees $T_1, T_2, \ldots, T_n$ can be secured with $R(v_1), R(v_2), \ldots, R(v_n)$ searchers, respectively. Then, using the Depth First Rule, the tree with root v can be secured with $R(v)$ searchers, as given by (1) if we allow contamination of the parent of v.

Proof of Lemma 1: There are three possible cases:

- If $n = 0$ then v is a leaf and $R(v) = 1$. It is trivial for one searcher to secure v.
- If $R(v) = R(v_1) + 1$ then the Depth First Rule leads to leaving one searcher in v and then securing all the subtrees in order. The searcher in v stops contamination between subtrees of v.
- If $R(v) = R(v_1)$ then the Depth First Rule uses all searchers in v for securing T_1 first. Unless T_1 is the only subtree, this results in contamination of the parent of v as well as the subtrees $T_2, \ldots, T_n$, since there is no searcher left in v. But according to (1), $R(T_1) > R(T_2) \geq \ldots \geq R(T_n)$. So after returning from T_1, at least one searcher can be left to secure v while the other searchers secure the other subtrees, without risk of contamination between subtrees.

■

We can now state a result on securing the whole pruned traversability tree:

Theorem 1 (Depth First Search): The tree $\bar{T}$ with root $\bar{v}$ can be secured by using the Depth First Rule and $R(\bar{v})$ searchers, as given by (1).

Proof of Theorem 1: Trivially, all leaves of $\bar{T}$ can be searched with one searcher. Then, by induction, Lemma 1 shows that $\bar{T}$ can be searched with $R(\bar{v})$ searchers. And since $\bar{v}$ has no parent, the possible parent contamination in the lemma does not matter. ■

We note that since the Depth First Rule does not require re-securing any subtrees and is applied only once to each vertex, the search will be completed in finite time if $\bar{T}$ has a finite number of vertices.

B. Simultaneous Search Strategy

In the simultaneous search strategy, we search the branches of the tree in parallel, which makes the search faster. This strategy uses another labeling rule, also applied recursively. We use the same notation as above, except the labels are denoted $S(v)$ instead of $R(v)$.

$$S(v) = \begin{cases} 1 & \text{if } n = 0 \ (v \text{ is a leaf}) \\ \sum_{j=1}^{n} S(v_j) & \text{otherwise} \end{cases} \tag{2}$$

The simultaneous search strategy can now be described by the following rule, starting from the root of $\bar{T}$:

Definition 4 (Simultaneous Search Rule): To search a tree with root v, simultaneously send $S(v_j)$ searchers, as given by (2), to search each subtree T_j. If v is a leaf, stay there.

Theorem 2 (Simultaneous Search): The tree $\bar{T}$ with root $\bar{v}$ can be secured by using the Simultaneous Search Rule and $S(\bar{v})$ searchers.

Proof of Theorem 2: If the subtrees can be secured by $S(v_i)$ searchers, there is no risk of recontamination of v, since the Simultaneous Search Rule secures all subtrees at the same time. Hence, the tree with root v can be secured with $S(v)$ searchers. And trivially, a leaf can be secured by one searcher. So by induction, $\bar{T}$ can be secured with $S(\bar{v})$ searchers. ■

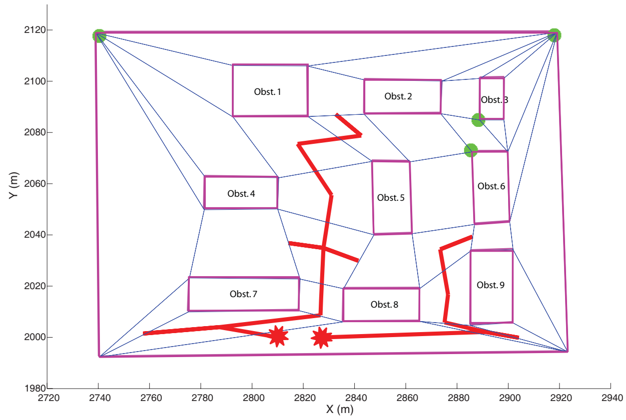

Fig. 6. Searching with two searchers (stars) and four blockers (circles), using the simultaneous search strategy. The thick lines denote the paths of each searcher, while the thin lines show the partitioning into convex regions.

C. Comparison of the Search Strategies

By recursively searching one subtree at a time, the depth first strategy uses more time but less searchers than the simultaneous strategy. The example tree in Fig. 5 would need two searchers if using the depth first strategy, but six searchers if the simultaneous strategy is used. If the search would be done in discrete time, with searchers making one move between vertices per time slot, the depth first search would take 24 time slots to secure the tree (not counting the maneuvers to search pruned leaves). The simultaneous search would take only 5 time slots. In the following sections, we will further illustrate this trade-off between the number of searchers and the search time, both by simulations and in the demonstration.

V. Simulations

In this section, we will illustrate the trade-off between the search time and the number of robots used. In all simulations, the speed of the robots is 1 m/s. As will be described later, the robots used for the physical system have cameras with a limited field of view. They therefore stop in each region and sweep the camera to provide omnidirectional sensing. This is included in the simulation as a small delay in each region, based on the rotational speed of 1 rad/s of the cameras.

A. Area with 9 Obstacles

Fig. 6 shows the result of the simultaneous search strategy, applied to an area with nine obstacles. The corresponding result of the depth first search strategy can be seen in Fig. 7. The simultaneous search strategy requires 6 robots and the total search time is 9 minutes. Using the depth first search strategy, 5 robots are needed and the total search time is 16 minutes and 45 seconds.

B. Area with 24 Obstacles

We now show solutions for a scenario with 24 obstacles. As can be seen in Fig. 8 and Fig. 9, the simultaneous search strategy requires 12 robots and the total search time is 13 minutes and 8 seconds. If the depth first search strategy is used,

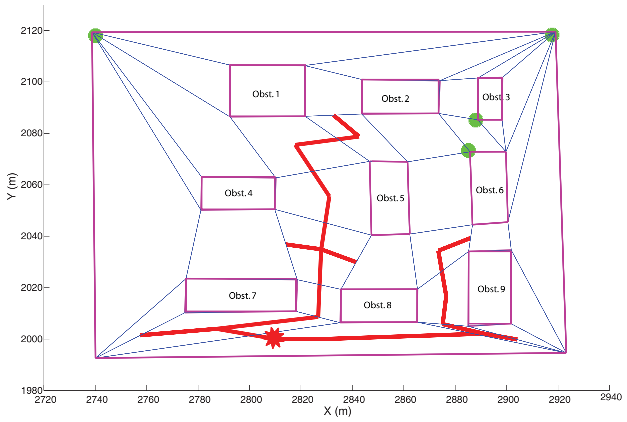

Fig. 7. Four blockers and one searcher, using the depth first search strategy. This takes almost twice as long as when using the simultaneous search strategy.

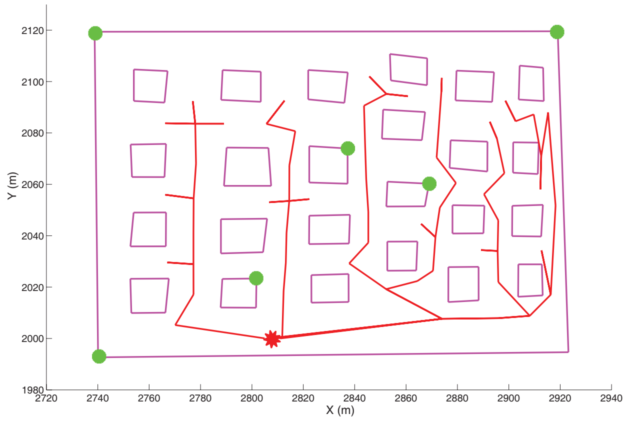

Fig. 8. Six blockers and six searchers, using the simultaneous search strategy.

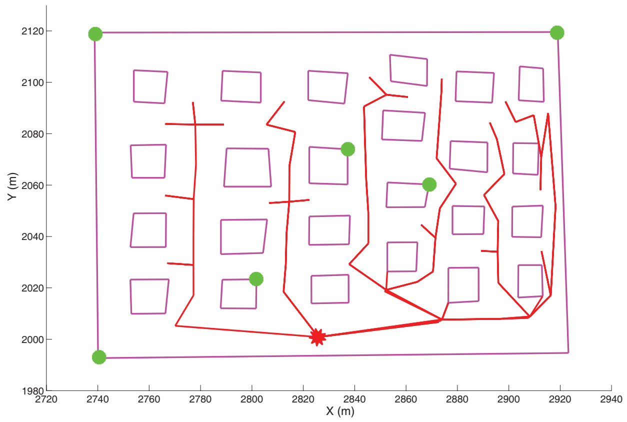

Fig. 9. If the depth first search strategy is used, only three searchers and six blockers are needed.

9 robots are needed and the total search time is 63 minutes and 14 seconds.

VI. Experimental Demonstration

As part of the AURES project, a physical multi-robot system was designed and used in a demonstration of the search and secure problem. In the experiment we assume that the infinite line of sight property is valid but in other parts of the AURES project it is taken into consideration [12]. The demonstration was held at at the premises of Saab Aerotech at Linköping, Sweden. The setup included tents as obstacles, a control room for the operator and spherical GroundBot robots equipped with cameras. One of the robots and part of the demonstration site can be seen in Fig. 10.

The spherical GroundBot is produced by Rotundus [13], has a height of 0.6 m and weighs 25 kg. Its shell is made of polycarbonate with high friction coating. It has two pan-tilt-zoom cameras, giving a 360° field of vision, that can both stream the recorded videos using a Wi-Fi connection. It can reach speeds up to 1.7 m/s and can follow waypoint paths autonomously or be controlled directly via a joystick. It uses GPS, a compass and dead reckoning for navigation and localization. The GroundBot was chosen because of its mechanical reliability (all moving components are contained inside the watertight shell), its relevant sensor suite and because it is safe to humans who enter the search area.

All robots are controlled over a Wi-Fi link, connecting them to a central computer which hosts the mission planning software and interacts with the operator console. The central computer can also run a simulation environment with a realistic physics engine, so the operator and planning software can use exactly the same interface to interact with simulated robots instead. In more complex demonstration scenarios one can use a mix of physical and simulated robots thanks to the modular design of the system.

Fig. 11 shows the operator console of the system (AURES controller software), where the operator can choose which task should be executed. Examples of tasks that were included in the project are autonomous camera positioning, minimum-time patrolling, perimeter surveillance and, presented here, searching and securing. The operator can also adjust the map of the area and set parameters for the robots. During execution, the positions of the robots are displayed on the map and the image streams from each camera are displayed on a separate screen. The user could manually search for intruders, record the streams for post-processing or, not included in the scope of this project, apply image processing for automatic detection.

Fig. 12 shows the result of one run of the demonstration, using the simultaneous search strategy. The total search time was 3 minutes and 36 seconds. When the depth first search strategy was used on the same scenario, only two robots were needed and the total search time was 5 minutes and 4 seconds.

The search times were relatively long for both cases since the robots were limited to a maximum speed of 1 m/s and the on-board cameras rotated with 1 rad/s. This was chosen so the camera streams were easier to watch for the users. Nevertheless, this illustrates the opportunity for the operator to use more robots if a quicker execution is necessary, or to accept a longer search time if the number of robots is limited.

Fig. 10. One of the robots at the demonstration site. The tents are used as obstacles.

Fig. 11. The AURES operator console. The operator can choose which task should be launched. It is also possible to change scenario parameters such as the speed of the robots and coordinates of the map. The map shows the location of the robots.

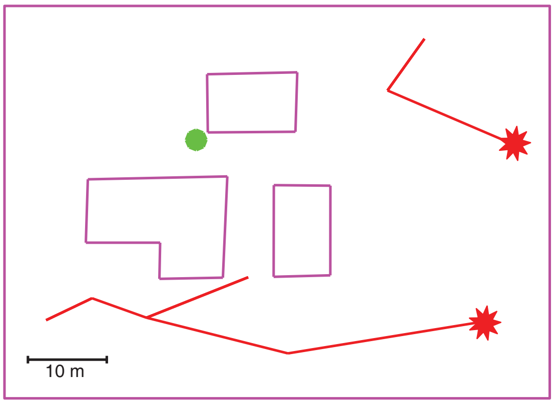

Fig. 12. Three robots searching and securing the demonstration area, using the simultaneous search strategy. One blocker is depicted as a circle and two searchers are shown as stars. Their trajectories are drawn as thick line segments.

VII. Conclusions

We have described a demonstration system of multi-agent robots, capable of solving the search and secure problem. The system abstracts the geometric map into a traversability graph on which it places stationary blockers to remove possible loops. The resulting tree can be searched with two different strategies, offering fast execution with many searchers or a slower alternative that uses fewer robots. Both strategies are proven correct.

As demonstrated, we have successfully integrated several robots and their sensors in a remotely operated system that can solve search and secure tasks using the proposed algorithm. The demonstration and simulations also show that the depth first strategy has the potential to reduce the number of robots used, compared to the more direct simultaneous strategy.

Our approach of static blockers and moving searchers does not guarantee using the minimum number of robots, but as previously shown [10], searching an arbitrary graph is an NP-hard problem. We believe that the presented strategies offer a system designer two strategies that span a wide range of needs and are readily implementable, even in very resource-constrained systems.

An interesting direction of future research would be to explore how the placement of the blockers affects the properties of the resulting tree graph. This would allow to create trees that require fewer searchers and to make a trade-off between the number of blockers and searchers. Similarly, refined procedures for merging regions could potentially reduce the number of robots and the search time.

Acknowledgment

The authors would like to thank the AURES project members from Saab Aerotech, FOI and Rotundus for their support with the experimental platform.

References

[1] T. Shermer, "Recent results in art galleries," in *Proceedings of the IEEE*, vol. 80, no. 9, 1992.

[2] A. Ganguli, J. Cortes, and F. Bullo, "Distributed deployment of asynchronous guards in art galleries," in *Proceedings of American Control Conference*, 2006.

[3] P. Brass, A. Gasparri, F. Cabrera-Mora, and J. Xiao, "Multi-robot tree and graph exploration," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2009.

[4] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 5, 1992.

[5] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," in *International Journal of Robotics Research*, 2004.

[6] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion in a polygonal environment," *IEEE Transactions on Robotics*, vol. 21, no. 5, Oct. 2005.

[7] A. Kolling and S. Carpin, "Probabilistic graph-clear," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2009.

[8] T. D. Parsons, *Theory and Applications of Graphs*, Y. Alani and D. R. Lick, Eds. Springer Berlin / Heidelberg, 1976, vol. Volume 642/1978.

[9] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, "The complexity of searching a graph," *Journal of the Association for Computing Machinery*, vol. 35, 1988.

[10] A. Kolling and S. Carpin, "The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[11] ——, "Multi-robot surveillance: An improved algorithm for the graph-clear problem," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2008.

[12] U. Nilsson, P. Ögren, and J. Thunberg, "Optimal positioning of surveillance UGVs," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[13] 2009.07.10. [Online]. Available: http://www.rotundus.se/