This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters

Automated Creation of Topological Maps in Unknown Environments Using a Swarm of Resource-Constrained Robots

Rattanachai Ramaithitima, Michael Whitzer, Subhrajit Bhattacharya and Vijay Kumar *

***Abstract*—We address the development of the theory and algorithms that can enable a swarm of inexpensive robots or mobile sensors to create topological maps of unknown environments without explicitly requiring metric information. Topological maps are sparse and faithful representations of environments, and are useful constructions for efficient navigation in GPS-denied environments. However, in the absence of prior knowledge of the environment, with extremely limited sensing capabilities, and in an environment without localization, the construction of a topological map is a challenging problem that has not received much attention in the literature. In this paper, we present the algorithmic tools for constructing a topological map of an unknown GPS-denied environments using a swarm of resource-constrained robots. This method involves covering the free space, and then constructing an approximate Generalized Voronoi Graph of the covered environment. We demonstrate the algorithm using ROS, first with simulations, and then with experiments using a combination of real and virtual (simulated) robots.**

***Index Terms*—Swarms, Sensor Networks, Mapping**

I. Introduction

TOPOLOGICAL map (also, a *topological graph*), by definition, is a roadmap [1], [2] which is a sparse representation of the configuration space capturing all its topological features. A topological map provides information for fast navigation in many human environments (urban/rural environments, homes, shops, office buildings), as well as hazardous environments (collapsed buildings, underground tunnels) and for time-critical applications like disaster response and search/rescue. In a resource-constrained environment and in the absence of global localization or metric information, a topological map not only provides a coarse topological representation of the environment, but also provides a roadmap for transporting physical robots, equipment or supplies from one point to another.

The topological map that is of particular interest to us is the Generalized Voronoi Graph (GVG, also called a Generalized Voronoi Diagram or a GVD), which is the locus of points in the configuration space which have more than one distinct "closest" points on the boundary of the configuration space. Given a configuration space, the algorithms for constructing a GVG are well-studied [3], [4], [5]. The practical motivation and applications of a GVG representation of an environment are diverse, including sensor based mapping [2], efficient motion planning [6], and computer graphics [7].

There has been very little work in literature on how small, resource-constrained sensors and processors can yield global information that is relevant to operation in large-scale environments. Topological representations that are robust to sensor and actuator noise, and have reliable local-to-global integrability properties, are an ideal choice when working with resource-constrained robots in GPS-denied environments. Existing works in the topological mapping literature ([2], [8], [9], [10]) require robot(s) equipped with sensors that yield metric information and are able to localize themselves in an inertial frame. However, in the scenario where robots are prone to failure, such as in hazardous environment, the swarms of resource-constrained robots would be preferred due to the higher rate of success. As a result, our approach focuses on using minimum sensing robots that yield topological information only to construct a coarse global representation of an environment. To the best of our knowledge, this is the first approach for constructing the physical representation of a topological map using GPS-denied swarms with limited sensing capability.

This paper, following the authors' previous work [11], considers the problem of building a topological map of an unknown 2-D environment using a swarm of limited sensing robots. Every robot is only equipped with a bearing sensor with limited range that allows a robot to detect the bearing of neighbors, and a touch sensor that allows a robot to perform obstacle avoidance. Additionally, robots are capable of communicating with their neighbors. As a result, robots do not have access to metric information and cannot localize themselves in any meaningful coordinate system. Our algorithm allows the swarm of robots to construct an approximate GVG of the environment. The graph must be approximate since robots lack the ability to make metric measurements. Further since all sensors are local, the graph must be constructed incrementally.

The essence of our approach is illustrated in Figure 1. The robot swarm enters a completely unknown area (Figure 1(a)). The robots navigate the environment, gather information, construct a simplicial complex representation (a Rips complex) [12] of the environment with sensor coverage, and attain a *hole-less* sensor coverage using all the available robots. Since

Manuscript received: August, 31, 2015; Revised December, 9, 2015; Accepted January, 10, 2016.

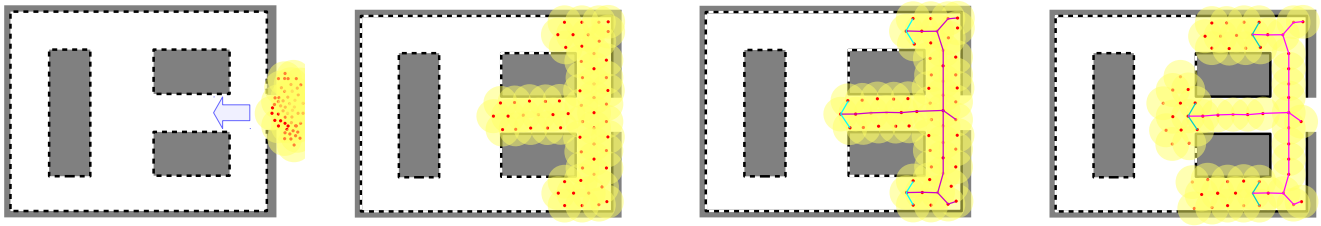
This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments. The authors gratefully acknowledge the support of AFOSR grant FA9550-10-1-0567, ONR grants N00014-07-1-0829, N00014-14-1-0510, and N00014-09-1-103

* All authors are with the GRASP Laboratory, University of Pennsylvania. [ramar, mwhitzer, kumar]@seas.upenn.edu subhrabh@sas.upenn.edu

Digital Object Identifier (DOI): see top of this page.

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters



(a) A finite swarm of robots enters an unknown environment.

(b) The swarm "floods" part of the environment and attains hole-less sensor coverage.

(c) A topological map of the partially covered environment (APGVG) is computed – magenta curve).

(d) Robots not constituting APGVG are re-deployed to the unexplored region and the process continues.

Fig. 1. **Overview:** Illustration of the main steps in construction of a physical representation of a topological map using a finite swarm of robots with limited local sensing.

there are a finite number of robots available, we can only ensure coverage of a subset of the environment (Figure 1(b)). Once the hole-less sensor coverage of the partial environment is attained, we run a discrete graph-based GVG construction algorithm (similar to [5]) on the 1-skeleton of the Rips complex formed by the robot sensor footprints to identify the robots that can be removed and the ones that need to be kept in order to retain the approximate physical representation of the GVG of the partially covered environment (which we call an *approximate physical/partial GVG* or an APGVG for simplicity – Figure 1(c)). The robots that are not tagged to be part of the APGVG can now be redeployed beyond the frontier to the unknown part of the environment to attain hole-less sensor coverage of a new portion of the environment, while the robots stationed on the APGVG of the previously covered environment maintain their position (Figure 1(d)).

The overall algorithm involves interleaving robot deployment cycles with the construction of the APGVG. Every new deployment cycle results in a new APGVG which must be stitched together with the old APGVG so that the stitched result is an approximate GVG of the entire environment. This process results in a sparse *topological map* for the free environment and is a deformation retract of the environment.

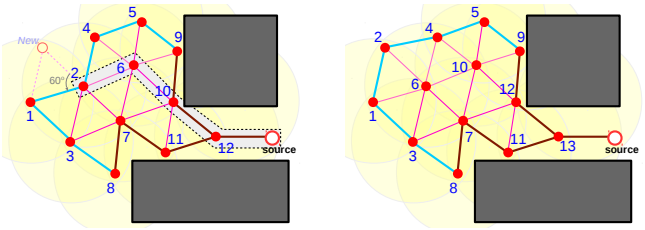
We call each of these cycles (involving robot deployment, hole-less sensor coverage of part of the environment, and computation of APGVG for identifying robots for deployment in next cycle) a *APGVG computation cycle* or a APGVGCC for simplicity.

Section II briefly describes the algorithm and control for deploying a robot swarm to attain a hole-less coverage of a part of the environment. The main contribution appears in Section III, where we describe the algorithm for computing the APGVGs and stitching them. In Section IV, we present simulation results and describe an experimental platform involving a heterogeneous team of real and virtual robots for demonstrating the proposed algorithm.

II. Preliminaries

For the sake of completeness of this paper, in this section we briefly describe the algorithm for robot deployment and control for attaining hole-less sensor coverage of an environment, as described in [11].

We assume that each robot has a disk-shaped sensor footprint of radius r_v, and a robot can see (and identify) another



(a) Partially covered environment, with new location for deployment identified in local coordinates of vertex '2' in $\mathcal{F}$. Shortest path $12 \rightarrow 10 \rightarrow 6 \rightarrow 2$ identified from a source to the vertex.

(b) Robots are "pushed" along the path. Notice how the new robot 13 appears near the source.

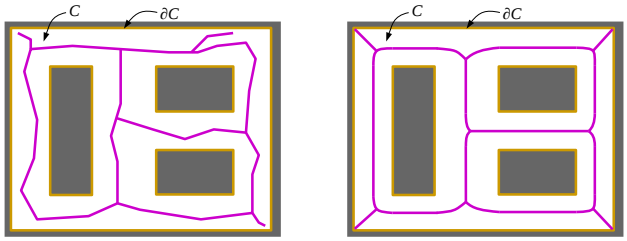
Fig. 2. The complex $\mathcal{R}_{r_v}$, and the subcomplexes $\mathcal{F}$ (cyan) and $\mathcal{O}$ (brown). Robots are transported along paths through the 1-skeleton of the complex using a "pushing" action in order to explore and cover regions beyond the frontier, while keeping the graph topology in the already covered region fixed.

robot, and measure the bearing to it in local coordinates (θ_i^j is the bearing of j in the local coordinates of i), only if the other robot is in its sensing disk. Each robot then sends the locally-sensed data to a server, where all centralized data is assimilated and decisions are made. We assume that the server can communicate directly with all robots through wireless communication. However, the same performance can be achieved through multiple hop communication along the swarm. The overall algorithm involves a sequence of deployment cycles, each of which consists of identifying *frontier robots*, locations (in local coordinates of the frontier robots) for deployment of new robots, and thus executing a "push" action through the graph so as to deploy a new robot into the coverage network. The key components of the algorithm are described next:

i. An *abstract simplicial complex* [13] representation of the environment is computed in the server through local communication with the neighbors of each robot. Robot i determines its set of neighbors, N_i, and sends that information to the server, where a *Rips complex* [14], $\mathcal{R}_{r_v}$, is constructed: A 0-simplex corresponds to every deployed robot in the environment, a 1-simplex exists between two 0-simplices if the corresponding robots are in each other's disk of visibility, and a 2-simplex exists for every 3-tuple of robots that can all see each other. Since $\mathcal{R}_{r_v}$ is dependent on the set of robot configurations, $X = [x_1, x_2, \cdots, x_n]$, we sometimes refer to it as

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters



(a) A topological map which is an *approximate* GVG. (b) The exact GVG of the free space.

Fig. 3. Topological graphs of a simple subset of $\mathbb{R}^2$.

$\mathcal{R}_{r_v}(X)$.

ii. Using the constructed Rips complex, we identify the *frontier and obstacle subcomplexes* and the robots constituting them. The basic idea is that if all the 2-simplices attached to a 1-simplex lie on the *same side* of the 1-simplex, (which can be determined from local bearing measurements), the 1-simplex should be part of the boundary to the unexplored region or is adjacent to an obstacle. A few pathological exceptions to this general rule also need to be considered. Such simplices, in general, constitute the *fence subcomplex*, $\mathcal{K} \subseteq \mathcal{R}_{r_v}$. Using this subcomplex and readings from the touch sensors, we perform additional checks to determine whether the *fence simplices* are a frontier to the uncovered region, or adjacent to an obstacle. These respectively constitute the *frontier subcomplex*, $\mathcal{F}$, and the *obstacle subcomplex*, $\mathcal{O}$, such that $\mathcal{K} = \mathcal{F} \cup \mathcal{O}$. For example, in Figure 2(a) we identify $\{8,3\}, \{3,1\}, \{1,2\}, \cdots \in \mathcal{F}$ and $\{7,8\}, \{7,11\} \in \mathcal{O}$.

iii. We perform a *breadth-first search* from the source to a frontier robot, and thus identify a *path* in the 1-skeleton of the complex from the source to a frontier robot (Figure 2(a)).

iv. The frontier robot at the end of the path identifies (in its local coordinates) the new location beyond the frontier for robot placement (Figure 2(a)). This is done so as to try and achieve a *hexagonal packaging* of the robots as much as possible, only to be interrupted by obstacles.

v. Robots are then transported using a *"push"* strategy along the path. In the absence of global localization or long-range sensing, the robot needs to control their movements using only the bearing to its neighbors. To that end, we use the bearing-based *visual homing controller* presented in [15]. The basic idea involves selecting the adjacent robots and use them as "landmarks" for bearing-based navigation (Figure 2(b)).

The algorithm comes with guarantees of hole-less coverage, stability (no indefinite switching behavior) and termination with complete coverage (in the presence of a sufficient number of robots).

III. Multi-stage Construction and Stitching of APGVGs

A. Generalized Voronoi Graph and its Approximate Discrete Construction

The topological map of a configuration space, C, is a 1-dimensional subspace such that it is topologically equivalent to C. In particular, the topology of the topological map captures all the "*holes*" in the space C as *loops* in the topological map (Figure 3(a)). If C is a subset of $\mathbb{R}^2$ (say, part of the plane with obstacles in it), the topological map can be more precisely described as a 1-dimensional *deformation retract* of the space, and can always be constructed [4], [16], [13]. The GVG is a special topological map with the property that each point on the GVG has two or more equidistant *closest points* on the obstacle boundary of the configuration space, ∂C (Figure 3(b)).

The 1-skeleton of the simplicial complex, $\mathcal{R}_{r_v}(X)$, formed by the robots as described in the previous section, can be considered as an approximate discrete graph representation of the covered subset of the workspace, with the robots being the vertices of the graph and the 1-simplices being the edges. Let's call this graph $\mathcal{G}_{r_v}(X)$ (or simply $\mathcal{G}_{r_v}$ for simplicity) for the set of robot positions, X. One can employ a *wave-front propagation* algorithm in $\mathcal{G}_{r_v}(X)$, as described in [5], [17], for identifying the vertices (the robots) in the graph which constitute an approximate GVG.

The overall idea is not very different from the continuous gradient flow method for the Computation of GVGs [4] – we employ a breadth-first search (Dijkstra's algorithm) with the initial *open list* containing all the vertices adjacent to the obstacles. Out of those initial vertices adjacent to obstacles, we mark the ones lying between two "*concave corners*" of an obstacle with the same label, so that the part of the wavefront originating from the vertices with the same label sweep a Voronoi cell of the approximate GVG. By the virtue of the Dijkstra's algorithm, the property of the wavefront is that at every instant all points on it are at an equal shortest distance from the closest obstacle. Thus, wherever the wavefronts with different labels *collide*, it ought to be a point on the GVG. The overall process is illustrated in Figure 5, and the pseudocode of the algorithm is provided later.

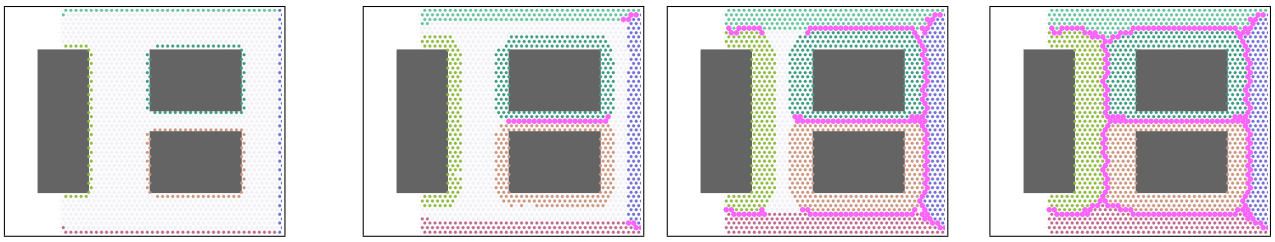
1) Segmentation of the Obstacle Subcomplex by Concave Corners: As described above, we need to segment the workspace boundary (boundary next to the obstacles) based on the presence of concave corners. However, since we do not have global knowledge of the environment, all that we can use to identify corners at the boundaries is the obstacle subcomplex, $\mathcal{O} \subseteq \mathcal{G}_{r_v}(X) \subseteq \mathcal{R}_{r_v}(X)$. This is achieved through communication between adjacent robots in the obstacle subcomplex. We choose the criteria on the angle at a corner in the environment to be γ, which, for example in environments with only right-angle corners will be $\frac{\pi}{2} + \epsilon$ (where the factor ϵ accounts for mismatch of the robot placement with the corners, and for all the simulations we choose $\epsilon = \frac{\pi}{4}$).

Suppose robots$\{i, j_1\}$ and $\{i, j_2\}$ are 1-simplices in $\mathcal{O}$, and let $\angle j_1 i j_2$ be the angle made by the 1-simplices at i (which, in the local frame of i, is the relative bearing between j_1 and j_2 as seen by i) on the side opposite to the obstacles. We identify i as a concave corner if $\angle j_1 i j_2 < \gamma$ (Figure 4(b)).

However, under some circumstances (as shown in Figure 4(c), where we mark ij_2 and $j_2 j_3$ as obstacle 1-simplices due to algorithm in [11]), it is not sufficient to consider only the angle made by two consecutive boundary 1-simplices at i. We also need to put some normality condition on angles made by the boundary 1-simplices at j_1 and j_2 in order to

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters

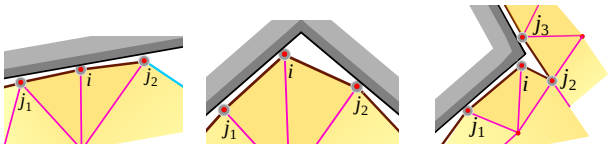


(a) The initial stage in the wavefront propagation, with different labels (marked by different colors) assigned to parts of the wavefront originating from segments of boundaries with no concave corner.

(b) The vertices at which the wavefronts with different labels (marked by different colors) "collide" constitute the approximate GVG (magenta disks) for the partially covered environment. Figures show the steps with 930 and 1430 vertices *expanded*.

(c) Final approximate APGVG shown in magenta.

Fig. 5. Illustration of the progress of wavefront algorithm for construction of APGVG using a discrete graph representation of the partially covered environment. The vertices of the graph are marked by the small disks and are representative of the physical robots (not to scale, and dense, ideal placement for illustration).



(a) Not a concave corner.

(b) A concave corner at i since $\angle j_1 i j_2 < \gamma$.

(c) Not a concave corner at i since $\angle i j_2 j_3$ does not satisfy normality condition.

Fig. 4. Detection of concave corners from the simplicial complex with $\gamma = \frac{\pi}{2} + \epsilon$.

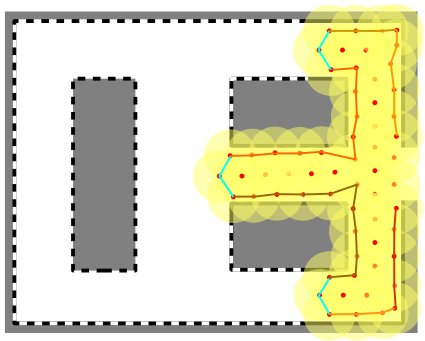


Fig. 6. Segmented obstacle subcomplex. The segments $\mathcal{O}_1, \mathcal{O}_2, \ldots$ are the curves in different hues of brown.

avoid too much spurious detection of concave corners. This method for corner detection is, nevertheless, mostly heuristic-based, given that we can only estimate the presence of the corners from relative bearings between the robots, and thus can lead to false positives or false negatives in concave corner detections. However, it is to be noted that the presence of concave corners only effects the "*branchiness*" of the Voronoi graph, and does not effect its more fundamental property of being a deformation retract.

Once we identify the concave corner robots on the obstacle subcomplex, we assign identical labels to all robots between two concave corners, while the corner robot is assigned either of the labels of the obstacle robots on its two sides. This gives a segmentation of the obstacle subcomplex, $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \cdots \cup \mathcal{O}_\sigma$, where $\mathcal{O}_m$ contains all the robots that are assigned label m (Figure 6). Note that $\mathcal{O}_m$ is itself a subgraph of $\mathcal{O}$ (which in turn is a subgraph of $\mathcal{G}_{r_v}(X)$), and we denote the vertex and edge sets of this subgraph by $V(\mathcal{O}_m)$ and $E(\mathcal{O}_m)$, respectively.

2) Wavefront Algorithm For Voronoi Graph Construction: We present the pseudocode, **GVG_Compute**, of the wavefront algorithm for computing the GVG in a discrete graph setting as described earlier (also see Figure 5). The basic framework of the algorithm is that of Dijkstra's [18]. The algorithm takes as input the 1-skeleton of the Rips complex – the graph $\mathcal{G}_{r_v}$, and the segmented boundary subgraphs, $\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_\sigma$. We assume that the *cost* of each edge in $\mathcal{G}_{r_v}$ is 1 (*i.e.*, "*distance*" is measured in hop count) since we do not have the inter-robot distances available. The algorithm outputs the set of vertices in the graph $\mathcal{G}_{r_v}$ which will constitute the discrete approximate GVG in the graph.

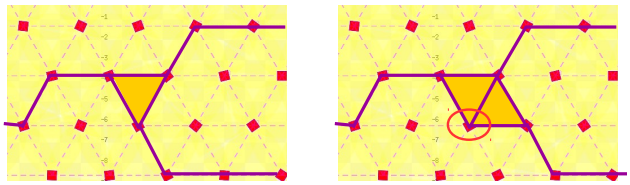
$\mathcal{V}$ = **GVG_Compute** $(\mathcal{G}_{r_v},\ \{\mathrm{O}_\alpha\}_{\alpha=1,2,\cdots,\sigma})$
Inputs: a. Graph $\mathcal{G}_{r_v}$, with vertex set $V(\mathcal{G}_{r_v})$ and edge set $E(\mathcal{G}_{r_v})$,
b. Segmented obstacle subgraphs, $\mathcal{O}_\alpha \subseteq \mathcal{O}$
Outputs: a. Vertex set constituting APGVG, $\mathcal{V} \subseteq V(\mathcal{G}_{r_v})$

```
1  Set g(v) := ∞, for all v ∈ V(G_{r_v}) // Distances to obstacle
2  Set l(v) := −1, for all v ∈ V(G_{r_v}) // Labels
3  Set V := ∅
4  for each k ∈ {1, 2, ··· , σ}
5      for each v ∈ V(O_k)
6          Set g(v) := 0
7          Set l(v) := k // Assign label k to vertices in O_k
8  Set Q := V(G_{r_v}) // Set of un-expanded vertices
9  while (Q ≠ ∅)
10     q := argmin_{q'∈Q} g(q') // Maintained by a heap
11     if (g(q) == ∞) // Open list is empty
12         break
13     Set Q := Q − q // Remove q from Q
       // Look at expanded neighbors with a different label
14     u := argmin_{u'∈N_{G_{r_v}}(q)} {g(u') | u' ∉ Q, l(u') ≠ l(q)}
15     if (g(u)+1 == g(q)  OR  g(u) == g(q))
16         Insert q into V // It's a GVG vertex!
       // un-expanded neighbors that need updating:
17     W := {w ∈ N_{G_{r_v}}(q) | w ∈ Q, g(w) > g(q)+1}
18     for each w ∈ W
19         Set g(w) := g(q) + 1 // Update to lower g-value
20         Set l(w) := l(q) // Copy label to neighbor
21 return V
```

Pseudocode for **GVG_Compute**

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters



(a) None of the robots in the marked GVG can be removed. (b) Robot marked by red ellipse in the GVG is redundant.

Fig. 7. Identifying robots that can be removed from the GVG computed by **GVG_Compute**.

We initiate the *open list* in the search algorithm with all the vertices in the obstacle subcomplex, set their g-value to zero (*i.e.*, they are at a distance of zero from the obstacle subcomplex) and attach a label to them based on the segmentation of $\mathcal{O}$ (lines 4-7). The rest of the algorithm is the usual breadth-first search — at every iteration we choose the vertex, q, in the open list with smallest g-value (line 10), place it in the closed list (*i.e.*, "expand" the vertex – line 13) and update its *un-expanded* neighbors if they will have better g-values if reached via q (lines 17-20).

We determine whether the vertex q, which is being expanded, is part of the GVG by looking at its expanded neighbors that have a label different to q's label. Precisely, q is equidistant from two different segments of the obstacle subcomplex if its g-value would have been the same (or one more) had it been expanded via a differently labeled vertex, and hence placed in $\mathcal{V}$ (lines 14-16). However, this process may end up including some redundant vertices (and corresponding 2-simplices from $\mathcal{R}_v$) in the GVG, which are not essential in maintaining the connectivity of the GVG (Figure 7(b)). We remove such vertices from the GVG through a simple post-check of the number of 2 and 1-simplices connected to a vertex belonging to the GVG. In particular, if a vertex in the GVG is connected to n edges (1-simplices) that form part of the GVG, which in turn form boundary of at least $n-1$ counts of 2-simplices, and also form part of the GVG, then the vertex is redundant in the GVG and can be removed from it (an explicit deformation retract can be constructed).

B. Robot Redeployment and Stitching the APGVGs

As illustrated in Section I, we construct the partial GVGs in the discrete setting (the APGVGs) at every APGVGCC.

For easy reference, for the i^{th} APGVGCC we will use superscripts of i to indicate the different objects described so far (*e.g.*, X^i will be the set of robot positions constituting the hole-less coverage of the partial environment at the i^{th} APGVGCC, $\mathcal{R}^i_{r_v}$ its Rips complex, $\mathcal{G}^i_{r_v}$ its 1-skeleton, $\mathcal{F}^i$ the frontier subcomplex, APGVG^i the GVG computed on $\mathcal{G}^i_{r_v}$ using segmented obstacle subcomplex $\mathcal{O}^i = \mathcal{O}^i_1 \cup \mathcal{O}^i_2 \cup \cdots \cup \mathcal{O}^i_{\sigma^i}$, etc.). Thus, we have APGVG^i = **GVG_Compute**$(\mathcal{G}^i_{r_v}, \{\mathrm{O}^i_\alpha\}_{\alpha=1,2,\cdots,\sigma^i})$.

By the virtue of its construction, APGVG^i and APGVG^{i+1} will be connected to $\mathcal{F}^i$ (Figure 8(a)). Following the computation of APGVG^{i+1}, we consider each connected component of $\mathcal{F}^i$, and identify the subcomplex, $\mathcal{S}^i$, necessary to keep the branches of APGVG^i and APGVG^{i+1} emanating from that

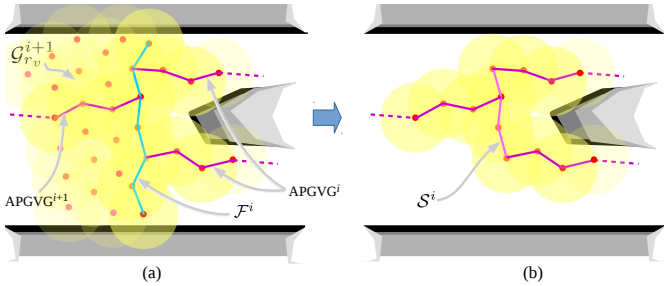


Fig. 8. Stitching APGVG^i and APGVG^{i+1} by considering each connected component of $\mathcal{F}^i$. The set of robots re-deployed/removed in going from (a) to (b) is $\Lambda^{i+1} = \mathcal{G}^{i+1}_{r_v} - (\text{APGVG}^{i+1} \cup \mathcal{F}^{i+1} \cup \mathcal{S}^i)$

component of $\mathcal{F}^i$ connected to each other (this essentially boils down to eliminating the robots at the trailing ends of the connected component of $\mathcal{F}^i$ – Figure 8(b)). We exclude the robots in $\mathcal{S}^i$ from the set of robots, Λ^{i+1}, for re-deployment in the $(i+1)^{th}$ APGVGCC (Figure 8(b)).

Thus we identify the set of robots that can be redeployed for the APGVGCC^{i+1} as $\Lambda^{i+1} = \mathcal{G}^{i+1}_{r_v} - (\text{APGVG}^{i+1} \cup \mathcal{F}^{i+1} \cup \mathcal{S}^i)$ (*i.e.*, we leave the robots on the partial GVG just computed, as well as those on the current frontier subcomplex and the subcomplex of the past frontier, $\mathcal{S}^i$).

In general, we use the "push" strategy through the 1-skeleton as described in Section II and [11] for finding paths to transport the robots one at a time to explore new locations outside the frontier $\mathcal{F}^{i+1}$. We then use the local bearing-based control described in [15] to move the robots. However, in some cases the path from a re-deployable robot to a frontier may contain parts of the approximate GVG which are not surrounded by neighboring robots any more. The "push" strategy does not work well under such circumstances due to lack of a sufficient number of landmark robots for the bearing-based controller. For that we need to use a separate controller, which we are in the process of implementing, that physically navigates the re-deployable robot along the single-robot chain constituting the GVG. In the simulations presented in this paper such a situation does not arise since we use sufficient number of robots in the swarm.

The overall algorithm for the multi-stage approximate GVG construction can thus be summarized as follows:

Algorithm: Multi-stage approximate GVG construction using a finite robot swarm

1 Set Λ^0 := the set of all robots
2 Set $\mathcal{F}^0$:= the initial frontier at the entrance
3 Set $i := 0$
4 **while** $\mathcal{F}^i \neq \emptyset$ **and** $\Lambda^i \neq \emptyset$ // APGVGCC^{i+1}
5 - Deploy robots in Λ^i to unexplored region outside $\mathcal{F}^i$ for hole-less coverage and construct the final $\mathcal{R}^{i+1}_{r_v}$ (which includes $\mathcal{F}^i$), with its 1-skeleton $\mathcal{G}^{i+1}_{r_v}$.
6 - Compute the obstacle and frontier subcomplexes, $\mathcal{O}^{i+1}, \mathcal{F}^{i+1} \subseteq \mathcal{R}^{i+1}_{r_v}$
7 - APGVG^{i+1} = **GVG_Compute**$(\mathcal{G}^{i+1}_{r_v}, \{\mathrm{O}^{i+1}_\alpha\}_{\alpha=1,\cdots,\sigma^{i+1}})$
8 - Identify robots $\mathcal{S}^i \subseteq \mathcal{F}^i$ to keep for proper stitching of APGVG^i and APGVG^{i+1}.
9 - $\Lambda^{i+1} = \mathcal{G}^i_{r_v} - (\text{APGVG}^{i+1} \cup \mathcal{F}^{i+1} \cup \mathcal{S}^i)$
10 Set $i := i+1$

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters

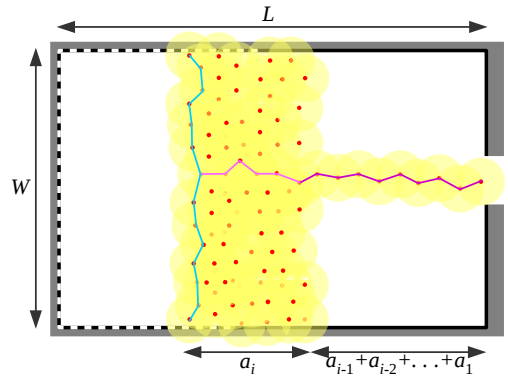


Fig. 9. Illustration of the number of robots required for an obstacle-free environment with $\beta = 1$.

C. Estimation of the Number of Robots Required

In general, the number of robots required for being able to completely construct the approximate GVG is highly dependent on the precise structure of the environment. However, under some assumptions of the environment, a very rough and informal estimate can be worked out. In this section, we provide an extremely simplified estimate of the number of robots that will be required for constructing the complete approximate GVG using the algorithm described in this paper. We consider the dimensions of the environment described in terms of number of average robot separations along X or Y directions (which, we approximately assume to be uniform, and equal to κr_v for some constant κ). Suppose the width of an environment (the dimension orthogonal to the main flow direction of the robots) is W times the average robot separation, and the length (dimension in the direction of robot flow) be L times the average robot separation (Figure 9 shows the obstacle-free case). Furthermore, in the presence of obstacles, let the average number of "branches" that the final GVG will have in the vertical direction be β.

Say we start with $n = n_1$ robots. In APGVGCC[1], those robots will be able to progress a distance of $a_1 = \frac{n_1}{W}$ average robot separations along the width of the environment. This will also be equal to the approximate number of robots that will constitute APGVG[1] with βa_1 robots. Thus, the remaining robots, $n_2 \simeq n_1 - \beta a_1 = n_1 \frac{W-\beta}{W}$ can be deployed for APGVGCC[2]. In general, using an inductive argument, at the beginning of the i^{th} APGVGCC, the number of robots available will be $n_k \simeq n_{k-1}\rho \simeq n_1 \rho^{k-1}$, where $\rho = \frac{W-\beta}{W}$. However, if the algorithm terminates at the m^{th} APGVGCC, we should at least have the final free robots span the entire width of the environment (so that there are enough robots to have the complete obstacle subcomplex and empty frontier subcomplex, for being able to compute the final APGVG effectively). This gives us

$$n_1 \rho^{m-1} \simeq W$$

Furthermore, we should be able to span the entire length of the environment using the APGVGs of length $a_1, a_2, \cdots, a_{m-1}$. Thus,

$$\frac{1}{W}(n_1 + n_2 + \cdots + n_{m-1}) \simeq L$$
$$\Rightarrow \quad \frac{1-\rho^{m-1}}{1-\rho} \simeq \frac{LW}{n_1}$$

Combining the above equations, and eliminating m, one gets $n_1 \simeq W + \beta L$. Thus, this simplified estimate puts the required number of robots at $W + \beta L$.

In practice we would surely like to keep a margin for safety and have more robots than what is presented in this simple estimate. For instance, the estimation for the environment in Figure 10 is about 60 robots, while we used about 100 robots in the simulation.

IV. Results

A. Simulation Result

We implemented the proposed algorithm on the Robot Operating System (ROS) [19] platform with the kinematic robots and the on-board sensors simulated by Stage robot simulator [20]. All the APGVG related computations are performed and kept by the server node, while each robot sends the locally-sensed data to the server and listens to commands from the server. Obstacle avoidance is performed individually by each robot.

Figure 10 shows a simple environment with an entrance at the top. A team of 100 robots construct a topological map (an approximate GVG) using the proposed algorithm in four APGVGCCs, with a total of 259 deployment iterations, which is the approximate number of robots required to cover the entire environment.

Figure 11 shows a simulation in a more complex indoor environment (a part of the 4^{th} floor plan of the Levine building at the University of Pennsylvania). We construct the approximate GVG of the environment with a swarm of 270 robots that is not sufficient to *fill* the entire environment (Figure 11(a)). The experiment was completed in three APGVGCCs with 535 deployment iterations.

B. Experiment with Heterogeneous team of Live and Virtual Robots

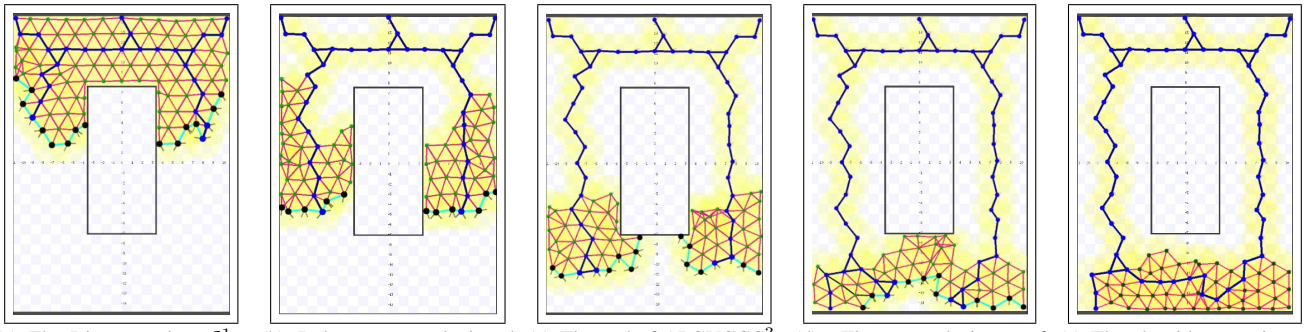
We also tested our algorithm on a real experimental platform. Because of limited number of available physical robots, as well as to demonstrate a new paradigm in combining physical robots with virtual/simulated ones in a real-time experiment, we use a heterogeneous team of virtual and real robots to constitute the swarm in the experimental setup.

We used Scarab robot [21] as the physical robot platforms for some of the robots, and used Stage robot simulator to simulate the remaining robots in the swarm. The Scarab is a differential drive robot, while the virtual robots consisted of holonomic robots simulated in Stage. Each real robot in the environment was coupled with a robot in the simulation environment (*virtual models of real robots*), besides the remaining robots in the swarm being simulated as well (*simulated robots*).

In order to make the real robot work coherently with all virtual robots, its corresponding simulated peer needs to

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters



(a) The Rips complex, $\mathcal{G}^1_{r_v}$, with frontier subcomplex, $\mathcal{F}^1$, marked in cyan and black, and the discrete GVG, APGVG^1, computed in $\mathcal{G}^1_{r_v}$ shown in blue.

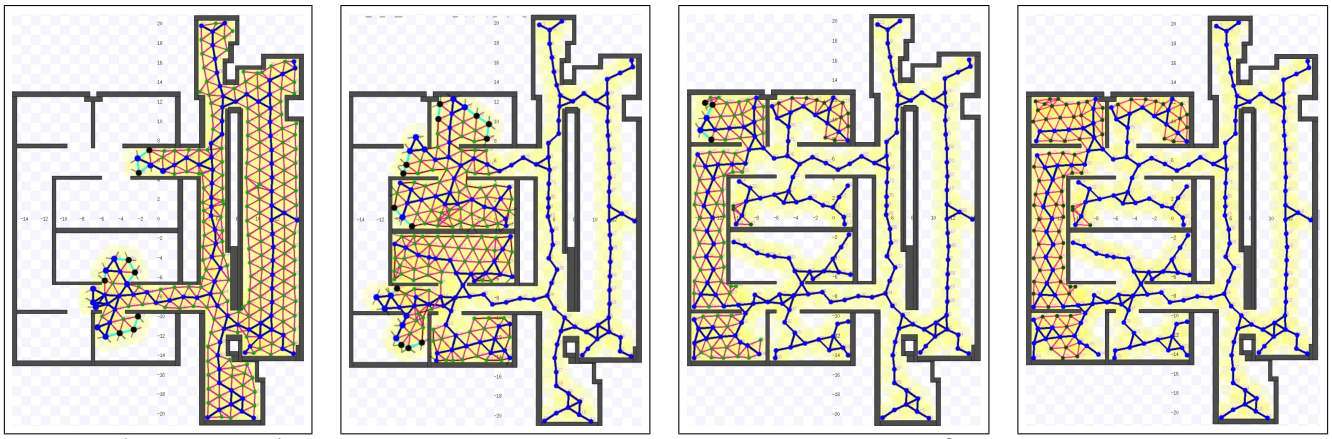
(b) Robots are re-deployed to construct $\mathcal{G}^2_{r_v}$. The new discrete Voronoi graph, APGVG^2, is also shown in blue, stitched with the earlier APGVG^1.

(c) The end of APGVGCC^3, showing the three subsequent APGVGs stitched together.

(d) The conclusion of APGVGCC^4.

(e) The algorithm terminates since there are no more unexplored frontiers ($\mathcal{F}^5 = \emptyset$).

Fig. 10. Demonstration of the proposed algorithm in ROS simulation using a simple environment.



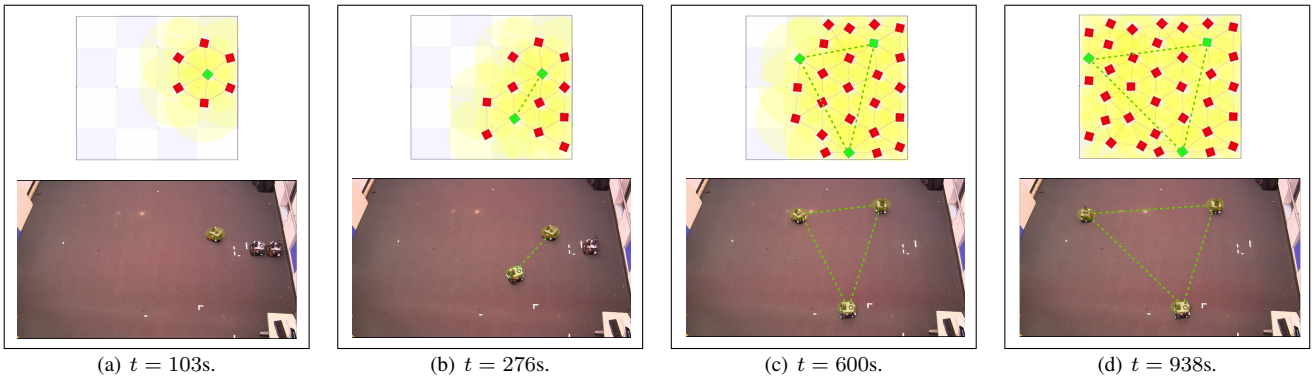
(a) APGVG^1, computed in $\mathcal{G}^1_{r_v}$.

(b) At the end of the computation of APGVG^2.

(c) The end of APGVGCC^3.

(d) Algorithm terminates since there are no more unexplored frontiers.

Fig. 11. ROS simulation in a more complex indoor environment. See the accompanying multimedia attachment for the video.



(a) $t = 103$s.

(b) $t = 276$s.

(c) $t = 600$s.

(d) $t = 938$s.

Fig. 12. A heterogeneous team of live (green) and virtual (red) robots covering a simple, obstacle-free environment. The dashed green lines are drawn for comparison between the formation of the live robots and the simulated version of the live robots. The accompanying multimedia attachment shows the video of the simulation environment overlaid on the experiment for better comparison.

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2016.2523600, IEEE Robotics and Automation Letters

be synchronized through the feedback loop as illustrated in Figure 13. Localization of the real robots was done through the use of a motion capture system. The motion capture system would broadcast the pose information of the physical robots in the environment. These poses were used to update the poses of the simulated versions of the real robots using a proportional position control on the pose. This is used in conjunction with the pose of the other simulated robots to emulate bearing sensors for all the robots (real as well as simulated). The real robots would then utilize this sensor data to compute and execute bearing-based visual homing control. In summary, a bearing-based visual homing control was implemented in a heterogeneous team of real and virtual robots.

As a proof of concept, an obstacle free environment was selected for exploration by the heterogeneous team. The results are illustrated in Figure 12. The top row shows the simulated environment, with virtual robots colored in red and the simulated version of the live robots colored in green. The snapshots of the experiment in the lower row shows the live robots. The heterogeneous team is able to explore the environment with the limited sensing and communication capabilities. Testing in more complex environments and construction of the GVG is within the scope of future work.

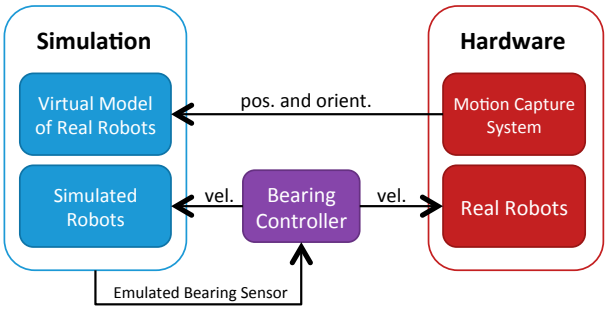


Fig. 13. Block digram describing the communication and feedback between the simulation platform, experimental (real) robots and the motion capture system.

Videos of the simulation and the experiment can be found in the accompanying multimedia attachment or at http://mrsl.grasp.upenn.edu/~tee/APGVG/.

V. Conclusion

In this paper, we proposed the basic theory and algorithms that allow a swarm of resource-constrained robots to automatically create a topological map, specifically a Generalized Voronoi Graph, of indoor environments without requiring metric information. This method involves covering part of the free space of an environment prior to constructing a Generalized Voronoi Graph from the covered space. The excess robots are then used to extend the covered space and further construct a GVG of the environment until a full topological representation is completed. We demonstrated the proposed algorithm in a ROS-Stage simulation, as well as introduced a new paradigm in experimental demonstration involving a heterogeneous team of real and virtual robots.

As part of future research we will demonstrate how the approximate GVG construction using physical robots, as described in this paper, can be used for fast and efficient transportation of other robots and resources from one region to another in unknown, GPS-denied environments.

References

[1] E. Rimon. Construction of c-space roadmaps from local sensory data. what should the sensors look for? *Algorithmica*, 17(4):357–379, 1997.

[2] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, 1996.

[3] D. Avis and B. K. Bhattacharya. Algorithms for computing d-dimensional Voronoi diagrams and their duals. *Advances in Computing Research*, 1:159–180, 1983.

[4] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick. Sensor-based exploration: Incremental construction of the hierarchical Generalized Voronoi Graph. *The International Journal of Robotics Research*, 19(2):126–148, 2000.

[5] E.G. Tsardoulias, A.T. Serafi, M.N. Panourgia, A. Papazoglou, and L. Petrou. Construction of minimized topological graphs on occupancy grid maps based on GVD and sensor coverage information. *Journal of Intelligent and Robotic Systems*, 75(3-4):457–474, 2014.

[6] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. Path planning for mobile robot navigation using Voronoi diagram and fast marching. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2376–2381, Oct 2006.

[7] R. Nascimento and T. Lewiner. Streamline-based topological graph construction with application to self-animated images. In *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on*, pages 296–303, Aug 2013.

[8] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (Vol. 2)*, AAAI'94, pages 979–984, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[9] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4845–4851 Vol.5, April 2004.

[10] J. Kim, F. Zhang, and M. Egerstedt. A provably complete exploration strategy by constructing voronoi diagrams. *Autonomous Robots*, 29(3-4):367–380, 2010.

[11] R. Ramaithitima, M. Whitzer, S. Bhattacharya, and V. Kumar. Sensor coverage robot swarms using local sensing without metric information. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[12] R. Ghrist, editor. *Elementary Applied Topology*. CreateSpace Independent Publishing Platform, 2014.

[13] A. Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001.

[14] V. de Silva and R. Ghrist. Coordinate-free coverage in sensor networks with controlled boundaries via homology. *The International Journal of Robotics Research*, 25(12):1205–1222, 2006.

[15] R. Tron and K. Daniilidis. Technical report on Optimization-Based Bearing-Only Visual Homing with Applications to a 2-D Unicycle Model. *ArXiv e-prints*, February 2014.

[16] H. Choset. *Sensor based motion planning: The hierarchical generalized Voronoi graph*. PhD thesis, California Institute of Technology, 1996.

[17] S. Bhattacharya, R. Ghrist, and V. Kumar. Multi-robot coverage and exploration on riemannian manifolds with boundary. *International Journal of Robotics Research*, 33(1):113–137, January 2014. DOI: 10.1177/0278364913507324.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2nd edition, 2001.

[19] 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 Robot. and Autom.*, Kobe, Japan, May 2009.

[20] The stage robot simulator. http://rtv.github.io/Stage/.

[21] N. Michael, J. Fink, and V. Kumar. Experimental testbed for large multi-robot teams: Verification and validation. *IEEE Robot. Autom. Mag.*, 15(1):53–61, March 2008.

2377-3766 (c) 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.