Guaranteed Search With Large Teams of Unmanned Aerial Vehicles

A. Kleiner* A. Kolling*

***Abstract*— We consider the problem of computing trajectories for a team of of coordinated unmanned aerial vehicles (UAVs) in large and complex 2D and 2.5D environments to guarantee the detection of any evading target. Our approach is based on the coordination of 2D sweep lines that move through the environment to clear it from all contamination, representing the possibility of a target being located in an area, and thereby detecting all targets. The trajectories of the UAVs are computed from the motion of these sweep lines. Low cost coordination strategies of the UAV sweep lines are computed in 2D and simply-connected polygonal environments and then converted to strategies capable of clearing multiply-connected 2.5D environments. We present simulation experiments with maps of real and artificial environments and demonstrate the execution of strategies with simulated quadrotors using the Robot Operating System (ROS) framework. The algorithms used for the experiments are made available on a public repository.**

I. Introduction

In many real-world situations it is necessary to locate moving targets within a larger perimeter, such as injured victims or smart intruders. In some applications such as surveillance it is crucial to also guarantee the detection of all targets, even when they move evasively, at unbounded speeds, omniscient. Unmanned aerial vehicles (UAVs) are offering a great deal in mobility to observe a larger terrain at comparably low costs and are ideal to solve such guaranteed search problems. In order to scale to very large environments the proper coordination of many UAVs becomes paramount, especially when detections guarantees are required. In this paper we present a solution to the problem of coordinating a guaranteed search with a team of UAVs searching for ground targets in environments represented by elevation maps. This is achieved by coordinating the motion of all UAVs through the environment with a line-based abstraction which aggregates multiple UAVs on a line spanned between obstacles. The environment is then cleared by moving these lines of UAVs in a synchronized manner. The goal is to minimize the number of UAVs needed for clearing the entire environment, to compute and execute their trajectories. To compute these low cost trajectories we use an algorithm that computes the best possible motion through a simply-connected polygonal environment, then adapt the resulting strategies to multiply-connected and 2.5D environments, and finally convert them to coordinated trajectories.

Figure 1 depicts an elevation map generated at the campus of the University of Freiburg and a snapshot during coordinated search by UAVs.

* Computer Science Department, Linköping University, 58738 Linköping, Sweden

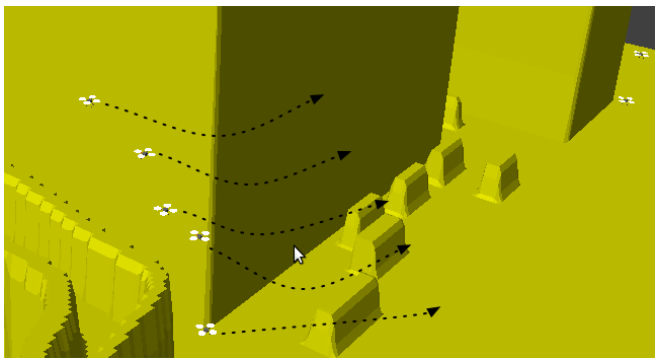


Fig. 1. Motivating picture: Trajectories of a coordinated guaranteed search for evaders by a team of UAVs.

The remainder of this paper is organized as follows. In Section II related work is discussed and in Section III we provide a formal description of the problem. Our approach is described in Section IV and Section V. In Section VI results from experiments are presented and we finally conclude with Section VII.

II. Related Work

In contrast to the presented work, several approaches have been introduced that are particularly tailored for non-moving targets [1] or targets moving at bounded speeds [2], [3]. Sujit *et. al* considered search and surveillance tasks based on uncertainty maps of an unknown region [1]. They used the k-shortest path algorithm for maximizing the effectiveness of the search when searching through uncertain regions while respecting several constraints such as flight time. More closely related to our work, Durham *et al.* presented a distributed algorithm guaranteeing complete coverage of the frontier between cleared and contaminated areas during expansion [2]. Their algorithm can be applied for multiply-connected planar environments which may be non-polygonal, but no coordination that guarantees the minimizing of the number of robots is computed. Shim *et al.* presented a nonlinear model predictive control (NMPC) for multiple autonomous helicopters [4]. More precisely, their presented approach combines the stabilization of vehicle dynamics and decentralized trajectory generation with optimizing a potential function that reflects possibly moving obstacles or other vehicles. Vidal *et al.* considered the problem in a probabilistic game theoretic framework and introduced two computationally feasible greedy pursuit policies, which are local-max and global-max [3].

Another closely related paper by Efrat *et al.* [5] considers an approach where multiple UAVs, each with an unlimited

range sensor, are arranged in a single movable polygonal chain operating in a simply connected environment. Their algorithm for computing motion strategies runs in $O(n^3)$, an improved version of the algorithms runs in $O(n^2)$ time [6]. Our work in contrast considers UAVs with a limited sensing range and multiple simultaneous lines in the environment. We further allow the number of lines to vary as the search mission unfolds.

Another interesting approach that also assumes limited sensing range is presented in [7] where the authors present a system where multiple UAVs trap faster intruders, once they detect them, by forming a surrounding chain. An approach using sweep lines for clearing an environments by coordinating lines on a Voronoi diagram has been presented by Kolling *et al.* [8]. The Voronoi Diagram of the environment induces a surveillance graph for which the weights, i.e. costs in terms of UAVs, of vertices are given by the distances to the three closest and equidistant obstacles. Lines then either wait on a Voronoi edge or move towards a new obstacle that is associated to a Voronoi vertex. This approach was shown in [9] to result in suboptimal coordination of lines for some problem instances, which motivated our generalized approach presented here. To overcome this it was suggested in [9] to apply the approach from [10], which dealt with computing optimal guaranteed search strategies on trees. This has been achieved in [11] and we shall use the same approach from [10] to enable the coordination of UAVs for our system. In contrast to prior work in [8] we consider more possibilities for the expansion of the lines and provide a more flexible data structure for the analysis of the environment leading to an improved algorithm. In addition we consider the adaptation of line-based strategies to 2.5D environments with complex visibility constraints and, in addition to work in [11] compute UAV trajectories, execute these with realistically simulated UAVs and provide further experiments and insights into the properties of such type of systems and algorithms.

III. Problem Formulation

We consider the problem of *guaranteed detection* by a team of UAVs searching for ground targets located in an environment. With guaranteed detection we refer to the general definition of guaranteed search, common in the pursuit-evasion literature, that guarantees the detection of any omniscient target moving at unbounded speed in a 2.5D environment $\mathcal{E}$. We represent $\mathcal{E}$ with a 2.5D map given by a height function $h : H \to \mathbb{R}^+$. The domain H is continuous and $H \subset \mathbb{R}^2$ which can be approximated by a 2D grid map that contains in each discrete grid cell the corresponding height value. We assume that $\mathcal{E} \subset H$ is connected and every point in $\mathcal{E}$ is reachable by the ground target.

The problem is to move a team of UAVs equipped with a target detection sensor through $\mathcal{E}$ to detect all targets that are potentially located therein. Since we assume that targets are omniscient and move at unbounded speeds within $\mathcal{E}$ we can represent the possibility of a target with the concept of contamination, which is ubiquitous in the guaranteed search and pursuit-evasion literature. Contamination spreads immediately from every contaminated point to all other points in $\mathcal{E}$ to which a path exists that is not covered by a sensor. This models the assumption that targets can immediately take advantage of any gap in the sensor coverage. Points that are covered by sensors are cleared and remain cleared if at no point an uncovered path to a contaminated point exists. In other words, the boundary of the cleared region must be continually guarded to prevent recontamination.

Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be the set of n UAVs navigating simultaneously in $\mathcal{E}$. The problem we consider is to find the paths and the minimum number of UAVs n, so that there exist for each UAV $R_i \in \mathcal{R}$ a path $\pi_i : [0, T_i] \to \mathcal{E}$ such that at $\max(T_i)$ an initially contaminated environment $\mathcal{E}$ is *cleared*. Note that T_i denotes the time needed by UAV R_i to travel path π_i. Write $\mathcal{C}$ for all cleared points and $\mathcal{E} \setminus \mathcal{C}$ for all contaminated points at time t.

IV. Global Clearing Strategy

In order to coordinate the motion of all UAVs through the environment we choose a line-based abstraction which coordinates the motion of lines. This approach was previously introduced in [9] and we will present a shortened and less formal introduction here. It assumes, however, that the environment is simply-connected and polygon. In Section V we describe how extend this to multiply-connected environments and how to polygonize the elevation map introduced in Section III.

Multiple UAVs are aggregated into a line spanned between obstacles. The environment is then cleared by moving these lines of UAVs through it. Every line that can be set up between any two obstacles has an associated cost that represents the number of UAVs that are needed to cover the area of the line. The goal is to coordinate the motion of multiple such lines moving through the environment while minimizing the overall cost at any point in time. The movement of these lines is required to clear the entire environment and every line separates the contaminated from the cleared area. We assume that the environment is given as simply-connected and simple polygon, $P = \{v_1, \ldots, v_n\}$, with n vertices and edges, written $e_i = [v_i, v_{i+1}]$, $i = 1, \ldots, n$. Throughout this section the indices of the obstacles, i.e. the polygon edges, are assumed to be circular, i.e. we identify $i + n$ with i.

Initially, all of P is contaminated and one sweep line starts at a point on an obstacle boundary and starts to clear contamination, as seen in Fig. 2. Every line either continues to sweep the environment to expand the cleared area or it stops and blocks contamination from entering the already cleared parts. Now, every intersection between $\delta\mathcal{C}$, the boundary of the cleared part, with the interior of P has to have a line to block contamination. Otherwise $\mathcal{C}$ will be recontaminated. The problem, defined more formally in [9], that we need solved is to find lines and move them through the environment to clear it at the lowest cost in terms of UAVs without allowing the recontamination of any point in $\mathcal{C}$. As in [9], we write $l_{i,j}$ for a sweep line between obstacles e_i and e_j and $c(l_{i,j})$ for its costs, which represents

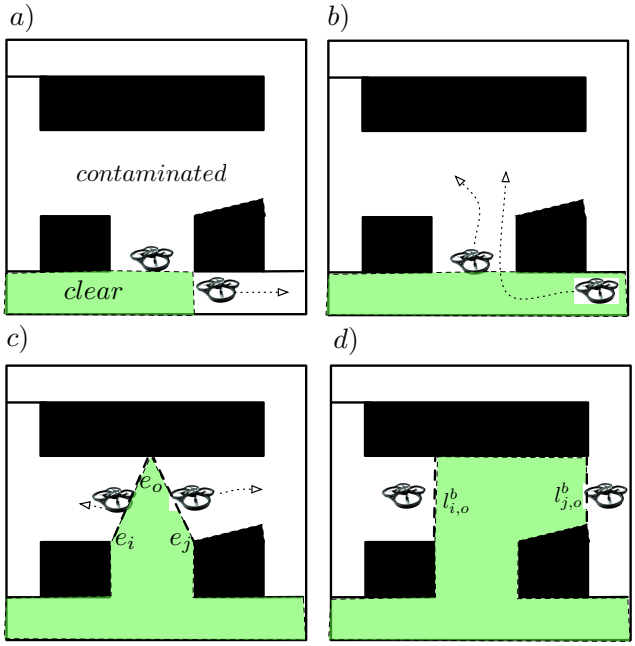


Fig. 2. An illustration of a sweep through a simple environment. UAVs separate clear and contaminated areas in a), one UAV proceeds to move a line in b), two UAVs split on an obstacle segment e_o towards blocking lines $l^b_{i,o}$, on the left, and $l^b_{j,o}$ on the right of d).

the number of UAVs needed to cover the line. For our experiments we set $c(l_{i,j}) = \frac{|l_{i,j}|}{2 \cdot s_{range}}$, where s_{range} is the sensor range of the UAVs, but other alternative cost functions could also be considered. In addition, we assume that two basic function are available. One that computes the shortest line between two edges e_i, e_j in P, called $shortest(e_i, e_j)$, and one that computes the shortest line between a point $p \in P$ and an edge e_i in P, called $shortest(e_i, p)$. To compute these shortest lines one can, for example, use the methods from [12] Chapter 6.2.4 for planning shortest paths between two points in polygons. To describe $\mathcal{C}$ we call an edge e_i of P cleared if $\mathcal{C} \cap e_i \neq \emptyset$, i.e. if any point on e_i is cleared. As a corollary every cleared e_i that has an adjacent contaminated obstacle must have a sweep line starting on it to block and prevent recontamination. For two segments e_i and e_j we write $l^b(i,j) = shortest(e_i, e_j)$ for the lowest cost line that blocks contamination between them.

Now, to clear the environment we have to move some of the sweep lines on the boundary of $\mathcal{C}$ forward and eventually clear more obstacles on the boundary. Let e_o be the first obstacle that is cleared in this process and let $l^b_{i,j}$ be the line moved forward to clear e_o. The lowest possible cost of this process can be computed by considering the cost of splitting the line $l^b_{i,j}$ onto e_o as seen in Fig. 2. This lowest cost is given by:

$$c(o|i,j) := min_{p \in e_o}\{|shortest(e_i, p)| + |shortest(e_j, p)|\} \quad (1)$$

In cases where the shortest sweep line is a simple straight line this equation becomes $min_{p \in e_o} = |p_i - p| + |p_j - p|$ with p_i, p_j, p subject to the constraints given by their segments. Even though these constraints are linear cost equation is non-linear and without an easy analytical solution. For our purposes we assume an oracle that determines p.

A line splitting on a new obstacle leads to new blocking lines between e_o and e_i and e_o and e_j. The cost to maintain these lines can be reduced by moving them to a locally shorter line, namely $l^b_{i,o}$ and $l^b_{j,o}$, as shown in Fig. 2. Notice that if e_o is adjacent to either e_j, e_i, or both, then the length of $l^b_{i,o}$, $l^b_{j,o}$, or both is zero. Now that we defined these simple local operations, to find a low cost blocking line or a low cost split, we describe the more difficult problem of coordinating many such operations in a larger environment. This problem reduces to the problem of determining which line should split onto which obstacles and in which sequence. Every choice of split further determines the possible choices and costs for future splits which can be described by a sequence containing all obstacle indices $o_1, \ldots, o_n$. The first splits on o_1 and o_2 simply set up the first line, possibly a zero length line if o_1 and o_2 are adjacent. The s-th split, $s > 2$, is then given by o_s. The line that is split on o_s is the blocking line between two indices from $o_1, \ldots, o_{s-1}$. These indices are next smaller and next larger index to o_s in $o_1, \ldots, o_{s-1}$, i.e. $o_l = \text{argmax}_{o \in \{o_1, \ldots, o_{s-1}\}}\{o < o_s\}$ and $o_r = \text{argmin}_{o \in \{o_1, \ldots, o_{s-1}\}}\{o > o_s\}$. So the line $l^b(o_l, o_r)$ is split on o_s while all other lines already present in P wait. The overall cost can be calculated by summing up all other blocking lines and the cost of the split onto o_s. In the next section we show how to compute obstacle sequence and choose the splits so that the overall cost is minimized.

A. Choice Sets

The application of the approach from [10] to the concept of a choice set, introduced formally in [9], requires the identification of a recursive relationship between subsequences of the above mentioned obstacle sequences. In this sections we briefly describe the concept of a choice set, and the relationship between obstacle sequences that allow an efficient recursive construction. More details on this are found in [11].

Formally a choice set is defined as $T^i_k := \{i, i+1, \ldots, i+k-1\}$ which corresponds to the contaminated area circumscribed by $e_i, \ldots, e_{i+k-1}$. This contaminated area guarded by a blocking sweep line going from edge e_{i-1} to edge e_{i+k}, written $l^b(T^i_k) := l^b_{i-1,i+k}$. As a shorthand write $b(T^i_k) := c(l^b(T^i_k))$ and call it the blocking cost for the contaminated area of choice set T^i_k.

The area corresponding to T^i_k can be cleared further by choosing to split on an index $o \in T^i_k$. Write $c(o|T^i_k) := c(o|i-1, i+k)$ using Eq. 1 for the cost of this split which separates the contaminated area of T^i_k into two disconnected contaminated areas. Each of these is again represented by a choice set, namely T^i_{o-i} to the left of o and $T^{o+1}_{i+k-o-1}$ to the right of o, written $T^l := T^i_{o-i}$ and $T^r := T^{o+1}_{i+k-o-1}$. This relationship between T^i_k, T^r, and T^l is the key to compute the cost of obstacle sequences recursively. It exploits the fact that $T^r, T^l \subset T^i_k$ and that their low cost obstacle sequences are subsequences for low cost obstacles sequences of T^i_k.

Let $O = \{o_1, \ldots, o_k\}$ be an obstacle sequence using all indices from T_k^i, i.e. $o_s \in T_k^i$ and $o_s \neq o_{s'} \iff s \neq s'$. Write $\bar{O}$ for all possible such obstacle sequences. Let $\mu_s(O)$ be the cost of all sweep lines located in the area for T_i^k when splitting on o_s and let $b_s(O)$ be the cost of all sweep lines after splitting on o_s, i.e. the blocking cost after step s. Let $T_{k_s}^{i_s}$ be the choice set so that o_s splits the line l_{i_s-1,i_s+k_s} onto e_{o_s}. Note that $i_s = \text{argmax}_{o \in \{o_1,\ldots,o_{s-1}\}}\{o < o_s\}$ and $k_s = \text{argmin}_{o \in \{o_1,\ldots,o_{s-1}\}}\{o > o_s\} - i_s - 1$. Now, $\mu_s(O)$ and $b_s(O)$ are given by:

$$\mu_s(O) = b_{s-1}(O) + c(o_s|T_{k_s}^{i_s}) - b(T_{k_s}^{i_s}) \quad (2)$$

$$b_s(O) = b_{s-1}(O) - b(T_{k_s}^{i_s}) + b(T^l) + b(T^r) \quad (3)$$

with $b_0(O) := c(l_{i-1,i+k})$, i.e. the blocking line for T_k^i, and $T^l = T_{o_s-i_s}^{i_s}$ and $T^r = T_{i_s+k_s-o_s-1}^{o_s+1}$, as before. The tradeoff between the cost clearing everything until step s, given by $\max s' \leq s\{\mu_{s'}(O)\}$, and the resulting blocking cost $b_s(O)$ is given by:

$$\rho_s(O) = \max_{o \leq s}\{\mu_o(O)\} - b_{s-1}(O). \quad (4)$$

This tradeoff $\rho_s(O)$, similar to the ρ values defined in [10] for graphs, determine how to order the subsequences when combining them recursively.

The possible index sequences for T_k^i that start with the choice of o can now be constructed by considering combinations from a sequence from T^l and from T^r written o_s^l and o_s^r respectively. Fortunately, only the combination we obtain by ordering all o_s^l and o_s^r by ρ_s^l and ρ_s^r has to be considered, i.e. the sequence $O = \{o_1 = o, o_2, o_3, \ldots, o_k\}$ which satisfies: if $o_s = o_{s'}^l$, then $\rho_{s'}^l < \rho_{s''}^r$ for all $o_{s''}^r = o_{s^*}$ with $s^* > s$ and vice versa for r. In colloquial terms, ordering all obstacles indices from the left and right according to their tradeoff value ρ determines the lowest combination. Going through the equations that determine the cost of O from the costs from the costs $\mu_s^l(O^l)$, $\mu_s^l(O^r)$, $b_s^l(O^l)$, and $b_s^l(O^r)$ confirms this. The detailed algorithm and further theoretical insights into the ordering by ρ are found in [10] and [11]. The algorithm builds $\tilde{O}(T_k^i) \subset \bar{O}(T_k^i)$ which only contains obstacle sequences constructed that are relevant for subsequent constructions. Note that one would still have to consider all possible combinations from sequences from $\tilde{O}(T^l)$ and $\tilde{O}(T^r)$. In order to reduce the number of sequences in $\tilde{O}(T_k^i)$ used for later constructions for larger choice sets we use the pruning criteria from [11] using a domination criteria that prunes all sequences that are dominated by another sequence, i.e. they never have a lower blocking μ cost that can be reached with at a lower overall cost. In other words, whenever a dominated sequence reaches a low blocking cost, the dominating sequences already did so but with a lower or equal overall cost.

At this point, it is still an open question whether the domination criteria suffices for reducing the growth of $\tilde{O}(T_k^i)$ for larger choice sets. We provide an indication of the effect of this pruning and an alternative pruning in Section VI. Once all obstacle sequences in all choice sets are computed the best sequence can be found by choosing the best sequence from all $\tilde{O}(T_{n-1}^i), i = 1, \ldots, n$.

V. Towards a Real System: From 2D to 2.5D

The above gives us a sequence of low cost lines that move between obstacles in a 2D environment. In this section we describe how to compute a local cover for these sweep lines in the 2.5D environment. This method considers visibility constraints as they arise from the underlying 2.5D representation. In addition, we briefly describe how we convert the 2.5D representation to a simply-connected polygonal environment.

For every line that needs to be covered we compute UAV a set of locations that guarantee that any target crossing the line will be detected by at least one of the UAVs. This is achieved by computing so called *detection sets* also used in [13], which are areas around a location in which targets are detectable from the respective location. Let $D(p) \subset \mathcal{E}$, the detection set of a point $p \in \mathcal{E}$, be the set of all points in $\mathcal{E}$ on which a target is detectable by a UAV located at p. In general, $D(p)$ depends on the sensor model, height of the sensor h_r relative to $h(p)$ and height of targets h_t. We consider a limited range three-dimensional and omni-directional sensor. A target at $p' \in \mathcal{E}$ is detectable by a UAV at p if at least one point on the line segment from $\{p', h(p')\}$ to $\{p', h(p') + h_t\}$ embedded in $\mathbb{R}^+$ is visible from $\{p, h(p) + h_r\}$ at distance s_r. Notice while $\mathcal{E}$ is considered as discretized into grid cells, height values and thus the z-component of the line segments are in $\mathbb{R}^+$. Here h_t can be understood as the minimum height of any target for which we seek to guarantee a detection. The computation is carried out by generating rays with the Bresenham algorithm [14] More details on the computation of detection sets can be found in [13].

With the ability to compute detection sets in the environment one can determine the minimal set of observation points needed to be covered simultaneously in order to detect any target within a specific region. However, since this problem, which is generally known as the *set cover problem*, is NP-complete [15] we are utilizing a fast heuristic solution based on a greedy approach.

Given a line with endpoints $[p_l, p_r]$ we compute the detection set $D(p)$ for all points $p \in [p_l, p_l + s_r \cdot (p_r - p_l)]$ and chose the point p with the longest contiguous segment in $D(p) \cap [p_l, p_r]$ that contains p_l. Then the new left endpoint p_l is set to the end of this contiguous segment and the procedure is repeated until all of $[p_l, p_r]$ is covered. Using this procedure we convert $L(t)$ to a set of locations $P(t) = \{p_1, \ldots, p_{m(t)}\}$, $p_1 \in H$ for every $t = 1, \ldots, T$. These locations are effectively the 3D poses that have to be occupied by a UAV at time t.

In principle, these methods can easily be replaced with custom methods that are adapted to the particular sensor or robot used, e.g. a forward facing camera will require the placement of UAVs behind the line in order to cover it, a considerably more difficult problem.

Finally, we use this discretized set of poses to compute trajectories for the UAVs. Here one still has to determine

which UAV moves from its location at time t to one of the new required locations at time $t+1$. To assign UAVs from their current location to a new location we use the Hungarian method [16] with the travel time to the new locations as the costs. This assigns UAVs to locations by minimizing the average cost, which can be distance or time. Any motion model or planner can be used to determine the motion cost.

A. Polygonization

In addition to computing the 2.5D cover of lines we also have to extract a polygonal representation of the height map to apply the algorithm from Section IV-A. For this every position in the domain H of h that is traversable by a ground target is marked as free space, using the same classification algorithm as in [13]. All free space reachable from a starting position then provides the connected free space that forms our search environment. This connected set $\mathcal{E} \subset H$ environment is then polygonized by computing an α shape using the CGAL library [17]. These shapes are frequently used to reconstruct the shape of a dense set of points. From the α shape we construct a polygon boundary and interior polygons. On these we apply the Ramer-Douglas-Peucker line-simplification algorithm [18] to strip redundant vertices and get a polygons with fewer edges. The loss of precision is given by a parameter ε which determines the degree of simplification.

B. Multiply-connected Environments

In order to apply the choice set approach to compute optimal obstacle sequences described in Section IV-A the polygon needs to be simply-connected. To achieve this the interior polygons are connected to the outer polygon boundary by connecting the closest polygon to the boundary first and then each subsequently closer polygon to the new combined boundary. This is a greedy approach that creates short connections with newly introduced artificial obstacle segments between interior polygons. This process is similar to the computation of graph searching strategies on graphs with cycles by removing all edges that cause cycles, computing a strategy on the tree and then adapting that strategy back to the graph. This approach has been used and discussed extensively in [19], [13],[11], and [9]. The strategies computed for the simply-connected environment can be adapted to the original and multiply-connected environment in a straightforward manner. Whenever one side of an artificial obstacle segment is cleared and the other contaminated, then that segment has to be covered by UAVs. The increase in cost due to these additionally covered lines can vary between environments. There are no significant theoretical insights known with regard to bounds on this additional costs, but some experimental investigations have bene carried out in [11], [9], and [13].

VI. Experimental Results and Discussion

The source code and maps used for the experiments in this paper are published online at http://code.google.com/p/guaranteed-search/ under a GNU GPLv3 license. We implemented all algorithms described above and used the popular Robot Operating System (ROS) [20] to integrate the various components. UAVs are simulated as quadrotors in Gazebo, as seen in Fig. 1 and are controlled via the move_base node available in ROS to measure the time to execute a set of guaranteed search trajectories. The primary concerns from a practical perspective that we addressed for our systems are: 1) implications of and adaptation to $2.5D$ environments; 2) feasibility and time of execution for search trajectories in realistic maps; We addressed these concerns using the maps shown in Fig 3.

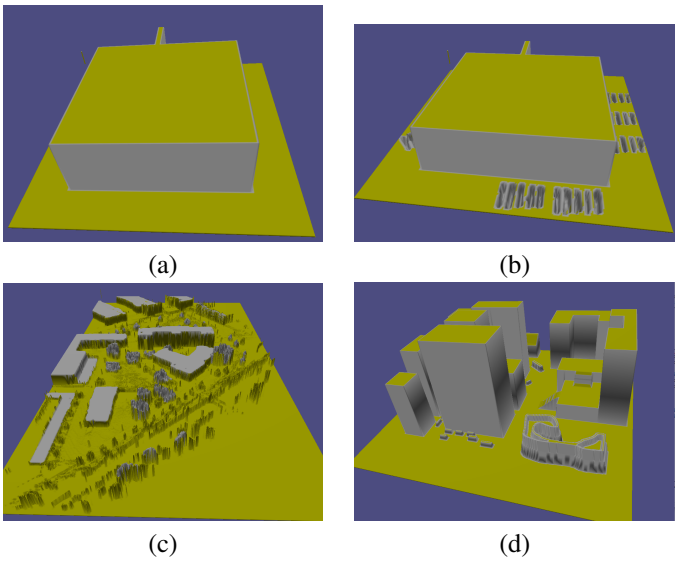


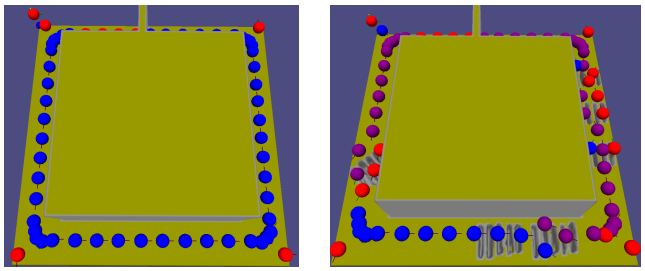
Fig. 3. The maps used in the simulation experiments: (a) **Simple Corridor** - elevation map of a simply-connected corridor around a square obstacle; (b) **Simple Corridor with grooves** - same as (a) but with small grooves that can hide targets; (c) **Freiburg** - elevation map constructed for the campus at the University of Freiburg; (d) **Town** - artificial elevation map of a simple town.

A. 2.5D visibility

The impact that 2.5D visibility can have on search trajectories is seen by comparing the trajectories in the two corridor environments seen in Fig. 4. In Fig. 4 a) two UAVs suffice to clear the simply connected environment with the second UAV being needed only in the corners. An additional UAV is needed when grooves are added in Fig. 4 b) so that they can be positioned to sense targets hidden in these grooves.

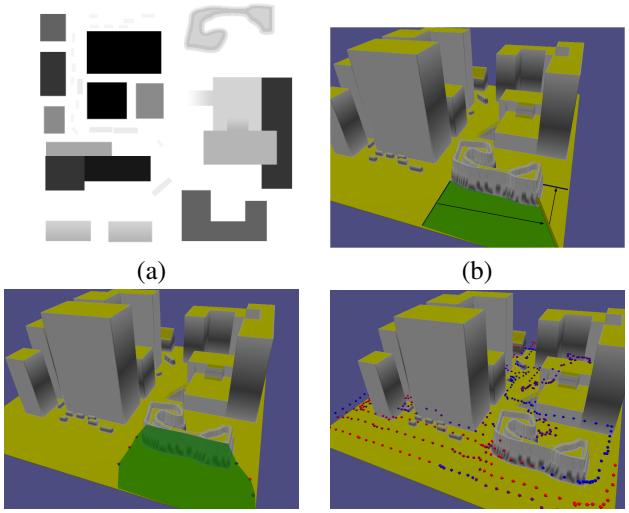
B. Executing Searches in Realistic Maps

The results from experiments in realistic scenarios are summarized in Table I. Obviously, the results document the increasing need for UAVs when the complexity of the environment increases. It can also be clearly seen from the waiting time of each UAV (Tot. Wait) that the load for searching parts of the environments is almost equally balanced. While there is some considerable variance for smaller scenarios such as the *Simple Corridor*, waiting times on larger maps such as *Freiburg* and *Town* are within similar range.



(a) (b)

Fig. 4. (a) Search trajectories in a simple corridor with UAVs drawn in different colors; (b) search trajectories in a simple corridor with small grooves that can hide targets.



(a) (b) (c) (d)

Fig. 5. (a) An elevation map of an artificial town; (b) A 3d display of the map from a) with 2d sweep lines illustrated; (c) the corresponding positions of UAVs for the lines seen in (b); (d) a discretized snapshot of the positions of all UAVs.

Map: **Si.Cor.** Time $101[s]$	**Total Dist. [m]**	**Tot. Wait [s]**	**A.Vel. [m/s]**
UAV-1	40.6 ± 0.1	9.5 ± 1.3	0.4 ± 0.0
UAV-2	34.9 ± 0.1	25.4 ± 0.1	0.5 ± 0.0
Map: **Si.Cor.Gro.** Time $109[s]$			
UAV-1	38.3 ± 0.1	14.9 ± 1.0	0.4 ± 0.0
UAV-2	35.2 ± 0.8	35.8 ± 1.7	0.5 ± 0.0
UAV-3	31.1 ± 0.1	32.3 ± 1.7	0.4 ± 0.0
Map: **Town** Time $3228[s]$			
UAV-1	496.2 ± 5.9	2122.8 ± 92.4	0.5 ± 0.0
UAV-2	268.3 ± 35.2	2713.9 ± 99.9	0.5 ± 0.1
UAV-3	330.9 ± 23.3	2518.6 ± 60.1	0.5 ± 0.1
UAV-4	374.3 ± 5.5	2490.5 ± 97.9	0.5 ± 0.1
UAV-5	563.1 ± 1.0	2033.2 ± 92.7	0.5 ± 0.0
UAV-6	542.4 ± 13.9	2146.8 ± 47.9	0.5 ± 0.0
UAV-7	562.3 ± 4.7	2147.7 ± 72.9	0.5 ± 0.0
Map: **Freiburg** Time $6896[s]$			
UAV-1	511.2 ± 54.5	5782.9 ± 117.7	0.5 ± 0.1
UAV-2	500.9 ± 40.9	5870.5 ± 108.9	0.5 ± 0.1
UAV-3	447.5 ± 35.9	6072.1 ± 161.7	0.5 ± 0.0
UAV-4	469.2 ± 38.2	5955.8 ± 70.37	0.5 ± 0.1
UAV-5	473.3 ± 2.9	5904.6 ± 195.1	0.5 ± 0.1
UAV-6	1383.9 ± 6.9	4070.8 ± 142.7	0.5 ± 0.0
UAV-7	677.2 ± 68.6	5577.6 ± 165.6	0.5 ± 0.0
UAV-8	1157.7 ± 19.4	4441.9 ± 173.9	0.4 ± 0.0
UAV-9	1257.6 ± 26.6	4213.9 ± 55.2	0.4 ± 0.0
UAV-10	1136.2 ± 24.2	4459.8 ± 84.6	0.4 ± 0.0
UAV-11	1103.1 ± 26.4	4578.6 ± 137.6	0.4 ± 0.0
UAV-12	742.6 ± 19.3	5382.8 ± 85.9	0.4 ± 0.0

TABLE I

STATISTICS RECORDED DURING REALISTIC EXPERIMENTS: THE TOTAL DISTANCE TRAVELLED BY EACH UAV *Total Dist.*, THE AVERAGE TOTAL WAITING TIME OF EACH UAV *Tot. Wait Time*, AND THE AVERAGE VELOCITY *Avg. Vel.*.

VII. CONCLUSION

The goal of this paper is to enable the building of a system of UAVs for guaranteed search in complex and large environments. To this end we presented an improved algorithm to coordinate the motion of sweep lines to clear a 2D environment from a polygonized elevation map. We conjecture that this algorithm runs with polynomial time complexity and is optimal for simply-connected environments, but the proof of this claim remains subject of further work. Furthermore, we provided methods to adapt the resulting strategies to multiply-connected and 2.5D environments.

In addition to the computation of trajectories in 2.5D from the motion of lines the consideration of motion constraints of UAVs is a natural next step towards building a real system, especially for fixed wing UAVs. Also, the fast and parallel execution of a sweep remains an open problem. In [13] and [21] the issue of minimizing the time to execute graph-based strategies was considered, but no such work for sweep schedules exists as of now. Especially with quad-rotors and their energy constraints such a fast execution will be crucial.

We have demonstrated the feasibility and scalability of our approach and successfully computed coordination strategies for complex and large environments. The next step is to apply this to a real system and demonstrate powerful search capabilities with a real team of UAVs.

Extensions to 3D environments can also be envisioned, since the primary role of our coordination algorithm is to compute sequences of splits and blocks. In principle, these splits and blocks can also be 2D planes in a 3D environment instead of lines. Albeit obtaining the costs and computing multiple possible such planes will pose a significant challenge.

VIII. ACKNOWLEDGMENTS

This work is partially supported by grants from the Swedish Foundation for Strategic Research and the Excellence Center at Linköping and Lund in Information Technology (ELLIIT), the Research Council (VR) Linnaeus Center CADICS, the Swedish Foundation for Strategic Research CUAS Project, and the EU FP7 project SHERPA, grand agreement 600958.

REFERENCES

[1] P. Sujit and D. Ghose, "Search using multiple uavs with flight time constraints," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 40, no. 2, pp. 491–509, 2004.

[2] J. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Autonomous Robots*, pp. 1–15, 2011.
[3] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 5, pp. 662–669, 2002.
[4] D. Shim, H. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *Decision and control, 2003. Proceedings. 42nd IEEE conference on*, vol. 4. IEEE, 2003, pp. 3621–3626.
[5] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali, "Sweeping simple polygons with a chain of guards," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2000, pp. 927–936.
[6] X. Tan, "Sweeping simple polygons with the minimum number of chain guards," *Information processing letters*, vol. 102, no. 2-3, pp. 66–71, 2007.
[7] S. Bopardikar, F. Bullo, and J. P. Hespanha, "On discrete-time pursuit-evasion games with sensing limitations," *IEEE Transactions on Robotics*, 2008, to appear.
[8] A. Kolling and S. Carpin, "Surveillance strategies for target detection with sweep lines," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 5821–5827.
[9] A. Kolling, "Multi-robot pursuit-evasion," Ph.D. dissertation, University of California, Merced, December 2009.
[10] A. Kolling and S. Carpin, "Pursuit-evasion on trees by robot teams," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 32–47, 2010.
[11] A. Kolling and A. Kleiner, "Multi-uav motion planning for guaranteed search," in *Proceedings of the Twelth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2013, accepted for publication.
[12] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.
[13] A. Kleiner, A. Kolling, M. Lewis, and K. Sycara, "Hierarchical visibility for guaranteed search in large-scale outdoor terrain," *Autonomous Agents and Multi-Agent Systems*, pp. 1–36, 2011.
[14] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.
[15] R. Karp, "Reducibility among Combinatorial Problems," *Complexity of Computer Computations*, 1972.
[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.
[17] T. K. F. Da, "2D alpha shapes," in *CGAL User and Reference Manual*, 3rd ed., C. E. Board, Ed., 2008. [Online]. Available: http://www.cgal.org/
[18] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, vol. 1, no. 2, pp. 244–256, 1972.
[19] G. Hollinger, A. Kehagias, and S. Singh, "Gsst: anytime guaranteed search," *Autonomous Robots*, vol. 29, no. 1, pp. 99 – 118, July 2010.
[20] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.
[21] A. Kolling, A. Kleiner, M. Lewis, and K. Sycara, "Computing and executing strategies for moving target search," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 4246–4253.