Surveillance strategies for target detection with sweep lines

Andreas Kolling and Stefano Carpin

Abstract— **In this paper we present a method to extract *surveillance graphs* from occupancy grid maps. Surveillance graphs are part of the Graph-Clear framework and model the problem of detecting targets using multiple robots with limited range sensors. Robots can only execute basic actions called *sweep* and *block* on vertices and edges, respectively. Sweep detects targets in vertices and block prevents them from crossing edges. The extracted graphs accurately model the complexity of the planar environment to be searched, and are constructed as duals of the Voronoi Diagram. We give a geometric embedding for *blocking* and *sweeping* actions of the graph into the environment by directly associating them to *sweep lines* that robots cover with their sensors. This paper solves two open problems, namely the generation of surveillance graphs and the implementation of actions on a robot team. Sweep lines can then be directly translated into control inputs to the robot team. The new method is superior to previous heuristics for the extraction of graphs not only through its direct geometric relationship to the environment, but also due to its increased performance in direct experimental comparisons. Additionally, it provides a basis for possible theoretical results regarding the optimal coordination of multiple robots to detect targets in an arbitrary planar environment.**

I. Introduction

Our interest in target detection is with the coordination of large robot teams with limited range sensors, each of which is not sufficient to achieve any part of the task alone. We assume the target is a worst-case adversary. This means no knowledge about the target is available beforehand, but instead a coordination solution is needed that will work for any possible motion of any target. This gives a guaranteed lower bound for the detection performance even when target properties are unknown. To this end we introduced a graph-theoretical problem coined Graph-Clear [1], [2], [3]. Graph-Clear is defined on *surveillance graphs* which are somehow similar to topological maps of the environment. A surveillance graph relates the detection capabilities of the robots to the spatial structure of a closed bounded planar environment. Building blocks for the framework are the *sweep* and *block* actions which correspond to the execution of distributed routines on the robot team which respectively: 1) guarantee the detection of intruders within the region associated to a vertex; 2) guarantee that no intruder crosses an edge undetected. Actions have a cost which is the number of robots needed to execute them. Graph-Clear asks to find a sequence of sweep and block actions that can be executed with the least robots and which guarantees detection of any target possibly located in the graph. The possibility of a target being located in a vertex or edge is modelled by the concept of *contamination*, an idea frequently used in the pursuit-evasion literature on graphs. An optimal strategy clears an initially contaminated graph with the least number of robots. Practical applications of Graph-Clear hinge on the ability to extract surveillance graphs from arbitrary planar environments, as well as the ability to execute the resulting strategies with the robot team. A first heuristic attempt to extract surveillance graphs from occupancy grid maps was given in [2]. This manuscript continues along this line of work and provides a better solution for graph extraction as well as better implementations for actions. More precisely our main contributions are: 1) implementations for block and sweep actions using sweep lines between obstacles; 2) an algorithm to construct surveillance graphs from any grid map and compute surveillance strategies as a sequence of sweep lines that can be executed by robot teams; 3) experiments on grid maps and comparison of these strategies to our previous work [2]. Additionally, we offer pointers on theoretical ramifications of the sweep line approach.

A major related body of work in robotic surveillance is concerned with visibility-based pursuit-evasion, introduced in [4]. Therein the pursuit-evasion problem is directly embedded into a planar environment and robots are equipped with a number of unlimited range beams which detect targets that are visible by these beams. The crucial point is that beams have unlimited range. This is the major distinction to the Graph-Clear approach. Throughout the variety of contributions to this topic, environments considered in visibility-based pursuit-evasion range from simple polygons to arbitrary curved environments. The later type of environment is assumed in [5]. Therein a simple omnidirectional gap sensor can be used to detect all intruders, even under imperfect control and without localization capabilities. Many other variations of sensors have been investigated and a survey is beyond the scope of this paper. However, most interesting to note is one of the few theoretical relationships between graph-searching and visibility-based pursuit-evasion. In [6] it was shown that for any instance of edge-searching one can construct a corresponding visibility-based instance. This transports many theoretical properties such as the NP-completeness proof from [7] to visibility-based pursuit-evasion. The relationship remains single-sided and does little to improve practicality of edge-searching for robotic surveillance. The work presented in this paper goes into the direction of establishing a relationship between planar environments and Graph-Clear, although in the opposite direction, which is more important from a practical point of view. We show that for any environment we can construct

A. Kolling and S. Carpin are with the School of Engineering, University of California. Address: 5200 North Lake Rd, 95343, Merced, CA (USA). E-mail: {akolling,scarpin}@ucmerced.edu

an appropriate surveillance graph. This ability enhances the usefulness since graph strategies can now be translated to a coordination of movements for robots in planar environments. It also presents a starting point for defining a problem which shares the useful property with visibility-based pursuit-evasion that it is naturally embedded in the environment. The problem is coined Line-Clear and defines how lines between obstacles can be used to restrict target movement. The basic idea is related to [8] where a single line is spanned between the edges of a simple polygon, and robots cover it as it moves to clear the environment. In a way, we are extending this approach to consider any environment and multiple independently moving lines. Other line following approaches include the capturing of intruders with a moving chain of robots in open space [9]. The chain moves onto a target once a single robot detects it, guaranteeing capture of the target. Line following has the added benefit that a control algorithm can be executed in a distributed fashion ensuring the line is followed properly. This translates line movement to individual robot movement. Various distributed control algorithms achieving such and other similar tasks are presented in [10]. Although the Line-Clear idea offers interesting theoretical aspects to investigate, in this manuscript we are primarily concerned with practical ramifications to model the sensing capabilities of a robot team as lines between obstacles. Theoretical investigations and definitions making the Line-Clear approach formally precise are work in progress. For now we consider a sweep line as a line covered by the sensors of multiple robots which ensure that no intruder can pass through. As the line moves through the environment, robots continue to cover it with their sensors. A line spanned between two obstacles can approach a third obstacle with one of its mid points and then split into two. Minimum length lines correspond to minima on an edge of the Voronoi Diagram. At each vertex of the Voronoi Diagram we can split the line on the third defining site for the vertex. Then one has to choose with which line to continue first. This question is answered by converting the Voronoi Diagram into a surveillance graph. The paper is organized as follows. We will first recap an algorithm for Graph-Clear in section II, followed by details on the construction outlined above in section III. Section IV presents a practical algorithm that takes an occupancy grid map and computes a sequences of lines based on the constructions from section II and III. Some practical implementation details are offered in section IV. Experimental results on two large maps are presented in V. A comparison of the cost for executing the resulting strategy and line movements to previous heuristic strategies from [2] is also included therein. Finally, in section VI we conclude with a discussion including the generation of robot paths from a sequence of moving lines.

II. Graph-Clear

Graph-Clear is a graph theoretic problem to model target detection in environments which are represented by *surveillance graphs*. Each vertex is associated to a connected region in the environment and vertices associated to adjacent regions are connected by edges. Vertices and edges are denoted as contaminated if a target could possibly be located therein. Robots detect targets executing actions on vertices and edges of the graph. A sweep action detects targets within the region of a vertex and clears it from contamination. A block action on an edge prevents targets from crossing between two regions and hence prevents contamination to spread. According to our worst-case assumption, contamination spreads every time there is a chance for it. Block and sweep actions are abstractions of algorithms implemented and executed on a robot team. A strategy for a robot team is a sequence of these actions on an initially fully contaminated graph. The cost of each action is given by the weight of the vertex or edge. The Graph-Clear problem asks to find strategies with the lowest cost, i.e. the least number of robots needed to executed them. We showed the decision version of this problem is NP-hard [1]. Hence, we developed algorithms for Graph-Clear for the special case of trees. There is a useful distinction between so called *contiguous* strategies and those that are not. A contiguous strategy guarantees that all cleared vertices always form a connected sub-graph at any time during the execution of the strategy. This property can be useful when one needs to guarantee that a path free from contamination between any two robots always exists. Furthermore contiguous strategies are simpler.

We now outline a modified version of the contiguous algorithm for trees from [1]. The main modification is that we drop the requirement that when a vertex is swept all its edges are blocked. This requirement was introduced to allow implementations of a sweeping action for a robot team which cannot guarantee that no target leaves the region of the vertex, but can guarantee detection if no target enters or leaves undetected. Hence the need for blocks on the edges. In this paper we shall waive this requirement and propose a different and more effective approach for our particular sweeping and blocking implementations. We shortly recap the basic notions for the algorithm.

The main idea is that contiguous strategies can be computed with the help of labels associated to each edge for both directions. The label shall represent the cost of clearing all vertices when crossing the edge in the given direction. Using these labels one can compare the cost of clearing subtrees rooted at all neighbors. Figure 1 illustrates this idea.

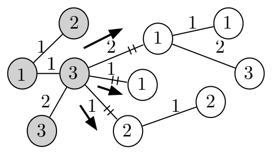

Fig. 1. In this simple graph gray vertices are cleared vertices and white vertices are contaminated. Arrows indicate possible continuations of a strategy each of which has a cost described by labels.

Formally, a surveillance graph is $G_{sg} = (V, E, w)$ with V being a set of vertices, $E \subseteq V \times V$ a set of edges, and $w : V \cup E \rightarrow \mathbb{N}$ a weight function. Let $\lambda_{v_x}(e)$ denote the label on an edge $e = [v_x, v_y]$ for the direction from

v_x towards v_y. Labels can be computed by first considering all v_y that are leaves. Originally the cost of clearing a leaf v_y was $w(v_y) + w(e)$, i.e. in order to clear a leaf it was also necessary to block its only edge. Here our modification comes into play. We extend the weight function to describe a directional weight, i.e. it matters from which edge a vertex is cleared. Let us redefine $w : (V \times E) \cup E \to \mathbb{N}$ and write $w(v_y, e)$ for the cost of clearing v_y entering from edge e. We also drop the requirement for the block cost $w(e)$, but assume that $w(v_y, e) \geq w(e)$. Now for all v_y that are leaves let $\lambda_{v_x}(e) = w(v_y, e)$. Once the labels towards leaves are computed we can consider vertices which are not leaves but for which all edges, except one, have an outgoing label. More precisely let v_x, v_y be neighbors and $m = degree(v_y) - 1$. Write neighbors of v_y different from v_x as $v_1, \dots, v_m$. When coming from v_x the first step is always to clear v_y, since the strategy has to be contiguous and v_x is assumed cleared. Then vertices of the contaminated subtrees rooted at the neighbors $v_1, \dots, v_m$ can be cleared. To simplify the matter we assume that once a subtree v_i, $i = 1, \dots, m$ is entered the robot team clears all its vertices, comes back and then clears another subtree. This assumption makes the approach simpler, but also leads to suboptimal solution in certain cases. Following this simple procedure the goal is now to clear the subtrees in an order that is least costly. Let $e_i = [v_y, v_i]$. It turns out that ordering $v_1, \dots, v_m$ s.t. $\rho_i = \lambda_{v_y}(e_i) - w(e_i)$ is descending and then clearing $v_m, \dots, v_1$ in this order has minimum overall cost [1]. The cost at step i is given by the cost of clearing subtree v_i and blocking all e_i towards subtrees that are still contaminated. Assuming that we order indices by ρ_i we can write this as:

$$c(v_i) = \lambda_{v_y}(e_i) + \sum_{l=1}^{i-1} w(e_l). \tag{1}$$

The new label for v_x then becomes the maximum of clearing v_y itself and the maximum cost occurred while clearing any one of the subtrees: $\lambda_{v_x}(e) = \max\{w(v_y, e), \max_{i=1,\dots,m}\{c(v_i)\}\}$. Given these definitions, computing all labels in a tree is straightforward. Once these are computed the overall cost of clearing the tree when starting at vertex v is determined by $ag(v) = \max\{w(v), \max_{1 \leq i \leq m}\{\lambda_v e_i + \sum_{l=1}^{i} w(e_i)\}\}$, where now $v_1, \dots, v_m$ are all neighbors of v, i.e. $m = degree(v)$ and $w(v) = \min_{e \in Edges(v)}\{w(v, e) + w(e)\}$[1]. Finding the best starting vertex leads to the best possible strategy that clears each subtree in a depth-first manner. To compute strategies on graphs with cycles one can apply the techniques presented in [1] to convert the graph into a tree and then extend the a strategies computed by the above algorithm to consider the cycles. We will now show how to construct a surveillance graph that is naturally embedded in the environment by a dual Voronoi Diagram, i.e. both have the same number of vertices and edges in free space. Furthermore, we will give implementations for block and sweep actions that lead to an embedding of strategies into the environment in form of moving lines between obstacles. These lines in turn lead to robot paths which a team can follow in a coordinated fashion. We will focus on practical considerations and refrain from a rigorous formalism on what it means to clear an environment from contamination with moving lines. Such a formalism is however of interest and the topic of work in progress.

[1]Since we redefined w we need this to denote the cost of clearing v coming from no edge. It emulates blocking e and then clearing v from there while keeping it blocked.

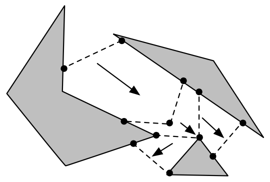

Fig. 2. Illustration of the concept of moving and splitting sweep lines. The arrows indicate the direction of movement of the sweep line on the left side until it splits into two sweep lines which continue independently.

III. Clearing with lines

At the core of our approach lies the idea to model the joint sensing capabilities of the robot team as a set of moving lines, each of which is spanned between obstacles. Recall that targets are assumed to be worst-case adversary and hence capable to exploit any weakness the robot team may exhibit. Hence the robot team needs to restrict all possible target movements until targets cannot possibly escape, which then guarantees a detection. The fundamental ability a robot team needs for this to succeed is to restrict target movement between obstacle boundaries. Moreover, it needs to move its sensor coverage continuously to extend the areas in which no target could possibly be located without having crossed a sensor. We shall denote a *s*weep line as any line spanned between two obstacles. A *m*oving sweep line is one that moves continuously. A splitting sweep line is one that is composed of two line segments with the mid point approaching a third obstacle. Figure 2 illustrates this idea. It is important to note that this sensing abstraction has serious implications with respect to applicability and usefulness of the algorithm presented in this manuscript. In particular, it is suitable for very limited sensing ranges, i.e. ranges shorter than most distances between obstacles. A robot with a large sensing range could cover the area of two disjoint sweep lines, which the model does not take into consideration.

The best sweep lines are obviously those with minimum length. This observation justifies the use of the Voronoi Diagram. We will use it to find short lines between obstacles. A rigorous formalization and generalization of Voronoi Diagrams to Generalized Voronoi Graphs (GVG) are found in [11]. The following definitions are useful and come from [11]. The environment is given as $\mathcal{E} \subset \mathbb{R}^2$ denoting free space and $C_1, \dots, C_{n_o}$, $n_o \in \mathbb{N}$ denoting n_o convex obstacles. Let $C = \bigcup_{i=1}^{n_o} C_i$ and δA be the frontier of a set A. The following functions define a distance function towards obstacles which is the basis for the GVG:

$$d_i(x) = min_{c_0 \in C_i} \|x - c_0\|, \quad \nabla d_i(x) = \frac{x - c_0}{\|x - c_0\|}.$$

With these one can construct *equidistant surfaces* and *2-equidistance surjective surfaces* via respectively:

$$\mathcal{S}_{ij} = \{x \in \mathbb{R}^2 : \ d_i(x) - d_j(x) = 0\}$$
$$\mathcal{SS}_{ij} = \{x \in \mathcal{S}_{ij} : \ \nabla d_i(x) \neq \nabla d_j(x)\}$$

Subsets of these then make up *2-equidistant faces* $\mathcal{F}_{ij} = \{x \in \mathcal{SS}_{ij} : \ d_i(x) \leq d_k(x) \ \ \forall k \neq i,j\}$ which are further restricted to *3-equidistance faces* $\mathcal{F}_{ijk} = \mathcal{F}_{ij} \cap \mathcal{F}_{ik}$. The 2-equidistant faces and 3-equidistant faces become the edges and vertices of the GVG, respectively. More precisely the GVG for two dimension is $G_{gvg} = (\mathcal{F}^2, \mathcal{F}^3)$ with $\mathcal{F}^2 = \bigcup_{i=1}^{n_o-1} \bigcup_{j=i+1}^{n_o} \mathcal{F}_{ij}$ and $\mathcal{F}^3 = \bigcup_{i=1}^{n_o-2} \bigcup_{j=i+1}^{n_o-1} \bigcup_{k=j+1}^{n_o} \mathcal{F}_{ijk}$. We will also make use of the function $q_i(x) = argmin_{c_0 \in C_i} \|x - c_0\|$ which returns the closest point to x from obstacle C_i. The Voronoi Diagram computed from a set of convex obstacles has useful properties to detect parts of the environment with small clearance and naturally provides a topological map. This fact has often been exploited for robot navigation [12]. Also Graph-Clear benefits from surveillance graphs with small edge weights and hence areas with small clearance provide good candidates for borders between regions leading to edges with small weights. But the relationship between Voronoi Diagrams and surveillance graphs goes further. We can construct a surveillance graph from a Voronoi Diagram by associating with each vertex of the Voronoi Diagram a corresponding vertex of the surveillance graph, and similarly for edges. For this paper we will restrict our attention to polygonal obstacles and consider each line segment of the polygonal obstacles as an independent obstacle, although there is no fundamental reason for requiring polygonal obstacles. It is however practical from an implementation point of view. Figure 3 shows a Voronoi Diagram that would be generated by considering each segment as its own obstacle. In fact, one segment leads to multiple obstacles since the endpoints are treated as obstacles as well.

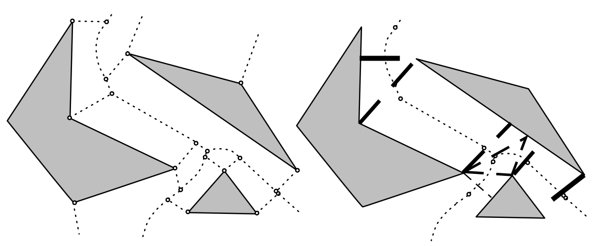

Fig. 3. Left: A Voronoi Diagram resulting from line segments of multiple polygonal obstacles by considering each open segment and their endpoints as independent obstacles. Right: Conversion of the Voronoi Diagram into a surveillance graph. Dashed lines indicate lines that are associated to vertices and edges and represent blocks and sweeps. The movement of lines is represented by their thickness, i.e. thin lines move towards thicker lines.

For each vertex in $\mathcal{F}^3$ which is in free space $\mathcal{E}$ we create a vertex for a surveillance graph G_{sg} and similarly add all edges between these vertices. We can assume[2] that any vertex of $\mathcal{F}^3$ has degree 3 and hence no vertex of G_{sg} will have degree larger than 3. The main problem of the construction is now to associate weights and moving liens to vertices. Figure 3 shows G_{sg} and a few lines we will associate to its vertices. With this figure in mind let us make this idea more precise.

Every vertex $v \in V$ has exactly three defining sites, i.e. distinct line segments or points from the polygon boundary. For each pair of these sites there can be an edge in G_{sg}. We need to create a sweeping routine and weights for v for each edge and direction. Let $e_{i,j}$ be one of the edges of v. Assume that $e_{i,j}$ has a sweep line spanned between its two defining sites C_i and C_j. Call this line the block line for $e_{i,j}$. To move this line further and sweep v we need to consider a splitting sweep line on the third defining site of v in the Voronoi Diagram, written C_k. Let $p_{k,i,j}$ denote lowest cost point for this split on C_k when coming from the line between C_i and C_j. The splitting sweep line is then composed of two line segments $[p_{k,i,j}, q_i(p_{k,i,j})]$ and $[p_{k,i,j}, q_j(p_{k,i,j})]$. Finding this point is simple with obstacles as line segments and points. The splitting sweep line can be reached from the block line by simply moving the endpoints of the block line to the two endpoints of the split line on C_i and C_j and then move a mid point towards $p_{k,i,j}$. Once $p_{k,i,j}$ is reached, the line splits into two lines which move independently towards the blocking line of the respective edges. For $degree(v) = 3$ these are two block line on edges e_2, e_3; for $degree(v) = 2$ there is only one sweep line and either $[p_{k,i,j}, q_i(p_{k,i,j})]$ or $[p_{k,i,j}, q_j(p_{k,i,j})]$ has zero length. For $degree(v) = 1$ there will be no blocking lines and the split line disappears on the boundary releasing the robots covering it. Doing this for every direction gives up to three critical points $p_{k,i,j}, p_{i,k,j}$ and $p_{j,k,i}$ each representing a sweep of v using their respective split lines.

We can now determine a precise location for the blocking lines. Given $q_i(p_{k,i,j})$ and $q_j(p_{k,i,j})$, the blocking line of $e_{i,j}$ is exactly the line between C_i and C_j of minimum length before $q_i(p_{k,i,j})$ and $q_j(p_{k,i,j})$ are reached. Hence it is either the line $[q_i(p_{k,i,j}), q_j(p_{k,i,j})]$ or a line with the minimum distance between C_i and C_j with endpoints further from C_k. Note that it may not intersect the Voronoi edge for $e_{i,j}$, but it certainly intersects $\mathcal{S}_{ij}$. Furthermore, the sum of distance function $d_i(x) + d_j(x)$ has exactly one minimum for $x \in \mathcal{S}_{ij}$. We can now associate this line with the edge block on $e_{i,j}$. The weights of edges can be computed by taking the length of its blocking line divided by $r - \delta$. Here r is the maximum line a robot can sense on, e.g the diameter of a disk if the robot senses with an omnidirectional limited range sensor. The parameter δ accounts for errors in localization, navigation or approximations in the map. Other footprints than disks could also be considered but for simplicity we refer only to omnidirectional sensors. For a vertex v we set

$$w(v, e_{i,j}) = \left\lceil \frac{\|[p_{k,i,j}, q_i(p_{k,i,j})]\|}{r - \delta} \right\rceil + \left\lceil \frac{\|[p_{k,i,j}, q_j(p_{k,i,j})]\|}{r - \delta} \right\rceil$$

which is the cost of covering the split line at the critical point, the most costly step during the sweep. A few properties of

[2]If this is not the case, the requirement can be enforced by adding a small random perturbation [13].

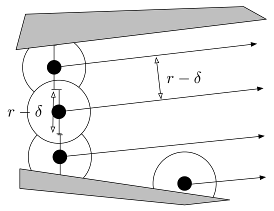

Fig. 4. Multiple robots spanning a line between two obstacles. As the distance between obstacles grows another robot is added.

the Voronoi Diagram are exploited in the above construction. The block line does not depend on the direction from which we compute it, i.e. with v_1 and v_2 as endpoints of $e_{i,j}$ it does matter whether we compute the block line using the above procedure for v_1 or v_2. This is due to the fact that obstacles are convex and line segments. In practice, block lines are often at the position right after the split on $p_{k,i,j}$, in particular when one or more of C_i, C_j, C_k are points. Fig. 4 shows how multiple robots maintain a line oriented towards the left site, i.e. robots are placed uniformly on a sweep line starting from the left site and placed at distance $r - \delta$ from each other with the first robot having distance $\frac{r-\delta}{2}$ from the left site. As the distance between the obstacles grows another robot will have to be added. Due to the left bias this is trivially achieved. Following the split lines is also easy and an illustration is shown in figure 5. A further discussion of the significance of δ and how to arrange robots on the line is found in VI.

IV. Implementation

The actual implementation to use the line clearing approach to construct a surveillance graph from an occupancy grid map proceeds in several stages. First, to smoothen the map we convert it to a polygon by computing its α shape using the CGAL library [14]. These shapes are frequently used to reconstruct the shape of a dense set of points. Once we get the polygon boundary from the occupied grip points we apply the Ramer-Douglas-Peucker line-simplification algorithm [15] to get a polygon boundary with less line segments. A parameter specifying the degree of simplification is required which we shall denote as ε. We will discuss the sensitivity to these parameters in section V. After these two steps the polygon segments provide obstacles sets that can be processed to compute the Voronoi Diagram. Once the Voronoi Diagram is computed we proceed by computing the critical points for each vertex and thereby its directional weights. Edges also receive their critical points at the minimum length line between its defining sites and get their block weights. Given the weights we apply the contiguous algorithm for trees described in section II as well as the minimum spanning tree based cycle blocking. This gives us a strategy in form of a sequence of vertices. We compute the overall cost of the strategy as well as the cost of clearing the tree without considering the cycle edges. From here on it is a small step towards the actual motion of the robots. Given the sequence of vertices we can construct a sequence of moving lines. The first vertex is cleared by blocking one edge and clearing it coming with a new sweep line starting from that edge, leading to blocks on all its edges. From there on every next vertex is a movement of one blocking line towards the critical point for the direction the line is coming from. These continuously moving lines can be followed by a team of robots. To account for localization and navigation errors robots can be spaced with sensors overlapping as seen in Figure 5. The δ parameter also helps to offset possible approximation errors during the conversion of the grid map into a polygon.

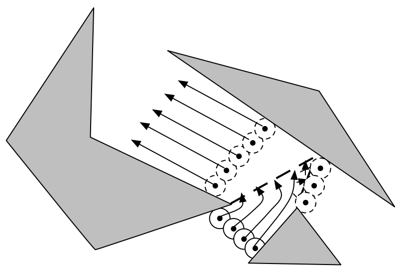

Fig. 5. Robots following a sweep line with δ overlap and splitting into two sweep lines at a critical point. Robots with solid disks are moving towards future positions marked as robots with dashed disks. Note that the four robots require additional robots to reach the dashed positions.

V. Experiments

To validate the algorithm from section IV we ran it in a variety of configurations on the grid maps from [2]. The first grid map was obtained with a Pioneer P3AT mobile platform equipped with a SICK PLS200 laser range finder. The robot was driven through the 2nd floor of the UC Merced Engineering and Science building to collect laser data. The data was then used to build the map with the Gmapping software [16]. Figure 6 shows the map after it was processed by computing its α-shape and simplifying the polygon boundary. We shall call this map UCM map. The second map is obtained from the Radish online robotics data repository [17] sdr_site_b data set. We will denote it as SDR map. It is shown after processing in Figure 7. We compare our algorithm to the one used in [2] which also extracts surveillance graphs. The extraction in [2] is based on a selection of local clearance minima on an approximation of the Voronoi Diagram and creating edges at these minima. For each minima a line is spanned between two closest obstacles which leads to a partition of free space. For each connected set in this partition a vertex is created. Weights of vertices are computed by assuming they are swept like a rectangle, i.e. the top-left and bottom-right point belonging to the vertex are used to define a rectangle which is then swept by moving robots on a horizontal or vertical line through it. This very crude sweeping routine does not acknowledge complexities in the environment within a vertex. On the graph, strategies are computed with the hybrid algorithm from [3]. These generally have lower cost than contiguous strategies, but they do not satisfy contiguity. The evaluation of the algorithm from [2] included the cost for strategies

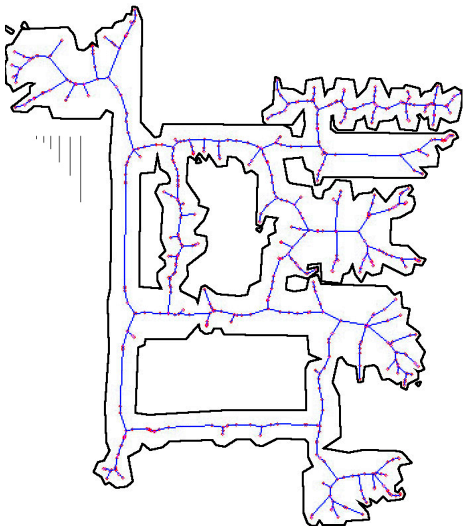

Fig. 6. UCM map obtained from [2] after applying the α-shape and line simplification with $\alpha = 10$ and $\varepsilon = 3$. The graph is embedded with thin lines as edges in free space. Distances are in pixel and horizontal lines of length 5,10,20,40,60 and 100 pixel on the left illustrates scale.

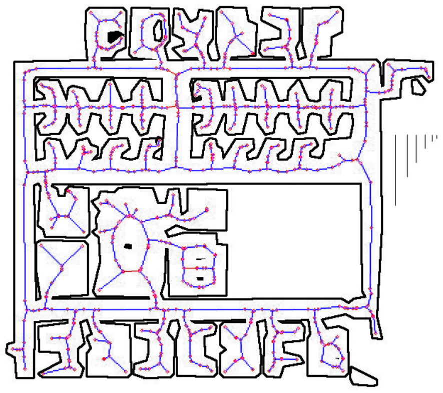

Fig. 7. The SDR Map obtained from [2] just as Figure 6.

on the generated tree denoted by ag. Let ag_r denote the cost of strategies extended to the entire graph. Since all non minimum spanning tree were assumed blocked in [2] we set ag_r to $ag + b_c$, where b_c is the cost for all cycle edges. Let b denote the number of cycles. In our case, ag_r denotes the cost for only blocking cycle edges when needed and corresponds to the actual number of robots needed to clear the environment by following the sweep lines associated to vertices and edges. We write r for the sensing range in pixels and δ for the parameter adjusting for errors.

An accompanying video shows a sequence of block and split lines in the UCM map ($\varepsilon = 3$, $\alpha = 10$, $\delta = 2$, $r = 40$) as red and green, respectively. Associated cost for lines is show in red for blocks, green for split lines and blue for block lines due to cycle edges. At each step part of the robot team is moving from a red block line to a green split line. Counters on the top left show the clearing step, current cost and maximum encountered cost.

Table I summarizes the results from [2] for comparison purposes. Of importance are primarily ag_r and ag as a function of r. Tables II and III present results from executing

	UCM map				SDR map			
r	ag_r	ag	b	b_c	ag_r	ag	b	b_c
5	73	58	3	15	67	36	8	31
10	37	28	3	9	36	19	7	17
20	20	14	3	6	19	9	7	10
40	11	8	3	3	14	6	8	8
60	9	6	3	3	11	5	6	6
100	7	4	3	3	10	4	6	6

TABLE I
SUMMARY OF THE EXPERIMENTAL RESULTS FROM [2].

the algorithm from section IV on the UCM and SDR maps. Table II can be compared to table I since $\delta = 0$. Despite the additional constraint to require contiguous strategies the new approach outperforms the one from [2] on the UCM map for sensing ranges from 5 to 60. Also on the SDR map the new approach produces better strategies on the tree (ag) for sensing ranges 5 to 60. The similar performance for sensing ranges at 100 result from the fact that the method from [2] ignores the complexities of the environments when merging it into a graph with 13 to 14 vertices each of which is swept with the rectangular sweep.

It is also interesting to note that the new approach removes the problem encountered in [2] in which two vertices are merged leading to the removal of a cycle edge. The cycle in the environment is then inside the vertex and effectively ignored. The new approach continues to capture the geometric complexity without such heuristics and the number of cycle edges remains relatively constant. There is, however, a slight variation resulting from the processing of the grid map. The produced alpha shapes are not deterministic and slightly different alpha shapes may lead to a different surveillance graph which can also lead to a slight variation in the number of cycles, as well as slight variations in the tree and hence the strategy cost. Usually these cycles are degenerate, resulting from the particular computation of the Voronoi Diagram as the dual of the Delaunay Graph. These, however, do not pose a problem neither for computation nor for the execution of the strategy.

		$\delta = 0$				$\delta = 2$			
Map	r	ag_r	ag	b	b_c	ag_r	ag	b	b_c
UCM	5	56	46	3	15	90	75	3	24
UCM	10	32	24	3	9	36	30	3	10
UCM	20	18	13	3	6	20	14	3	6
UCM	40	9	7	3	3	10	8	3	3
UCM	60	9	6	3	3	8	6	3	3
UCM	100	7	5	3	3	6	4	3	3
SDR	5	45	31	14	48	72	47	14	73
SDR	10	26	16	14	27	30	18	15	31
SDR	20	14	8	14	17	15	9	15	18
SDR	40	11	6	14	11	10	6	14	11
SDR	60	10	5	14	10	8	5	14	10
SDR	100	9	5	14	10	7	4	15	10

TABLE II
SUMMARY OF THE EXPERIMENTAL RESULTS WITH $\alpha = 10$, $\epsilon = 7$. NOTE THAT SOME MST-EDGES ARE DEGENERATE AND HAVE 0 WEIGHT.

As expected, increasing δ leads to slightly more costly strategies as seen in table II. Obviously, robots with small sensing ranges are affected more than those with large sensing range since the ratio of δ to r matters. But already a relatively small δ of 2 leads to cost increases across all sensing ranges. Table III shows the effect of varying the degree of simplification for the Ramer-Douglas-Peucker algorithm. Setting ε from previously 7 (in table II) to 3 leads to no significant differences in ag. The differences in ag_r can be explained by an unfortunate selection of cycle edges due to a different selection via the minimum spanning tree. Also note that degenerate cycle edges disappeared for the SDR map. All in all the experiments show that the algorithm is simple to implement and robust while maintaining consideration for the geometric complexities. The only approximations are made during the polygon creation and simplification which are marginal in comparison to the heuristics employed in [2].

	UCM map				SDR map			
r	ag_r	ag	b	b_c	ag_r	ag	b	b_c
5	97	76	3	24	73	50	10	69
10	38	30	3	9	30	19	10	28
20	18	14	3	6	15	9	10	15
40	10	8	3	3	10	6	10	10
60	8	6	3	3	7	5	10	10
100	7	5	3	3	11	5	10	10

TABLE III

SUMMARY OF THE EXPERIMENTAL RESULTS WITH $\alpha = 10$,$\delta = 2$, $\epsilon = 3$. NOTE THAT DEGENERATE MST-EDGES FROM TABLE II DO NOT APPEAR.

VI. DISCUSSION AND CONCLUSION

We provided an improved method to relate surveillance graphs to the planar environment they represent. This leads to an efficient extraction of surveillance graphs from any robotic grid map or polygonal environments. It also gives a starting point for investigating the relationship between surveillance graphs and line sweeping in 2d planar environments. Regarding the computation of Graph-Clear strategies it is apparent that the reduction to a tree is not entirely satisfactory and motivates the study of approximation algorithms to the problem on the graph. Furthermore, an optimal algorithm for contiguous strategies on trees is obviously desirable and hence subject of current work. From a practical point of view the presented vertex sweeps and edge blocks can be executed by a robot team which follows the lines that are associated with the sweeps and stops at the blocks. Here a wide body of literature is available and a control theoretic approach in the spirit of [10] is a viable direction to pursue. The robots have to follow a moving boundary which can be achieved with an event-driven asynchronous robotic network as described in Chapter 6 of [10]. Therein issues such as communication are also addressed. Furthermore, the robot team requires some coordination to assign paths that result from following the lines to individual robots. Here performance parameters such as time and travelled distance can start to play a role. A robot which is not needed for a few vertex sweeps could already travel to the vertex where it is needed next, speeding up the overall execution. Also an interesting question is whether a minimalist approach such as in [5] can be employed for following sweep lines. Another important direction is the consideration of probabilistic sensing and errors in control. In [18] Graph-Clear is extended to a probabilistic sensing model. The algorithm therein can be combined with the δ parameter and together give a probabilistic guarantee that no intruder passes through a sweep line. Failure to detect may result from the sensor or an errors in following a sweep line which can open up a gap. Increasing δ and using more robots reduces this probability. Of greater interest, however, is the geometric relationship of surveillance graphs with 2d environments via Voronoi Diagrams. This opens up further theoretical possibilities to study optimal movements of sweep lines between obstacles.

REFERENCES

[1] A. Kolling and S. Carpin, "The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance," in *Proc. of IEEE/RSJ Intl. Conf. On Int. Robots and Systems*, 2007, pp. 1003–1008.

[2] ——, "Extracting surveillance graphs from robot maps," in *Proc. of IEEE/RSJ Intl. Conf. On Int. Robots and Systems*, 2008, pp. 2323–2328.

[3] ——, "Multi-robot surveillance: an improved algorithm for the graph-clear problem," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, 2008, pp. 2360–2365.

[4] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM J. on Computing*, vol. 21, no. 5, pp. 863–888, 1992.

[5] S. Sachs, S. Rajko, and S. M. LaValle, "Visibility-based pursuit-evasion in an unknown planar environment," *Int. J. Robotics Research*, vol. 23, no. 1, pp. 3–26, Jan. 2004.

[6] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *Int. J. of Comp. Geom. and Appl.*, vol. 9, pp. 471–494, 1999.

[7] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou, "The complexity of searching a graph," *J. ACM*, vol. 35, no. 1, pp. 18–44, 1988.

[8] 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 *Proc. of the 11th ACM-SIAM symp. on Disc. Alg.*, 2000, pp. 927–936.

[9] S. D. Bopardikar, F. Bullo, and J. P. Hespanha, "Cooperative pursuit with sensing limitations," in *ACC*, 2007, pp. 5394–5399.

[10] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, to appear. Electronically available at http://coordinationbook.info.

[11] H. Choset and J. Burdick, "Sensor based planning, part I: The generalized voronoi graph," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1995, pp. 1649 – 1655.

[12] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[13] M. de Berg, M. van Kreveld, M.Overmars, and O. Schwarzkopf, *Computational Geometry*. Springer, 2000.

[14] 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/

[15] 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.

[16] G. Grisetti, C. Stachniss, and W. Burgard, "Gmapping - openslam.org." [Online]. Available: http://www.openslam.org/gmapping.html

[17] Radish: The robotics data set repository. [Online]. Available: http://radish.sourceforge.net/

[18] A. Kolling and S. Carpin, "Probabilistic graph clear," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 3508–3514.