# Near-Optimal Area-Coverage Path Planning of Energy-Constrained Aerial Robots With Application in Autonomous Environmental Monitoring

Katharin R. Jensen-Nau, Tucker Hermans, *Member, IEEE*, and Kam K. Leang, *Senior Member, IEEE*

***Abstract*— This article describes a Voronoi-based path generation (VPG) algorithm for an energy-constrained mobile robot, such as an unmanned aerial vehicle (UAV). The algorithm solves a variation of the coverage path-planning problem where complete coverage of an area is not possible due to path-length limits caused by energy constraints on the robot. The algorithm works by modeling the path as a connected network of mass-spring-damper systems. The approach further leverages the properties of Voronoi diagrams to generate a potential field to move path waypoints to near-optimal configurations while maintaining path-length constraints. Simulation and physical experiments on an aerial vehicle are described. Simulated runtimes show linear-time complexity with respect to the number of path waypoints. Tests in variously shaped areas demonstrate that the method can generate paths in both convex and nonconvex areas. Comparison tests with other path generation methods demonstrate that the VPG algorithm strikes a good balance between runtime and optimality, with significantly better runtime than direct optimization, lower cost coverage paths than a lawnmower-style coverage path, and moderately better performance in both metrics than the most conceptually similar method. Physical experiments demonstrate the applicability of the VPG method to a physical UAV, and comparisons between real-world results and simulations show that the costs of the generated paths are within a few percent of each other, implying that analysis performed in simulation will hold for real-world application, assuming that the robot is capable of closely following the path and a good energy model is available.**

***Note to Practitioners*—For autonomous mobile-robotics-based applications where a robot equipped with a tool or sensor is required to survey an area for inspection, monitoring, cleaning, and so on, effectively covering the area is desirable. However, for energy-constrained systems such as aerial vehicles with limited flight time, complete coverage is not possible. Presented here is a new Voronoi-based path generation algorithm that takes energy constraints into account to generate waypoints for the robot to follow in a near-optimal configuration while maintaining path-length constraints. The approach is applied in simulation and experiments for an application in environmental monitoring using unmanned aerial vehicles.**

***Index Terms*— Coverage path planning (CPP), mobile robot, Voronoi-based path generation (VPG).**

## I. Introduction

COVERAGE path planning (CPP) is the process of planning a path that covers all points in an area [1] and has application to robotics where a tool or sensor is displaced or distributed to survey an area for inspection, monitoring, cleaning, and so on. The research and application of CPP is extensive, including uses for agriculture [2]–[4], structure inspection [5], [6], floor cleaning [7], [8], terrain surveying [9], [10], environmental monitoring [11] (see Fig. 1), lawn mowing [12], demining [13], and industrial processes such as painting or sanding [14]. The vast majority of prior works assume that the robot will have enough available energy to cover the entire area. Only recently have energy usage and potential energy limitations been considered when planning coverage paths [15]–[19]. Limited energy may lead to partial area coverage; however, partial coverage can still provide useful information if the planning process is executed appropriately. Thus, energy-constrained CPP is an increasingly important consideration with the proliferation of robots such as multirotor unmanned aerial vehicles (UAVs) [20], which tend to have short battery life, often less than 20 min when carrying a sensor payload and computational hardware for survey, inspection, and monitoring applications [11].

This article presents a Voronoi-based path generation (VPG) algorithm that solves a variation of the CPP problem where complete coverage of an area is not possible due to energy constraints on the robot which limit the total path length. The algorithm distributes path waypoints to achieve near-optimal coverage given the path-length limitations, where the optimal path is defined as one that gets as close as possible to every point in the area, and a near-optimal path is one that is close to the optimal path but not close enough to be considered

Manuscript received January 21, 2020; revised April 29, 2020; accepted June 15, 2020. This article was recommended for publication by Associate Editor S. Rathinam and Editor K. Saitou upon evaluation of the reviewers' comments. This work was supported in part by the University of Utah, in part by the National Science Foundation's Partnership for Innovation Program under Grant 1430328, and in part by the U.S. Army STTR Program under Grant W9132T-16-C-0001. *(Corresponding author: Kam K. Leang.)*

Katharin R. Jensen-Nau and Kam K. Leang are with the Design, Automation, Robotics and Control (DARC) Laboratory, Department of Mechanical Engineering, University of Utah, Salt Lake City, UT 84112 USA, and also with the Robotics Center, University of Utah, Salt Lake City, UT 84112 USA (e-mail: kam.k.leang@utah.edu).

Tucker Hermans is with the School of Computing, University of Utah, Salt Lake City, UT 84112 USA, and also with the Robotics Center, University of Utah, Salt Lake City, UT 84112 USA.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2020.3016276

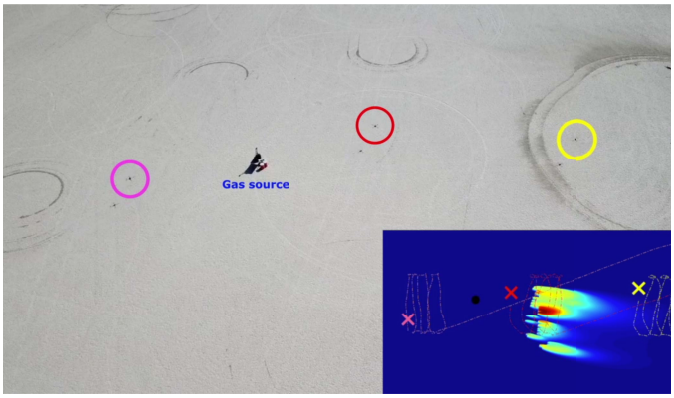


Fig. 1. Three quadcopter aerial robots with chemical sensors scanning an area to generate chemical concentration maps [11]. In this case, the source is a simulated propane leak. CPP methods are important in such applications to optimize the mapping process when robot flight time (energy) is limited.

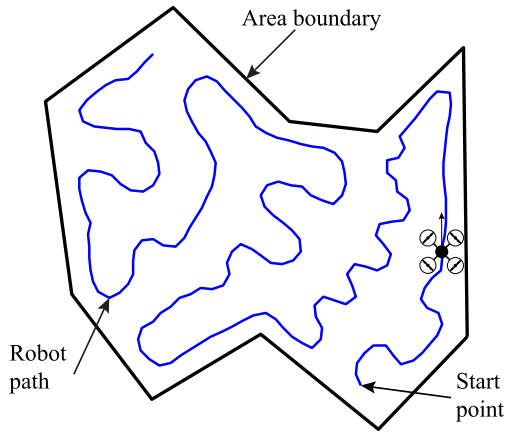


Fig. 2. Example of a near-optimal partial coverage path created by the VPG algorithm given a path-length constraint based on an energy model of a mobile robot.

truly optimal. Fig. 2 shows an example of one such path. The algorithm uses a model of the path as a connected network of mass-spring-damper systems, in combination with a Voronoi diagram, to generate a potential field that distributes the path waypoints into a near-optimal configuration. Such an algorithm has utility in applications, such as mapping chemical concentrations [11] over an area of interest while accounting for the highly limited battery capacity inherent in multirotor UAV systems. As illustrated by the gas concentration mapping example in Fig. 1, energy-constrained CPP is important to ensure that the robot traverses the terrain in an optimal manner while performing inspection/monitoring tasks. However, there is limited work on optimizing coverage paths for partial coverage of an area [6], [21].

The primary novelty of the proposed VPG method is the dynamic path model consisting of a chained mass-spring-damper system in combination with the use of Voronoi diagrams to generate a potential field, which distributes preconnected path waypoints. In this method, the waypoints are represented as masses, and the forces acting on them come from spring forces maintaining desired spacing between neighboring waypoints, spring forces pulling the waypoints toward the centroids of their Voronoi areas, and associated damping forces. The VPG method produces near-optimal coverage paths given path-length limitations due to energy constraints on the robot, and, unlike most CPP methods, the VPG method is designed to be able to easily account for path-length limitations when generating the path. The use of potential fields makes the method generalizable to both convex and nonconvex areas with no modifications required and allows the algorithm to generate its paths significantly faster than direct optimization.

In the literature, a similar method using a combination of Voronoi diagrams and potential fields is proposed by Han *et al.* [22] for the purpose of deploying a wireless sensor network (WSN). Since WSNs use large numbers of sensors to provide coverage of an area, their focus is on placing individual points representing sensors in their optimal locations. A way of connecting the points into a path is not necessary for them and is thus not considered, making their method as presented not easily applicable to generating coverage paths. Also, a path-planning method presented by Soltero *et al.* [23] uses distances between the neighboring waypoints and between waypoints and the centroids of their Voronoi areas for optimizing coverage paths. However, their method uses a cost function based on those distances and creates a gradient descent-based velocity controller to move the path points to their optimal locations. Their method considers a set number of waypoints with variable spacing between them, so while it does generate good paths in the absence of path-length constraints, it is difficult to impose those constraints on the paths it generates. It is pointed out that extensive work on trajectory optimization for various applications can be found in the literature (see [24], [25]). The proposed VPG method, in contrast, is designed to easily account for path-length constraints.

There are three main contributions of this work. First is the development of a Voronoi-based algorithm that helps solve the issue of path planning for an energy-limited mobile robotic sensor system by generating near-optimal length-constrained coverage paths. Second is the detailed simulation and characterization of the algorithm to study its performance and compare it with other path generation algorithms. Third are the physical experiments performed to validate the applicability of the algorithm on a real-world mobile robotic platform, such as the quadrotor UAV described in [11]. Overall, simulation results show that the algorithm has distinct advantages over other path generation methods in the partial coverage scenario, with good runtimes, scalability, and path costs. Also, results from physical experiments show that the algorithm can successfully be applied to a real-world mobile robotic platform and that the physical experimental results are in good agreement with simulation results.

## II. Related Prior Works

### A. Coverage

In general, the problem of area coverage is defined as getting a sensor or sensors to take measurements that cover an area or getting a tool or tools to perform a task covering an area.

This article specifically considers applications that involve area coverage with sensors. Methods for sensor coverage of an area can be grouped into two broad categories. One category is coverage using networks of many sensors that remain stationary, each taking measurements in one location. The other category, which is the focus of this article, is coverage using a single or a small number of mobile sensors that follow a path within an area taking measurements in multiple places.

*1) Coverage With Sensor Networks:* WSNs are networks of stationary sensors distributed over an area. Coverage is one of the main uses of WSNs, and how to best distribute the sensors is an important consideration [26], [27]. Algorithms for distributing WSNs generally try to place sensor nodes in a way that maximizes area coverage. One common method is the potential field approach. For example, a potential field is proposed in [28] that causes the sensors to repel each other and spread out over an area. In [29], three methods are presented that use Voronoi diagrams of nodes to find coverage holes and potential field methods to pull the nodes toward the coverage holes. The vector-based algorithm (or VEC for short) simply repels each node from its Voronoi neighbors, the Voronoi-based algorithm (VOR) pulls nodes to their farthest Voronoi vertex, and Minimax pulls nodes toward the point that minimizes the distance to their farthest Voronoi vertex. Likewise, in [22], a potential field pulls nodes to the centroids of their Voronoi areas while repelling them from neighboring nodes. The main effect of these various potential field methods is that the nodes spread out fairly evenly and approach uniform coverage of the area. Other recent work in this area focuses on bioinspired methods, such as the bee colony algorithm used in [30] or the immune algorithm described in [31]. While WSNs are not the focus of this article, some of their distribution methods can be useful for distributing waypoints in a path, as will be discussed further in Section II-B.

*2) Coverage Path Planning:* CPP encompasses all methods of having one or several mobile sensors cover an area by following a path. CPP methods for single robots have been studied extensively and can generally be divided into classes based on how they decompose the area [1]. One class, called grid-based decomposition, involves discretizing the coverage area into a grid and planning a path that visits each grid cell. The other common class, called cellular decomposition, uses an exact cell decomposition to divide the area into cells that can then be covered by a back and forth lawnmower-style pattern. The order in which to visit the cells is found by solving a traveling salesman problem on their adjacency graph.

CPP algorithms using grid-based decomposition include spiral spanning trees [32] [see Fig. 3(a) for an example], a wavefront algorithm based on distance transforms [33], and an adaption of the D* path-planning method [34]. Some work has also been done with different grid discretizations, including triangles [35] and fractal-based decompositions [36]. An optimization method is proposed in [9], which uses a depth-limited search and tries to minimize the turn angles along the path. Another optimization method using a genetic algorithm with path templates is given in [37]. Grid-based methods have the advantage that it is easy to represent grids and mark whether cells have been covered. However, they only give approximate decompositions of the area and can require prohibitively large amounts of memory for large areas or fine-resolution grids.

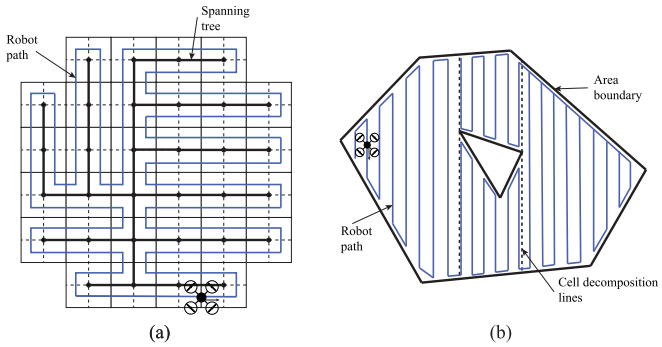


Fig. 3. Example coverage paths. (a) Coverage path that would be generated with the spiral spanning-tree method [32], which uses grid-based decomposition. The thin black lines make up the grid, the thick black line is the spanning tree, and the blue line is the coverage path. (b) Coverage path that would be generated with the boustrophedon decomposition method, which uses cellular decomposition [38]. The solid black lines are the area boundaries, the dashed lines are the cell decomposition lines, and the blue line is the coverage path.

Common cellular decomposition methods include, for example, boustrophedon decomposition [38] [see Fig. 3(b) for an example] and Morse-based decomposition [39]. Optimization of these methods tends to involve choosing the cell decomposition and path direction that minimize the number of turns that the covering robot must make [4], [40], [41]. Online methods that build the adjacency graphs incrementally have been proposed in [42] and [43]. A method is described in [44] for general 3-D surfaces. Compared with grid-based decomposition methods, cellular decomposition methods require significantly less memory, but with the tradeoff that covering individual cells becomes more complicated.

*3) Energy-Constrained Coverage:* While both grid-based and cellular decomposition CPP methods work well for complete area coverage, they do not consider the case where the robot is not capable of covering the entire area due to limited energy, which is of particular concern when path planning for UAVs [20]. The most common way to deal with energy constraints is to add more robots and partition the area such that no robot is assigned a larger area than it can cover [45], [46]. Multiagent CPP is a vast field of research in and of itself, but a few methods include partitioning the areas using Voronoi diagrams [47], [48], using multiple spanning trees [49], [50], and applying genetic algorithms [51], [52]. While some recent work has been done to optimize single-robot coverage paths with respect to energy consumption [15]–[19], most methods still require that the robot has enough power to completely traverse the path. Papachristos *et al.* [21] proposed a coverage method for inspecting multiple structures that assigns an importance weight to each structure and finds a coverage path that maximizes the inspection reward, allowing for some structures to be left uncovered if their UAV's flight time constraints do not allow full coverage. The CPP method presented in [6] for 3-D structure inspection allows for partial coverage if it would result in significant energy savings, using

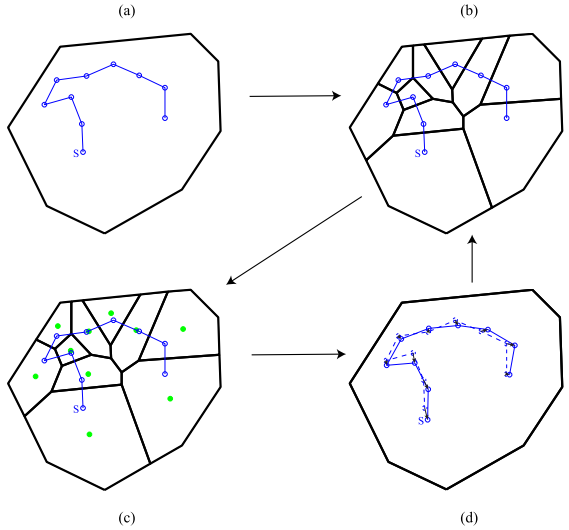


Fig. 4. Illustration of one iteration of the path generation process. (a) Path (blue line) defined by waypoints (blue dots), with the starting point marked with an "S." (b) Algorithm draws a bounded Voronoi diagram using the path waypoints. (c) Centroids of the Voronoi polygons (green dots) are found. (d) Waypoints are displaced, by forces pulling the waypoints toward the centroids and intrawaypoint forces, from the location marked by the dashed line to the location marked by the solid line. The process repeats until all waypoints converge to their final locations.

multiobjective optimization with respect to both coverage and energy consumption.

### *B. Point Distribution for Path Planning*

A path can be generated by defining it as a sequence of waypoints that the robot must visit in a specified order. One possible way of creating a partial coverage path could be by optimally distributing the points and then connecting them. The WSN deployment algorithms described in Section II-A1 could be used for waypoint distribution since they are designed to place the points in a configuration for optimal coverage. However, optimally connecting the points into a path can be difficult. Finding the shortest path to connect a collection of points is a well-studied combinatorial optimization problem called the traveling salesman problem. It is an NP-hard problem, and a large body of research exists in finding feasible solutions [53]. However, a simpler solution is to distribute points that are already connected to a path. This idea is used in [23], which begins with a preconnected coverage path. The method is designed for a robot to patrol a particular area of interest based on sensor knowledge of the area, representing how interesting different parts of the area are as a density function. The method tries to put path waypoints at the centers of mass of their Voronoi areas based on that density function while also reducing the distance between waypoints. It optimizes the path with a gradient descent-based controller using a cost function that combines the distances between waypoints and their Voronoi centroids and the distances between the neighboring waypoints and directly controlling the velocity.

### *C. Advantages Over Current Methods*

Unlike the CPP methods discussed in Section II-A2, the method proposed in this article treats the path as a dynamic system, allowing the path to essentially distribute itself to provide the best coverage it can to an area given its specified path length. The method is not dependent on the area geometry and does not require decomposing the area into cells or grids, eliminating the need to decide how to visit and cover those cells or grid spaces in an optimal way with a path-length constraint. Treating the path as a dynamical system and allowing it to self-distribute based on a potential field also makes it unnecessary to use time-consuming direct optimization methods. The VPG algorithm also does not require any additional adjustments or considerations to work in nonconvex areas of interest as well as in convex areas.

The most conceptually similar method to the proposed Voronoi-based path distribution algorithm is the method presented by Soltero *et al.* [23]. One primary difference between the two methods is in how the waypoints' movements are controlled. Unlike the Soltero method, the method presented in this article models the path as a chained mass-spring-damper system, with springs connecting waypoints to their neighbors and to the centroids of their Voronoi areas. The points are moved based on the spring forces and a viscous damping force. The main advantage of this force model over the gradient descent-based control model used in [23] is that it eliminates the need to take any area integrals, which the Soltero method requires due to the use of a nonuniform density function over the area. The VPG algorithm is intended to be used for mapping an area given no prior knowledge of the area and thus can use a simple equation for finding the centroid of a polygon instead of taking an integral to find the centroid. Numerical area integration is slow, so eliminating the need to do so speeds up computations considerably. In addition, the Voronoi-based path distribution method is designed to easily account for path-length limitations imposed due to energy constraints. Since the Soltero method does not consider energy constraints, the method has no need to consider how to control the path length. As a result, the only way to do so is to change weightings in the computation of the cost function, which must either be tuned for individual areas and path lengths or changed iteratively until the path-length constraint is met. Either method adds additional time and effort into applying this method to an energy-constrained problem, in which the VPG method avoids by being designed with energy constraints in mind.

## III. Path Generation Algorithm

The energy-constrained VPG algorithm is described next and applied to an aerial robot moving a sensor for measurement over an area of interest. In the most generic case, kinematic constraints on the motion of the robot may limit its ability to precisely follow a generated trajectory, and optimal control may be used to ensure that the path is one that the robot can follow. However, the kinematics of a quadrotor UAV allows several simplifying assumptions to be made. It is assumed that the path can be defined by a sequence of waypoints. The acceleration and velocity profiles for the robot when moving between waypoints are assumed to be consistent such that the same amount of energy is consumed traveling between two different pairs of points the same distance apart.

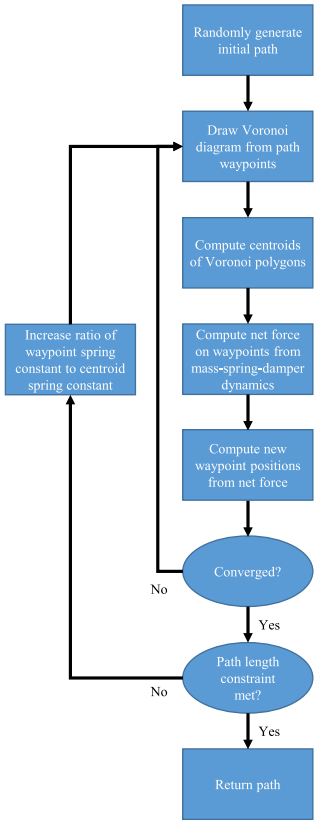


Fig. 5. Flowchart illustrating the high-level VPG algorithm process.

Rotations are assumed to use approximately the same amount of energy regardless of angle, and the robot is assumed to be able to turn sharply enough to closely follow the path. The number of waypoints on the path is then based on the maximum allowable energy usage, the desired distance between path waypoints, the amount of energy required to fly that distance in a straight line, and the average energy used when turning.

### A. Algorithm Overview

At a high level, the VPG algorithm works as follows, where Fig. 4 shows an example of one iteration process and Fig. 5 shows a flowchart of the path generation process.

1) Start with a randomly generated sequence of path waypoints where adjacent waypoints are separated by the desired spacing and the path does not cross itself.
2) Draw a Voronoi diagram based on the locations of the waypoints.
3) Compute the centroid of each Voronoi polygon.
4) Set the resultant force acting on each waypoint to the sum of spring forces attracting or repelling the waypoint to the desired distance from its adjacent waypoints, a spring force attracting the waypoint toward the centroid of its associated Voronoi area, and a viscous friction force.
5) Use the force from the previous step to compute new positions for each waypoint.
6) Repeat steps 2–5 until the path converges.
7) If the path is too long, increase the ratio of the spring constant maintaining waypoint spacing to the spring constant pulling waypoints toward the centroids and repeat steps 2–6.

### B. Determining Total Path Length and Number of Waypoints

The VPG algorithm requires the number of path waypoints as an input, which means that the number of waypoints must be determined prior to running the algorithm. The waypoints are all spaced a user-defined distance $d_w$ away from their adjacent waypoints in the path. If there are $n$ total waypoints, the path length is given by

$$l_{\text{total}} = (n-1)d_w. \tag{1}$$

The maximum possible path length depends entirely on the energy consumption of the particular robot being used. In general, computation of the path length requires a function $f_m(d, v)$ that gives the energy used to travel a distance $d$ with a velocity $v$, a function $f_r(t)$ that gives the energy consumed when resting for an amount of time $t$, and an average energy used when rotating, $e_t$. Assuming a consistent travel velocity $v_t$ and spacing $d_w$ between each pair of adjacent path waypoints, $e_m = f_m(d_w, v_t)$ will be a constant. The robot is assumed to stop at each waypoint for the same amount of time $t_r$, so $e_r = f_r(t_r)$ will also be a constant. If there are $n$ total waypoints, the energy $e_{\text{path}}$ consumed as the robot travels along the path is

$$e_{\text{path}} = n(e_t + e_r) + (n-1)e_m. \tag{2}$$

If the total energy capacity $e_{\text{tot}}$ is known, the number of possible waypoints can be found by setting $e_{\text{path}} \leq e_{\text{tot}}$ and solving for $n$, resulting in

$$n \leq \frac{e_{\text{tot}} + e_m}{e_m + e_r + e_t}. \tag{3}$$

The maximum possible number of waypoints will be the largest integer satisfying (3).

### C. Voronoi Diagram Generation

The first step in the main loop of the VPG algorithm is to create a Voronoi diagram based on the path waypoints. A Voronoi diagram is a method of dividing an area given a set of discrete points or sites, such that each site is associated with one subarea and all points in that site's subarea are closer to it than to any other site. Mathematically, given a set of sites $S = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n\}$, where $\mathbf{s}_i \in \mathbb{R}^2$, the Voronoi area $V_j$ associated with site $\mathbf{s}_j$ can be defined as

$$V_j = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x} - \mathbf{s}_j\| \leq \|\mathbf{x} - \mathbf{s}_i\|, \forall i \neq j\}. \tag{4}$$

The algorithm presented in this article generates a Voronoi diagram using the associated Delaunay triangulation. The Delaunay triangulation for a set of points $S$ connects them into nonoverlapping triangles such that the circumcircle of any triangle does not contain any of the other points. The circumcenters are the vertices of the Voronoi diagram, and the vertices of Voronoi polygon $V_j$ are the circumcenters of all Delaunay triangles that have point $\mathbf{s}_j$ as one of the vertices.

Because the VPG algorithm has a particular area of interest to cover, the Voronoi diagram must be bounded to that area.

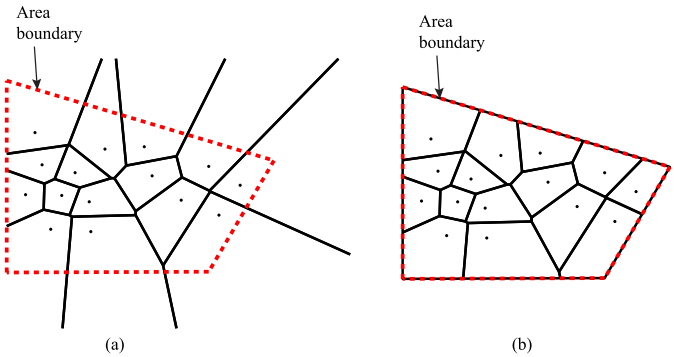


Fig. 6. Illustration of (a) unbounded Voronoi diagram and (b) bounded Voronoi diagram, where the red dashed line represents the area the Voronoi diagram is being bounded to.

This means that for each Voronoi polygon that extends beyond the area boundary, the vertices that lie outside the boundary are replaced with the Voronoi polygon's intersection points with the boundary edge and any boundary corners between those intersection points, as shown in Fig. 6.

### D. Optimality

The goal of the algorithm is to generate a coverage path that will best cover an area if only partial coverage can be achieved. For many coverage problems, this means attempting to maximize coverage. However, when considering path-length constraints, maximizing coverage only ensures that the path does not overlap itself. It does not consider that there may be two paths that achieve the same amount of coverage—by meeting the path-length constraint and not overlapping themselves—but still achieve different qualities of coverage. An example of this idea is shown in Fig. 7. If the sensor footprint is smaller than the smallest spacing between the lines of the path, both of these paths will cover the same amount of area, but with different quality of coverage. In Fig. 7(a), sensor measurements taken along the coverage path will provide very good information about the left side of the area but none about the right side. In Fig. 7(b), sensor measurements taken along the coverage path will provide moderately good information over the entire area. It is assumed that having moderately good information over the entire area is better than having very good information in one part of the area and no information in another part. To this end, a cost function is imposed that when minimized results in a path more similar to Fig. 7(b) than 7(a) since Fig. 7(b) gets closer to all points in the area than Fig. 7(a). Thus, given $R \subset \mathbb{R}^2$ as the area of interest, if $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, $\mathbf{p}_i \in R$ is a set of $n$ waypoints defining the path, the total cost is

$$J(P) = \iint_R \min_{\mathbf{p}_i} \|\mathbf{x} - \mathbf{p}_i\| dA \tag{5}$$

where $\mathbf{x}$ is a point represented by a vector $[x, y]^T \in R$, $dA$ is the differential area $dydx$, and $\|\cdot\|$ is the L2-norm. This expression sums the distances between every point $\mathbf{x} \in R$ and its closest waypoint. Applying cost $J(P)$, the problem becomes finding the optimal locations of all of the path waypoints $P^*$, where

$$P^* = \underset{P}{\operatorname{argmin}} \; J(P) \tag{6}$$

subject to the constraints

$$\|\mathbf{p}_j - \mathbf{p}_{j-1}\| = d_w, \quad j = 2, 3, \ldots, n \tag{7}$$

$$\|\mathbf{p}_{j+1} - \mathbf{p}_j\| = d_w, \quad j = 1, 2, \ldots, n-1. \tag{8}$$

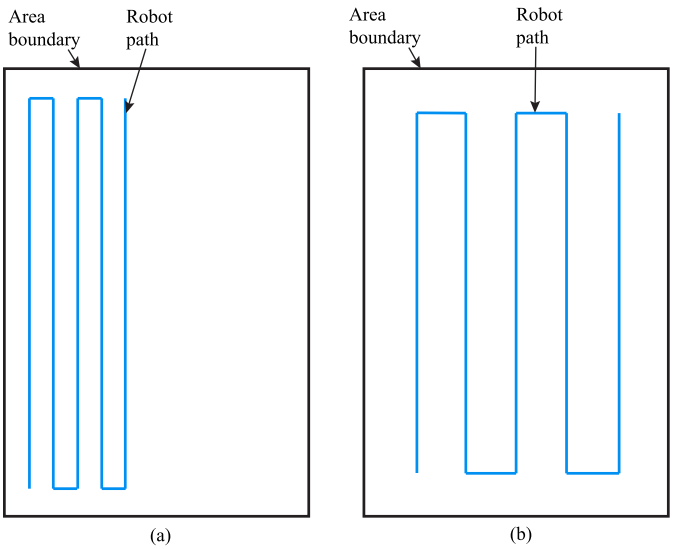


Fig. 7. Rationale behind the choice of cost function, if the goal is to get the best coverage of the entire area. (a) If sensor measurements are taken along the coverage path, there will be a lot of information about the right side of the area but absolutely no information about the left side. (b) Better coverage path of roughly the same length, where the path gets closer to all points in the area, and thus, taking sensor measurements along the path will give a moderate amount of information about the entire area.

The difficulty with optimizing this cost function is that it is expensive to compute and requires numerically computing an area integral, which in turn requires computing for every point in the area the inner minimization that finds its closest path waypoint. However, it is possible to leverage Voronoi diagrams to simplify the problem.

To begin, consider a single point $\mathbf{p}$ in a polygonal area. For this particular case, the location of $\mathbf{p}$ that minimizes the cost function is the centroid of the polygon, which can be computed using

$$c_x = \frac{1}{6A_p} \sum_{i=0}^{n_v-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{9}$$

$$c_y = \frac{1}{6A_p} \sum_{i=0}^{n_v-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{10}$$

where $c_x$ and $c_y$ are the $x$- and $y$-coordinates, respectively, of the centroid, $n_v$ is the number of polygon vertices, and $A_p$ is the signed area of the polygon, which is defined as

$$A_p = \frac{1}{2} \sum_{i=0}^{n_v-1} (x_i y_{i+1} - x_{i+1} y_i) \tag{11}$$

with $(x_0, y_0)$ being the same vertex as $(x_{n_v}, y_{n_v})$. The vertices are numbered in order around the polygon.

Next, consider a set of multiple unconnected waypoints. In this case, the cost function uses the distance between each point $\mathbf{x} \in R$ and its closest waypoint $\mathbf{p}_j$. If $V_j$ is the Voronoi area associated with $\mathbf{p}_j$, then by definition, $\mathbf{x} \in V_j$. Given this, the cost function can be rewritten as

$$J(P) = \sum_{i=1}^{n} \left( \iint_{V_i} \|\mathbf{x} - \mathbf{p}_i\| dA \right). \tag{12}$$

Minimizing each individual term of the sum will minimize the entire cost, and due to the nature of Voronoi diagrams, each individual term of the sum is simply the case of a single point in a polygonal area. For a set of unconnected points, if the centroid of Voronoi area $V_j$ is denoted $\mathbf{c}_{V_j}$, the optimal configuration for the set of waypoints thus becomes the one where $\mathbf{p}_j = \mathbf{c}_{V_j}$, $\forall \mathbf{p}_j \in P$.

For a sequence of connected waypoints, adjacent points in the path must be kept evenly spaced a distance $d_w$ apart, making each waypoint $\mathbf{p}_j$ subject to the constraints (7) and (8). The optimal configuration becomes the one that gets the closest to the optimal configuration for unconnected waypoints while maintaining the constraints. The optimization problem can then be rewritten as finding $P^*$ such that

$$P^* = \underset{P}{\operatorname{argmin}} \sum_{i=1}^{n} \|\mathbf{p}_i - \mathbf{c}_{V_i}\| \tag{13}$$

subject to the constraints (7) and (8).

### *E. Potential Field*

Directly optimizing the path based on the cost function and constraints from Section III-D requires numerically taking an area integral, which is a time-consuming process if the integral is to be computed with a small enough resolution for any reasonable accuracy. To avoid taking that integral and thus speed up the algorithm, a potential field method is used to move the connected path waypoints toward the optimal configuration described in Section III-D. To create the forces for the potential field, the path is modeled as a chained mass-spring-damper system. The locations of the masses are the waypoints, there are springs between the adjacent waypoints and their Voronoi centroids, and there is a viscous damping element between the waypoints and ground. Fig. 8 shows a small-scale system model. The force applied to each waypoint $\mathbf{p}_j$ is defined as

$$\mathbf{f}_j = \mathbf{f}_{p_{j-1}} + \mathbf{f}_{p_{j+1}} + \mathbf{f}_{c_j} + \mathbf{f}_{b_j} \tag{14}$$

where $\mathbf{f}_{c_j}$ is an attractive force toward $\mathbf{c}_{V_j}$, $\mathbf{f}_{b_j}$ is a viscous damping force, and $\mathbf{f}_{p_{j-1}}$ and $\mathbf{f}_{p_{j+1}}$ are the spring forces that try to maintain the constraints (7) and (8), respectively. A diagram showing these forces on the small-scale system model is shown in Fig. 9.

The forces are defined as follows:

$$\mathbf{f}_{c_j} = k_c(\mathbf{c}_{V_j} - \mathbf{p}_j) \tag{15}$$

$$\mathbf{f}_{p_{j-1}} = k_p(\|\mathbf{r}_{j-1}\| - d_w)\frac{\mathbf{r}_{j-1}}{\|\mathbf{r}_{j-1}\|} \tag{16}$$

$$\mathbf{f}_{p_{j+1}} = k_p(\|\mathbf{r}_{j+1}\| - d_w)\frac{\mathbf{r}_{j+1}}{\|\mathbf{r}_{j+1}\|} \tag{17}$$

$$\mathbf{f}_{b_j} = -b\dot{\mathbf{p}}_j \tag{18}$$

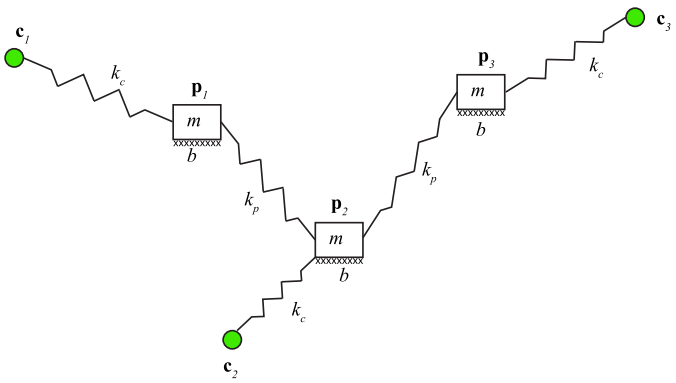


Fig. 8. Illustration of how the path is modeled as a mass-spring-damper system.

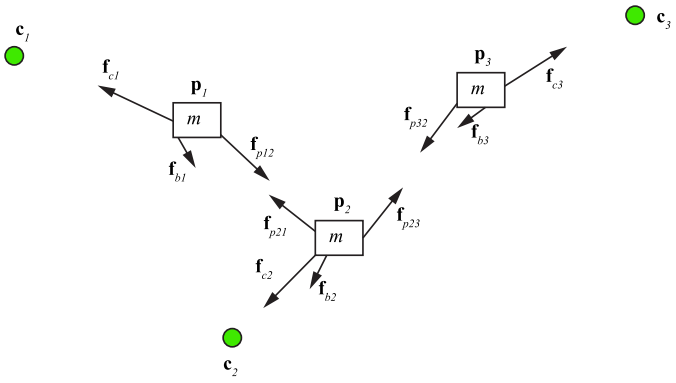


Fig. 9. Free body diagram for a path modeled as a mass-spring-damper system.

where $\mathbf{r}_{j-1} = \mathbf{p}_{j-1} - \mathbf{p}_j$ and $\mathbf{r}_{j+1} = \mathbf{p}_{j+1} - \mathbf{p}_j$, $b$ is a damping coefficient, and $k_c$ and $k_p$ are spring constants associated with the centroid and the neighboring waypoints, respectively. In the special cases of the endpoints, there is only one adjacent waypoint, which means that for $\mathbf{p}_1$, the force $\mathbf{f}_{p_{j-1}} = 0$, and for $\mathbf{p}_n$, the force $\mathbf{f}_{p_{j+1}} = 0$.

To determine the movement of each waypoint, the acceleration for each is calculated using

$$\ddot{\mathbf{p}}_j = \frac{\mathbf{f}_j}{m} \tag{19}$$

where $m$ is the mass value assigned to the waypoints. The differential equation (19) can then be solved numerically using Euler's method to find the next velocity $\dot{\mathbf{p}}_j$ and position $\mathbf{p}_j$ for each waypoint.

### *F. Path Generation Summary*

Putting together all of the elements discussed in this section, the energy-constrained VPG algorithm works as follows. To begin, an initial random path is generated, with the waypoints spaced to satisfy constraints (7) and (8). In addition, to avoid local optima caused by the path crossing itself, the algorithm ensures that the initial path does not cross at any point. The algorithm then loops through several actions. First, it draws a Voronoi diagram based on the current waypoint locations, as described in Section III-C. Next, it computes the

centroids of the Voronoi polygons using (9) and (10). Then, it calculates the force on each waypoint based on (14)–(17). The acceleration of each particle is found using (19), and the resulting differential equation is numerically solved using a 0.01-s time step to get the position and velocity of each waypoint at the next time step. This process repeats until the system converges, defined as all points moving less than 1/1000th of the largest dimension of the area at any given time step. The path length $l_{\text{total}}$ is computed when the system converges, and if it is too long, the ratio between $k_p$ and $k_c$ is doubled. The algorithm finishes when the system has converged and the path meets the length constraint. The full VPG algorithm is given as follows.

Initialize $P$ to a random path
Initialize $\mathbf{p}_i \leftarrow 0$, $\forall \mathbf{p}_i \in P$
Initialize $k_{ratio} \leftarrow 10$ $\{k_{ratio} \equiv \frac{k_p}{k_c}\}$
**while** $l_{total} > d_w(n-1)$ **do**
  $k_c \leftarrow k_p / k_{ratio}$
  $b \leftarrow \sqrt{k_c}/2$
  **while** P.NotConverged() **do**
    $V \leftarrow P.GetVoronoiDiagram()$
    $C \leftarrow V.FindCentroids$
    **for all** $\mathbf{p}_j \in P$ **do**
      $\mathbf{f}_j \leftarrow$ Eq. (14)
      $\dot{\mathbf{p}}_j \leftarrow \frac{\mathbf{f}_j}{m}\Delta t + \dot{\mathbf{p}}_{j,prev}$
      $\mathbf{p}_j \leftarrow \dot{\mathbf{p}}_j \Delta t + \mathbf{p}_{j,prev}$
      **if** $\mathbf{p}_j$.NotInArea() **then**
        Find crossed boundary edge $e$
        $\dot{\mathbf{p}}_j \leftarrow e.FindParallelVelocityComponent()$
        $\mathbf{p}_j \leftarrow \dot{\mathbf{p}}_j \Delta t + \mathbf{p}_{j,prev}$
      **end if**
    **end for**
  **end while**
  Multiply $k_{ratio}$ by 2
**end while**
**return** $P$

## IV. Simulations and Results

Simulations are presented to assess the performance of the VPG algorithm, beginning with several examples of the sorts of paths that the algorithm generates in both convex and nonconvex areas. Tests are performed to characterize the time complexity of the algorithm as the number of waypoints increases and to compare the algorithm to other path generation methods, looking at the runtimes of the algorithms and the optimality of the paths they generate. The simulations assume that an arbitrary energy model has already been used to determine the number of waypoints, which is used as the input to the algorithms. All simulations were programmed and run using MATLAB R2013a on a Dell Inspiron 7559 Laptop running Windows 10 with an Intel Core i5 6300HQ CPU and 8 GB of RAM. The built-in MATLAB delaunayTriangulation class is used to generate the Delaunay triangulations used in creating the Voronoi diagrams. For the best algorithm performance, the values of $k_c$, $k_p$, and $b$ are set to 900 N/m, 9000 N/m, and 15 N·s/m, respectively, assuming unit masses.

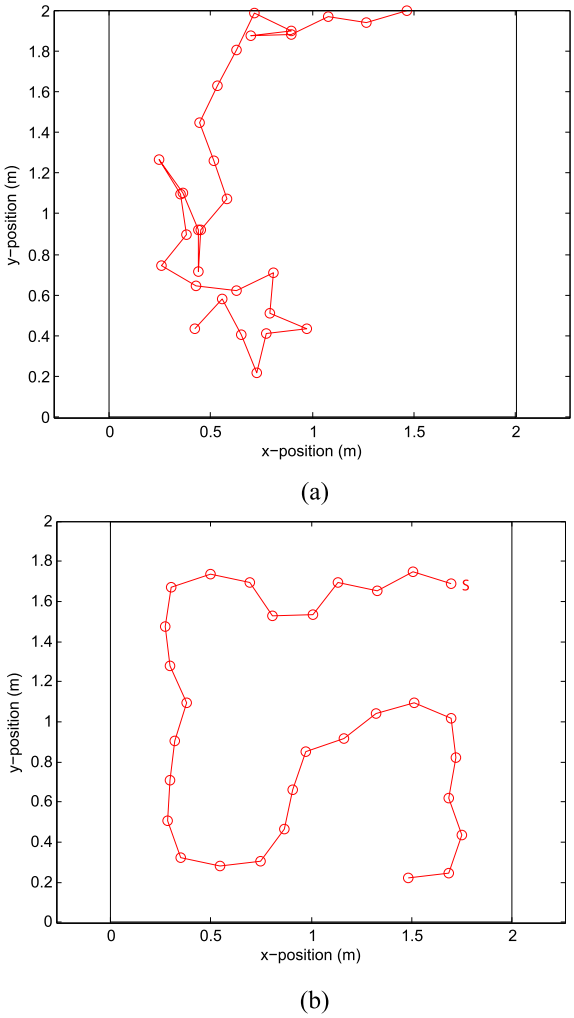


Fig. 10. Path generated using a 2 m × 2 m area and 30 waypoints with the spacing of 0.2 m. (a) Initial waypoint locations. (b) Final path with the starting point marked with an "S."

### A. Path Generation Examples

In the first example, the algorithm was run in a 2 m × 2 m square using 30 waypoints with a spacing of 0.2 m. The initial locations of the waypoints for the path are shown in Fig. 10(a). The system converged to the path shown in Fig. 10(b). If the cost function is computed numerically, discretizing the square into a 75 × 75 cell grid when taking the area integral, the total cost of the generated path is 1032.

The remaining examples show the paths generated with both convex and nonconvex polygonal areas of interest, illustrating the ability of the algorithm to generate paths in both types of polygons. Fig. 11 shows these paths. Each was generated using 20 waypoints, with the same waypoint spacing for all.

### B. Algorithm Characterization

Several different types of tests were run in simulation to characterize the VPG algorithm. One set of tests was used to determine the time complexity of the algorithm with respect to the number of waypoints in the path. Another set of tests was run to compare the algorithm runtime and optimality

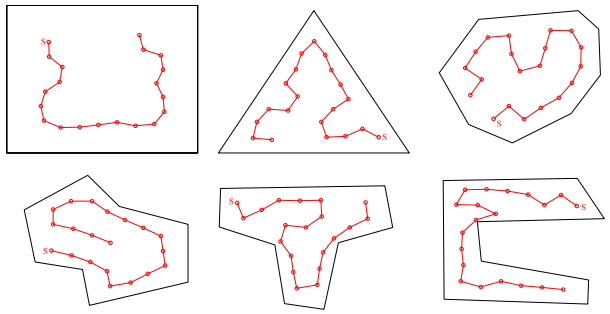

Fig. 11. Examples of paths generated by the VPG algorithm in both convex and nonconvex polygons, using 20 waypoints. Start points are denoted with an "S."

of the VPG algorithm against three other path generation methods. The first comparison method is the method described by Soltero *et al.* [23]. This method was chosen because it is conceptually the most similar to the method presented in this article. However, due to the fact that the Soltero method does not allow direct control of the path length, a slight modification was added to iteratively change the weights in the cost function to ensure that the final path would meet the length constraints. The amount by which to change the weights each iteration to ensure that the fastest convergence was determined by trial-and-error. Because no initial information about the area of interest is assumed, the density function is set as a uniform density. The second comparison method is direct optimization using MATLAB's built-in active set algorithm provided by the function *fmincon*. This method was chosen both to provide a baseline for the true optimal path because direct optimization is one obvious solution to the problem of generating an optimal partial coverage path. The third path generation method is simply adding a lawnmower-type path to the area, with the optimal sweep direction chosen as described in [41] and with even spacing between the sweep lines. This method was chosen because it is commonly used for generating coverage paths, with much of the work in CPP involving creating these sorts of coverage paths for a variety of areas and applications.

*1) Time Complexity:* To determine the time complexity of the VPG algorithm, it was run in a 2 m $\times$ 2 m box for paths with numbers of waypoints ranging from 5 to 100, with the same waypoint spacing regardless of a number of waypoints. One-hundred trials were run for each different number of waypoints, and both the total time to converge and the number of iterations required to converge were recorded and used to compute the average runtime per iteration. The results are presented in Fig. 12, showing that as the number of points increases, the time per iteration increase appears to be linear.

*2) Optimality Comparison Tests:* To assess the optimality of the VPG algorithm, it was compared with the three previously described comparison methods. Five different path lengths were tested, in a 75 m $\times$ 100 m box, using different numbers of waypoints but the same waypoint spacing of 5 m. All algorithms were run ten times for each number of waypoints. For the iterative algorithms, all three used the same initial randomly generated path for a given trial, with a different initial path generated for each trial. The cost function was computed over the final path, and the average path cost was

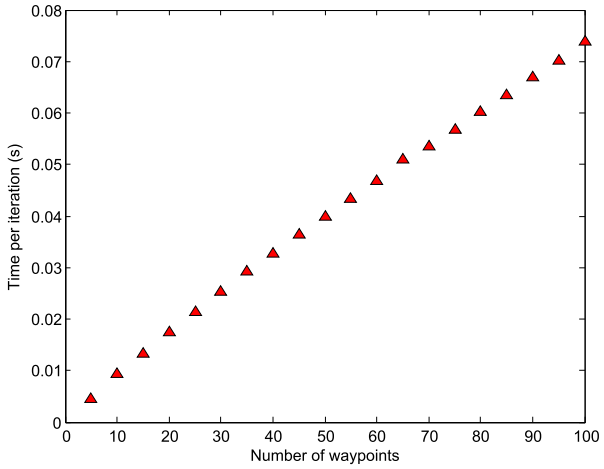


Fig. 12. Average runtime per iteration for different numbers of waypoints in a path.

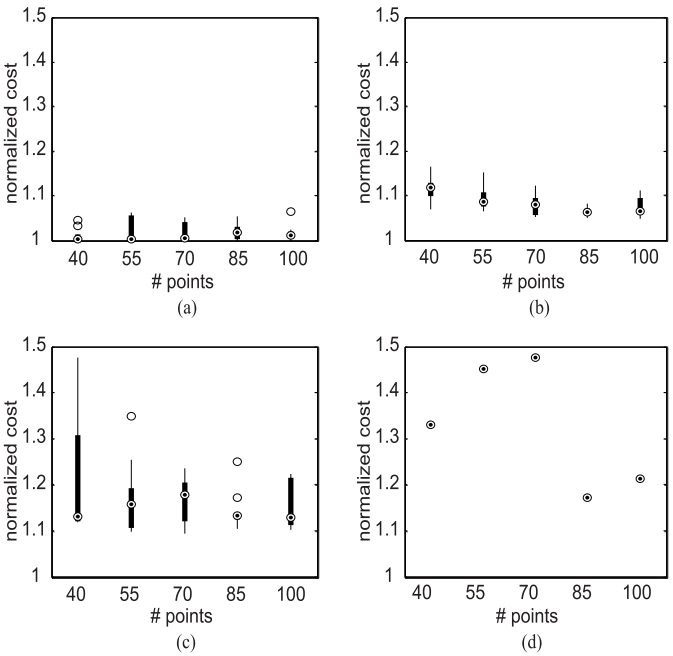


Fig. 13. Costs of the final paths generated by (a) fmincon, (b) VPG method, (c) Soltero method, and (d) lawnmower path generation method, normalized by the cost of the actual optimal path for each number of path waypoints, which is assumed to be the lowest cost path found by direct optimization.

computed for each method for each number of path waypoints. The cost of the true optimal path for each number of waypoints was assumed to be the lowest cost found by the direct optimization, and the average costs for each different number of waypoints were normalized by that optimal cost. The results are shown in Fig. 13. The VPG algorithm-generated paths between 45% and 55% closer to the optimal cost than the method in [23] and between 62% and 83% closer to the optimal cost than the lawnmower path generation.

The standard deviations of the path costs for each method for each number of waypoints, normalized by the optimal path cost, are shown in Table I. Note that the lawnmower path generation method is not included in this table because it is a deterministic method, so its standard deviation is zero.

TABLE I

STANDARD DEVIATIONS OF THE PATH COSTS NORMALIZED BY THE OPTIMAL PATH COST FOR EACH NUMBER OF WAYPOINTS, FOR THE VPG METHOD, THE SOLTERO METHOD, AND DIRECT OPTIMIZATION

| | 40 pts | 55 pts | 70 pts | 85 pts | 100 pts |
|---|---|---|---|---|---|
| Voronoi | 0.026 | 0.026 | 0.023 | 0.010 | 0.023 |
| Soltero | 0.127 | 0.078 | 0.049 | 0.041 | 0.052 |
| *fmincon* | 0.016 | 0.028 | 0.022 | 0.018 | 0.018 |

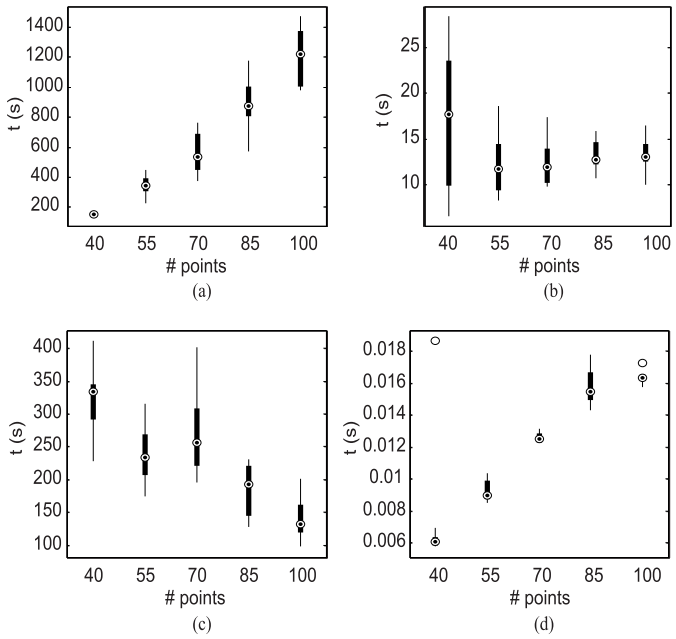


Fig. 14. Average total times to converge for (a) fmincon, (b) VPG method, (c) Soltero method, and (d) lawnmower path generation method for different numbers of path waypoints.

*3) Runtime Comparison Tests:* The total time required to converge was also measured for each method in the trials described in Section IV-B2, and the average convergence time for each number of waypoints using each method was computed. The results are shown in Fig. 14.

## V. PHYSICAL EXPERIMENTS AND RESULTS

This section describes the details of the experiments to test the applicability of the VPG algorithm to a real-world mobile robotic platform, as well as to determine how the simulation results compare to physical tests. The path generation tests are described, and the results of those experiments are given. First, it is pointed out that the physical experiments were run on the Enif autonomous chemical-sensing aerial robot platform, where full details of the vehicle, command station, and software are described in [11] (see Fig. 15). The robot system was designed for autonomous chemical-source mapping and localization. Because of this, the vehicle was optimized for maximizing flight time, to allow it to cover and map large areas. The vehicle has a 50-cm carbon fiber frame and can carry two lithium-polymer batteries. It has a GPS for position tracking, a LiDAR sensor for collision avoidance, and a high-performance Mass Property Spectrometer (MPS,

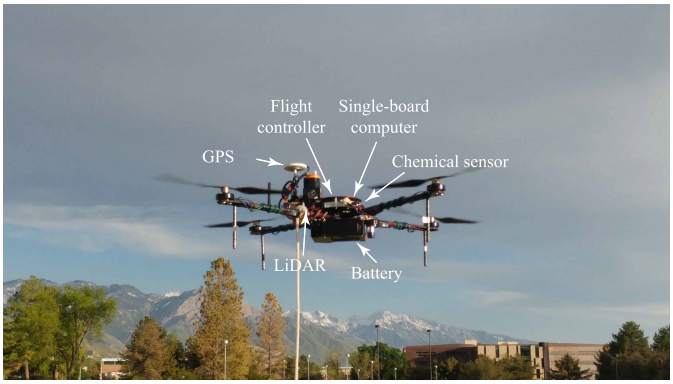


Fig. 15. Enif autonomous chemical-sensing UAV [11] photographed during an in-flight path generation test. Image also shows key components of the system.

Nevada Nano Tech) chemical sensor for airborne chemical detection. It uses a DJI A3 flight controller integrated with an Odroid C2 single-board computer to run everything necessary for autonomous flight. For the gross vehicle weight of 2.95 kg, the vehicle can hover for approximately 45 min [11].

While the platform is capable of flying with up to two batteries, one battery was used for the tests in this section, so the platform achieved approximately 30 min of flight time for each test. The VPG algorithm was programmed in C++ to interface with the robot operating system (ROS) running on the robot, which is used to communicate between the different algorithms that control the Enif platform. The MATLAB version of the VPG algorithm used the built-in MATLAB delaunayTriangulation function, which is not available in C++, so it was replaced with an open-source Delaunay triangulation program [54].

### A. Path Generation Tests

To test the performance of the path generation algorithm on the robot platform and to evaluate the applicability of simulation results to the real world, the VPG and lawnmower path generation algorithms were tested in two different areas. The first area, shown in Fig. 16(a), is a 20 m $\times$ 40 $m_2$. The second area, shown in Fig. 16(b), is a roughly 150 m $\times$ 80 m trapezoid. Both areas were chosen because they were accessible for flight testing, open and relatively flat. Both a rectangular and a nonrectangular area were chosen to demonstrate paths for different shapes as well as sizes. After each test, the total path costs were computed based on the actual path flown. Due to the long convergence times for the Soltero and direct optimization methods used in the simulation tests, they were not used as comparison methods in the physical tests.

As a means to compare results, each real-world flight test was also run in simulation. The GPS coordinates of the polygon corners were extracted from the flight test data, converted to coordinates in meters, and passed to the VPG algorithm in simulation, along with the associated waypoint spacing and a number of waypoints. The path costs were analyzed in the same way as the costs of the real-world flight paths, to provide a quantitative comparison between simulation

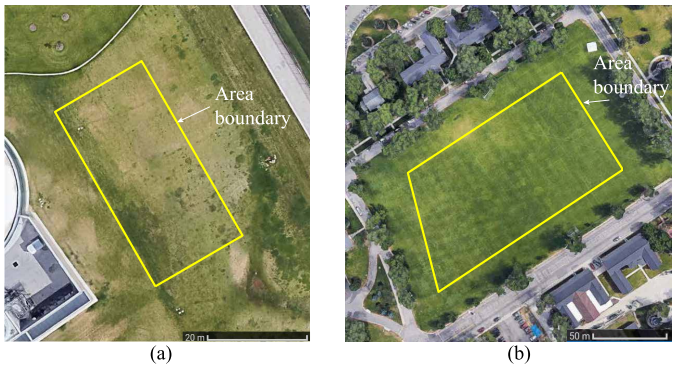


Fig. 16. Overhead view of (a) first flight test area (small area) and (b) second flight test area (large area), as seen from Google Maps.

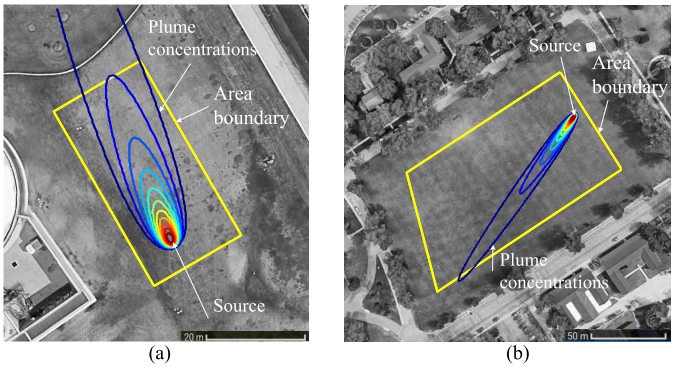


Fig. 17. Gaussian plume model in (a) small and (b) large areas.

and reality. Based on the experimental characterization of the energy model for the Enif platform, the values of $e_{\text{tot}}$ and $e_t$ for (3) were determined to be 115 and 0.0653 Wh, respectively.

To provide an illustration of how the paths would be used for sensing a chemical plume, the generated paths were overlaid on a standard Gaussian model of the chemical concentrations in a gaseous plume. The Gaussian plume model captures the spatial variation of the chemical concentration [55], [56]. Colormaps of the concentration readings were generated, going from blue to red as concentration increases. The plume in the small area is shown in Fig. 17(a). The plume in the large area is shown in Fig. 17(b).

*1) Small Area Experiments:* In the first smaller flight test area, the waypoint spacing was chosen to be 1 m. The wait time at each waypoint was set to 15 s, based on the time it takes the chemical sensor on the Enif platform to take a good reading. From experimental characterization of the energy model for this particular platform, $e_m = 0.0708$ Wh for a 1-m waypoint distance and $e_r = 1.025$ Wh for a 15-s wait time. Using (3), the number of path waypoints was determined to be 99. Both the VPG method and the lawnmower path generation method were each tested three times in both the real world and simulation. For each physical test, the Enif platform generated its path and then flew along the path until the low-battery safety feature automatically landed the platform. The GPS coordinates of the flight path and the area polygon corners were recorded. Postprocessing was performed on the

TABLE II

AVERAGE COSTS OF THE PATHS GENERATED IN SIMULATION, THE AVERAGE COSTS OF THE ACTUAL FLIGHT PATHS FLOWN, AND THE PERCENT DIFFERENCES BETWEEN THEM FOR THE FIRST AREA

| | Simulation | Real-world | % difference, real vs. sim |
|---|---|---|---|
| Voronoi avg. cost (m) | 4973 | 5032 | 1.2 |
| Lawnmower avg. cost (m) | 6596 | 6465 | 2.0 |
| % difference, methods | 24.6 | 22.2 | |

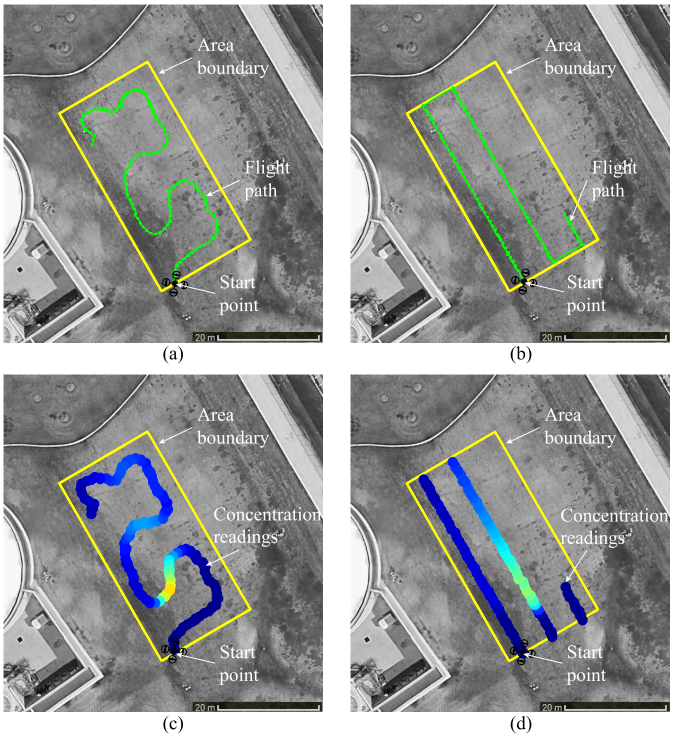


Fig. 18. Flight paths from small area experiments. (a) Flight path flown by the chemical-sensing robot platform during real-world testing based on the path generated using the VPG method. (b) Flight path based on the generated lawnmower path. (c) Concentrations of the Gaussian plume along the flight path generated using the VPG method. (d) Concentrations along the lawnmower flight path.

GPS coordinates to convert them to locations in meters. There was no easy way to reliably detect where the platform had stopped at a waypoint based only on the raw numbers, but the waypoint locations were obvious when the flight path was plotted, so the MATLAB data tip tool was used to carefully select and save each waypoint location. Once the waypoints were extracted, the costs of the paths were determined and averaged. The costs of the simulated paths generated with the same input parameters were also determined and averaged. The results are shown in Table II. One of the real-world flight paths generated with the VPG method is shown in Fig. 18(a). A real-world lawnmower flight path is shown in Fig. 18(b). The concentration maps that each of these paths would have generated are shown in Fig. 18(c) and (d).

*2) Large Area Experiments:* In the second, larger flight test area, the waypoint spacing was chosen to be 1.7 m, just above 1% of the largest area dimension, which has been found through qualitative testing to be roughly the lower limit

TABLE III

COSTS OF THE PATHS GENERATED IN SIMULATION, THE COSTS OF THE ACTUAL FLIGHT PATHS, AND THE PERCENT DIFFERENCES BETWEEN THEM FOR THE SECOND AREA

| | Simulation | Real-world | % difference, real vs. sim |
|---|---|---|---|
| Voronoi avg. cost (m) | 2799 | 2711 | 3.2 |
| Lawnmower avg. cost (m) | 2837 | 2796 | 1.5 |
| % difference, methods | 1.3 | 3.0 | |

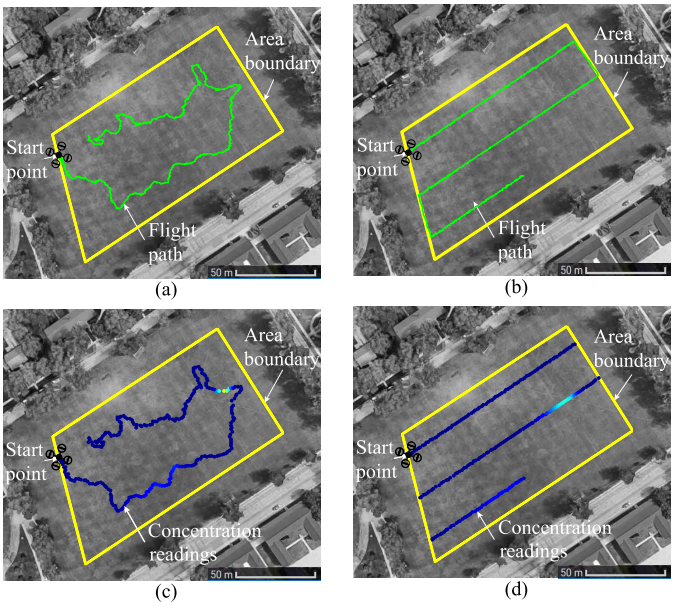


Fig. 19. Flight paths from large area experiments. (a) Flight path flown by the chemical-sensing robot platform during real-world testing based on the path generated using the VPG method. (b) Flight path based on the generated lawnmower path. (c) Concentrations of the Gaussian plume along the flight path generated using the VPG method. (d) Concentrations along the lawnmower flight path.

for waypoint spacing before the difference in the stiffnesses of the modeled springs between the adjacent waypoints and between the waypoints and their Voronoi polygon centroids to begin to cause difficulties with path convergence. The wait time at each waypoint was set to 4 s, based on the time it takes the Enif platform to come to a complete stop and take a measurement, assuming that it uses a faster generic sensor. For these tests, $e_m = 0.120$ Wh for a 1.7-m waypoint spacing and $e_r = 0.273$ Wh for a 4-s wait time. Using (3), the number of path waypoints was determined to be 250. As with the smaller area, the Enif platform generated its path and then flew along it until it was forced to land by the built-in safety feature. The same postprocessing of the flight path data was performed, converting the GPS coordinates into meters, plotting the flight path, extracting the waypoint locations, and determining the path costs. The costs of the paths generated in simulation using the same input parameters were also determined. The results are shown in Table III. The flight path generated with the VPG method is shown in Fig. 19(a). The lawnmower flight path is shown in Fig. 19(b). The concentration maps that each of these paths would have generated are shown in Fig. 19(c) and (d).

## VI. DISCUSSION OF RESULTS

### A. Algorithm Characterization

The results of the simulations show that the VPG algorithm strikes a good balance between optimality and runtime. It is able to generate near-optimal paths while still running quickly enough to be usable for planning on a mobile robot in the field. In addition, as shown in Fig. 11, the method is capable of generating paths in both convex and nonconvex areas without having to make adjustments to the algorithm.

*1) Runtime Results:* As can be seen in Fig. 12, the time complexity of the VPG method appears to be linear. This allows the method to extend to relatively large numbers of points before hitting the upper limit where it becomes prohibitively slow to run. While absolute runtimes will obviously differ somewhat depending on the hardware of the mobile robot, the fact that the runtimes are on the order of seconds for the VPG method is promising for the ability of a mobile robot to run the algorithm on its own hardware in the field.

As shown in Fig. 14, it would be practically impossible to run direct optimization on a mobile robot in the field for large numbers of path waypoints. It would be difficult to run the Soltero method with iteratively changing weights on a mobile robot in the field without some efficiency improvements, but it is possible that such improvements could be made. However, based on the data in Fig. 14, the VPG algorithm runs an order of magnitude faster and will thus be easier for a mobile robot to use. The lawnmower path generation method is the only tested method faster than the VPG algorithm, likely due to the lack of iteration or complex computation required by the algorithm. However, this is only guaranteed to be the case for a convex area, as applying lawnmower-style paths to a nonconvex area becomes significantly more complex, likely involving methods similar to boustrophedon decomposition methods for CPP, where the area would be decomposed into multiple convex areas, a traveling salesman problem would be solved to determine the order in which to visit the subareas, and lawnmower-style paths would be placed in each individual subarea, with additional consideration as to how to space the sweep lines such that the path-length constraints are met and the partial coverage path does not cover any subareas more than the others. This process would take significantly more time than the simple lawnmower path generation for a convex area used as a comparison method in this article.

*2) Optimality Results:* Based on the optimality comparisons in Fig. 13, direct optimization results in the best paths with respect to path cost, followed by the VPG algorithm and the Soltero method, with the worst being the lawnmower pattern. The VPG algorithm outperforms both the simple lawnmower path and the method from [23] in terms of path cost. From Fig. 13, the cost of the lawnmower path is anywhere between 17% and 50% higher than the cost of the optimal path for the path lengths tested, and the method in [23] produces paths with costs between 15% and 25% higher. The VPG algorithm creates the paths that cost between 5% and 11% more than the optimal path. The VPG algorithm creates the paths with costs between 63% and 83% closer to the cost of the optimal path than the basic lawnmower paths and between 45% and 55%

closer than the paths generated using the method from [23], indicating that from an optimality standpoint, it is much better to use for generating partial coverage paths.

One further note is that while the VPG algorithm does significantly better than the lawnmower path generation method over the range of waypoints tested, the normalized cost of the lawnmower path decreases overall as the number of waypoints increases, suggesting that as the path becomes long enough to approach complete coverage, the lawnmower paths could become just as good as the paths generated using the Voronoi-based method. However, for partial coverage, which is the focus of this article, the VPG algorithm is the better of the two methods to use, and its ease of application in nonconvex areas could still make it a reasonable alternative to a lawnmower path in such areas even as the path approaches a length where the costs of the paths from both methods become comparable.

From Table I, the standard deviations for the VPG algorithm are between 1% and 2.6% of the optimal path cost, indicating that despite using random initial paths, the final paths generated by the algorithm are fairly consistent in cost. This is on par with the direct optimization method, which has standard deviations for most tests between 1.6% and 2.8%. There is a bit more variability in the Soltero method, with standard deviations ranging from 4.1% to 12.7% of the optimal path cost. The lawnmower path generation method has no standard deviation because it is deterministic, not iterative, and thus generates the same path every time when given the same area and path-length constraint.

*3) Comparison Analysis:* The VPG algorithm has different strengths when compared individually to the other methods. While it runs slower than a simple lawnmower path generation, the paths from the VPG method have costs that are significantly closer to the optimum than the lawnmower path. The VPG method is also significantly simpler to apply to a nonconvex area than the lawnmower path generation method. Large amounts of research have been done just to find the best way of putting a lawnmower-style path onto a nonconvex area for complete coverage, and extending these methods to partial coverage is so complex as to be beyond the scope of this article. Compared to direct optimization, the VPG algorithm generates paths whose costs are not as close to the optimal path cost, which is to be expected, but the loss in optimality is made up for in the drastic improvements in runtime as the number of path waypoints increases. The method from [23] performs moderately worse than the VPG on both metrics, primarily because it is not designed to have hard path-length limits applied to it. The path length can only be controlled indirectly by changing the weights in the cost function, with the required weights varying based on multiple considerations such as path length, area size, and area shape. The easiest way to find the correct weights is by changing them iteratively, which is not particularly efficient. It also causes the phenomenon seen in the plot of total runtimes where the total runtime actually decreases with the number of waypoints. More waypoints allow for a longer path, which means having to iteratively change the weights fewer times, resulting in fewer iterations to converge. The method from [23] also requires taking area integrals numerically, increasing computation time compared to the VPG algorithm, which is specifically designed to avoid taking area integrals.

### *B. Real-World Tests*

The results of the real-world tests show two things. First, the method can be implemented on an actual robotic platform to successfully generate coverage paths for the robot to follow, with better coverage than lawnmower-style paths. Second, given a reasonably accurate energy model and a robot capable of closely following the generated path, there is a little difference between simulated and real-world results.

*1) Comparison of Path Generation Methods:* For both test areas, the VPG method produced paths better than a lawnmower path, as shown in Tables II and III. In the smaller area, the path generated using the VPG method had a 22.2% lower cost in the real-world flight tests. In the larger area, the path generated using the Voronoi method had a 3.0% lower cost in the real-world flight tests. The difference in cost improvement between the two areas can be explained by the difference in area size, wait time at each waypoint, and distance between waypoints, all of which affect the length of the generated path compared with the area size and thus affect the ability of the generated path to cover the area. In addition, if the waypoint spacing is very small with respect to the area size, the interaction between the different types of spring forces can cause the path having difficulty getting close to the optimal path. For closely spaced waypoints, the Voronoi areas for points along the straight lines become long narrow rectangular shapes, with the centroid almost on top of the waypoint, whereas the Voronoi areas for waypoints along the curved sections of the path end up being large triangles or trapezoids with centroids far from the waypoints. If there are large numbers of closely spaced waypoints, the few waypoints along the curves must pull a large number of waypoints along with them if they are going to get close to the centroids of their Voronoi areas, in which they are not always able to do, causing the system to converge to a less good path.

*2) Comparison With Simulation:* For both test areas, the results of the physical flight tests track closely with the simulated results. As can be seen from Tables II and III, the computed costs for the final flight paths were similar to those for the paths in simulation. In the smaller area, the paths generated using the VPG method had a 1.2% cost difference between reality and simulation, and the lawnmower paths had a 2.0% difference. In the larger area, the paths generated by the VPG method have a 3.2% difference between reality and simulation, and the lawnmower paths had a 1.5% difference. Since the lawnmower path generation method is deterministic, the percent differences in path costs between the real world and the simulation tests must come from slight discrepancies in the waypoint following on the physical robotic platform. The fact that the VPG method had similar differences between the real-world and simulated paths suggests that the difference is also accounted for primarily by slight discrepancies in the waypoint tracking of the robotic platform.

In the smaller area, the VPG method path had a 24.6% lower cost than the lawnmower path in simulation compared with a

22.2% lower cost in the real-world flight tests. In the larger area, the path generated using the VPG method had a 1.3% lower cost than the lawnmower path in simulation compared with a 3.0% lower cost in the real-world flight tests. The similarity between the cost improvements seen in simulation and the cost improvements seen in the real-world flight paths, along with the small percent differences between real-world and simulation results, indicate the applicability of simulation results to physical robotic systems for different area sizes and shapes and different numbers of waypoints, suggesting that all of the conclusions drawn through the simulation tests will hold for physical robotic systems.

In summary, there are a few key points from the simulations and physical experiments. First is that the VPG algorithm strikes a good balance between optimality and runtime, with none of the comparison algorithms performing better on both counts. The linear nature of the measured time complexity suggests that the algorithm would scale relatively well to large numbers of path waypoints. The paths created using the VPG algorithm are near-optimal, with average costs falling between 5% and 11% of the cost of the true optimal path. The method can successfully be applied to a physical mobile robotic platform to generate actual coverage paths, given a robot capable of closely following paths with sharp turns. The results from the physical experiments track closely with those from the simulations using the same areas and path parameters, with path cost differences of 1%–3%, indicating that the simulation analysis is applicable to real-world mobile robotic systems.

## VII. Conclusion and Future Work

This article presented the VPG method for planning near-optimal coverage paths for a mobile robot given path-length constraints due to energy limitations. The path generation algorithm distributes path waypoints using a potential field created by modeling the path as a chained mass-spring-damper system. In this system, the waypoints are represented by masses, spring forces between adjacent waypoints maintaining waypoint spacing, other spring forces pulling waypoints toward the centroids of their Voronoi polygons, and a damping force acting between waypoints and the ground. Little work has been done previously exploring how to best generate paths that can only provide partial area coverage due to energy constraints, and the path-planning method described in this article is a novel approach to solving that problem.

Simulations were run to compare the VPG method with other methods that can generate partial coverage paths. The algorithm presented in this article is much faster than directly optimizing the waypoint locations, and it provides paths with costs much closer to the optimal path cost than the common method of applying a simple lawnmower pattern path to an area. It performs moderately better in terms of both time and final path cost than the conceptually somewhat similar method proposed in [23], due in part to differences in design focuses between the two methods, which resulted in different methods of modeling the system and generating the potential fields that move the path waypoints. The VPG method creates near-optimal coverage paths, with costs within about 10% of the cost of the optimal path. The method's runtime scales linearly with the number of waypoints, allowing it to feasibly handle paths with large numbers of waypoints. Finally, the algorithm can also handle generating paths for both convex and nonconvex areas with no additional modifications required.

The VPG algorithm was also successfully used to plan paths for a real-world mobile robotic platform in two different areas with different sizes and shapes. The VPG algorithm performed better than a lawnmower path in the physical experiments in both areas. Simulations were run with the same sizes and shapes of areas as the real-world tests, with the same number of path waypoints and waypoint spacing, and the results of the simulations were compared with the results from the physical experiments. This indicates that the results from the simulations and the analysis performed on those results hold for real-world mobile robotic systems, assuming a reasonably accurate energy model and a robotic platform capable of traversing the generated paths.

Some limitations of the algorithm include requiring a fairly accurate energy model, assuming that the robot can rotate in place, and the algorithm struggles when the waypoint spacing is less than about 1% of the largest area dimension.

There are a few possible directions for future research to build on the work done for this article. One avenue could be physical testing of this method with a live sensor and analysis of the maps it generates of the measured quantity. A second potential direction could involve extending the VPG method to 3-D spaces, whether for volumetric coverage or coverage of 3-D surfaces. A possible avenue of future research could also involve adapting the VPG algorithm to plan multiagent coverage paths, allowing multiple mobile robotic platforms to cover the area of interest. Another interesting research direction could be finding ways to modify the mass-spring-damper model of the system to generate paths that meet the nonholonomic motion constraints of robotic systems such as cars or fixed-wing UAVs, for example, using optimal control [57].

## Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

## References

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092188901300167X

[2] I. A. Hameed, D. Bochtis, and C. A. Sørensen, "An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 5, p. 231, May 2013, doi: 10.5772/56248.

[3] K. Zhou, A. L. Jensen, C. G. Sørensen, P. Busato, and D. D. Bothtis, "Agricultural operations planning in fields with multiple obstacle areas," *Comput. Electron. Agricult.*, vol. 109, pp. 12–22, Nov. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169914002129

[4] J. Jin and L. Tang, "Optimal coverage path planning for arable farming on 2D surfaces," *Trans. ASABE*, vol. 53, no. 1, p. 283, 2010.

[5] B. J. Englot and F. S. Hover, "Sampling-based coverage path planning for inspection of complex structures," in *Proc. AAAI*, May 2012, pp. 1–9.

[6] K. O. Ellefsen, H. A. Lepikson, and J. C. Albiez, "Multiobjective coverage path planning: Enabling automated inspection of complex, real-world structures," *Appl. Soft Comput.*, vol. 61, pp. 264–282, Dec. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494617304714

[7] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 718–724, Feb. 2004.

[8] J. S. Oh, Y. H. Choi, J. B. Park, and Y. F. Zheng, "Complete coverage navigation of cleaning robots using triangular-cell-based map," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 718–726, Jun. 2004.

[9] J. Valente, A. B. Cruz, J. del Cerro Giner, and D. S. Muñoz, "A waypoint-based mission planner for a farmland coverage with an aerial robot—A precision farming tool," in *Proc. Precis. Agricult.*, 2011, pp. 427–436. [Online]. Available: http://oa.upm.es/12710/

[10] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction," *Expert Syst. Appl.*, vol. 55, pp. 441–451, Aug. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417416300306

[11] X. He *et al.*, "Autonomous chemical-sensing aerial robot for urban/suburban environmental monitoring," *IEEE Syst. J.*, vol. 13, no. 3, pp. 3524–3535, Sep. 2019.

[12] M. Weiss-Cohen, I. Sirotin, and E. Rave, "Lawn mowing system for known areas," in *Proc. Int. Conf. Comput. Intell. Modelling Control Automat.*, Dec. 2008, pp. 539–544.

[13] M. Đakulovic and I. Petrovic, "Complete coverage path planning of mobile robots for humanitarian demining," *Ind. Robot, Int. J.*, vol. 39, no. 5, pp. 484–493, Aug. 2012, doi: 10.1108/01439911211249779.

[14] M. Hassan, D. Liu, S. Huang, and G. Dissanayake, "Task oriented area partitioning and allocation for optimal operation of multiple industrial robots in unstructured environments," in *Proc. 13th Int. Conf. Control Automat. Robot. Vis. (ICARCV)*, Dec. 2014, pp. 1184–1189.

[15] C. Di Franco and G. Buttazzo, "Coverage path planning for UAVs photogrammetry with energy and resolution constraints," *J. Intell. Robot. Syst.*, vol. 83, nos. 3–4, pp. 445–462, Sep. 2016, doi: 10.1007/s10846-016-0348-x.

[16] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "UB-ANC planner: Energy efficient coverage path planning with multiple drones," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 6182–6189.

[17] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *J. Intell. Robot. Syst.*, vol. 74, nos. 3–4, pp. 965–983, Jun. 2014, doi: 10.1007/s10846-013-9834-6.

[18] G. Sharma, A. Dutta, and J.-H. Kim, "Optimal online coverage path planning with energy constraints," in *Proc. 18th Int. Conf. Auton. Agents MultiAgent Syst. (AAMAS)*. Richland, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1189–1197.

[19] T. M. Cabreira, C. D. Franco, P. R. Ferreira, and G. C. Buttazzo, "Energy-aware spiral coverage path planning for UAV photogrammetric applications," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3662–3668, Oct. 2018.

[20] T. Cabreira, L. Brisolara, and P. R. Ferreira, Jr., "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, Jan. 2019. [Online]. Available: https://www.mdpi.com/2504-446X/3/1/4

[21] C. Papachristos, K. Alexis, L. R. G. Carrillo, and A. Tzes, "Distributed infrastructure inspection path planning for aerial robotics subject to time constraints," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 406–412.

[22] Y.-H. Han, Y.-H. Kim, W. Kim, and Y.-S. Jeong, "An energy-efficient self-deployment with the centroid-directed virtual force in mobile sensor networks," *Simulation*, vol. 88, no. 10, pp. 1152–1165, Oct. 2012, doi: 10.1177/0037549711411314.

[23] D. E. Soltero, M. Schwager, and D. Rus, "Decentralized path planning for coverage tasks using gradient descent adaptive control," *Int. J. Robot. Res.*, vol. 33, no. 3, pp. 401–425, Oct. 2013, doi: 10.1177/0278364913497241.

[24] X. Wu, Z. Li, Z. Kan, and H. Gao, "Reference trajectory reshaping optimization and control of robotic exoskeletons for human–robot comanipulation," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3740–3751, Aug. 2020.

[25] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using CNN-based fusion of vision and LiDAR in autonomous vehicle environment," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4224–4231, Sep. 2018.

[26] G. Fan and S. Jin, "Coverage problem in wireless sensor network: A survey," *J. Netw.*, vol. 5, no. 9, pp. 1033–1040, Sep. 2010.

[27] I. Khoufi, P. Minet, A. Laouiti, and S. Mahfoudh, "Survey of deployment algorithms in wireless sensor networks: Coverage and connectivity issues and challenges," *Int. J. Auton. Adapt. Commun. Syst.*, vol. 10, no. 4, pp. 341–390, 2017. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJAACS.2017.088774

[28] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems*. Tokyo, Japan: Springer, 2002, pp. 299–308.

[29] G. Wang, G. Cao, and T. F. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 640–652, Jun. 2006.

[30] H. A. Hashim, B. O. Ayinde, and M. A. Abido, "Optimal placement of relay nodes in wireless sensor network using artificial bee colony algorithm," *J. Netw. Comput. Appl.*, vol. 64, pp. 239–248, Apr. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516000527

[31] M. Abo-Zahhad, S. M. Ahmed, N. Sabor, and S. Sasaki, "Coverage maximization in mobile wireless sensor networks utilizing immune node deployment algorithm," in *Proc. IEEE 27th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2014, pp. 1–6.

[32] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Ann. Math. Artif. Intell.*, vol. 31, no. 1, pp. 77–98, Oct. 2001, doi: 10.1023/A:1016610507833.

[33] A. Zelinsky, R. Jarvis, J. C. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, 1993, pp. 533–538.

[34] M. Dakulović, S. Horvatić, and I. Petrović, "Complete coverage D* algorithm for path planning of a floor-cleaning mobile robot," in *Proc. 18th IFAC World Congr.*, Jan. 2011, vol. 44, no. 1, pp. 5950–5955. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S147466701644557X

[35] F. Balampanis, I. Maza, and A. Ollero, "Area decomposition, partition and coverage with multiple remotely piloted aircraft systems operating in coastal regions," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2016, pp. 275–283.

[36] S. A. Sadat, J. Wawerla, and R. Vaughan, "Fractal trajectories for online non-uniform aerial coverage," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 2971–2976.

[37] P. A. Jimenez, B. Shirinzadeh, A. Nicholson, and G. Alici, "Optimal area covering using genetic algorithms," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics*, Sep. 2007, pp. 1–5.

[38] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London, U.K.: Springer, 1998, pp. 203–209.

[39] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 331–344, Apr. 2002, doi: 10.1177/027836402320556359.

[40] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, vol. 1, May 2001, pp. 27–32

[41] Y. Li, H. Chen, M. J. Er, and X. Wang, "Coverage path planning for UAVs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, Aug. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957415810001893

[42] B. Dugarjav, S.-G. Lee, D. Kim, J. H. Kim, and N. Y. Chong, "Scan matching online cell decomposition for coverage path planning in an unknown environment," *Int. J. Precis. Eng. Manuf.*, vol. 14, no. 9, pp. 1551–1558, Sep. 2013, doi: 10.1007/s12541-013-0209-5.

[43] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-based coverage with extended range detectors," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 189–198, Feb. 2006.

[44] Y.-Y. Lin, C.-C. Ni, N. Lei, X. D. Gu, and J. Gao, "Robot coverage path planning for general surfaces using quadratic differentials," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 5005–5011.

[45] G. Avellar, G. Pereira, L. Pimenta, and P. Iscold, "Multi-UAV routing for area coverage and remote sensing with minimum time," *Sensors*, vol. 15, no. 11, pp. 27783–27803, Nov. 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/11/27783

[46] I. Maza and A. Ollero, "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Distributed Autonomous Robotic Systems*, R. Alami, R. Chatila, and H. Asama, Eds. Japan: Springer, 2007, pp. 221–230.

[47] L. Wu, M. A. Garcia, D. Puig, and A. Sole, "Voronoi-based space partitioning for coordinated multi-robot exploration," *J. Phys. Agents*, vol. 1, no. 1, pp. 37–44, 2007.

[48] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Trans. Cybern.*, vol. 44, no. 3, pp. 305–314, Mar. 2014.

[49] K. S. Senthilkumar and K. K. Bharadwaj, "Multi-robot exploration and terrain coverage in an unknown environment," *Robot. Auton. Syst.*, vol. 60, no. 1, pp. 123–132, Jan. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889011001783

[50] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, "DARP: Divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robot. Syst.*, vol. 86, nos. 3–4, pp. 663–680, Jun. 2017, doi: 10.1007/s10846-016-0461-x.

[51] K. S. Senthilkumar and K. K. Bharadwaj, "An efficient global optimization approach to multi robot path exploration problem using hybrid genetic algorithm," in *Proc. 4th Int. Conf. Inf. Automat. Sustainability*, Dec. 2008, pp. 7–12.

[52] W.-M. Chong, C.-L. Goh, and Y.-T. Bau, "A practical framework for cleaning robots," in *Proc. 6th Int. Conf. Bio-Inspired Comput., Theories Appl.*, Sep. 2011, pp. 97–102.

[53] G. Gutin, *The Traveling Salesman Problem and Its Variations* (Combinatorial Optimization), vol. 12. Boston, MA, USA: Springer, 2007.

[54] S. Zeni. (2015). *Delaunay-Triangulation*. [Online]. Available: https://github.com/Bl4ckb0ne/delaunay-triangulation

[55] J. R. Bourne, E. R. Pardyjak, and K. K. Leang, "Coordinated Bayesian-based bioinspired plume source term estimation and source seeking for mobile robots," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 967–986, Aug. 2019.

[56] X. He, J. R. Bourne, J. A. Steiner, and K. K. Leang, "Gaussian-based kernel for multi-agent aerial chemical-plume mapping," in *Proc. ASME Dyn. Syst. Control Conf. (DSCC)*, 2019, pp. 1–10.

[57] X. Wang, J. Liu, Y. Zhang, B. Shi, D. Jiang, and H. Peng, "A unified symplectic pseudospectral method for motion planning and tracking control of 3D underactuated overhead cranes," *Int. J. Robust Nonlinear Control*, vol. 29, no. 7, pp. 2236–2253, May 2019.

**Katharin R. Jensen-Nau** received the B.S. and M.S. degrees in mechanical engineering from the University of Utah, Salt Lake City, UT, USA, in 2018.

She is currently a Control Systems Engineer with American Oxygen, West Valley City, UT, USA, where she is working on thermal and electrochemical system analysis and control. Her research interests include path planning and control of mobile robotic systems.

**Tucker Hermans** (Member, IEEE) received the A.B. degree in German and computer science from Bowdoin College, Brunswick, ME, USA, in 2009, the M.Sc. degree in computer science with specialization in computational perception and robotics from Georgia Tech, Atlanta, GA, USA, in 2012, and the Ph.D. degree in robotics from the Computational Perception Laboratory, Georgia Tech, under the supervision of Aaron Bobick and Jim Rehg in 2014. His dissertation research dealt with robots learning to discover and manipulate previously unknown objects.

He was a Post-Doctoral Researcher with the Intelligent Autonomous Systems Laboratory, Technical University of Darmstadt, Darmstadt, Germany, where he worked with Jan Peters on tactile manipulation and robot learning while serving as the Team Leader at TUDa for the European Commission project TACMAN. He was with the School of Interactive Computing, Georgia Tech, from 2009 to 2014. He is currently an Assistant Professor with the School of Computing, University of Utah, Salt Lake City, UT, USA. He is also with the Robotics Center, University of Utah.

**Kam K. Leang** (Senior Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from the University of Utah, Salt Lake City, UT, USA, in 1997 and 1999, respectively, and the Ph.D. degree from the University of Washington, Seattle, WA, USA, in December 2004.

From 2008 to 2014, he was with the University of Nevada at Reno, Reno, NV, USA. He is currently a Professor with the Mechanical Engineering Department, University of Utah, where he joined in July 2014 and directs the Design, Automation, Robotics and Control (DARC) Laboratory. He is also a member of the Robotics Center, University of Utah. His current research interests focus on three main areas: modeling and control for high-speed nanopositioning and scanning probe microscopy; modeling, control, and manufacturing of electroactive polymers for applications in soft robotics; and design, control, and motion planning of mobile robotic systems, including aerial robotic systems.

Dr. Leang is a fellow of the ASME. He has been involved with conference organization and editorialship, including the American Control Conference (ACC), the IEEE International Conference on Robotics and Automation (ICRA), and the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). He also serves as an Associate Editor for *Mechatronics* (Elsevier), *ASME Letters on Dynamic Systems and Control*, and the *International Journal of Intelligent Robotics and Applications* (IJIRA).