2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Extracting Surveillance Graphs from Robot Maps

Andreas Kolling and Stefano Carpin

Abstract— **GRAPH-CLEAR is a recently introduced theoretical framework to model surveillance tasks accomplished by multiple robots patrolling complex indoor environments. In this paper we provide a first step to close the loop between its graph-based theoretical formulation and practical scenarios. We show how it is possible to algorithmically extract suitable so-called surveillance graphs from occupancy grid maps. We also identify local graph modification operators, called contractions, that alter the graph being extracted so that the original surveillance problem can be solved using less robots. The algorithm we present is based on the Generalized Voronoi Diagram, a structure that can be simply computed using watershed like algorithms. Our algorithm is evaluated by processing maps produced by mobile robots exploring indoor environments. It turns out that the proposed algorithm is fast, robust to noise, and opportunistically modifies the graph so that less expensive strategies can be computed.**

I. Introduction

We recently introduced a theoretical framework called GRAPH-CLEAR to model surveillance tasks performed by teams of robots [1][2]. One of its peculiar aspects resides in a conceptual mechanism that allows to model and study different search and clear strategies abstracting from the underlying robotic platforms used. GRAPH-CLEAR in particular aims to model surveillance tasks where multiple robots with limited sensing capabilities cooperate to detect intruders in complex environments. The name GRAPH-CLEAR stems from the fact that environments to be cleared are modeled as surveillance graphs, i.e. a special graph class that will be later defined. Theoretical properties of GRAPH-CLEAR, as well as solving algorithms have been extensively studied in an earlier stage of this research, and will be summarized in the remaining part of the paper. In this manuscript we present the first steps towards the practical deployment of robot teams that clear complex environments using the formalism we formerly investigated. In particular we address the problem of automatic extraction of surveillance graphs from occupancy grid maps. Informally speaking, this task entails allocating graph vertices on rooms and graph edges on corridors or connections between rooms. The reader will realize that this step is similar to creating topological maps from occupancy grid maps. However, our task is more complex and our contribution more articulated. In order to enable the GRAPH-CLEAR framework it is also necessary to determine certain *weights* associated with edges and vertices. These weights measure the effort, i.e. the number of robots, needed to enforce relevant properties on the graph, like for example preventing intruders from passing through a door, or from hiding in a room. The method to assign weights presented in this paper is parametric with regard to the sensing capabilities of the robots used to implement the clearing strategies, and has therefore general applicability. In addition, while extracting graphs from occupancy grid maps we have identified certain opportunistic operations that modify the graph so that the algorithm generating clearing strategies produces solutions requiring less robots. While most ideas herein generalize to varying implementations of the basic GRAPH-CLEAR actions, i.e. blocking edges and clearing vertices, we will present experimental results for particular implementations of these actions on two realistic robot maps, one collected for a P3AT at UCMerced and one generated from the Radish online robotics data repository [3] called "sdr_site_b". The remaining part of the paper is organized as follows. Section II shortly revises related work in the area of robot aided surveillance systems, and pursuit evasion games. GRAPH-CLEAR and its solving algorithm are informally presented in section III. Section IV illustrates how graphs are created starting from occupancy grids, and how these graphs can be modified in order to yield better solving strategies. Strategies to block edges and clear vertices are presented in section V. Finally, experimental results are shown in section VI, and conclusions are drawn in section VII.

School of Engineering, University of California,Merced, CA, USA

II. Related Work

Surveillance tasks have been studied in a great number of variations. One version with strong theoretical results are visibility-based pursuit-evasion games, first investigated by Suzuki and Yamashita for detecting targets with an unlimited range beam sensor in [4]. Subsequently many variants of this problem were studied, most notably the variant of a robot with an unlimited range omnidirectional gap sensor [5] which can detect intruders robustly and in manifold environments. An entirely different approach was taken by Parker who investigated the surveillance of multiple moving targets in simple planar environments by large robot teams in [6]. Also in open planar environments we find a capturing strategy presented in [7] in which robots form a so called trapping chain to ensure that the target once detected by any robot in the chain will subsequently be caught. Probabilistic detection guarantees are given for traversing an environment with such chains for varying conditions. A probabilistic approach which works in cluttered environments is presented in [8].

The GRAPH-CLEAR problem was formalized in [2] in which we presented algorithms to compute strategies for GRAPH-CLEAR on trees and heuristics to apply tree strategies to graphs. Improved algorithms were presented in [1]. GRAPH-CLEAR is an extension of edge-search another

978-1-4244-2058-2/08/$25.00 ©2008 IEEE.

variant of graph searching first presented in [9]. Another related contribution to edge-search is [10] in which weights on edges and vertices for edge-search were considered. In colloquial terms the major distinction between edge-search and graph-clear is that instead of edges vertices are cleared.

Extracting topological maps, usually represented as graphs, from environments has been widely studied. Foremost, Voronoi-based or fuzzy topology based approaches have found their usage in robotics. In particular, Voronoi Diagrams have been used extensively. In [11] a rigorous formalization of the Generalized Voronoi Diagram (GVD) and the Generalized Voronoi Graph (GVD) are given, which are valuable when extending our results to higher dimensions. Since we are in a robotics context we shall mean the GVD whenever we refer to a Voronoi Diagram. In robotics the construction from sensor data also received considerable attention such as in [12] and [13]. Voronoi Diagrams were used in path planning [14] and have also proven to be useful in localization [15]. While we can benefit from the previous research on constructing the diagrams the application of Voronoi Diagrams for constructing surveillance graphs and improving these is, however, novel.

III. GRAPH-CLEAR

This section introduces the basic terminology and ideas behind the GRAPH-CLEAR problem in colloquial terms. A rigorous and formal introduction can be found in [1][2] and the reader is referred to those papers for more details about the results stated in this section. The problem is defined on a weighted graph with weights on edges and vertices, which we term Surveillance Graph (SG). Intuitively, edges correspond to narrow connections between wide and open regions which in turn correspond to vertices. Rooms or corridors/doors are said to be *contaminated* if they may hide one or more intruders, and are said to be *clear* otherwise. Robots can block edges between vertices thereby preventing contamination from spreading. Otherwise contamination spreads from all contaminated elements through all edges without a block. Vertices can be cleared by the robots which corresponds to detecting all intruders present within the region associated with the vertex. Clearing vertices is hence the basic action to remove contamination from a SG. The costs in terms of the number of robots for these actions are the associated weights on edges and vertices. If e is an edge, its blocking cost will be indicated as $w(e)$, whereas the clearing cost of a vertex v will be indicated as $w(v)$. An ordered sequence of blocking and clearing operations that turn all vertices and egdes from contaminated to clear makes up a strategy for a SG. Note that multiple actions can be applied simultaneously. The largest number of robots a strategy uses at any one time is its cost. Our goal is to determine strategies of minimal cost. The decision variant of the general GRAPH-CLEAR problem is NP-complete. There are, however, efficient methods to find strategies for trees and heuristics to reduce the SG to a tree. All algorithms that compute strategies on trees are based on labels attached to the edges of the SG. Labels have a direction, i.e. two per edge, and represent the number of robots needed to clear the part of the tree the team enters when moving along the edge in the respective direction. The label $\lambda_{v_x}(e)$ for an edge e between vertices v_x and v_y and in the direction of v_y is computed as follows. If v_y is a leaf, then $\lambda_{v_x}(e) = w(v_y) + w(e)$, otherwise we denote all neighbors of v_y different from v_x by $v_2, \ldots, v_m$ where m is the degree of v_y. We denote the edge from v_y to v_i by e_i and w.l.o.g. assume the neighbors are ordered w.r.t. $\lambda_{v_y}(e_i) - w(e_i)$ in decreasing order. For each neighbor we define a cost:

$$c(v_i) := \lambda_{v_y}(e_i) + \sum_{2 \leq l < i} w(e_l), \tag{1}$$

Using the cost for each neighbor we define the label on e when coming from v_x as:

$$\lambda_{v_x}(e) = \max\{s(v_y), \max_{i=2,\ldots,m}\{c(v_i)\}\}. \tag{2}$$

where $s(v) := w(v) + \sum_{e \in Edges(v)} w(e)$ is the safe clearing cost for a vertex, i.e. a clearing action on the vertex and blocking actions on all edges connected to vertex being cleared. Fig. 1 illustrates the label computation. The labels can be computed recursively starting from the leaves of the tree, i.e. vertices of degree one.

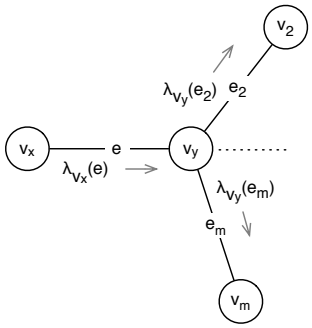

Fig. 1. An illustration of the computation of the label $\lambda_{v_x}(e)$ from v_x towards v_y. This involves the labels λ_{v_y} on edges $e_2, \ldots, e_m$, where $m = degree(v_y)$, and the edge and vertex weights. The direction of a label is further indicated by the arrow.

One of the main problems that remains to be solved in order to use GRAPH-CLEAR for a realistic surveillance scenario is to construct a SG from a given environment, which is the main focus of this paper.

IV. FROM MAPS TO SURVEILLANCE GRAPHS

Our proposed approach is based on a Voronoi Diagram for the given environment. Voronoi Diagrams can readily be constructed from sensor data, occupancy grid maps or vectorized maps. For our purposes we will assume a two dimensional grid map. Once a Voronoi Diagram is given we construct a SG which leads to strategies with low costs. Before we tackle this problem let us first discuss how blocking on edges and clearing on vertices can be implemented.

1) Blocking: The requirement from GRAPH-CLEAR is that a block detects intruders as they attempt to pass through the edge. This can be fulfilled by continuously covering the area corresponding to the edge with sensors. In practice no sensor can give a 100% guarantee that an intruder will be detected and one may wish to cover the area with multiple sensors, or if speed constraints apply to intruders then one might just need one robot patrolling fast enough along the edge while not covering it all at once. The actual choice of implementation can differ widely depending on the application. Our basic assumption is that any implementation will benefit from edges placed at narrow section of the environment.

2) Vertex clear: For implementations of vertex clearing the choices are even more manifold. In open uncluttered regions one could use the sweep-pursuit-capture strategy presented in [7]. In cluttered but not too large regions the approach from [8] could be applied. For vertices in which the sensor range is larger than the diameter of the region one could use the approach from [5]. These methods are shortly presented in section II. A good choice for an implementation of the clearing action is heavily dependent on the type of robot used and the shape of region that the vertex represents. That being said, however, one of the advantages of GRAPH-CLEAR is not only that it can scale local clearing methods to very large environments, but that it allows us to use simple clearing methods for the vertices which could be implemented by very simple robots. In section VI we will demonstrate the graph construction with a very simple bounding box sweeping. Our approach for the graph construction is in two stages. The initial construction places edges at every possibly beneficial position, i.e. every narrow section of the environment. Already in a simple case such as in fig. 2 one can see that not every narrow part should receive an edge, depending on the type of vertex clearing and edge block implementation and the type of robot. Hence the construction proceeds with *contracting* (i.e. merging) vertices from the initial construction when an edge between two vertices is not beneficial. The next two sections present this two-staged construction.

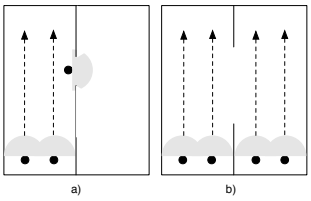

Fig. 2. Illustrating the advantage of narrow connections between open regions. For robots with a limited sensing range the environment in part a) can be cleared with 3 robots, while the one on part b) requires 4. The reader should note that the surface is the same.

A. Initial Graph Construction

Given a map, we first compute the Voronoi Diagram. Similar to [14], in which the graph was used for fast path planning, we use the minima of the local clearance function defined on the edges of the Voronoi Diagram to create the SG. The local clearance function on the edges of the Voronoi Diagram is simply the distance from the point on the edge to the nearest obstacle point. A minima is considered any point for which $\exists \epsilon > 0$ s.t. within its ϵ-neighborhood there are no other points with strictly smaller clearance and $\forall \delta > 0$ there is at least one point within its δ-neighborhood with strictly larger clearance. In colloquial terms, one very close neighbor should be larger and within any small neighborhood none of the neighbors should be smaller. Here we differ from the construction in [14] which considers all points for which all points within an ϵ-neighborhood are not smaller, which would also include entire plateaus and also those of maxima. With our definition, if a minimum value is achieved on a compact subset of the edge we select the two end points of the set. Fig. 3 shows a Voronoi edge and two minima on it. It is worth to note that minima cannot lie

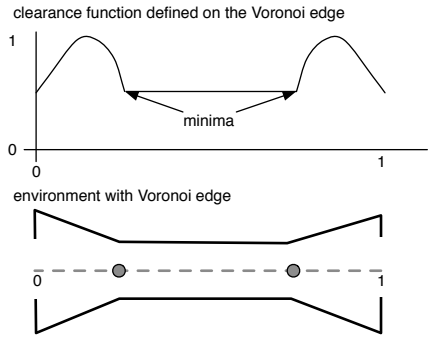

Fig. 3. A simple environment with a Voronoi edge in the center as a dotted line and the clearance function in the graph on top. The minima on the Voronoi edge are marked by grey circles.

on vertices of the Voronoi Diagram, so by only considering edges we do not miss any minimum. For each minimum we consider the lines from the minimum to the two nearest obstacle points, which since we are on a Voronoi edge lie in two different obstacles. These lines will be represented by an edge in the SG, i.e. we are partitioning free space into regions based on these lines and each region becomes a vertex in the SG. In most cases this construction yields a valid partitioning and hence a valid graph. Fig. 4 shows such a construction. Once the edges and vertices of the SG are constructed we use the implementation of the edge block and the vertex clearing actions to compute the weights for the SG. This concludes the initial construction. The resulting surveillance graph will be the starting point to find a graph with better strategies. In most environments the presented construction will introduce many more edges than would be beneficial, as seen in fig. 2 in which an edge between the two regions is only of advantage when it is sufficiently narrow. Since this depends on the implementations of the actions and the type of robot we will have to introduce a method that considers this when improving the surveillance graph. Gladly, we can use a general approach that merely calls the weight computation of the implementations. After presenting this in the next section we will demonstrate it

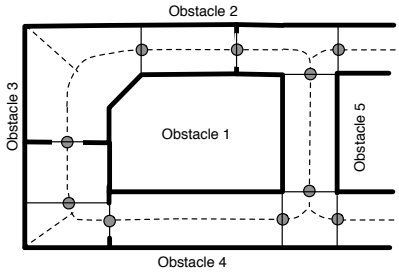

Fig. 4. A Voronoi Diagram and its minima. The Voronoi Diagram is marked with grey dashed lines, the minima with grey circles and the lines to the closest obstacle points with thin black lines. Obstacle boundaries are thick black lines. Corners in corridors tend to produce minima, unless a narrow part proceeds it as seen in the upper left corner of the figure.

with two realistic examples.

B. Improving the graph

The initial construction does not guarantee the existence of good strategies for the GRAPH-CLEAR problem. Hence, the next step is to contract vertices wherever this may lead to a better strategy. This will also remove spurious minima that may be introduced by noise in the map or approximations of the Voronoi Diagram, as seen in fig. 5. The simplest type of

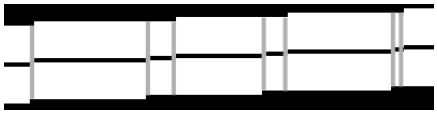

Fig. 5. An example in which a discrete approximation of the Voronoi Diagram in a grid map leads to introduction of unwanted edges. The black line in the center is the Voronoi Diagram edge and minima are marked by grey lines

contraction is between a leaf vertex l and its sole neighbor v. Let e be the edge connecting them and v' be l and v merged. Contracting l and v cannot make the strategies for the SG worse if $w(v') - w(v) \leq w(e)$. It is easy to verify from the equations from [2] that under these conditions none of the labels in the graph can get larger. Furthermore, from [1] it follows that if e is not in the tail of another a label on an edge coming from the neighbors $v_2, \ldots, v_m$ of v, then this labels necessarily improves due to the removal of the edge. The tail of a label in colloquial terms is the set of edges whose ending vertices are cleared before clearing the one with the highest cost $c(e_i)$, a concept formalized in [1]. The second contraction is for vertices of degree two. Let c be such a vertex, v_y, v'_y its neighbors and e, e' the respective edges. If $w(e) \geq w(e')$ and $w(v^c_y) \leq w(v_y) + w(e) - w(e')$ then a contraction of v_y and c into a single vertex v^c_y is of advantage. Fig. 6 shows this contraction. If either of these two conditions is satisfied then a contraction is guaranteed not to lead to worse strategies on the SG, regardless of the value of the labels on the edges. Analyzing general contractions is beyond the scope of this paper and involves a careful and formal consideration of the recursive nature of the label computation, occurrences of the maxima in

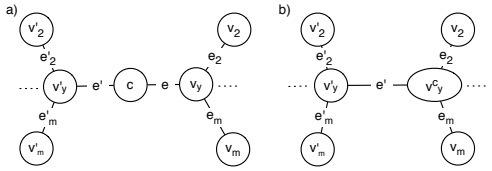

Fig. 6. A contraction of a vertex with degree two and its neighbor. Part a) shows the initial graph and part b) the graph after the contraction.

batches and the role of the tails as it is done in [1]. We will, however, demonstrate the potency of already the simple types of contractions in our experimental section.

V. Implementing blocking and clearing actions

As previously indicated, blocking and clearing actions of a strategy for the SG can be implemented in manifold variations depending on the particular needs of the application. The GRAPH-CLEAR abstractions can be of benefit in any environment in which occlusions can help the pursuers to restrict contamination from spreading. The requirements for the implementation is merely that the computation of the weights on vertices and edges is possible and that the action can be executed. In particular when using vertex clearing strategies that have strict assumptions such as the region of the vertex being simply connected this has to be considered when building the graph. In this case one needs to detect vertices in which this is not given and subdivide them into further vertices. Issues with particular types of sensors, their range and error rates are all aspects that come into play when designing the implementation details. Once the implementation gives satisfactory guarantees for the detection of intruders, then GRAPH-CLEAR can be applied. An interesting property of GRAPH-CLEAR solution strategies is that all paths from the cleared vertices of the graph to those that are contaminated have always at least one block. In fact, if a probabilistic guarantee of e.g. 95% to detect an intruder trying to pass through a blocked edge can be given and a close to certain probability that one will be detected during a vertex clearing, then the final cleared graph will contain no intruder with 95% certainty. While it is beyond the scope of this paper to introduce a probabilistic variant of GRAPH-CLEAR there is already quite something one can achieve w.r.t. the implementation of the actions and there are some immediate relationships between a detection probability for a vertex sweep or edge block and the entire graph.

VI. Experimental Results

To demonstrate some the presented ideas in an application we constructed a robot grid map of part of the UC Merced Science and Engineering building with a Pioneer P3AT mobile platform equipped with a SICK PLS200 laser range finder. The map is built using the GMapping software [16]. Figure 7 shows this map, which we will further denote as UCM map. As a second map we used the sdr_site_b data set from the Radish online robotics data repository

[3]. This latter set will be denoted as SDR map, and the correspondingly generated map is seen in figure 8.

To both maps a low-pass filter was preliminary applied to remove noise and get smoother boundaries. Both maps have a complicated structure, many occlusions, circles and noisy artifacts. The resolution of the grid map for the UCM map is 690x790 pixels while the SDR map is 645x573. The free space in the UCM map and SDR is approx. 35.8% and approx. 46.4% respectively, which corresponds to 442^2 pixels for the UCM map and 414^2 pixels for the SDR map. In the SDR map there are two small pockets at the bottom of the map which were included in the open space calculation but are not accessible and hence not part of the graph construction. Since the maps are both grid maps a simple wave propagation algorithm to compute an approximation of the Voronoi Diagram has been used. The algorithm implicitly assumes that points on a straight line belong to the same obstacle and a diagonal marks a new obstacle. Diagonal collisions are permitted, complicating implementation slightly. Since our focus is on the graph construction we will spare the remaining details of the Voronoi Diagram construction noting that good algorithms have been developed in the vast body of literature on the topic.

Based on the crude approximation of the Voronoi Diagram we construct the SG graph by detecting the minima on the Voronoi Diagram edges. We mark every point on the Voronoi Diagram as a minima which has several neighbors in at least one direction being larger and no other point on the Voronoi edge within 3 steps in any direction being smaller. To avoid too many minima on the initial graph we set a minimum distance between initial minima to 10 steps on the diagram. The selection of minima can be application dependent, e.g. for some applications it may be desirable to have minima that are guaranteed to be further apart. Whilst very close edges are likely to be merged in the contraction stage it is still more convenient not to clutter the initial graph with many spurious edges when those are easy to avoid. The resulting graphs for the two maps are displayed in figures 7 and 8.

Once the SG graph is constructed we compute the weights on edges by computing the distance d between the two closest obstacle points of the minimum. The weight on the edge becomes $w(e) = ceil\left(\frac{d}{r}\right)$ whereby r is the maximum the sensor can cover between any two points. For example, for an omnidirectional sensor this will be the diameter of its disk, while and for a 45 degree laser range scanner it will be the maximum range of one beam. For vertices we assume a simple bounding box clearing method, i.e. we compute a rectangular bounding box around region of the vertex. Let s be the length of the shorter side of the bounding box, then the vertex weight becomes $w(v) = ceil\left(\frac{s}{r}\right)$ where r is as before. We assume only horizontal and vertical lines for the bounding box. This simple vertex clearing implementation should lead to less contractions in the given environments as it penalizes merging vertices that lead to complicated regions in which such a bounding box is a poor clearing method.

Once the initial graph is given we compute GRAPH-CLEAR strategies on it. First we convert the graph into a tree by computing the Minimum Spanning Tree (MST) w.r.t. to the inverse of the edge weight to yield those edges in the MST with the highest weight. Edges that are not in the MST will be blocked continuously to reduce the graph to a tree. In [2] details of this method are presented and it is shown that not all of the non-MST edges have to be blocked simultaneously, i.e. the total cost of clearing the environment can be further reduced. We then used the currently best known variant of the hybrid strategy algorithm from [1] on the tree. The partitioning problem for the hybrid algorithm mentioned in [1] was solved with brute force, albeit not hindering computational performance as the solutions were computed in the order of milliseconds. Labels in all directions were computed and the vertex with the best cost was chosen as the starting vertex. Finally, we start the contraction process which proceeds in loops, contracting all vertices satisfying the criteria from section IV-B at each iteration until no more such contractions are found. The resulting graph can be seen in fig. 7 and 8. On this graph we compute new GRAPH-CLEAR strategies. The graph construction has been carried out for varying sensing range. A summary of the results is found in table I, where r denotes the sensing range, n_0 the initial number of vertices, n the number of vertices in the final graph. The number of robots needed to execute the computed strategies are ag_0 for the initial graph and ag for the final graph. The number of non-MST edges, each corresponding to a cycle in the graph, as well as their total weight, is also given as b and b_c respectively. In the graphs analyzed in [2] it turned out that to successfully apply the strategy from the tree to the graph we required about $0.5 \cdot b_c$ additional robots to what the tree strategy requires. Furthermore, we included the total area one can cover with the robots assuming they have an omnidirectional sensor. As c_1 we denote the area all robots needed to executed the tree strategy could cover as a percentage of the total area of the free space of the given map. For c_2 we also include all robots needed to block all non-MST edges continuously. These percentages would be reduced if we assumed a 180 degree sensor as we only need sensor coverage to maintain sweep lines, i.e. in theory even a single beam would already suffice, albeit in practice a 180 degree or omnidirectional sensor can give repeated observations of the same target which is more robust considering the erroneous nature of the sensors and the need to integrate the observations to obtain robust target detections.

VII. Discussion and Conclusion

The experiments demonstrate a successful construction of a SG for two complicated environments. The number of robots needed is significantly reduced for the contracted variant of the graph. Furthermore, we can see that we can detect all intruders in the environment for just a minor fraction of the total area of the environment. More importantly, the approach scales well to large teams with each robot having only limited capabilities. The number of robots needed increases linearly w.r.t. to the sensing range. Another minor observation is that the number of non-MST edges in the SDR

Map	r	n_0	n	ag_0	ag	b	b_c	c_1	c_2
UCM	5	108	40	68	58	3	15	0.6%	0.7%
UCM	10	108	25	37	28	3	9	1.1%	1.5%
UCM	20	108	22	18	14	3	6	2.3%	3.2%
UCM	40	108	13	10	8	3	3	5.1%	7.1%
UCM	60	108	19	9	6	3	3	8.7%	13.0%
UCM	100	108	13	6	4	3	3	16.1%	28.1%
SDR	5	172	79	42	36	8	31	0.4%	0.8%
SDR	10	172	72	22	19	7	17	0.9%	1.6%
SDR	20	172	60	12	9	7	10	1.7%	3.5%
SDR	40	172	42	7	6	8	8	4.4%	10.3%
SDR	60	172	32	6	5	6	6	8.3%	18.1%
SDR	100	172	14	6	4	6	6	18.3%	45.8%

TABLE I

SUMMARY OF THE EXPERIMENTAL RESULTS.

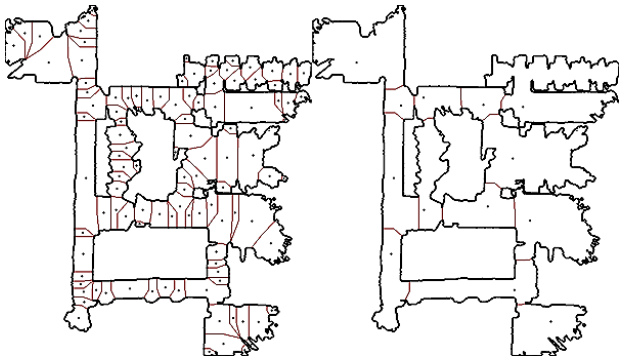

Fig. 7. The map created by the P3AT at UC Merced with initial graph construction on the left. The thick black lines are boundaries between free and occupied space. The small black points are vertices placed in their corresponding region which are separated by thin lines. On the right is the final graph resulting from contractions.

map varies, which is a result of the different contractions applied to the initial graph due to the different weights. Some of the cycles are then contained within a vertex and hence do not appear in the SG. We have hence demonstrate the applicability of GRAPH-CLEAR and provided a valuable method for the construction of Surveillance Graphs from maps which already works well in practice, despite leaving open many more directions for further improvements such as the Voronoi-based vertex clearing implementation which more accurately reflects the clearing costs of a vertex with limited range sensors. Furthermore, we showed that already simple criteria for contractions lead to significant improvements of the strategies. Also, the construction process is robust against errors in the approximation of the Voronoi Diagram as we did not use a state of the art algorithm for this purpose. Yet, contractions are by no means exhaustive and ideally a comprehensive theory of these contractions should be put into place. For all practical purposes, however, current contraction methods seem to suffice.There are open questions w.r.t. to an extension to a probabilistic variant, an incremental construction of SGs and strategies based on an incremental construction of a Voronoi Diagram and local optimization techniques. Such techniques could entail using a vertex sweep that finishes at an edge s.t. the edge does not have to be blocked during the process. The final steps

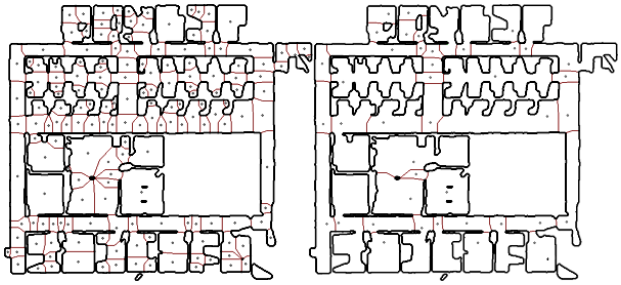

Fig. 8. The sdr_site_b from Radish [3] with initial graph construction. The thick black lines are boundaries between free and occupied space. The small black points are vertices placed in their corresponding region which are separated by thin lines. On the right is final graph resulting from contractions.

towards a demonstration with real robots is to implement and demonstrate GRAPH-CLEAR strategies on a specific robotic platform. With the results presented here such a demonstration is now within reach.

REFERENCES

[1] A. Kolling and S. Carpin, "Multi-robot surveillance: an improved algorithm for the graph-clear problem," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 2360–2365.

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

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

[4] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal 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. Journal of Robotics Research*, vol. 23, no. 1, pp. 3–26, Jan. 2004.

[6] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Autonomous Robots*, vol. 12, pp. 231–255, 2002.

[7] S. D. Bopardikar, F. Bullo, and J. P. Hespanha, "Cooperative pursuit with sensing limitations," in *American Control Conference*, New York, July 2007, pp. 5394–5399.

[8] M. Moors, T. Röhling, and D. Schulz, "A probabilistic approach to coordinated multi-robot indoor surveillance," in *Proc IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2005, pp. 3447–3452.

[9] T. Parsons, "Pursuit-evasion in a graph," in *Theory and Application of Graphs*, Y. Alavi and D. R. Lick, Eds. Springer Berlin / Heidelberg, 1976, vol. 642, pp. 426–441.

[10] L. Barriere, P. Flocchini, P. Fraigniaud, and N. Santoro, "Capture of an intruder by mobile agents," in *SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*. New York, NY, USA: ACM Press, 2002, pp. 200–209.

[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] H. Choset, "Incremental construction of the generalized voronoi diagram, the generalized voronoi graph, and the hierarchical generalized voronoi graph," in *Proc. of the First CGC Workshop on Computational Geometry*, October 1997.

[13] R. Mahkovic and T. Slivnik, "Constructing the generalized local voronoi diagram from laser range scanner data," *IEEE Transactions on Man and Cybernetics, Part A*, vol. 30, no. 6, pp. 710–719, 2000.

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

[15] L. E. Moreno and D. Blanco, "Localization by voronoi diagrams correlation," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2001, pp. 4232–4237.

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