Revisiting Boustrophedon Coverage Path Planning as a Generalized Traveling Salesman Problem

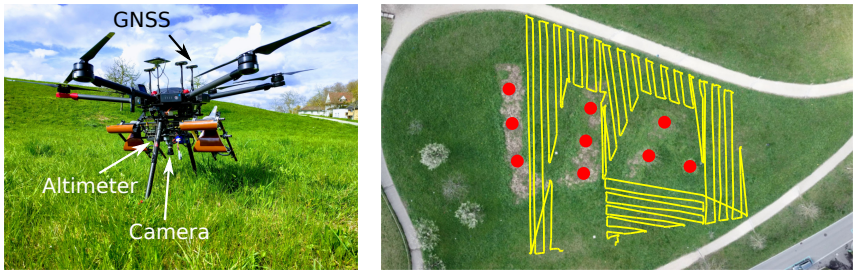
Rik Bähnemann, Nicholas Lawrance, Jen Jen Chung, Michael Pantic, Roland Siegwart, and Juan Nieto

Abstract In this paper, we present a path planner for low-altitude terrain coverage in known environments with unmanned rotary-wing micro aerial vehicles (MAVs). Airborne systems can assist humanitarian demining by surveying suspected hazardous areas (SHAs) with cameras, ground-penetrating synthetic aperture radar (GPSAR), and metal detectors. Most available coverage planner implementations for MAVs do not consider obstacles and thus cannot be deployed in obstructed environments. We describe an open source framework to perform coverage planning in polygon flight corridors with obstacles. Our planner extends boustrophedon coverage planning by optimizing over different sweep combinations to find the optimal sweep path, and considers obstacles during transition flights between cells. We evaluate the path planner on 320 synthetic maps and show that it is able to solve realistic planning instances fast enough to run in the field. The planner achieves 14 % lower path costs than a conventional coverage planner. We validate the planner on a real platform where we show low-altitude coverage over a sloped terrain with trees.

1 Introduction

MAVs such as the DJI M600 shown in Figure 1 present ideal platforms for supporting demining efforts and other critical remote sensing tasks. These commercially available platforms are fast to deploy and can collect useful data that are often inaccessible to other modes of sensing [19]. For demining applications, this means that an MAV carrying a GPSAR system can be flown over a minefield to collect in-situ, high precision measurements for locating buried landmines [22].

Authors are with Autonomous Systems Lab, ETH Zurich, e-mail: {brik, lawrancn, chungj, mpantic, rsiegwart, nietoj}@ethz.ch



(a) The MAV platform. (b) Coverage path over sloped terrain (trees in red).

Fig. 1 Deployment of our landmine detecting MAV. The low-altitude coverage path was generated in the field. The operator defined obstacle boundaries to avoid trees and the planner found an optimal path. Video: `https://youtu.be/u1UOqdJoK9s`

At present, the target coverage area is 1 ha per battery charge. Furthermore, since the MAV must fly at relatively low altitudes to emit enough energy into the ground, structures in the environment such as trees or buildings can force no-fly-zones (NFZs) across the space. These obstacles can be especially problematic since they are often omitted from maps and must be dealt with at deployment time. This results in a need for an autonomous coverage planning and execution solution that can rapidly replan trajectories in the field.

Many existing commercial mission planners, e.g., Ardupilot Mission Planner[1], Pix4DCapture[2], Drone Harmony[3], and DJIFlightPlanner[4] provide implementations of 2D coverage planners which allow specifying polygon flight corridors and generating lawnmower patterns. However, these mission planners aim at high-altitude, open space, top-down photography and thus neither allow specifying obstacles within the polygon nor consider the polygon edges as strict boundaries. Furthermore, due to safety limitations of commercial platforms, most commercial planners enforce a minimum altitude which is above the required 3 m of our GPSAR system.

While solutions for generating coverage paths in general polygon environments exist, the implementations are either not publicly available, computationally unsuitable for complex large environments or lack the full system integration that would allow a user to freely designate the flight zone.

In this work we develop a complete pipeline for rapidly generating 2D coverage plans that can account for NFZs. Our proposed solution takes as input a general polygon (potentially containing NFZs) and performs an exact cell decomposition over the region. Initial sweep patterns are computed for each cell from which feasible flight trajectories are computed. Finally, the Equality

[1] `http://ardupilot.org/planner/`

[2] `https://www.pix4d.com/product/pix4dcapture`

[3] `https://droneharmony.com`

[4] `https://www.djiflightplanner.com/`

Generalized Traveling Salesman Problem (E-GTSP) across the complete cell adjacency graph is solved to minimize the total path time.

Results comparing our proposed coverage planner to exhaustively searching a coverage adjacency graph demonstrate an order of magnitude speedup in the computation time for each trajectory. Specifically, we show that our solution's computation time grows reasonably with increasing map complexity, while the exact solution grows unbounded. Furthermore, the flight path computed by our method is 14 % shorter than those found by solving a regular traveling salesman problem (TSP) while providing the same level of coverage.

The contributions of this work are:

- An E-GTSP based fast 2D coverage planning algorithm that allows specifying polygonal flight zones and obstacles.
- Benchmarks against existing solutions.
- An open source robot operating system (ROS) implementation.[5]

We organize the remainder of the paper as follows. Section 2 presents related work. In section 3 we present the background in computational geometry to generate the sweep permutations. In section 4 we present our coverage planner formulation to solve for the optimal coverage pattern. Finally, we validate our method in section 5 before we close with concluding remarks in section 6.

2 Related Work

Coverage planning in partially known environments is an omnipresent problem in robotics and includes applications such as agriculture, photogrammetry and search. It is closely related to the art gallery problem (AGP) and TSP, which are NP-hard [18, 20], and for which heuristic solutions have been developed to cope with high dimensional problems. The coverage planning problem can be stated as: given a specified region and sensor model, generate a plan for a mobile robot that provides complete coverage, respects motion and spatial constraints, and minimizes some cost metric (often path length or time). The underlying algorithms to solve sweep planning in a 2D polygon map are usually either based on approximate or exact cellular decomposition. The state of the art in robotic coverage planning is summarized in [3, 9].

Early work in robot coverage planning developed approximate cellular decomposition methods. In [27], the target region is first decomposed into connected uniform grid cells such that coverage is achieved by visiting all cells. The authors presented a coverage wave front algorithm that obeyed the given starting and goal cells. Unfortunately, grid based methods grow linearly in memory with the area to be covered and exponentially in finding an optimal path, making them unsuitable for large areas with small sensor footprints. Furthermore, guiding the

[5] `https://github.com/ethz-asl/polygon_coverage_planning`

wave front algorithm to pick a route that meets all mission requirements involves designing and tuning a specific cost function for each application. Spanning-trees are an alternative method to solve this problem [7]. However, by default those result in a rather large number of turns which is undesirable for MAVs.

A simple approach stems from the idea that a polygonal region can be covered using sequential parallel back-and-forth motions, i.e. lawnmower or boustrophedon paths. This approach for 2D top-down lawnmower patterns in polygonal maps dominates commercial aerial remote sensing. The user defines a polygonal target region and a specified sweep direction, and the planner generates a path consisting of a sequence of equally-spaced sweep lines connected by turns. The generated coverage paths are time-efficient, complete, and intuitive to the user. However, these planners cannot account for NFZs nor do they satisfy our targeted flight altitude requirements. Furthermore, except for Ardupilot Mission Planner, these commercial systems do not allow modification as they are closed source.

Exact cellular decomposition is a geometric approach that can handle NFZs by dividing a configuration space into simpler component cells, whose union is the complete free space. A full coverage path can then be found by creating a boustrophedon path in each cell and connecting all cells. The individual cell coverage plans are connected in an adjacency graph and solving the resulting TSP solves the coverage problem. Compared to grid-based methods, exact decomposition generally results in significantly fewer cells in simple large environments and thus a smaller TSP. Choset and Pignon [4] present the standard solution in 2D environments, which has been adapted by many planners for robot coverage [1,8]. Several improvements have also been proposed to minimize the number of cell traversals and the number of turns along the coverage path. Namely, methods for optimizing over the sweep line direction rather than using a fixed direction for all cells have been proposed [16, 25]. However, these come at the cost of increased complexity when solving for the optimal path through the cell-connectivity graph.

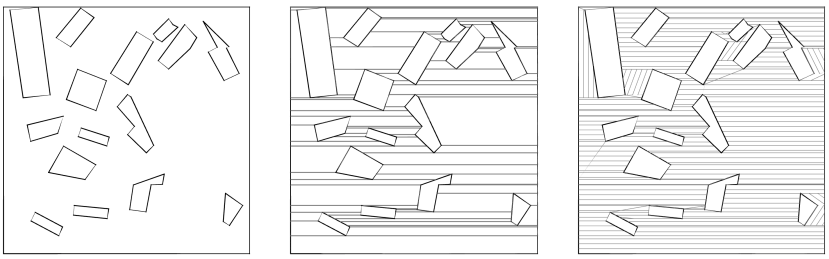
In our work we revisit the exact cellular decomposition method and also generate multiple possible sweep directions per cell, however we formulate the resulting search as an E-GTSP, allowing us to use a state of the art genetic solver that handles significantly larger problem sizes. The E-GTSP, also known as TSP with neighborhoods, is a generalization of the classical TSP [17]. The goal is to find a shortest tour that visits exactly one node in each of a set of k neighborhoods. As first proposed by Waanders [26], we cluster all possible sweep patterns of a cell in a neighborhood and search for the shortest path that includes exactly one sweep pattern per cell.

Bochkarev and Lewis [2, 15] took this E-GTSP formulation even further. Instead of using per-cell predefined sweep patterns, they precompute a set of globally good sweeping directions for each monotone cell and build a graph over all individual straight segments where a neighborhood is defined by traversing a segment in either direction. Compared to our approach their approach can give better solutions where the robot traverses from one cell to another cell without covering it completely first at the expense of a more complex E-GTSP and a predefined sweep direction.

To solve the E-GTSP, several options exist. Exact solvers like [6] only work reliably for small problem sizes. Converting the problem into a directed graph using a product graph [21] and solving it with an optimal graph solver, e.g. Dijkstra, basically falls back to solving it exhaustively as in [4] and [25], which remains intractable for larger problem sizes. Thus a practical solution for our problem sizes is to use heuristic solvers. Helsgaun [12] transforms the E-GTSP into a TSP and uses an approximate TSP solver. The Gutin and Karapetyan solver (GK), on the other hand, is a memetic algorithm that approximately solves the E-GTSP directly and is faster but with reduced performance for large problems [10].

3 Geometric Path Generation

In order to solve the coverage path problem we follow the route of exact cellular decomposition techniques outlined in Figure 2. We decompose a general polygon with holes (PWH) into cells, in a way that guarantees that each cell can be fully covered by simple boustrophedon paths. Our algorithm creates a permutation of sweeping directions for each cell and finds a shortest route that connects and covers every cell to define the *coverage path*.



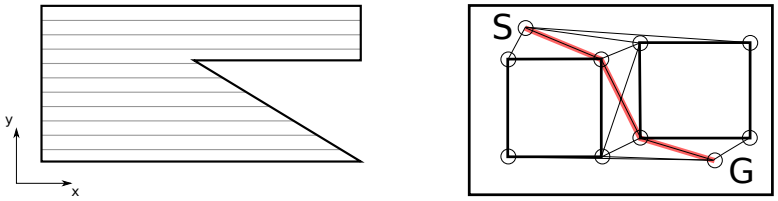
(a) Input PWH (b) Cell decomposition (c) Optimal sweep pattern

Fig. 2 The coverage algorithm on a synthetic map with 15 obstacles and 52 hole vertices.

3.1 Sweep Pattern Permutation

A *sweep pattern* describes the combination of *straight segments* and *transition segments* that covers a single polygon cell. A continuous parallel sweep pattern can be generated perpendicular to any monotone direction of a simple polygon. Without loss of generality we can consider the monotone direction the y-direction of the polygon. A polygon P is considered y-monotone, if any line in the x-direction intersects P at a single point, a segment or not at all. In other words the line intersects P at most twice [5].

Figure 3 (a) shows the straight segment generation in a y-monotone polygon. We initialize the first straight segment at the bottommost vertex parallel to the x-axis, which we refer to as *sweep direction*. In general, we restrict the sweep directions to be collinear to one of the edges of our polygon, as these directions have been proven to lead to a minimum number of straight segments to cover the polygon [13]. The individual straight segments are generated by alternating between intersecting a line in the x-direction with the polygon and offsetting the line from the bottommost towards the topmost vertex. The sensor footprint hereby defines the offset distance.



(a) Straight segments in a y-monotone polygon. (b) Shortest path transition segments.

Fig. 3 A sweep pattern consists of a set of parallel offsetted straight segments sequentially connected via transition segments. The straight segments are generated along edges that are perpendicular to a monotone direction (left). The transition segments are Euclidean shortest paths along the reduced visibility graph (right).

Based on this construction criterion, the straight segments can start clockwise or counter-clockwise at the bottom vertex and go from the bottom to the top or from top to bottom. Thus we generate four possible sweep patterns per *sweepable* direction as shown in Figure 4.

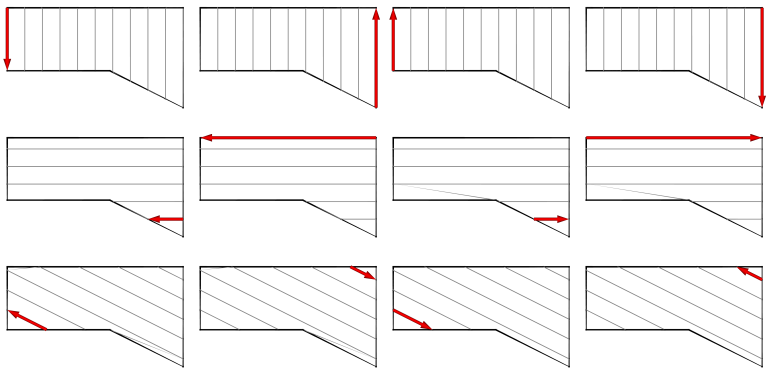
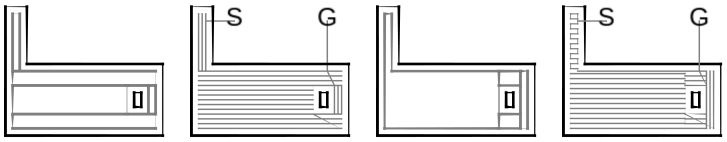


Fig. 4 Each *sweepable* direction (rows) has four *sweep permutations* (columns) based on the start vertex and start direction (red arrow).

To connect two straight segments (and later to connect two sweep patterns) we calculate the Euclidean shortest path that avoids collision with the NFZ. The Euclidean shortest path in a PWH is computed along the *reduced visibility graph* [14]. Figure 3 (b) shows an example solution (red) using A* to search the graph [11]. The graph node set (circles) consists of all non-convex hull vertices and convex hole vertices.

3.2 Polygon Decomposition

To cover a general PWH, we partition it into monotone non-intercepting cells whose union is again the original PWH. In general any monotone decomposition would be feasible but we only consider the trapezoidal cell decomposition (TCD) and the boustrophedon cell decompositon (BCD) [4, 5]. Both decompositions have different advantages as shown in Figure 5. The TCD provides a partitioning that adjusts well in rectangular scenes with multiple directions of extension. The BCD also adjusts well to rectangular scenes and usually leads to fewer cells and thus fewer redundant sweeps and traversal segments [4]. On the downside it can lead to degenerate sweeping behaviour in cells with narrow protrusions.



(a) TCD (b) BCD

Fig. 5 Qualitative comparison of TCD and BCD. While both adjust well to rectangular environments, the BCD leads to fewer cells and redundant sweeps but can have degenerate sweeping behaviour in narrow regions.

Since both decompositions result from scan line algorithms, the scan line direction determines the set of cells. To find a good decomposition direction we calculate the decomposition for every individual edge direction. A potentially good decomposition is the decomposition with the smallest altitude sum w, where altitude refers to the monotone extension of a cell. The minimum altitude sum corresponds to a minimum number of sweeps in the case of a fixed sweep direction [13].

$$w = \sum_{i=1}^{m} y_{\max,i} - y_{\min,i}, \tag{1}$$

where m is the number of monotone cells, $y_{\max,i}$ is the y-coordinate of the uppermost vertex and $y_{\min,i}$ is the y-coordinate of the lowermost vertex in a y-monotone polygon cell.

4 Coverage Path Planning

After decomposing the input PWH into simple cells and generating a set of sweep patterns for each cell the planner has to find a shortest sequence of sweep patterns such that every cell, and thus the whole PWH, is covered. To solve this problem efficiently we formulate it as an Equality Generalized Traveling Salesman Problem.

Figure 6 sketches the adjacency graph $G=(N,A)$, where $N=\{n_1 \ldots n_n\}$ is the node set and $A=\{(n_i,n_j): n_i,n_j \in N, i \neq j\}$ is the set of directed arcs. The node set N is divided into m mutually exclusive and exhaustive clusters $N_1 \ldots N_m$, i.e., $N=N_1 \cup \ldots \cup N_m$ with $N_i \cap N_j = \emptyset \ \forall i,j \in \{1 \ldots m\}, i \neq j$.

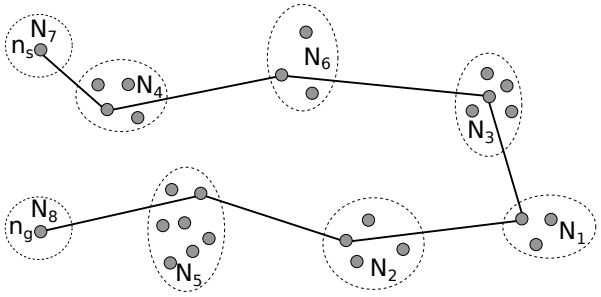


Fig. 6 Visualization of the coverage planning E-GTSP. The goal is to find the shortest path that visits exactly one sweep pattern (grey dot) in each polygon cell (dotted ellipsoid) while traveling from the start node n_s to the goal node n_g.

A node n_i represents an individual sweep pattern that covers a single monotone polygon cell as shown in Figure 4 or the start or goal point. Every monotone cell (and the start and goal point) represents an individual cluster N_i. The arcs are the shortest path connecting the end of one sweep pattern with the start of the next sweep pattern in another cluster. The start node has outgoing arcs to all nodes, and the goal node has incoming arcs from all nodes.

Every arc (n_i,n_j) has a non-negative cost c_{ij} which is the sum of the shortest path cost t_{ij} from n_i to n_j and the cost t_j to execute sweep pattern n_j.

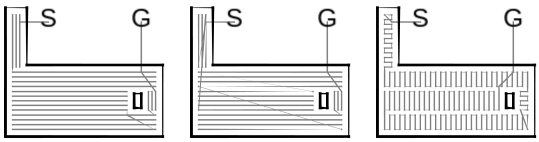
$$c_{ij} = t_{ij} + t_j \tag{2}$$

Because the start and the end of a sweep pattern do not coincide, the cost matrix $\boldsymbol{C}=(c_{ij})$ is asymmetric, i.e., $c_{ij} \neq c_{ji}$.

Since MAVs can hover and turn on the spot, we can plan rest-to-rest segments between waypoints which obey the straight-line assumption of our coverage planner. The trajectories are modeled with velocity ramp profiles with instantaneous acceleration and deceleration and a maximum velocity. The cost of a trajectory is the sum of all segment times. The segment time t between two waypoints is a function of the distance d, the maximum velocity v_{max}, and the maximum acceleration a_{max},

$$t = \begin{cases} \sqrt{\frac{4d}{a_{max}}}, & \text{for } d < 2d_a \\ 2t_a + \frac{d-2d_a}{v_{max}}, & \text{for } d \geq 2d_a \end{cases}, \quad \text{where } t_a = \frac{v_{max}}{a_{max}}, \quad d_a = \frac{1}{2} v_{max} t_a. \tag{3}$$

t_a is the time to accelerate to maximum velocity and d_a is the distance travelled while accelerating. Figure 7 shows that by tuning v_{max} and a_{max} with respect to our platform and sensor constraints the optimization finds a good compromise between minimizing distance and minimizing the number of turns.



(a) Time (b) Waypoints (c) Distance

Fig. 7 Qualitative comparison of different optimization criteria. Minimizing time results in elongated trajectories with short transition segments. Minimizing the number of segments also leads to long trajectories, but does not necessarily lead to good transitions. Minimizing only the Euclidean distance can lead to undesired sweeps since turns are not penalized.

Algorithm 1 summarizes the process of setting up the E-GTSP problem. First we decompose the polygon into monotone cells. For each cell we compute the sweep permutations and make each sweep pattern a node in the graph where the neighborhood is defined by the cell id. Finally, we create the edges between all sweeps patterns of different cells using the precomputed reduced visibility graph. Once the cost matrix is fully defined, we solve the E-GTSP using GK[6] as an off-the-shelf open source solver [10].

Algorithm 1 Generating the E-GTSP adjacency graph

Require: pwh, sweep_distance, wall_distance, cost_func, decomposition_type
Ensure: adj_graph
▷ Decompose polygon into monotone cells.
pwh = pwh.OFFSETPOLYGON(wall_distance)
cells = pwh.COMPUTEDECOMPOSITION(decomposition_type)
▷ Compute all sweep patterns per cell and create E-GTSP nodes.
vis_graph = pwh.COMPUTEVISIBLITYGRAPH()
for all cell ∈ cells **do**
    sweep_patterns = cell.COMPUTESWEEPPATTERNS(sweep_distance, vis_graph)
    for all sweep_pattern ∈ sweep_patterns **do**
        n.sweep_pattern = sweep_pattern
        n.cell_id = cell.id
        n.start_vis = pwh.COMPUTEVISIBILITY(sweep_pattern.start)
        n.goal_vis = pwh.COMPUTEVISIBILITY(sweep_pattern.goal)
        adj_graph.ADDNODE(n)
▷ Densely connect all nodes via Euclidean shortest paths.
adj_graph.PRUNE(vis_graph)
for all from ∈ adj_graph.nodes **do**
    for all to ∈ adj_graph.nodes ∧ from.cell_id ≠ to.cell_id **do**
        e.path = vis_graph.SOLVE(m.sweep.goal, m.goal_vis, n.sweep.goal, n.start_vis)
        e.cost = cost_func.COMPUTE(n.sweep_pattern) + cost_func.COMPUTE(e.path)
        adj_graph.ADDEDGE(e)

[6] https://csee.essex.ac.uk/staff/dkarap/?page=publications&key=Gutin2009a

4.1 Pruning

While our problem size (tens of clusters, hundreds of nodes, hundreds of thousands of edges) is no problem for the GK, generating the edges is the bottleneck of the algorithm because every sweep pattern needs to be connected to almost any other sweep pattern through a Euclidean shortest path. The total number of arcs grows quadratically with the number of nodes.

Fortunately, the optimization problem is modular, i.e., any sweep pattern combination that visits every cell will achieve full coverage and the path cost is the only optimization criterion. We can safely prune a node $n_i \in N_k$ if it is cheaper to first traverse from n_i's start point to the start point of $n_j \in N_k$, perform coverage n_j and return to n_i's goal point.

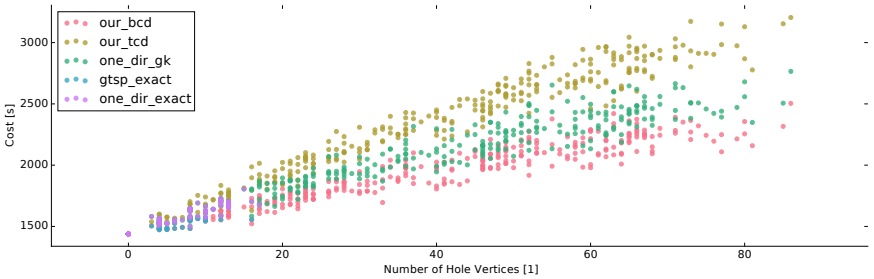
5 Results

The algorithm has been implemented in C++ using the Computational Geometry Algorithms Library (CGAL) [24]. CGAL provides efficient, reliable, and exact geometric algorithms which we extended to generate the sweep lines and the BCD. In order to find an exact solution we implement [21] to convert the E-GTSP into a directed graph for an exhaustive exact solution using Dijkstra.

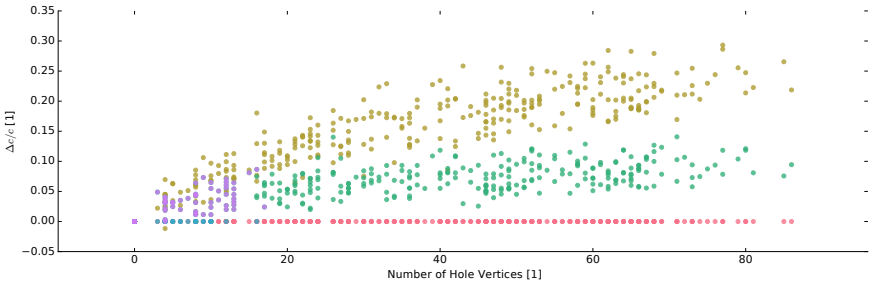
5.1 Simulation Benchmark

We setup a simulation benchmark to evaluate the performance of our algorithm (*our*) against the classical sweep planner with one sweep direction (*one_dir*) [4], to compare BCD and TCD, and to demonstrate the superiority of a designated E-GTSP solver over *exact* brute-force search. Our test instances, e.g. Figure 2, are automatically generated from the EPFL aerial rooftop dataset [23] to provide polygon maps with realistic obstacles and dimensions. Our synthetic dataset consists of 320 rectangular worlds with an area of 1 ha and 0 to 15 rooftop obstacles. The benchmark was executed on an Intel®Core™i7-7820HQ CPU @ 2.9 GHz.

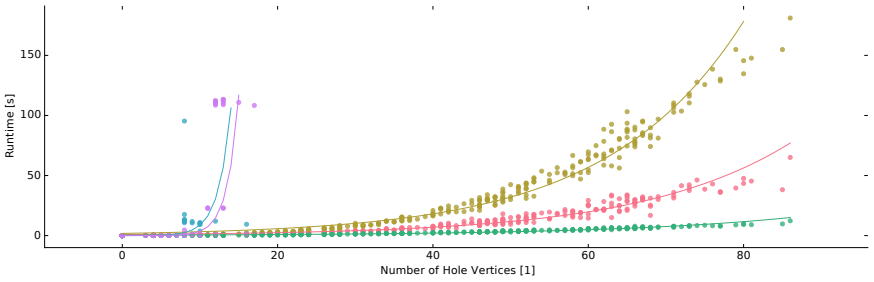
To relate our solution to the complexity of the maps, we plot coverage path cost and algorithm runtime against the number of hole vertices, as these are the events of the decomposition scan line algorithms. Figure 8 (a) shows the trajectory cost of the different planner configurations. Because our planner takes advantage of different sweep directions and start points, it gives better results than the classic solution with only one fixed sweep pattern per cell. Furthermore, BCD leads to shorter trajectories than TCD because it generates fewer cells and thus fewer redundant sweeps at the adjacent edges and transition segments between the cells. On a side note, whenever the exact solution is available it coincides with the approximate solution from GK.



(a) The absolute path cost.



(b) The relative improvement of our planner.



(c) The computation times.

Fig. 8 Benchmark results for increasing map complexity. Our E-GTSP with BCD generally has the lowest path cost and reasonable computation times even for complex maps. Exact solutions only work for maps with few holes. The TCD is generally worse than BCD.

The relative differences in path cost between our planner and the other configurations is shown in Figure 8 (b). We observe up to 14 % improvement over the optimal fixed direction and 29 % improvement of BCD over TCD. Figure 8 (c) shows the computation time of the planners. Exact solutions fail to solve within 200 s for any of our scenarios with more than twenty hole vertices. Generating the product graph from the adjacency graph G and Boolean lattice of possible cluster sequences grows exponentially in the number of clusters [21]. The reduced problem *one_dir_gk* achieves the best computation time. At most it takes 12 s for

the most complex map. The full E-GTSP with BCD solves it within 65 s which is still reasonable for employing the planner in field experiments. Table 1 reveals that generating the dense E-GTSP graph is the greatest computational burden.

	Graph elements			Computation time (s)						Path cost (s)
	Cells	Nodes	Edges	Cells	Sweeps	Nodes	Pruning	Edges	Solve	
our_bcd	**52**	254	63056	5.7	1.1	3.0	2.4	22.2	1.2	**2391.2**
our_tcd	87	512	258704	6.9	1.1	4.9	3.7	84.7	6.6	2879.2
one_dir_gk	**52**	**52**	**2652**	5.5	0.1	0.2	0.0	**0.8**	0.5	2665.0

Table 1 Detailed computation times for the instance in Figure 2 with 15 holes (52 hole vertices). Creating all edges is the greatest computational effort. The reduced problem thus has the smallest computation time. Our planner results in the lowest path cost.

5.2 Experiment

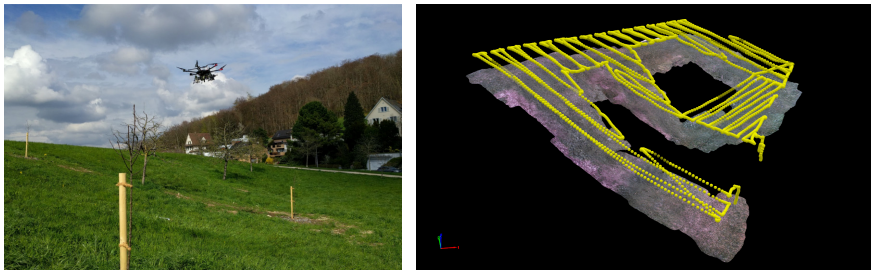
We validate our planner in a real flight on a DJI M600 Pro. Our drone covers a non-convex open area with sparse trees, depicted in Figure 1, in a low altitude mapping scenario. The flight corridor is selected in a georeferenced map, an optimal sweep path is calculated with our planner and executed under GNSS control. The drone follows a velocity ramp profile as described in section 4 and additionally turns at every waypoint in the flight direction. The slope of the terrain is not considered during planning. To regulate the height above ground level (AGL), we fuse Lidar and Radar altimeter data into a consistent altitude estimate. Table 2 shows the general experiment setup. The sweep distance is chosen based on image overlap. The velocity and acceleration are chosen to meet controller constraints and avoid motion blur. To validate coverage with a nadir configuration sensor we record QXGA top-down imagery and generate the digital terrain model (DTM) shown in Figure 9 using the Pix4D mapping tool. The DTM shows good coverage of the designated area but imperfect reconstruction due to instantaneous acceleration movements. Collision-free trajectory smoothing may improve the turning maneuvers both in speed and smoothness and consequentially increase overall performance of the coverage planner.

Area	Flight Time	AGL	Sweep Offset	$v_{\max}$	$a_{\max}$
$1950\,\mathrm{m}^2$	1050s	4 m	1.5 m	$3.0\,\mathrm{ms}^{-1}$	$0.5\,\mathrm{ms}^{-2}$

Table 2 Flight experiments parameters.

6 Conclusion

In this work we presented a boustrophedon coverage path planner based on an E-GTSP formulation. We showed in comprehensive benchmarks on realistic syn-



(a) Terrain following using altimeter data. (b) A Pix4D DTM reconstruction.

Fig. 9 The platform is capable of covering sloped terrain using the proposed planner.

thetic polygon maps that our planner reliably solves complex coverage tasks in reasonable computation time, making it suitable for field deployment. Furthermore, we showed that our planner outperforms the classic boustrophedon coverage planner in terms of path cost. We validated in a field experiment the usability of our coverage algorithm on a real MAV and show that we can cover a 1950 m^2 area with obstacles at low altitude. Future work includes optimizing coverage for a side looking GPSAR configuration, collision-free trajectory smoothing to improve turning times, and integration into an airborne mine detection system.

Acknowledgment

This work is part of the FindMine project and was supported by the Urs Endress Foundation. The authors would like to thank their student Lucia Liu for her initial work on the BCD, Florian Braun and Michael Riner-Kuhn for their hardware support, and the reviewers for their constructive advice.

References

1. Bähnemann, R., Schindler, D., Kamel, M., Siegwart, R., Nieto, J.: A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects. In: IEEE International Symposium on Safety, Security and Rescue Robotics, pp. 123–128. IEEE (2017)
2. Bochkarev, S., Smith, S.L.: On minimizing turns in robot coverage path planning. In: IEEE International Conference on Automation Science and Engineering, pp. 1237–1242. IEEE (2016)
3. Cabreira, T., Brisolara, L., R Ferreira, P.: Survey on coverage path planning with unmanned aerial vehicles. Drones **3**(1), 1–38 (2019)
4. Choset, H., Pignon, P.: Coverage path planning: The boustrophedon cellular decomposition. In: Field and Service Robotics, pp. 203–209. Springer (1998)
5. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational geometry. In: Computational Geometry, pp. 1–17. Springer (1997)

6. Fischetti, M., Salazar González, J.J., Toth, P.: A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Operations Research **45**(3), 378–394 (1997)
7. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Annals of Mathematics and Artificial Intelligence **31**(1-4), 77–98 (2001)
8. Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M., Ridao, P.: Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. Journal of Field Robotics **32**(7), 952–983 (2015)
9. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robotics and Autonomous Systems **61**(12), 1258–1276 (2013)
10. Gutin, G., Karapetyan, D.: A memetic algorithm for the generalized traveling salesman problem. Natural Computing **9**(1), 47–60 (2010)
11. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
12. Helsgaun, K.: Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm. Mathematical Programming Computation **7**(3), 269–287 (2015)
13. Huang, W.H.: Optimal line-sweep-based decompositions for coverage algorithms. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 27–32. IEEE (2001)
14. Latombe, J.C.: Robot Motion Planning. Springer Science & Business Media (1991)
15. Lewis, J.S., Edwards, W., Benson, K., Rekleitis, I., O'Kane, J.M.: Semi-boustrophedon coverage with a Dubins vehicle. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5630–5637. IEEE (2017)
16. Li, Y., Chen, H., Er, M.J., Wang, X.: Coverage path planning for UAVs based on enhanced exact cellular decomposition method. Mechatronics **21**(5), 876–885 (2011)
17. Mitchell, J.S.: Geometric shortest paths and network optimization. Handbook of Computational Geometry **334**, 633–702 (2000)
18. O'Rourke, J., Supowit, K.: Some NP-hard polygon decomposition problems. IEEE Transactions on Information Theory **29**(2), 181–190 (1983)
19. Pajares, G.: Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs). Photogrammetric Engineering & Remote Sensing **81**(4), 281–329 (2015)
20. Papadimitriou, C.H.: The Euclidean travelling salesman problem is NP-complete. Theoretical Computer Science **4**(3), 237–244 (1977)
21. Rice, M.N., Tsotras, V.J.: Exact graph search algorithms for generalized traveling salesman path problems. In: International Symposium on Experimental Algorithms, pp. 344–355. Springer (2012)
22. Schartel, M., Burr, R., Mayer, W., Docci, N., Waldschmidt, C.: UAV-based ground penetrating synthetic aperture radar. In: IEEE MTT-S International Conference on Microwaves for Intelligent Mobility, pp. 1–4. IEEE (2018)
23. Sun, X., Christoudias, C.M., Fua, P.: Free-shape polygonal object localization. In: European Conference on Computer Vision, pp. 317–332. Springer (2014)
24. The CGAL Project: CGAL User and Reference Manual, 4.13 edn. CGAL Editorial Board (2018). URL `https://doc.cgal.org/4.13/Manual/packages.html`
25. Torres, M., Pelta, D.A., Verdegay, J.L., Torres, J.C.: Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. Expert Systems with Applications **55**, 441–451 (2016)
26. Waanders, M.: Coverage path planning for mobile cleaning robots. In: 15th Twente Student Conference on IT, pp. 1–10 (2011)
27. Zelinsky, A., Jarvis, R.A., Byrne, J., Yuta, S.: Planning paths of complete coverage of an unstructured environment by a mobile robot. In: International Conference on Advanced Robotics, vol. 13, pp. 533–538 (1993)