Auton Robot (2014) 36:365–381
DOI 10.1007/s10514-013-9364-x

Efficient complete coverage of a known arbitrary environment with applications to aerial operations

Anqi Xu · Chatavut Viriyasuthee · Ioannis Rekleitis

Received: 20 July 2012 / Accepted: 8 August 2013 / Published online: 8 September 2013

Abstract The problem of coverage of known space arises in a multitude of domains, including search and rescue, mapping, and surveillance. In many of these applications, it is desirable or even necessary for the solution to guarantee both the complete coverage of the free space, as well as the efficiency of the generated trajectory in terms of distance traveled. A novel algorithm is introduced, based on the boustrophedon cellular decomposition technique, for computing an efficient complete coverage path for a known environment populated with arbitrary obstacles. This hierarchical approach first partitions the space to be covered into non-overlapping cells, then solves the Chinese postman problem to compute an Eulerian circuit traversing through these cells, and finally concatenates per-cell seed spreader motion patterns into a complete coverage path. Practical considerations of the coverage system are also explored for operations with a non-holonomic aerial vehicle. The effects of various system parameters are evaluated in controlled environments using a high-fidelity flight simulator, in addition to over 200 km of in-field flight sessions with a fixed-wing unmanned aerial vehicle.

Keywords Path planning · Coverage · Unmanned aerial vehicles

A. Xu · C. Viriyasuthee · I. Rekleitis (✉)
School of Computer Science, McGill University, Montreal, QC, Canada
e-mail: yiannis@cim.mcgill.ca

A. Xu
e-mail: anqixu@cim.mcgill.ca

C. Viriyasuthee
e-mail: pvirie@cim.mcgill.ca

1 Introduction

The task of covering a bounded region of space is an essential component of many robotic activities, such as automated painting, humanitarian de-mining, and search and rescue. The importance and impact of the coverage task is highlighted, for instance, by the immense commercial success of robotic vacuum cleaners, which have been integrated within the modern household. These vacuuming robots use diverse motion strategies, such as random walk and wall following, to achieve coverage of the entire floor space.

In all of these applications, the problem of coverage is defined as follows:

> The robot must pass an end-effector or a sensor, which might correspond to the entire body of the vehicle, over all of the available free space within a bounded region.

For example, a de-mining robot must scan for the presence of mines over the entirety of the terrain that is not obstructed by obstacles (Acar et al. 2003). In such an application, it is of paramount importance to ensure the completeness of coverage, meaning that no accessible area should be left uncovered at the end of the coverage session. It is equally vital in virtually all domains for the coverage path taken by the robot to be efficient, so as to reduce operational costs.

Coverage strategies can be categorized according to several criteria. One important differentiating factor is whether a certain technique requires access to a global obstacle map of the environment prior to performing coverage, or if it is capable of achieving coverage of an unknown environment. This characterization is often differentiated as offline coverage versus online coverage (Choset 2001).

When operating in unknown environments, it is impossible to enforce a criterion of optimality for deterministic

Fig. 1 Our fixed-wing UAV landing after a coverage session

strategies. This can be shown by modeling a given coverage strategy as a reactive decision tree, determining the choices needed to satisfy a specific optimality criterion, and then constructing a counter-example environment that, by design, forces the strategy to make the opposite choice and hence to violate optimality. Nevertheless, during coverage of unknown environments, it is important in terms of efficiency to avoid repeat coverage as much as possible.

On the other hand, an efficient coverage algorithm for known environments should be able to generate a coverage path that completes the task in the minimum amount of time or the shortest traveled distance, depending on the specific requirements of the application domain. The distinction between distance and time has practical importance when operating under the influence of external disturbances, for example, when flying an aerial robot in strong wind conditions. This distinction will be highlighted in the empirical evaluations of the current work.

Another distinction relates to the ability to ensure complete coverage, i.e. whether a strategy can guarantee that the entire free space within a bounded region will be covered. Complete coverage strategies often employ rigorous representations of the free space, such as exact cellular decompositions (Choset and Pignon 1997), or approximate (e.g. grid-based) cellular decompositions (Gabriely and Rimon 2001).

In this work we present an algorithm for complete and efficient coverage of a two-dimensional known region with arbitrarily-shaped obstacles. Our approach uses the boustrophedon cellular decomposition (BCD) method (Choset and Pignon 1997) to divide the free space into non-overlapping cells, and then uses the solution to the Chinese postman problem (CPP) (Guan 1962) to determine the optimal order in which cells should be covered.

We extend the classical boustrophedon decomposition method by allowing certain cells to be divided into half-cells, so as to ensure that no individual cell will be covered twice. Our solution also guarantees that the coverage session will terminate at the starting location, which makes it convenient for practical applications. As in most previous treatments of coverage, our strategy assumes that the robot operates with perfect localization.

Our coverage algorithm was first presented in Mannadiar and Rekleitis (2010) and assumed originally that the robot could perform in-place rotations. We have subsequently extended this strategy in Xu et al. (2011) to accommodate the general class of non-holonomic vehicles as well. Our system represents the computed coverage path using a sequence of waypoints, which is then realized by a robot-specific motion controller that accounts for the vehicular dynamics of the target platform. Because waypoint-based control can result in less maneuverability compared to velocity-based steering, our approach therefore cannot guarantee coverage optimality, in terms of minimal travel distance, for all possible types of robotic vehicles.

The key contributions of our work include a polynomial-time complete coverage algorithm for known two-dimensional regions, and extensions of this core algorithm to account for non-holonomic vehicular dynamics. In addition, we present thorough experimental assessment of our system to validate its correctness, as well as to investigate the effects of various system parameters on the overall coverage quality. Results are shown through over 200 km of in-field flight sessions using a fixed-wing unmanned aerial vehicle (UAV), see Fig. 1, in addition to extensive testing using a six degrees of freedom (DOF) high-fidelity flight simulator operating within various controlled environments.

2 Background

2.1 Coverage

The seed spreader algorithm (Lumelsky et al. 1990) describes an efficient, deterministic, and complete coverage strategy for simple regions, by having the robot move in back-and-forth, "lawn mower", sweeping motions. Choset and Pignon introduced a rigorous extension to the seed spreader algorithm under the name of BCD (Choset and Pignon 1997; Choset 2000), which guaranteed complete coverage of bounded environments. The work was further developed by Acar et al. (Acar et al. 2002; Acar and Choset 2002) with experimental verification and for a variety of control Morse functions (Forman 1998). The boustrophedon family of algorithms ensures the complete coverage of an unknown environment, although none of these algorithms provide any guarantees on the optimality of the coverage path, in contrast to our proposed approach.

Butler (1998) described a complete coverage strategy for unknown rectilinear environments using a square robot with contact sensing. This algorithm performed an online decom-

position of free space, where each resulting rectangular cell could be covered using back-and-forth seed spreader motions that are parallel to the walls of the environment.

Huang (2001) employed a different strategy towards coverage: the environment is partitioned with the objective of minimizing the amount of rotation the robot has to perform, rather than minimizing the total traveled distance. Yao (2006) subsequently proposed an improved strategy that further reduced the total length of the coverage path, although only illustrative examples were presented without a formal proof of optimality. More recently, Kang et al. (2007) devised an approach where a set of pre-calculated motion strategies are chosen in order to minimize repeat coverage. All of the aforementioned works presented their results in simulation, and thus sidestepped pragmatic and implementation-related concerns using actual robotic hardware.

Gabriely and Rimon (2001, 2002) employed a grid-based approach for planning a complete coverage path using a spanning tree formulation. The primary requirement for a solution to exist is that the environment must be decomposable into a grid with a pre-determined resolution. For known terrain, the grid-based coverage algorithms by Zheng et al. (2005) guaranteed a performance of at most eight times the optimal cost. Similar to Gabriely and Rimon (2002), coverage was also achieved by partitioning the environment into a grid, where each cell had the size of four footprints of the robot. The coverage strategy presented in our work eliminates the grid restriction on the environment and also guarantees that the generated path will be efficient in terms of minimizing path overlap over previously covered areas.

Agmon et al. (2008) extended the spanning tree techniques discussed previously to achieve efficient grid-based coverage using a team of multiple robots. A similar multi-robot spanning tree coverage formulation by Fazli et al. (2010) was applied over a general cell-based representation of free space.

Others have applied genetic algorithms (Jimenez et al. 2007) and visual landmarks (Weiss-Cohen et al. 2008) to improve the speed of coverage. Paull et al. (2010) investigated coverage for underwater vehicles by applying an information-theoretic path planner on a hexagonal grid representation of free space. Easton and Burdick (2005) presented a variant of the CPP to solve the problem of boundary coverage, in which the objective was to cover the immediate area around the boundary of obstacles within the environment. Additional coverage algorithms are outlined in Choset (2001) and also in a more recent survey on multi-robot coverage (Rekleitis et al. 2008).

The aerial robotics community has developed a number of systems that either directly achieve coverage, or use similar techniques to address related applications such as search. Gaudiano et al. (2003) investigated swarm intelligence and UAV control, with extensions to collaborative aerial coverage. Agarwal et al. (2006) proposed a multi-UAV coverage solution for a rectilinear polygonal environment that focused on assigning partitions of the environment proportionally based on each vehicle's capabilities, although computing explicit trajectories was not within their scope. Maza and Ollero (2007) proposed a similar technique for distributed coverage of a polygonal environment with no obstacles. Their algorithm divided the free space into simple regions, and focused on selecting a per-region coverage pattern that minimized the number of turns.

Ahmadzadeh et al. (2006a,b) investigated the problem of coverage-related aerial surveillance, with the goal of minimizing the amount of uncovered area given specific time constraints. Their work addressed explicit concerns for vehicle dynamics and sensor range, and explored a wide variety of different solutions to achieve surveillance. Their particular problem formulation precluded, however, any analysis on the completeness of the terrain coverage. In related work, Cheng et al. (2008) presented a solution to 3-D complete coverage by projecting the environment into non-planar surfaces. Their method also focused on designing a time-optimal trajectory within each surface manifold, which is devoid of obstacles.

Several solutions to search-related problems generated motion strategies that can be readily extensible to achieve coverage (Furukawa et al. 2006; DasGupta et al. 2006). These methods employed probabilistic models of the abstract problem to ensure robustness, however their use of approximation techniques could potentially impede coverage efficiency in practical implementations.

In the literature, the term 'coverage' has also been applied to the problem of distributing a group of mobile sensor units within an environment such that their positioning achieves maximum coverage of an area of interest. Most of these approaches assume that the mobile units do not move after reaching their desired positions, unless the configuration of the environment is altered during runtime. Cortes et al. (2004), and Martinez et al. (2007) proposed a sensor unit placement strategy that is based on utilizing the centers of Voronoi cells. Howard et al. (2002) employed artificial potential fields to force sensor units to move away from each other in order to increase the total coverage area. More recently, Schwager et al. (2009) devised an unifying scheme for multi-robot coverage which combines the previous techniques. The primary emphasis of these works is on determining the end-poses for a team of sensor units, with surveillance or similar applications in mind. In contrast, motion planning coverage strategies, such as the one presented in this paper, focus on generating a motion path through the environment that increases the coverage of a given environment over time, and are geared towards a different set of applications such as de-mining, vacuum cleaning, and search and rescue.

2.2 Graph theory

Various algorithms from graph theory have been applied in robotics to guide exploration (Meger et al. 2008), mapping (Rekleitis et al. 2001; Choset and Burdick 1995), and coverage (Choset 2000). Edmonds and Johnson (1973) presented an overview of a number of graph algorithms that are directly applicable to the problem of complete and efficient coverage of a known environment. Several of these concepts presented in Edmonds and Johnson (1973) are used as core components to our cell-based coverage technique, and are outlined next.

An *Eulerian circuit* is a cyclic graph traversal that visits every edge in a given graph exactly once, and terminates at the starting vertex. Euler demonstrated that a necessary and sufficient condition for the existence of such a cycle is that all vertices must have an even degree—such graphs are termed *Eulerian*. The task of computing an Eulerian circuit is related to the *CPP*, which aims to find the shortest closed path that traverses through every edge *at least* once. If a graph satisfies the Eulerian property, then all of its Eulerian circuits are solutions to the CPP. In contrast, for non-Eulerian graphs, a standard approach to solving the CPP is to first duplicate certain edges in the graph in order to satisfy the Eulerian property, and then choose one of the resulting Eulerian circuits as the solution.

Different strategies can be applied to determine which edges to duplicate, with the objective of minimizing the sum of individual costs associated to every edge in the resulting graph. The constraints and objective function of one such strategy based on a linear programming formulation is described in Edmonds and Johnson (1973). An alternative group of edge duplication strategies use algorithms from *matching theory*: one straight-forward approach is to compute the pairwise shortest paths between every node of odd degree in the original path, and then identify edges to be doubled based on the total costs for each of the competing path options.

After duplicating the selected edges, there are multiple methods to generate an Eulerian circuit. A simple but inefficient technique is to compute a connected edge sequence by greedily visiting and then removing contiguous edges that do not render the graph disconnected; this process continues until all edges have been removed. More efficient algorithms operate by iteratively constructing disjoint cycles that collectively span all the edges, and then merging these cycles together.

3 Efficient complete coverage algorithm

This section presents an algorithm for generating a piecewise-linear path that results in efficient and complete coverage of a known, bounded 2-D environment with arbitrary obstacles. The proposed approach operates hierarchically in two stages: first, an offline analysis is carried out by decomposing all the free space to be covered into cells and computing a traversal ordering through these cells using the solution to the CPP. Second, an online per-cell coverage planner produces motion strategies to completely cover individual cells based on the robot's current state, while following the previously computed cell-level cyclic path. Each of these components will be described in detail below.

3.1 Boustrophedon cellular decomposition

The input to the offline analysis stage is a binary rasterized obstacle map, where black and white pixels differentiate between obstacles and free space that needs to be covered. The free space is partitioned into *cells* using the BCD (Choset and Pignon 1997), which was originally applied to achieve coverage of unknown environments. BCD is a type of Morse decomposition that perceives the environment as a sequence of columns, termed *slices*; in our setup each slice consists of a 1-pixel wide vertical strip of the map. Each slice is composed of one or more non-overlapping sections of free space that are separated by obstacles, and the total number of sections in a slice is called the slice connectivity count. As these disjoint free space sections are identified, *cells* are formed by grouping connected regions of free space spanning across multiple slices. Points on obstacles which cause a change in the connectivity count are called *critical points* (Acar and Choset 2002), as illustrated at the left end of the obstacle in Fig.2a.

Algorithmically, cells and critical points are determined by scanning a slice virtually across the obstacle map and looking for changes in the slice connectivity count; without loss of generality, the direction of this scan is defined along the horizontal axis and is termed the *coverage direction*. In this work the notion of a critical point has been extended to include line segments as well, since they can also be responsible for causing a change in the connectivity within adjacent slices. Figure 2b shows the boustrophedon decomposition for a simple environment.

A common assumption in most cellular decomposition methods is that the connectivity across two adjacent slices can change by *at most* a single critical point. Based on our slice-scanning algorithm, this is equivalent to assuming that no two critical points can be aligned vertically, which is often violated in structured or indoor environments. Thus, care must be taken to properly handle the coverage of zero-width cells that result from vertically-aligned critical points.

After applying the BCD, critical points and cells are encoded respectively as vertices V and edges E of a graph representation $G = \langle V, E \rangle$ known as the *Reeb graph* (Acar and Choset 2002; Fomenko and Kunii 1997); see Fig. 2b. The Reeb graph captures the connectivity information between

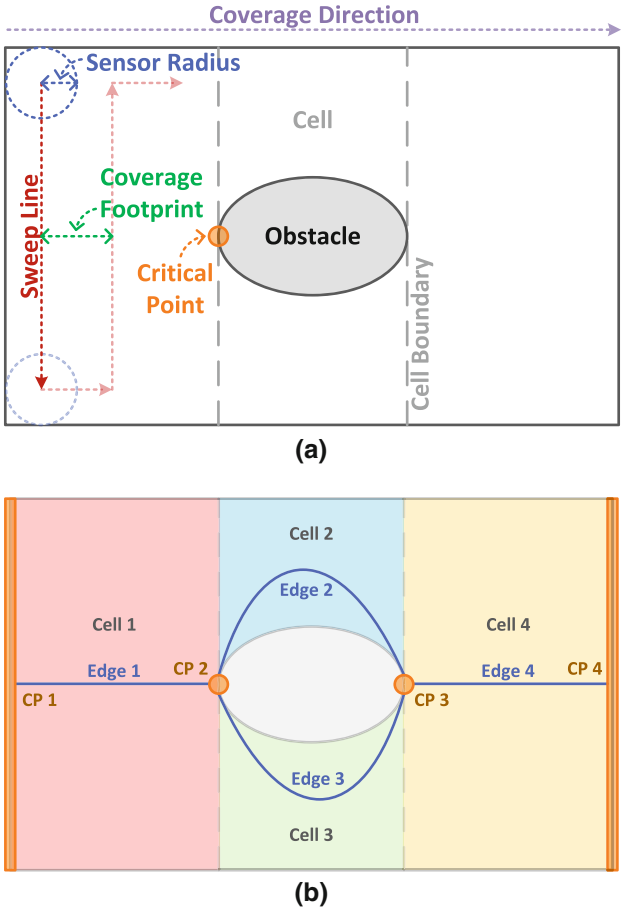


Fig. 2 **a** The entire *free space* is partitioned into *cells* by identifying *critical points* (CP), which are point- and line-intersections with obstacles that are perpendicular to the *coverage direction*. Per-cell coverage is achieved using the seed spreader pattern, i.e. back-and-forth motions composed of parallel *sweep lines* that are uniformly separated by the *coverage footprint width*. **b** Critical points and cells are mapped respectively into vertices and edges of the *Reeb graph* structure. Critical points consist either of point intersections, e.g. CP 2 & CP 3, or line segment intersections, e.g. CP 1 & CP 4

vertices and edges, which is used to compute a cyclic path through all the edges (i.e. cells). Every vertex V in the Reeb graph has a degree of one or three, since by construction each corresponding critical point is connected to exactly one or three cells.

3.2 Construction of the Eulerian circuit

After partitioning all available free space into cells using the BCD, the resulting Reeb graph is then used as input to the CPP. The goal of CPP is to compute a cyclic path that traverses through all the edges *at least once*. To ensure that the solution to the CPP will be an Eulerian circuit, i.e. a traversal that moves through every cell *exactly once*, certain edges in the Reeb graph must be duplicated. It has been shown that no edge needs to be doubled more than once (Edmonds and Johnson 1973). The solution to the CPP consists of two stages: first identify and duplicate a subset of edges, and then find an Eulerian cycle through the updated graph. The first stage can be posed as the following integer linear programming formulation:

$$\text{Minimize } z = \sum_{e \in E} (c_e x_e)$$

subject to the following constraints:

$$\sum_{e \in E} (a_{ne} x_e) - 2w_n = k_n, \quad \forall\, n \in V;$$

$$x_e \in \mathbf{Z}^+, \quad \forall\, e \in E;$$

$$w_n \in \mathbf{Z}^+, \quad \forall\, n \in V;$$

where $\sum_{e \in E} (a_{ne} x_e)$ is the number of all the edges connected to the node $n \in V$. For the result to be Eulerian, an odd number of edges must be connected to nodes with odd degree and an even number of edges must be connected to nodes with even degree; a_{ne} is 1 if node n is connected to edge e, and 0 otherwise; x_e is the total number of copies of edge e in the solution; w_n is an integer variable that forces $\sum_{e \in E} (a_{ne} x_e)$ to be odd for odd nodes and even for even nodes; k_n is 1 for nodes with odd degree, and 0 otherwise; c_e is a real number representing the cost of edge e, where an edge with high cost would be less likely to be duplicated.

This integer programming formulation can be solved in polynomial time since it satisfies the conditions of total dual integrality (Edmonds and Johnson 1973).

It is worth noting that edges in the Reeb graph correspond to free space cells in the obstacle map. A traversal of the Eulerian circuit will follow every edge of the Reeb graph, which means that the robot will cover every cell of the free space and therefore achieve complete coverage of the environment. To prevent repeat coverage and to improve efficiency, cells corresponding to duplicated edges are divided into non-overlapping half-cells. This strategy ensures that no free space area will be covered more than once. One straight-forward approach for splitting cells is to divide along the mid-points between the top and bottom cell boundaries. Unfortunately, because the mid-points at the left and right cell extremities do not necessarily correspond to the locations of critical points, in certain cases the robot will be forced to inefficiently backtrack through previously covered area after reaching a half-cell's end point.

A more efficient strategy for splitting cells operates by interpolating the ratio of the lengths between the top and bottom portions of each vertical slice through the cell. This method ensures that both half-cells will be physically connected to the critical points of their parent cell. The interpolation process is naturally constrained on the left and right cell extremities by the fixed positioning of the two critical points, and therefore guarantees that the robot will never traverse

through a half-cell when moving from the other half-cell to a critical point. The results of this cell-splitting strategy can be observed in Fig. 3a.

The edge cost c_e is a configurable parameter that determines which edges should be duplicated, so as to ensure that the integer programming solution will produce an Eulerian circuit. It is inefficient to split narrow cells since that would introduce an excessive number of additional turns in the coverage path. It is also preferable to avoid splitting small cells whenever possible. Based on these two criteria, the edge cost is chosen to be:

$$c_e = \frac{(cell\ width)^2}{cell\ area} = \frac{cell\ width}{average\ height\ of\ cell}$$

The final output of the offline analysis stage is an Eulerian circuit that traverses through all connected cells in the environment. Figure 3 illustrates this cell-level traversal ordering for both an outdoor region and an indoor environment.

3.3 Per-cell coverage pattern

After completing the offline analysis, our algorithm generates motion paths that cover each individual cell, following the order indicated by the Eulerian circuit. This hierarchical path-generation approach allows per-cell coverage trajectories to be computed dynamically during an actual coverage session, which is beneficial in domains where the efficiency and completeness properties may be affected by dynamically changing factors such as wind or other environmental conditions.

Per-cell coverage can be achieved efficiently using back-and-forth sweeping motions, which are conventionally referred to as *Boustrophedon*, *Seed Spreader*, or *Lawn mower* patterns, and are well documented in the literature (Lumelsky et al. 1990; Choset and Pignon 1997; Acar and Choset 2002). This trajectory can be represented as a set of waypoints forming parallel and uniformly spaced *sweep lines*, as illustrated in Fig. 2a. The resulting path completely sweeps through the cell, starting from one critical point and ending at the other. Note that these sweep lines are perpendicular to the *coverage direction*, which is the orientation of the coverage progression as the robot travels through free space.

A crucial parameter of the seed spreader algorithm is the *coverage footprint*, which measures the width between consecutive sweep lines. The footprint width should ideally be adjusted to match the width of the sensor that is used to achieve coverage, so as to prevent gaps or overlaps in the sensor data collected during consecutive sweep lines. For example, if a UAV is to record atmospheric readings inside a bounded region of space, then the footprint width should be computed as a function of the atmospheric sensor's range. Note that for non-circular sensor swaths, the sensor width is defined to be the minimal horizontal width of the swath as it traverses vertically through a sweep line, which accounts for the worst-case setup.

3.4 Analysis of completeness

The proposed algorithm guarantees the complete coverage of the free space $\mathcal{FS}$. We will prove the completeness of the coverage in three steps. First, we will prove that the BCD completely divides the free space in cells. Second, we will show how the solution of the CPP on the Reeb graph guarantees that every cell is covered and finally, we will prove that every cell is completely covered using the cycle algorithm (Acar and Choset 2002), a variant of the lawnmower pattern.

Lemma 1 *The BCD is an exact cellular decomposition of the bounded free space.*

Proof $\forall p \in \mathcal{FS}$, p is encountered only once by the slice during the construction of BCD. The placement of p on the slice assigns the point to a single cell (Acar and Choset 2002). Therefore, every point in $\mathcal{FS}$ will be associated with one cell. □

Definition 1 [*Reeb graph*]: is an one to one representation of the BCD, where each cell is represented by an edge, and each critical point is represented by a vertex of the graph.

Lemma 2 *The solution of the CPP results in covering every cell of the BCD.*

Proof The solution of the CPP is an Eulerian circuit that traverses every edge at least once and not more than twice. When an edge of the Reeb graph is traversed, the robot covers that cell. Consequently, each cell will be covered at least once. □

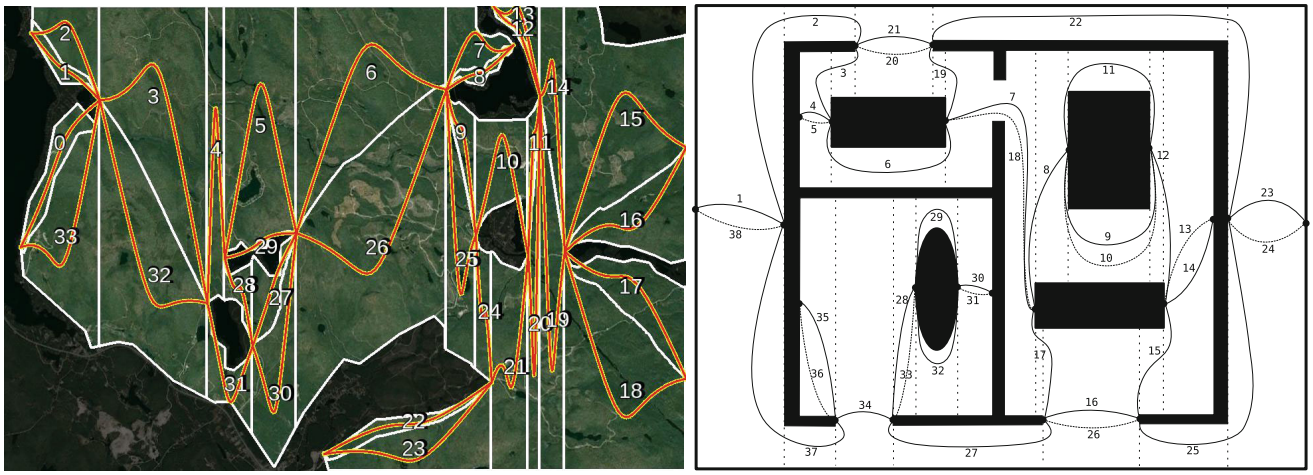
By construction of the algorithm, doubled edges represent two half cells, each one of them covered when the corresponding edge is traversed.

Lemma 3 *Each cell is covered completely when the cycle algorithm is used.*

Proof By construction, see Acar and Choset (2002), the lawnmower pattern when performed over the free part of the configuration space (Choset et al. 2005) $\mathcal{FS}$, will produce complete coverage. □

Proposition 1 *The proposed algorithm will result in the complete coverage of the free space.*

Proof Let us assume that the algorithm did not produce a complete coverage pattern, therefore, a point $p \in \mathcal{FS}$ is not covered during the execution of the algorithm. A point p that is not covered results from either p was not assigned to a cell, contradicting Lemma 1; or the cell that p was assigned to was not covered, contradicting Lemma 2; or during coverage of the cell that point was not covered, contradicting Lemma 3. □



(a) **(b)**

Fig. 3 Results of the offline analysis stage are shown for two different environments: **a** A 13 km × 10 km outdoor region, where *darkened areas* delineate obstacles, *white lines* depict cell boundaries, and *red curves along with cell numbers* indicate the Eulerian circuit. **b** An indoor office space, where *black areas* delineate obstacles, *dashed lines* depict cell boundaries (half-cells are not shown), *solid numbered edges* indicate the Eulerian circuit, and *dotted numbered edges* represent duplicated Reeb graph edges (Color figure online)

In summary, the completeness of our coverage algorithm is ensured by the construction of both the cell-level traversal ordering and the per-cell coverage pattern.

3.5 Analysis of efficiency

The algorithm presented above produces a complete coverage trajectory that systematically covers each cell, or half-cell, in the boustrophedon decomposition of the free space. Nevertheless, a number of fundamental and practical factors will affect the *optimality* of the resulting coverage path, where optimality is defined as minimal path length. This section discusses properties relating to efficiency and optimality that are induced by the two stages of the proposed coverage algorithm, namely the cell traversal ordering and the per-cell coverage pattern.

3.5.1 Efficiency of the cell traversal ordering

After covering a cell, the robot may need to backtrack in order to re-position itself at a reachable corner of the next cell to be covered. This problem arises due to a discontinuity in either the top or bottom boundary curves between two adjacent cells, which might in turn be caused by a line-segment-based critical point. One example of this phenomenon can be observed in Fig. 3a: if the robot finished covering cell 4 at the cell's bottom-right corner, then it would need to backtrack up to the lower-left corner of cell 5 in order to continue coverage. Note that this backtracking process can cost at most one extra sweep line in length.

For certain cell configurations, some of the backtracking can be avoided by switching the order of traversal through two loops in the Eulerian circuit that are connected at the same critical point. Building on this strategy, one option to minimize repeat coverage is to assume a static footprint width and find the best configuration out of all possible Eulerian circuits, i.e. the one with the shortest length disjunctions in the concatenated path. Unfortunately, enumerating through *all possible* Eulerian circuits is a #P-complete problem (Brightwell and Winkler 2004), and even then the best configuration may still contain disjunctions. Given practical considerations, our implementation dynamically selects which subcycle in the Eulerian circuit to traverse next when given a choice, so as to greedily minimize the amount of backtracking required.

It is important to observe that our coverage formulation solves for the optimal cell traversal ordering in an efficient manner, by mapping cells to edges and critical points to vertices. This can be contrasted to the alternative mapping of cells to vertices and critical points to edges that is employed by a number of other coverage formulations in the literature. In our representation, the optimal traversal ordering is equivalent to the Eulerian circuit, and as discussed previously, can be computed efficiently in polynomial time by solving the CPP (Edmonds and Johnson 1973).

3.5.2 Efficiency of per-cell coverage pattern

It is known that the footprint width of the seed spreader algorithm, i.e. the spacing between consecutive sweep lines, can affect the overall quality of coverage (Mannadiar and

Rekleitis 2010). Therefore, the number of sweeps required to cover a cell can be calculated by dividing the length of the cell by the footprint width. Since the remainder of this division is usually not zero, the last sweep is thus often narrower than the coverage footprint width, which results in some repeated coverage. To address this concern, the footprint width is reduced to ensure that the cell's area can be covered using an integer number of sweep lines. Although this strategy reduces coverage efficiency by increasing the overall trajectory length, its benefits include increased robustness of the coverage pattern to small motion perturbations, and even distribution of the redundant coverage across all of the sweep lines in the generated path. Furthermore, the increased overlap in sensor readings may also be beneficial to post-processing activities such as mosaicking and temporal data filtering. Section 4.1 will discuss practical heuristics for minimizing the amount of coverage overlap, by considering different orientations for the coverage direction.

3.5.3 Asymptotic coverage optimality

As discussed above, the cell traversal ordering and the per-cell coverage pattern can each generate a partially redundant sweep line in the worst case. Although these sweep lines are detrimental to the coverage optimality for a given environment and setup, the percentage of repeat coverage per cell will asymptotically go to zero as the ratio of all area to footprint goes to zero, e.g. as the size of the environment grows. Consequently, the proposed coverage algorithm is asymptotically optimal.

More formally, let us consider the case of a single cell with an length w, the distance along the direction of coverage, average height h, and a sensor of radius r, which results in a footprint width $d = 2 \times r$. The number of stripes required to cover this cell are $N = \lceil w/d \rceil$.

Definition 2 redundant coverage ratio RCR is the ratio of the distance traveled in the worst case, over the minimum distance required to cover the cell. $RCR = \frac{(N+2)\times h}{N\times h}$

Lemma 4 *As the area to be covered increases asymptotically in relation with the footprint width (d), RCR decreases asymptotically to 1.*

Proof By definition $RCR = \frac{(N+2)\times h}{N\times h} = \frac{(N+2)}{N} = 1 + \frac{2}{N}$. Therefore, $\lim_{N\to\infty} RCR = 1$. □

4 Aerial coverage

This section presents a number of pragmatic aspects to consider when deploying the proposed coverage algorithm on a vehicle with non-holonomic constraints. The computed piecewise-linear waypoint trajectory cannot be readily executed by a non-holonomic vehicle, such as an automobile, a fixed-wing aircraft, or a boat, since these vehicles cannot reliably carry out the on-the-spot rotations required by these trajectories. Therefore, supplementary motion control strategies are needed to ensure that such robotic vehicles will be able to follow the specified coverage path in its entirety. Another important aspect to consider is the effect of environmental factors, such as wind or current, for airplanes and boats respectively, which may cause unexpected deviations from the designated path. The coverage efficiency can be improved in this case by choosing an appropriate direction of coverage while taking these factors into consideration. Changing the coverage direction may cause the cell decomposition to produce a different result, which may subsequently lead to a lower quality coverage trajectory, where quality relates to the total path length and the total number of turns in the path. In the following sections, these pragmatic considerations will be discussed for the target application of visual terrain coverage using a fixed-wing aerial vehicle.

The task of visual coverage using an aerial robot consists of gathering "bird's-eye view" pictures of the terrain within a bounded region of space. In this setup, the coverage footprint width depends on the camera's field of view, on the UAV's current altitude, and, optionally, on the amount of image overlap needed to reliably stitch the collected images together during post-processing. In these types of aerial applications, one typically deals with environments with known obstacles, since satellite maps of the terrain can be readily obtained.

During aerial operations, the definition of obstacles is more flexible compared to the terrestrial context. Certain terrains and structures such as mountains, high-rise buildings, and even restricted airspace still appear as (pseudo-) physical obstructions. More commonly however, emphasis is given to application-specific areas for which coverage is not required: for example, during a wilderness search operation for a lost hiker, villages and large bodies of water do not need to be covered since they are irrelevant to the search task. In the following sections, only the latter type of obstacles will be considered.

4.1 Direction of coverage

The BCD algorithm presented in Sect. 3.1 scans through the obstacle map along a specific orientation known as the *coverage direction*. This parameter directly impacts the shape of the cells resulting from the BCD, and subsequently, both the length of sweep lines as well as the number of turns in the seed spreader pattern. Thus, the quality of the trajectory can be improved by searching through different orientations of the coverage direction, while attempting to minimize the number of sweep lines and maximize the individual lengths of each sweep line. This can be especially desirable when

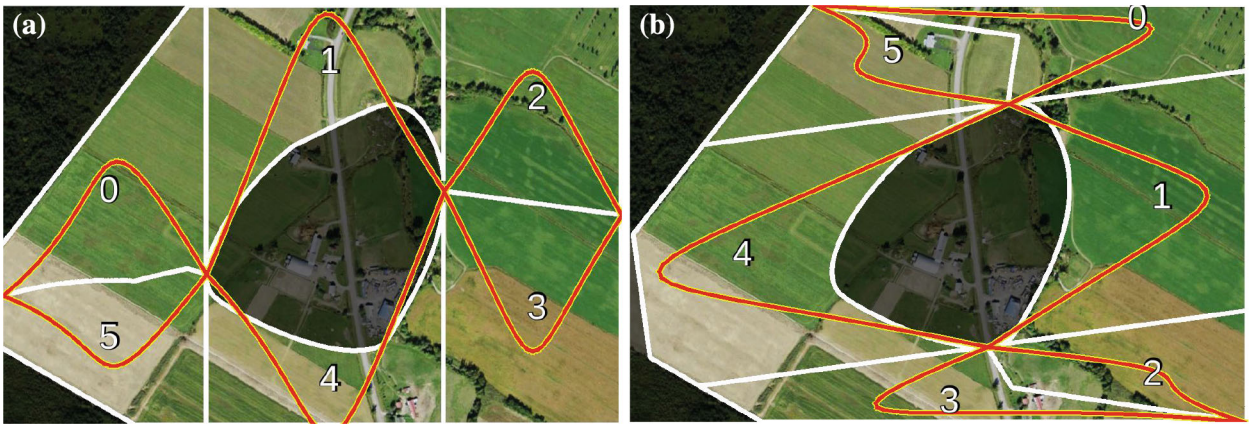


Fig. 4 Results of offline analysis using **a** the default direction of coverage, and **b** after aligning with the dominant axis of the free space, demonstrating a difference in the cell shapes produced by the BCD

working with non-holonomic vehicles (Huang 2001), since they cannot perform in-place rotations and also exhibit poor turning dynamics in general. Figure 4 illustrates that changing the direction of coverage prior to the construction of the BCD can reduce the presence of sharp cusps and produce more elongated cells, thus decreasing the number of turns. We next present three different strategies for selecting the direction of coverage, each of which rotates the map before running the BCD algorithm.

The first strategy for choosing a favorable coverage direction is inspired by scenarios that require the robot to navigate through a rectilinear ("Manhattan-like") environment, where most obstacles are aligned at right angles with each other. If the coverage direction is not aligned with these obstacles, then the boustrophedon decomposition process will generate small triangular cells for many of the room corners in the environment. This is problematic because some of cells may be too small to be covered efficiently using the seed spreader pattern; in these cases, skipping over such cells would lead to tiny amounts of missed coverage. This issue can be mitigated by aligning the direction of coverage to the dominant edge orientation for obstacle boundaries prior to decomposing the environment. In outdoor aerial coverage applications, this alignment strategy can produce notable improvements in the coverage quality when operating above man-made terrains such as farm plots or cities, which often contain straight or near-straight obstacle boundaries.

Another related strategy is to align the direction of coverage with the distribution of the free space, under the assumption that the length of the resulting sweep lines will be maximized along the dominant axis of the free space for environments with either few or small obstacles. Specifically, given the eigenspace decomposition of the entire reachable free space, the coverage direction is set to be orthogonal to the dominant eigenvector. This ensures that cells produced by the BCD are mostly elongated and *parallel* to the coverage direction, which in turn minimizes the number of turns and also leads to longer sweep lines in the coverage path. This is illustrated by the sample decomposition shown in Fig. 4b.

The third strategy directly addresses effects of certain environmental disturbances that can affect the quality of coverage: for example, when operating within a windy zone, strong crosswind can constantly divert the UAV from the designated sweep line trajectories, and will lead to uncovered gaps in the sensor readings. In these situations, aligning the direction of coverage to be perpendicular to the average wind heading prior to launch will shift most of the crosswind force into headwind. Although it may consequently be more challenging to fly against stronger headwinds during parts of the coverage session, this solution mitigates the presence of course deviations leading to violations of coverage completeness.

4.2 Non-holonomic vehicle control

The coverage trajectory produced by our algorithm is represented as a piecewise-linear path, or equivalently as a sequence of waypoints. Since this path contains point-based turns, some of which may be at very sharp angles, it is best geared towards a holonomic robot, i.e. one that can perform in-place rotations. In contrast, non-holonomic vehicles such as fixed-wing UAVs lack the maneuverability needed to accurately follow the designated coverage path in its current waypoint-based format. If a robot is incapable of traversing exactly through the designated trajectory, then the sensor readings collected during coverage may miss certain portions of the desired coverage area. To address this concern, our system employs a robot-specific motion controller to implement the computed coverage path, and in the case of a non-holonomic robot, additional motion segments are used to guide the vehicle along the designated path accurately and efficiently. Although the following steering strategies are

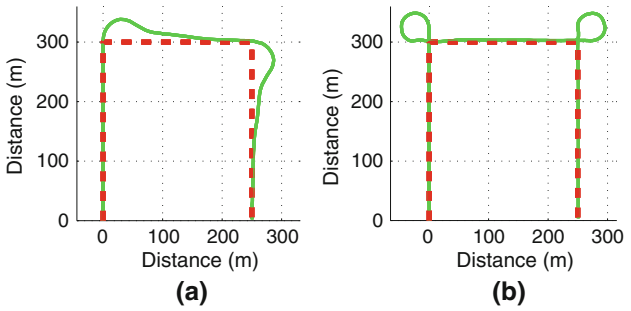


Fig. 5 Strategies for steering non-holonomic vehicles along the designated path (in *red*): follow waypoints from the path directly using **a** a *greedy* path planner, or **b** by adding *curlicue* orbits at corners (Color figure online)

Fig. 6 Our robotic platform is a commercial fixed-wing UAV with an on-board autopilot microprocessor and a gimbal-mounted camera

presented for the control of fixed-wing aircrafts, they can also be readily extended to other non-holonomic robots such as autonomous airboats, as discussed in Girdhar et al. (2011).

Certain robots, including fixed-wing aircraft, offer integrated motion planners based on waypoint commands. These planners typically employ a *greedy* strategy, by issuing a maximum turning rate until the vehicle aligns its heading to the destination. Figure 5a illustrates this behavior as the vehicle turns around to transition between consecutive sweep lines. Unfortunately, implementing the generated coverage path using a greedy path planner directly may compromise coverage completeness. In the illustrated example, although the UAV had a minimum turning radius of 45 m, the greedy planner caused a course deviation of about 80 m at the beginning of the second sweep line. One straight-forward method to resolve this issue is to reduce the footprint width in order to compensate for these deviations. Unfortunately, this uniform increase in sensor overlap would also result in redundant and wasteful coverage efforts.

Figure 5b depicts an alternative corner-steering strategy that allows non-holonomic robots to remain on the designated coverage path. Rather than carrying out a turn directly, the robot is steered in a circular orbit, in order to align it with the upcoming line segment. The orbit's size is determined by the vehicle's minimum turning radius, and the orbit is positioned appropriately to be tangential to the two line segments forming the turn. This *curlicue* strategy is well suited for most turning angles, although it may not be required for mild turns where the robot can adjust its course with negligible deviation.

5 Experimental validation

This section presents extensive evaluations of our efficient complete coverage solution. These evaluations are meant to validate the efficiency of the realized coverage trajectory, as well as to investigate the impact of various system parameters on the quality of coverage. Our evaluation is carried out using three separate methods: an exhaustive analysis of the waypoint-based path produced by our two-stage coverage algorithm; an assessment of aerial coverage in controlled environments carried out using the Aviones (2013) 6-DOF UAV simulation software; and finally, through field trials with a fixed-wing UAV, demonstrating the correctness of our entire coverage system in practical field deployment scenarios.

In these evaluations, performance is quantified by the total trajectory length, as well as the elapsed time for each coverage session. Although these two metrics are correlated, the total flight distance provides a more theoretical assessment of the coverage path, whereas the flight duration relates to the practical quality of a given coverage session, since it is influenced by circumstantial factors like robot battery levels and external conditions such as wind force.

5.1 Hardware platform

The UAV shown in Fig. 6 is a rigid body fixed-wing plane commercially available from Procerus® Technologies.[1] Its 1 m wingspan is built using expanded polypropylene foam, which is useful for absorbing impact upon touchdown. A brushless electric motor powered by lithium polymer batteries can drive the UAV at average ground speeds of 14 m/s and for flight durations of up to 30 min.

This vehicle is controlled by an on-board autopilot microprocessor, which receives instructions from the ground control software wirelessly. The autopilot is connected to various sensors, including a three-axis accelerometer, a three-axis gyroscope, a pitot tube, and a GPS unit. In addition, an on-board camera attached to a pan-tilt gimbal device transmits live analog video at 30 Hz via a separate radio frequency. This camera is equipped with a lens that has a standard 46° field of view. The UAV can operate in a number of modes, ranging from joystick-based manual control to fully autonomous waypoint-based navigation.

[1] www.procerusuav.com.

The proposed coverage algorithm is implemented in C++ under a Linux environment on-board a 1.66 GHz dual-core laptop, and communicates via Ethernet with a second computer acting as the UAV's ground control unit. During flight, the gimbal's orientation is continuously regulated, in order for the camera to be oriented parallel with the ground plane. This removes the need to transform the acquired images using projective geometry and also establishes a uniform pixel density. The reader is referred to Xu and Dudek (2010) for details on the gimbal control algorithm.

All of the UAV experiments presented in this paper were conducted at fixed altitudes, so as to facilitate the analysis of results. It is important to note however that the online nature of our per-cell coverage strategy is perfectly capable of adjusting the coverage footprint dynamically, in order to optimize the seed spreader pattern based on the UAV's altitude at the beginning of each cell.

5.2 Large scale simulation

This experiment evaluates the scalability of our algorithm by carrying out complete aerial coverage sessions over a large 13 km × 10 km environment, filled with many curved and polygonal obstacles in both convex and concave shapes. The two sessions shown in Fig. 7 were carried out in simulation, because our UAV platform lacked sufficient battery capacity to sustain such a flight, and also more importantly to ensure that the greedy and curlicue motion controllers can be evaluated under a controlled environment without any external disturbances such as wind or rain. Nevertheless, to remain faithful to our UAV's capabilities, the coverage footprint was calculated based on a typical operational altitude of 300 m and with our camera's field of view of 46°.

As discussed in Sect. 4.2, a naive way to compensate for the potential missed coverage caused by executing the greedy path planner on the computed coverage path is to decrease the *effective* coverage footprint width. Given the operational altitude and camera settings, and knowing that our UAV's minimum turning radius in the absence of wind is approximately 50 m, the effective coverage footprint should ideally be set to 70 % of the original width, in order to prevent gaps in the sensor readings. Unfortunately, simulation results using this setting nearly doubled the length of the original path, even in the absence of wind. In light of this result, we opted to compare the two control techniques using the same footprint width. This effectively compares the difference in performance between a non-holonomic vehicle such as a fixed-wing aircraft executing curlicue patterns, versus a holonomic robot such as a helicopter following the original waypoint-based path. Since this experiment was conducted using the fixed-wing UAV simulator, the latter results are only an approximation of the true performance of a holonomic vehicle.

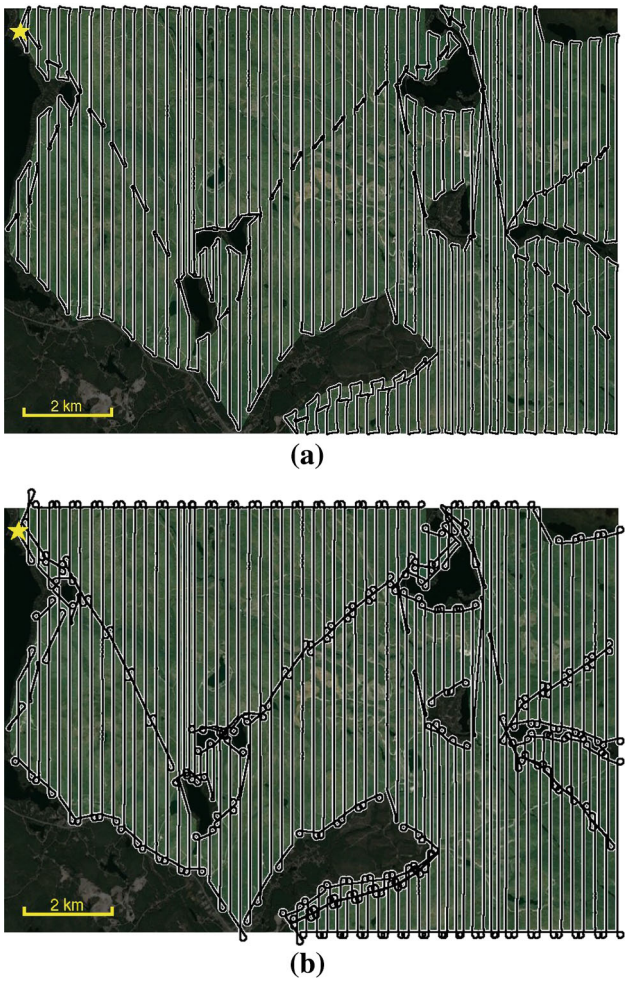


Fig. 7 Simulated coverage paths for a 13 km × 10 km region at 300 m altitude with no wind. *Star* denotes beginning and end of flight. The UAV traveled **a** 590.9 km in 11 h 56 m using the greedy path planner, and **b** 740.7 km in 14 h 56 m using the curlicue strategy

The flight paths recorded from both sessions are shown in Fig. 7. Under ideal experimental conditions, the elapsed times and total flight distances both indicate a 25 % increased penalty for using the curlicue strategy. Depending on the target application domain, it may be reasonable to accept this increase in additional resources needed to achieve complete coverage of a large-scale environment.

For the flight session using the curlicue controller, there were multiple occasions where the UAV unnecessarily performed curlicue turns at corners that were located between two free space cells. This can be observed from the flight trajectories in Fig. 7, and notably by comparing the flight behavior along the boundaries of half-cells on the right side of the environment. One simple heuristic to prevent such redundancies would be to switch to using the greedy waypoint controller at these corners, since the portions of missed coverage resulting from a given turn would be covered later while moving through the adjacent cell.

Table 1 Simulated coverage results for a 1 km × 0.6 km region at 150 m altitude operating under different coverage directions

	Alignment of coverage direction			
	Random	Obs. edges	Free space	Wind
Time (no wind)	21 m 54 s	21 m 59 s	21 m 26 s	–
Dist. (no wind)	13.61 km	13.69 km	13.36 km	–
Time (7 m/s wind)	28 m 57 s	25 m 23 s	27 m 13 s	27 m 39 s
Dist. (7 m/s wind)	15.14 km	13.24 km	14.46 km	15.19 km

Finally, it is worth noting that the penalty for using the curlicue controller will depend on the size of the coverage region. This can be seen by comparing the length of the sweep lines with that of the circular orbits introduced by the curlicue controller, leading to the conclusion that this controller would be inefficient for small-sized environments. Consequently, in these situations it may be more cost effective to decrease the effective footprint width and use the greedy waypoint controller instead.

5.3 Alignment strategies for coverage direction

The best direction of coverage, assuming that such a value exists, is highly dependent on the obstacle layout for a given environment. In order to obtain some intuition on the overall effect of different alignment strategies previously introduced in Sect. 4.1, several coverage sessions using the Aviones flight simulator were conducted over a simple terrain with one obstacle, while operating along different coverage directions.

Table 1 enumerates the elapsed times and flight distances resulting from changing the direction of coverage using various alignment methods. In the first set of experiments, the UAV simulations were carried out in the absence of wind, in order to determine the isolated effects of the different alignment strategies. The resulting paths exhibited nearly identical performance, arguably due to the simplicity of the chosen terrain. On the other hand, these results also suggest that the direction of coverage had minimal impact on coverage efficiency in practice, at the scale of operations relevant to our particular UAV platform, and in the absence of strong external disturbances.

Next, the same three coverage sessions were repeated after introducing a constant 7 m/s wind force in a random direction. These simulations produced similar flight behaviors compared to field trials operating under comparable wind conditions. A separate coverage session was also carried out after aligning the direction of coverage to be perpendicular with the wind heading. Looking at the second row of elapsed time results in Table 1, the presence of wind appeared to have consistently increased the duration of coverage in all four sessions.

Interestingly, in the very last flight session, where the coverage direction was aligned to be perpendicular to the wind direction, the vehicle's speed oscillated between 20 m/s while flying along the wind and 10 m/s while flying against the wind. Although these ground speeds were still within operational range during this particular session, a stronger wind force might have destabilized the UAV critically. Also, even when the strong headwind conditions did not cause the UAV to stall, it would nevertheless significantly impede the UAV from executing the sweep lines in a timely manner. Therefore, this flight session revealed that the wind alignment strategy may exhibit undesirable properties in practice when operating under adverse environments, especially during time-critical scenarios.

As another notable result, looking at the flight distances for the latter set of experiments, the coverage session using alignment with obstacle boundaries produced a noticeably shorter trajectory compared to previous results under no wind conditions. Upon further inspection of the data however, the crosswind can be seen in Fig. 8a to have significantly warped the back-and-forth coverage patterns. In contrast, the flight trajectory resulting from alignment with the wind direction in Fig. 8b displayed minimal deviation from the original set of waypoints generated by the coverage algorithm. This difference in the quality of coverage can also be observed from the amount of free space area that was not covered in each session, which consisted of 8.2 % missed coverage when using the obstacle edge alignment strategy, compared to 3.3 % missed coverage when using the wind alignment strategy.

The results of this experiment suggest that no single alignment method dominated in terms of performance improvements, even for a simple one-obstacle terrain. Summarizing the results, when operating in environments with little or no environmental disturbances, the choice of alignment ultimately depended on whether the cellular decomposition lead to a reduction in the total number of turns for the generated coverage trajectory. On the other hand, it was observed in practice that aligning the direction of the sweep lines to be parallel to a strong wind force will not result in a significant reduction in flight duration and distance, although it does ensure that the vehicle will follow the designated path more accurately.

5.4 Exhaustive coverage direction analysis

The previous section demonstrated experimentally that the different coverage alignment strategies all lead to some improvements in the quality of the generated trajectory. A related concern that was not addressed however is whether these alignment strategies are capable of choosing a

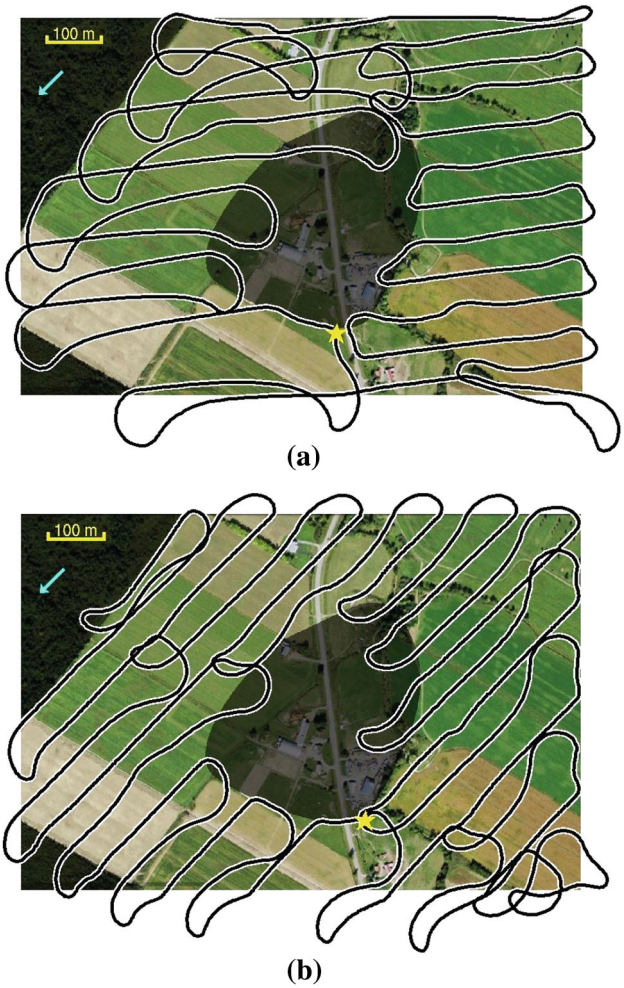


Fig. 8 Simulated coverage paths for a 1 km × 0.6 km region at 100 m altitude with 7 m/s wind. The *arrow* denotes the wind direction, whereas the *star* denotes the start and end of the flight. Our UAV traveled **a** 14.46 km in 27 m 13 s under alignment with obstacle edges, and **b** 15.19 km in 27 m 39 s under alignment with wind direction

coverage direction that will result in the *best* coverage quality. To answer this question, the quality of paths generated using different coverage directions were compared exhaustively, for a number of distinct environments. This evaluation was carried out by contrasting the generated waypoint-based paths directly rather than comparing the executed flight sessions, in order to exclude influences from environmental and other pragmatic factors.

During this analysis, the quality of coverage was measured quantitatively by looking at the number of turns in the generated path. This metric was chosen over the percentage of repeated coverage even though the latter measured coverage efficiency, since a path with minimal repeated coverage could also result in increased missed coverage and thus would not correspond to a higher quality path. In contrast to the previous experiment, the total path length was not used to measure the quality of the trajectory here, since the path length could be affected by pragmatic factors such as wind, which were ignored for this analysis.

Figure 9 depicts the coverage quality as a function of coverage direction, for three different environments. The results for the outdoor environment with one single obstacle depicts multiple near-optimal valleys, whereas the plots for the other two environments with multiple obstacles reveal landscapes that are approximately unimodal. One hypothesis to explain this discrepancy is that whereas each obstacle contributes to the quality of the generated path at different orientation angles, aggregating these individual contributions for an environment with many obstacles leads to a smoother and almost unimodal response in terms of coverage quality. This hypothesis can also be used to explain the contrast between the oscillatory and potentially unstable quality for the single obstacle environment, and the smoother response for the two multi-obstacle terrains.

In addition, although the two alignment strategies produced comparable results to that of the global optimal orientation, in all three cases the obstacle edge alignment strategy showed slightly better performance compared to the free space distribution alignment strategy. One explanation is that the latter variant did not take into account the positions of the obstacles in the environment, which were influential in determining the shapes of individual cells.

In summary, aligning the coverage direction to be orthogonal to the dominant obstacle edge direction has the potential to generate trajectories with near-minimal number of turns, and is thus an effective heuristic for avoiding an exhaustive search for the globally optimal coverage direction.

5.5 Global versus local optimality

When operating in an *unknown* environment, the cell traversal ordering can only be chosen using a *locally optimal* strategy (Choset and Pignon 1997), which is to select the closest unexplored cell to cover next. In this experiment, the local cell selection strategy was compared to the globally optimal cell traversal ordering generated by our coverage algorithm, for the two environments shown in Fig. 9a, b. In particular, the same footprint width was used to generate coverage paths for our globally optimal cell exploration strategy, as well as for the exhaustive set of locally optimal cell traversals corresponding to using different critical points as starting locations.

Table 2 indicates that the globally optimal coverage trajectories were about 10 % shorter compared to the average locally optimal path for both outdoor regions. In fact, for these two sets of results, the globally optimal solutions were below three standard deviations away from the average result using the locally optimal exploration strategy, suggesting that our coverage approach performs better than the vast majority of paths generated using a locally optimal strategy.

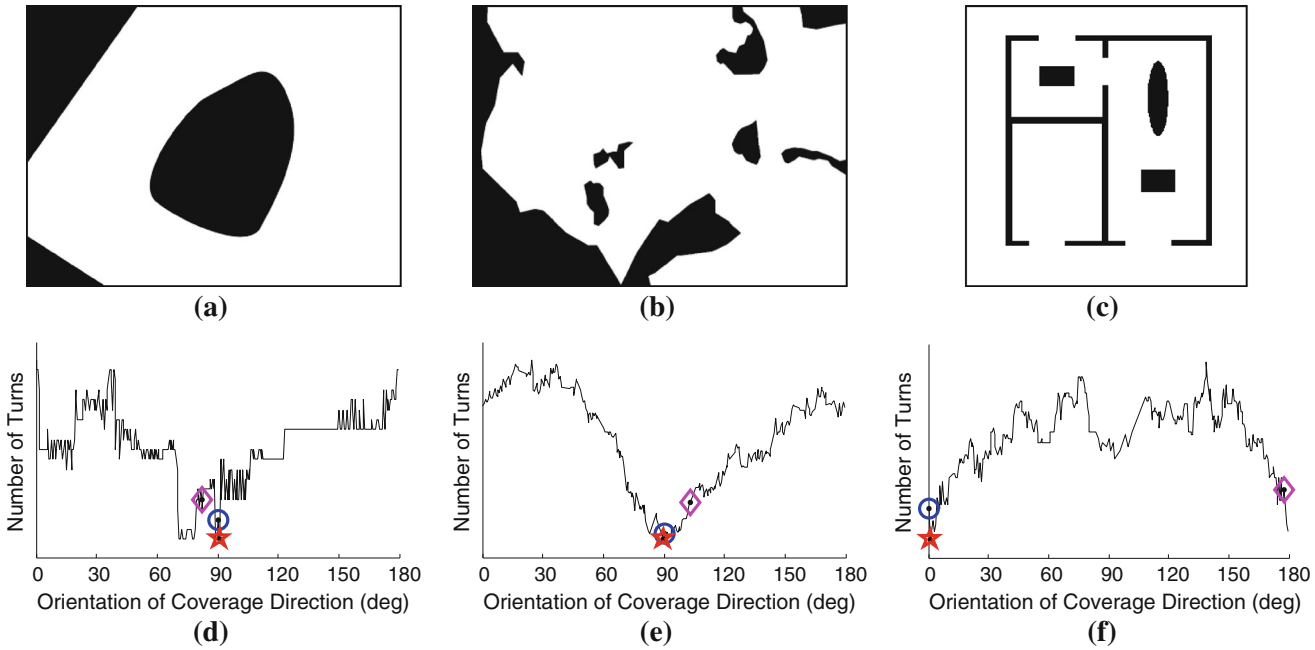


Fig. 9 Various obstacle maps: **a** A 1 km × 0.6 km simple outdoor region with one obstacle, **b** a 13 km × 10 km complex outdoor region with multiple obstacles, and **c** a 0.6 km × 0.6 km structured office space. **d–f** Plots the number of turns in the computed path plotted against coverage direction for these regions: *star* indicates the globally optimal result; *circle* and *diamond* indicate the angles chosen by the obstacle edge alignment and free space alignment strategies

Table 2 Total coverage distances at 200 m altitude, comparing global and local cell-selection strategies

	Simple outdoor region (Fig. 9a)	Complex outdoor region (Fig. 9b)
Dist. (optimal)	8.377 km	974.0 km
Avg. dist. (local)	9.109 km	1008 km
Std. dev. (local)	0.2158 km	5.243 km

5.6 UAV field trial

We carried out extensive field evaluations over farmland terrain resulting in a total flight distance of 213.8 km during numerous coverage sessions. All of the configuration parameters used in our simulated coverage sessions were obtained from real data collected during these field trials.

This study revisits the comparison between the greedy waypoint controller and the curlicue controller in a field deployment setting, with a fixed-wing UAV operating under 5–8 m/s wind conditions. We collected eight *complete* coverage runs for this experiment, resulting in a total of 118.9 km flight distance. Our discussions below focus on two flight instances among our collected field data set.

Observing the resulting trajectories and flight data in Fig. 10, the elapsed times and distances both indicate an 85 % increased penalty for using the curlicue strategy. One of the main causes for the large performance gap can be seen in the curlicue path: rather than flying in circular orbits, the strong wind force persistently pushed the UAV off of its designated coverage path.

The UAV's telemetry collected during these runs were used to estimate the completeness of coverage, by analyzing the percentage of visual overlap among video frames. The flight session using the greedy controller resulted in 66.6 % single coverage, 28.3 % repeated coverage, and 5.1 % missed coverage, as shown in Fig. 10c, whereas the curlicue session resulted in 35.1 % single coverage, 64.1 % repeated coverage, and 0.8 % missed coverage, as shown in Fig. 10d. The increased repeated coverage in the latter session was primarily due to the circular orbits generated by the curlicue controller, which represented a significant addition to the overall path length given the small size of the target environment. In addition, since our per-cell coverage algorithm reduced the effective footprint width in order to establish an integer number of sweep lines per cell, this resulted in a notable amount of coverage overlap in both field sessions. These deficiencies are related to a number of pragmatic factors, such as the size of the coverage environment and the severity of environmental disturbances. The illustrative flight sessions in this experiment were configured specifically to demonstrate the potential severity of these practical concerns.

The two presented coverage sessions were also replicated in simulation using a static wind factor computed

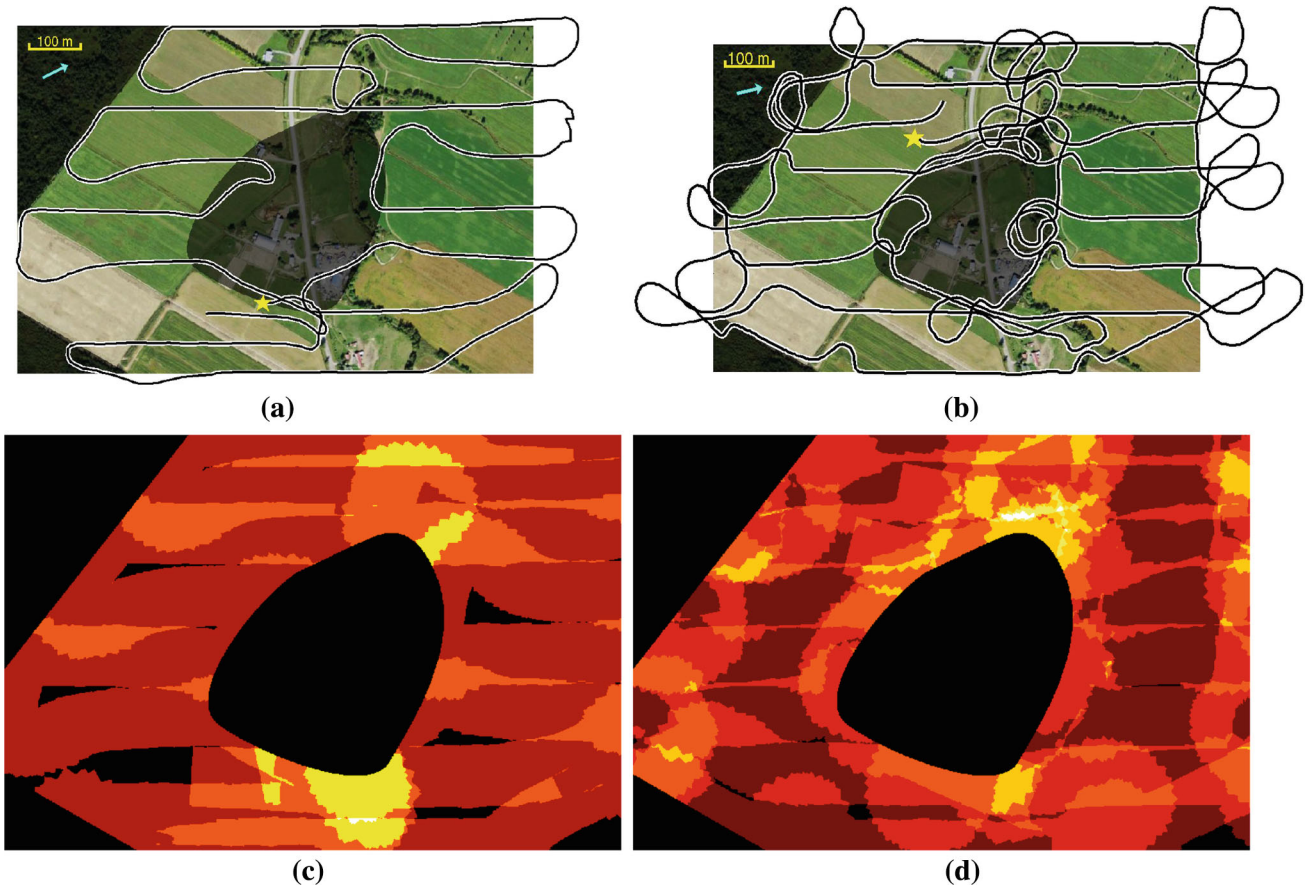


Fig. 10 Actual coverage paths from an UAV field trial for a 1 km × 0.7 km region at 150 m altitude with 5–8 m/s wind. The *arrow* denotes the wind direction, whereas the *star* denotes the start and end of the flight. Both directions of coverage were pre-aligned with obstacle edges. Our UAV traveled **a** 9.10 km in 12 m 48 s using the greedy path planner, and **b** 17.25 km in 23 m 42 s using the curlicue strategy. The corresponding coverage "heat maps" are shown in (**c**) and (**d**), where *bright regions* correspond to repeated coverage, and *black zones* depict obstacles as well as areas with missed coverage

from field data. During these sessions, the UAV traveled 10.2 km in 16 m 19 s using the greedy path planner, and 19.3 km in 30 m 38 s using the curlicue controller. Compared to the results of the field trial, the slightly increased distance and duration values can be attributed to a combination of the inaccuracies in the replication of the setup and of the simulation. More importantly, the performance gap between the two motion controllers remained at about 87 %, which is quite similar to the results obtained in the field trials, and thus further corroborates our field deployment findings.

6 Conclusions

We presented a new path planning algorithm and implementation for the efficient complete coverage of a known environment with arbitrary obstacles. This approach guides a mobile robot through a sequence of areas to be covered, while minimizing the amount of repeated coverage through previously traveled areas. The solution to the CPP was adapted to the computation of the cell traversal. Also, the single cell coverage used in the BCD algorithm was modified to eliminate repeat coverage, by splitting certain cells into two components.

Our experiments comprised of over 200 km of visual coverage flights using a fixed-wing aerial vehicle, together with thousands of kilometers of flight in simulation. Extensive testing validated the robustness and efficiency of the proposed approach, and also highlighted the effects on the quality of coverage of different low-level motion planners, of the direction of coverage, and of environmental factors.

We are currently investigating the effectiveness of alternative motion patterns such as sawtooth and spiral shapes for covering free space regions. We are also continuously expanding our experimental repertoire for a wider range of operating conditions. Finally, we are interested in deploying this solution on-board different types of autonomous robots, in order to extend this solution to heterogeneous multi-robot teams (Shkurti et al. 2012).

Acknowledgments We would like to thank both Microsoft Research and the National Science and Engineering Research Council of Canada (NSERC) for their generous financial support towards this work. We also would like to thank Prof. D. Avis for the useful discussions on graph theory and the CPP.

References

Acar, E. U., Choset, H., Rizzi A. A., Atkar P. N., & Hull D. (2002). Morse decompositions for coverage tasks. *The International Journal of Robotics Research (IJRR '02)*, *21*(4), 331–344 (2002).

Acar, E. U., & Choset, H. (2002). Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *The International Journal of Robotics Research (IJRR '02)*, *21*(4), 345–366.

Acar, E. U., Choset, H., Zhang, Y., & Schervish, M. (2003). Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International Journal of Robotics Research (IJRR '03)*, *22*, 441–466.

Agarwal, A., Hiot, L., Nghia, N., & Joo, E. (2006). Parallel region coverage using multiple UAVs. In *IEEE Aerospace Conference* (p. 8). Big Sky, MT.

Agmon, N., Hazon, N., & Kaminka, G. (2008). The giving tree: Constructing trees for efficient offline and online multi-robot coverage. *Annals of Mathematics and Artificial Intelligence*, *52*, 143–168.

Ahmadzadeh, A., Jadbabaie, A., Kumar, V., & Pappas, G. (2006a). *Multi-UAV cooperative surveillance with spatio-temporal specifications* (pp. 5293–5298). San Diego, CA: Proceedings of the 45th IEEE Conference on Decision and Control.

Ahmadzadeh, A., Keller, J., Jadbabaie, A., & Kumar, V. (2006b). *An optimization-based approach to time critical cooperative surveillance and coverage with unmanned aerial vehicles*. Rio de Janeiro: International Symposium on Experimental Robotics.

Aviones. (2013). *UAV Flight Simulator*. Retrieved June 6, 2013 from http://aviones.sourceforge.net.

Brightwell, G., & Winkler, P. (2004). Note on counting Eulerian circuits. CoRR cs.CC/0405067.

Butler, Z. (1998). *CC R : A complete algorithm for contact-sensor based coverage of rectilinear environments*. Technical Report. CMU-RI-TR-98-27. Pittsburgh, PA: The Robotics Institute, Carnegie Mellon University.

Cheng, P., Keller, J., & Kumar, V. (2008). *Time-optimal UAV trajectory planning for 3D urban structure coverage* (pp. 2750–2757). Nice: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08).

Choset, H. (2000). Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, *9*, 247–253.

Choset, H. (2001). Coverage for robotics—a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, *31*, 113–126.

Choset, H., & Burdick, J. (1995). Sensor based planning, part ii: Incremental construction of the generalized voronoi graph. In *Proceedings of the IEEE conference on robotics and automation (ICRA '95)* (pp. 1643–1648). Los Alamitos, CA: IEEE Computer Society Press.

Choset, H., & Pignon, P. (1997). *Coverage path planning: The boustrophedon cellular decomposition*. Leuven: Proceedings of the International Conference on Field and Service Robotics.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., et al. (2005). *Principles of robot motion: Theory, algorithms, and implementations*. Boston: MIT Press.

Cortes, J., Martinez, S., Karatas, T., & Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics*, *20*(2), 243–255.

DasGupta, B., Hespanha, J., Riehl, J., & Sontag, E. (2006). Honeypot constrained searching with local sensory information. *Nonlinear Analysis*, *65*(9), 1773–1793.

Easton, K., & Burdick, J. (2005). *A coverage algorithm for multi-robot boundary inspection* (pp. 727–734). Barcelona: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '05).

Edmonds, J., & Johnson, E. L. (1973). Matching, Euler tours and the Chinese postman. *Mathematical Programming*, *5*, 88–124.

Fazli, P., Davoodi, A., Pasquier, P., & Mackworth, A. (2010). *Complete and robust cooperative robot area coverage with limited range* (pp. 5577–5582). Taipei: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10).

Fomenko, A., & Kunii, T. L. (1997). *Topological modeling for visualization*. Tokyo: Springer-Verlag.

Forman, R. (1998). Morse theory for cell complexes. *Advances in Mathematics*, *134*, 90145.

Furukawa, T., Bourgault, F., Lavis, B., & Durrant-Whyte, H. (2006). *Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets*. Orlando, FL: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06).

Gabriely, Y., & Rimon, E. (2001). *Spanning-tree based coverage of continuous areas by a mobile robot*. Seoul: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '01).

Gabriely, Y., & Rimon, E. (2002). *Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot* (pp. 954–960). Washington, D.C: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02).

Gaudiano, P., Shargel, B., Bonabeau, E., & Clough, B. T. (2003). *Swarm intelligence: A new C2 paradigm with an application to control of swarms of UAVs*. Copenhagen: ICCRTS Command and Control Symposium.

Girdhar, Y., Xu, A., Dey, B. B., Meghjani, M., Shkurti, F., Rekleitis, I., & Dudek, G. (2011). *MARE: Marine Autonomous Robotic Explorer* (pp. 5048–5053). Algarve: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '11).

Guan, M.-K. (1962). Graphic programming using odd or even points. *Chinese Mathematics*, *1*(3), 273–277.

Howard, A., Matarić, M. J., & Sukhatme, G. S. (2002). *Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem* (pp. 299–308). Fukuoka: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems.

Huang, W. (2001). *Optimal line-sweep-based decompositions for coverage algorithms* (pp. 27–32). Seoul: Proceedings the IEEE International Conference on Robotics and Automation (ICRA '01).

Jimenez, P., Shirinzadeh, B., Nicholson, A., & Alici, G. (2007). *Optimal area covering using genetic algorithms* (pp. 1–5). Zurich: Proceedings of the IEEE/ASME International Conference on Advanced Intelligent, Mechatronics.

Kang, J. W., Kim, S. J., Chung, M. J., Myung, H., Park, J. H., & Bang, S. W. (2007). *Path planning for complete and efficient coverage operation of mobile robots* (pp. 2126–2131). Harbin: Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA '07).

Lumelsky, V. J., Mukhopadhyay, S., & Sun, K. (1990). Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, *6*(4), 462–472.

Mannadiar, R., & Rekleitis, I. (2010). *Optimal coverage of a known arbitrary environment* (pp. 5525–5530). Anchorage: Proceedings of IEEE International Conference on Robotics and Automation (ICRA '10).

Martinez, S., Cortes, J., & Bullo, F. (2007). Motion coordination with distributed information. *IEEE Control Systems Magazine*, *27*(4), 75–88.

Maza, I., & Ollero, A. (2007). Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6* (pp. 221–230). Japan: Springer.

Meger, D., Rekleitis, I., & Dudek, G. (2008). *Heuristic search planning to reduce exploration uncertainty* (pp. 3382–3399). Nice: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08).

Paull, L., Saeedi, S., Li, H., & Myers, V. (2010). *An information gain based adaptive path planning method for an autonomous underwater vehicle using sidescan sonar* (pp. 835–840). Toronto, ON: Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE '10).

Rekleitis, I. M., Dudek, G., & Milios, E. (2001). Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, *31*(1–4), 7–40.

Rekleitis, I. M., New, A. P., Rankin, E. S., & Choset, H. (2008). Efficient multi-robot coverage: An algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, *52*(2–4), 109–142.

Schwager, M., Slotine, J. J., & Rus, D. (2009). *Unifying geometric, probabilistic, and potential field approaches to multi-robot coverage control*. Lucerne: Proceedings of the IEEE International Symposium on Robotics Research (ISRR '09) (2009).

Shkurti, F., Xu, A., Meghjani, M., Higuera, J. C. G., Girdhar, Y., Giguère, P., Dey, B. B., Li, J., Kalmbach, A., Prahacs, C., Turgeon, K., Rekleitis, I., & Dudek, G. (2012). *Multi-domain monitoring of marine environments using a heterogeneous robot team*. Algarve: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '12).

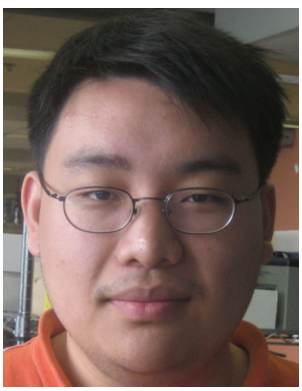
Weiss-Cohen, M., Sirotin, I., & Rave, E. (2008). *Lawn mowing system for known areas* (pp. 539–544). Vienna: Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation.

Xu, A., & Dudek, G. (2010). *A vision-based boundary following framework for aerial vehicles* (pp. 81–86). Algarve: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '10).

Xu, A., Viriyasuthee, C., & Rekleitis, I. (2011). *Optimal complete terrain coverage using an unmanned aerial vehicle* (pp. 2513–2519). Anchorage: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11).

Yao, Z. (2006). *Finding efficient robot path for the complete coverage of a known space* (pp. 3369–3374). Orlando, FL: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '06).

Zheng, X., Jain, S., Koenig, S., & Kempe, D. (2005). *Multi-robot forest coverage* (pp. 3852–3857). Edmonton, AB: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '05).



Anqi Xu is a Ph.D. student studying at McGill University's School of Computer Science under the supervision of Professor Gregory Dudek. He is a member of the Mobile Robotics Laboratory at the Centre for Intelligent Machines at McGill University. He obtained a B.Eng. degree in Computer Engineering and a minor in Software Engineering from McGill University in 2008 and graduated with great distinction. His primary research interests include Human–Robot Interaction, Computer Vision, and robot programming methodologies.

Chatavut Viriyasuthee is a M.Sc. student studying at McGill University's School of Computer Science under guidance of Professor Gregory Dudek. He is a member of the Mobile Robotics Laboratory, at the Centre for Intelligent Machines. He received his bachelor's degree in Computer Engineering from Chulalongkorn University, Thailand where he spent some time developing intelligent machines and participating in many robot competitions. His research interests span Robotics, Computer Vision, and certain topics in machine learning especially graphical models and reinforcement learning.

Ioannis Rekleitis is currently an Adjunct Professor at the School of Computer Science, McGill University. Between 2004 and 2007 he was a visiting fellow at the Canadian Space Agency. Between 2002 and 2003, he was a Postdoctoral Fellow at the Carnegie Mellon University in the Sensor Based Planning Lab with Professor Howie Choset. His research has focused on mobile robotics and in particular in the area of cooperating intelligent agents with application to multi-robot cooperative localization, mapping, exploration and coverage. He has worked with underwater, terrestrial, aerial, and space robots. His interests extend to computer vision and sensor networks. Ioannis Rekleitis has published more than 50 journals and conference papers. He has an H-Index of 23 (using Google Scholar). He has served as Program Chair, Associate Editor, and Program Committee member in several journal and conferences. He is the Principal Investigator for a five year Natural Sciences and Engineering Research Council of Canada (NSERC) discovery grant and a Co-PI on Microsoft Research and NSERC grants. Ioannis Rekleitis was granted his Ph.D. from the School of Computer Science, McGill University, Montreal, Quebec, Canada in 2002 under the supervision of Professors Gregory Dudek and Evangelos Milios. Thesis title: "Cooperative Localization and Multi-Robot Exploration". He finished his M.Sc. in McGill University in the field of Computer Vision in 1995. He was granted his B.Sc. in 1991 from the Department of Informatics, University of Athens, Greece.