

Coverage Path Planning for Holonomic UAVs via Uniaxial-Feasible, Gap-Severity Guided Decomposition

Pedro Antonio Alarcon Granadeno¹ and Jane Cleland-Huang²

Abstract—Modern coverage path planning (CPP) for holonomic UAVs in emergency response must contend with diverse environments where regions of interest (ROIs) often take the form of highly irregular polygons, characterized by asymmetric shapes, dense clusters of concavities, and multiple internal holes. Modern CPP pipelines typically rely on decomposition strategies that overfragment such polygons into numerous subregions. This increases the number of sweep segments and connectors, which in turn adds inter-region travel and forces more frequent reorientation. These effects ultimately result in longer completion times and degraded trajectory quality. We address this with a decomposition strategy that applies a recursive dual-axis monotonicity criterion, with cuts guided by a cumulative gap severity metric. This approach distributes clusters of concavities more evenly across subregions and produces a minimal set of partitions that remain sweepable under a parallel-track maneuver. We pair this with a global optimizer that jointly selects sweep paths and inter-partition transitions to minimize total path length, transition overhead, and turn count. We demonstrate that our proposed approach achieves the lowest mean path-length and completion-time overhead among 15 other CPP pipelines.

I. INTRODUCTION

Coverage Path Planning (CPP) generates trajectories that fully observe a Region of Interest (ROI) with minimal cost, typically measured in path length, turns, execution time, or energy use. A dominant strategy in aerial and ground robotics is parallel-track coverage, where back-and-forth sweeps (e.g., boustrophedon) tile the region. For convex ROIs, selecting an optimal sweep direction is straightforward. Huang [1] showed that the direction perpendicular to the polygon's minimal altitude minimizes turns, a widely accepted surrogate for maneuvering effort, execution time, and energy consumption [2].

Real-world ROIs, however, are rarely convex. This leads to the need for polygon decomposition, where the region is subdivided into simpler subregions (often convex or monotone in a chosen direction), each assigned a local sweep pattern. Early strategies such as trapezoidal decomposition [3] and boustrophedon cellular decomposition (BCD) [4] provided simple and topologically sound methods for this. Trapezoidal decomposition projects vertical rays from vertices to the

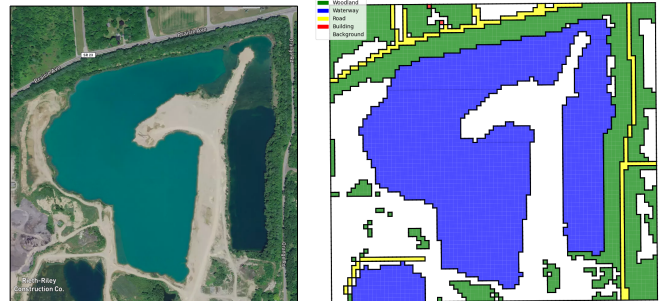


Fig. 1: **Left:** Raw satellite image. **Right:** Region-of-interest polygons derived via landcover segmentation and DFS-based component tracing. The extracted polygons vary widely in size and exhibit irregular, asymmetric, and elongated geometries typical of challenging coverage scenarios.

boundary, yielding convex but often overfragmented partitions when edges are nearly parallel or reflex vertices are dense. BCD improves upon this by introducing a sweep-line that only places cuts at connectivity events (openings, closures, splits, merges) which yields fewer and more meaningful partitions. Both methods rely on a fixed global sweep direction, which limits flexibility and path efficiency [5].

Several contemporary CPP methods build upon these foundations but aim to improve adaptability and overall efficiency. Three recurring themes characterize much of this recent work. First, many algorithms perform decomposition across multiple sweep orientations rather than committing to a single axis from the outset. This is often paired with width-based surrogates, such as minimal altitude, to estimate turn cost [2]. Subregions are split to reduce cumulative altitude and, by extension, the number of passes [6], [7]. Second, several approaches introduce post-decomposition merging strategies to attenuate overfragmentation inefficiencies. Some rely on simple heuristics that merge adjacent partitions sharing the same sweep direction [6], [7], while others apply optimization routines that merge subpolygons when the combined shape yields a more optimal width cost [1], [8], [9]. Third, a subset of approaches incorporate inter-partition connectivity directly into the planning objective. These methods generate multiple candidate paths per region and solve a global sequencing problem (often modeled as a TSP or GTSP) to jointly minimize coverage and transition costs [6], [10], [9].

These strategies have shown strong performance on concave regions that are geometrically simple, such as polygons with rectangular outlines, moderate symmetry, or limited branching. These types of geometries commonly appear in synthetic benchmarks or agricultural settings where the

*This work was supported by the USA National Science Foundation under Grant 1931962

¹Pedro Antonio Alarcon Granadeno is a PhD student in the Computer Science and Engineering department at the University of Notre Dame, IN, USA palarcon@nd.edu

²Jane Cleland-Huang is a Professor in the Department of Computer Science and Engineering, University of Notre Dame, IN, USA JaneHuang@nd.edu

Website and open-source code found at <https://sites.google.com/view/cpp-holonomic-uav>

spatial layout remains relatively uniform. However, this assumption fails in real-world aerial scenarios, particularly in applications like emergency response. Here, the ROI is often derived from landcover annotations, which are semantic segmentations that tightly trace the contours of natural and man-made features (Fig. 1). The resulting polygons are highly asymmetric, with winding boundaries, branched extensions, holes, and dense clusters of concavities. Classical and modern decomposition pipelines often overfragment these inputs, producing small partitions that yield abrupt changes in sweep alignment and excessive inter-region connectors downstream. Several core issues drive this overfragmentation: (i) **width-guided decomposition** (e.g., [6], [7]) introduces competing objectives, where minimizing per-region turn cost often comes at the expense of a larger, more fragmented partition set; (ii) **fixed global sweep axes** (as in [3], [4]) misalign with local geometry, forcing unnecessary cuts; and (iii) **post-hoc merging is tightly constrained**, typically allowed only when the merged region remains convex and yields a strictly better width cost (as in [6], [1], [9], [8]).

To address these limitations, we introduce a recursive decomposition strategy built around a relaxed geometric criterion. A region is considered sweepable if it is monotone with respect to at least one cardinal axis, a property we refer to as uniaxial feasibility. This criterion replaces stricter assumptions such as convexity, global sweep alignment, or width minimization, and is used both to determine when a region must be split and whether adjacent regions can be merged. When a region violates feasibility, we apply a recursive partitioning process guided by a cumulative gap severity metric that identifies and distributes clusters of concavity. Unlike prior methods that prioritize turn cost or axis alignment during decomposition, we aim to distribute regions of discontinuity evenly across resulting partitions. This yields a smaller partition set composed of larger, sweepable regions, leading to lower global path length and execution time as compared to contemporary CPP pipelines.

II. RELATED WORK

Decomposition is a core strategy in modern Coverage Path Planning (CPP) for simplifying coverage over complex regions. Recent CPP surveys note the recurring use of polygonal width as a practical surrogate for turn-related maneuvering cost [2]. Measured as the shortest distance between parallel supporting lines orthogonal to a candidate sweep direction, width affects the number of parallel coverage passes (for fixed sweep spacing) and thus lane transitions and turns, making it useful for decomposition, sweep-direction selection, and merging decisions. One influential early example is Huang’s Multi-Strip Approximation (MSA) [1], which extends classical Boustrophedon decomposition. MSA sweeps orthogonal to each polygon edge, places dividers at topological changes, overlays them to form monotone partitions, and uses dynamic programming to evaluate merges and assign sweep directions that minimize total width.

Later approaches introduced width surrogates at different stages of the planning pipeline. Li et al. [6] use a greedy

recursive strategy that places axis-aligned cuts at reflex vertices, scores each candidate by the cumulative width of the resulting sub-polygons, and selects the minimum-cost cut until all partitions are convex. Yu and Hung [7] instead extend trapezoidal decomposition with four event types (floor/ceiling, convex/concave) and apply it across multiple orientations orthogonal to polygon or obstacle edges, selecting the convex decomposition with the lowest total width. Both methods optionally merge adjacent convex regions that share a common sweep direction. Other methods postpone width optimization to the merging stage. Nielsen et al. [8] generate convex partitions by extending incident edges at reflex vertices until they intersect the boundary, then enumerate all valid convex unions and compute their widths. An integer linear program selects a subset of merged regions that together minimize total width. Bochkarev and Smith [9] start from an existing convex decomposition and iteratively refine it. At each reflex vertex, they define a feasible cut space within the visible cone formed by adjacent edges, sample chords, and evaluate each by the sum of minimum altitudes. Cuts that lower total width replace previous ones, and the process continues until convergence.

Other approaches emphasize global optimization over existing partitions instead of decomposition. Torres et al. [10] define four path alternatives per region, combining sweep orientation and traversal direction to determine entry and exit points. A global planner selects visitation order and local configuration to minimize total path length, jointly considering in-region coverage and inter-region transitions, while heuristically pruning the solution space. Jimenez et al. [11] take a genetic approach, encoding the entire plan as a chromosome. The head specifies region order and entry/exit configurations, while the body selects local paths from a template library. A fitness function combining travel and estimated turn cost guides the evolution of efficient plans.

Despite their effectiveness on well-structured regions, width-guided strategies struggle on irregular geometries due to competing objectives. Cuts that reduce turn cost often increase partition count, sweep-axis churn, and connector complexity. Post-hoc merging offers limited relief but is highly sensitive to early partitioning decisions and is typically constrained by stringent admissibility tests that accept unions only when surrogate width metrics improve. In contrast, our method avoids width-based optimization during decomposition and merging. We introduce a cumulative gap severity metric that evaluates sweep inefficiencies along both axes and guides recursive cuts to spread clustered concavities across subregions. A dual-axis monotonicity criterion governs both decomposition and merging: a region is preserved if it is monotone along at least one cardinal axis, and is split only when both fail. This flexibility retains complex but sweepable regions that traditional convex or width-driven methods would prematurely divide. Sweep direction is deferred to a final stitching phase that jointly optimizes path length, turn cost, and inter-region transitions across all partitions.

III. CPP PROBLEM FORMULATION

Coverage path planning (CPP) seeks a trajectory that enables a robot to observe every part of a planar region of interest (ROI), denoted $\tilde{R} \subset \mathbb{R}^2$. Unlike ground-based robots that must remain within strict workspace boundaries due to physical constraints, this work considers a holonomic multi-rotor UAV, capable of concurrent linear and rotational motion at any planar heading. In our holonomic setting, heading is not a constraint on planar motion. The UAV may translate laterally and transition between successive path segments without first stopping to reorient its body. Consequently, corner transitions do not impose curvature or minimum turning radius constraints. Instead, the dominant time cost at segment changes arises from translational acceleration and deceleration limits, rather than from nonholonomic turning-radius constraints or higher-order continuity requirements such as bounded curvature ($k_{\max} = 1/r_{\min}$) or C^3 path smoothness assumptions as in [12].

The UAV is equipped with a nadir-pointing camera with an effective symmetric field-of-view (FOV) half-angle $\phi/2$ at altitude h . Under a flat-ground, pinhole-camera model, the cross-track footprint width becomes $w = 2h \tan(\phi/2)$. We adopt an approximate cellular decomposition setting [13], where the ROI is modeled as a bounded polygonal region \tilde{R} that may contain holes. Let $\mathcal{O} = \{O_1, O_2, \dots, O_K\}$ denote the set of polygonal holes, each representing an obstacle or exclusion zone. A uniform grid of square cells with resolution equal to the UAV's footprint ($w \times w$) is overlaid onto \tilde{R} . All cells that intersect \tilde{R} but do not lie entirely within any hole $O_k \in \mathcal{O}$ are retained. The union of these retained cells defines an expanded rectilinear region R . By construction, $R \supseteq \tilde{R}$, which guarantees full coverage albeit at the cost of some negligible excess area near the boundary.

A region is considered fully covered once every retained cell in R has been observed. A cell is marked as covered when the cumulative camera footprint intersects it beyond a threshold ratio α (typically close to 1). This defines the core principle of complete coverage in CPP. Additional objectives usually include minimizing path length (L), execution time (T), and energy use (E). In emergency response scenarios, which motivate this work, fast execution is essential. Victim survival rates decline sharply with time; for instance, 93.6% of live extrications after earthquakes occur within the first 24 hours [14]. Accordingly, after guaranteeing complete coverage, this study adopts execution time as the *primary evaluation metric* when comparing coverage strategies.

IV. COVERAGE PATH PLANNING ALGORITHM

A. Uniaxial-Feasible, Gap-Severity Guided Decomposition

We operate on a rectilinear polygonal region $P \subset \mathbb{R}^2$, defined over a uniform grid of square cells with resolution equal to the UAV's camera footprint width w . The region may contain interior holes, which define non-observable zones. Our objective is to determine where P can be swept directly using a parallel-track maneuver, or whether we must decompose into simpler subregions. To do so, we apply a recursive, line-sweep-inspired strategy that partitions a polygon

only when it violates dual-axis monotonicity. Cut placements are guided by a cumulative *gap severity* metric that quantifies how inefficient a traversal without decomposition would be inside regions of discontinuity.

Many decomposition pipelines rely on stringent conditions: they fix a global sweep direction a priori (e.g., TD, BCD), decompose whenever a region is non-convex (e.g., Li), and merge only when convexity is preserved and width cost improves (e.g., Li, Huang). These constraints fragment regions that could otherwise be swept directly. To avoid premature cuts and preserve larger sweepable regions, we instead adopt a more permissive geometric rule called the uniaxial feasibility criterion, defined as follows:

Criterion 1 (Uniaxial Feasibility Test). *A polygon P is **acceptable** if it is monotone with respect to at least one of the two cardinal axes (horizontal or vertical). This criterion governs both decomposition and merging, that is, a region is decomposed only if it violates this condition, and adjacent regions are merged only if their union satisfies it.*

This condition retains many concave shapes that would be discarded under convexity or width-based tests, while still guaranteeing parallel-track coverage. To evaluate axis monotonicity, we construct a finite family of axis-aligned sweep lines over the polygon boundary. Note that a rectilinear polygon is composed of individual grid cells of width w , corresponding to the UAV's footprint as defined in Section III. Let ∂P denote the polygon boundary. We extract the unique set of x - and y -coordinates from all cell vertices that lie on ∂P , and sort them in increasing order:

$$X = \{x_1, x_2, \dots, x_n\}, \quad Y = \{y_1, y_2, \dots, y_m\} \quad (1)$$

Candidate sweep lines are then placed at midpoints between consecutive values. For the horizontal and vertical directions, these lines are:

$$L_i^H = \left\{ y = \frac{y_i + y_{i+1}}{2} \right\}, \quad i = 1, \dots, m-1,$$

$$L_j^V = \left\{ x = \frac{x_j + x_{j+1}}{2} \right\}, \quad j = 1, \dots, n-1$$

For each sweep line L , we compute its intersection with the polygon boundary:

$$I(L) = \{p \in \partial P \cap L\}$$

A polygon is monotone with respect to a given axis if and only if $|I(L)| = 2$ for all L in that axis. If any sweep line satisfies $|I(L)| > 2$, the region is non-monotone in that direction, typically due to concavities or internal holes. Note that by construction, the number of candidate sweep lines is at most $n + m$, corresponding to the grid columns and rows intersecting P , hence monotonicity test scales linearly with grid size.

In accordance with the Feasibility Criterion 1, if a polygon violates monotonicity in both directions, we initiate decomposition. To determine the appropriate axis and placement for the cut, we use a cumulative gap severity metric, which

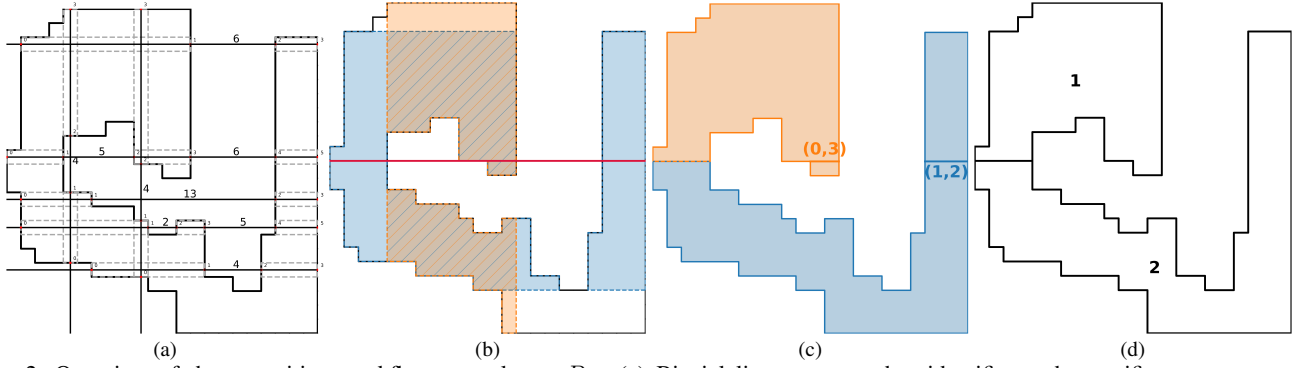


Fig. 2: Overview of decomposition workflow on polygon P_{12} . (a) Biaxial line sweep probes identifies and quantifies non-monotonic regions severities. (b) Cumulative regions of discontinuity (gap bands) guide partition placement. (c) Adjacent partitions are merged if their union satisfies the feasibility criterion. Merge candidates are labeled by partition indices, e.g., $(0,3)$ and $(1,2)$ denote proposed merges between partitions 0 and 3, and 1 and 2 (d) Final partitions are stored for downstream coverage planning.

quantifies the traversal inefficiency along each direction. For a given sweep line L , let $\{p_1, p_2, \dots, p_{2r}\}$ denote the ordered intersection points of line sweep L with the polygon boundary. Then, the total gap severity in that axis is defined as:

$$G = \sum_{L \in \mathcal{L}} \sum_{k=1}^{r-1} \|p_{2k+1} - p_{2k}\| \quad (2)$$

where \mathcal{L} is the set of candidate sweep lines in that axis. Each inner term measures the distance between two consecutive intersection points that fall outside the ROI. These intervals appear either due to internal holes or from disconnected interior spans that arise from the geometry of the polygon. Since a back-and-forth sweep would be forced to cross these intervals, the cumulative sum reflects the total excess distance a naive traversal would incur in the absence of decomposition.

For each sweep line $L \in \mathcal{L}$ where $|I(L)| > 2$, we define a rectangular cross-section bounded between adjacent coordinates in X or Y , depending on the axis (Equation 1). These cross-sections identify local areas where sweep continuity is interrupted. We denote them as

$$\mathcal{B}_H = \{B_i^H = P \cap \{(x, y) : y_i \leq y \leq y_{i+1}\} \mid |I(L_i^H)| > 2\}$$

$$\mathcal{B}_V = \{B_j^V = P \cap \{(x, y) : x_j \leq x \leq x_{j+1}\} \mid |I(L_j^V)| > 2\}$$

We compare the cumulative severities G_H and G_V (see Eq. 2) and choose the axis with greater total. The cut is then placed through the midpoint of the bounding box of the union of cross-sections along that axis. Specifically, letting $U_H = \bigcup \mathcal{B}_H$ and $U_V = \bigcup \mathcal{B}_V$, we define the cut line as:

$$\ell = \begin{cases} y = \frac{1}{2}(\min_y U_H + \max_y U_H) & \text{if } G_H \geq G_V \\ & \text{and } \mathcal{B}_H \neq \emptyset \\ x = \frac{1}{2}(\min_x U_V + \max_x U_V) & \text{if } G_V > G_H \\ & \text{and } \mathcal{B}_V \neq \emptyset \end{cases} \quad (3)$$

The decomposition proceeds recursively. After applying a cut ℓ (Equation 3), the polygon P is partitioned into child regions $\{P_1, P_2\}$, each of which is re-evaluated using the same monotonicity test (Criterion 1) and gap severity

analysis. The process recurses on any subregion that remains non-monotone in both directions. Recursion terminates when all subregions satisfy the Feasibility Criterion 1.

The purpose of the cut ℓ is to distribute the concentration of concavities across the children, so that each inherits a smaller share of the non-monotone support. This tends to reduce concavity complexity of each constituent and encourages larger sweepable regions while keeping the number of partitions low. However, complex geometries can still lead to over-fragmentation. A common failure case occurs when the troublesome cross-sections are highly skewed or clustered. In such cases, a midpoint cut may intersect a narrow indentation and produce thin boundary slivers. To recover from such scenarios, we apply a post-processing merge. Let P_1 and P_2 be two subpolygons produced by the recursion that share a boundary. If their union $P_1 \cup P_2$ is acceptable under the Uniaxial Feasibility Criterion 1, we replace $\{P_1, P_2\}$ with $P_1 \cup P_2$ in the partition. This merge operation is applied iteratively to all adjacent pairs until no further merges are possible. Figure 2 summarizes the main stages of this decomposition workflow on an example polygon.

B. Global Optimization

The decomposition and merge procedures produce a set of N subpolygon partitions $\{P_1, P_2, \dots, P_N\}$ that satisfy Criterion 1. Our goal in this stage is to (i) generate locally efficient candidate coverage paths for each partition, and (ii) globally sequence and connect these into a full, continuous route with minimal overall cost.

For each partition P_i , we generate a small set of *candidate* coverage paths $\{\pi_i^{(1)}, \dots, \pi_i^{(M_i)}\}$, where each $\pi_i^{(j)}$ denotes a *parallel track* traversal that guarantees full coverage of the partition. In this work, we consider up to eight corner-based variants per partition, derived by permuting the sweep direction (horizontal or vertical) and the start corner (bottom-left, bottom-right, top-left, top-right). This gives $M_i = 4$ when P_i is monotone with respect to a single axis, and $M_i = 8$ when it is monotone to both.

Each candidate $\pi_i^{(j)}$ is represented as a tuple

$$\pi_i^{(j)} = (e_i^{(j)}, x_i^{(j)}, L_i^{(j)}, T_i^{(j)})$$

where $e_i^{(j)}, x_i^{(j)} \in \mathbb{R}^2$ are entry and exit coordinates, $L_i^{(j)}$ is the total Euclidean path length, and $T_i^{(j)}$ is the number of turns. We define the local cost of a candidate as

$$\tilde{c}_i^{(j)} = L_i^{(j)} + \rho T_i^{(j)} \quad (4)$$

where $\rho > 0$ is a lightweight penalty applied per heading change within the sweep pattern. This term serves as a heuristic bias against excessive direction changes, since each turn typically introduces additional translational acceleration and deceleration overhead in practice.

For any pair of candidate paths $\pi_i^{(j)}$ and $\pi_k^{(\ell)}$, we define a connector cost (path that connects two partitions) as the Euclidean distance from the exit of the first to the entry of the second. This assumes a vehicle that may traverse brief segments outside the ROI when necessary. The connector cost is:

$$d(x_i^{(j)}, e_k^{(\ell)}) = \|x_i^{(j)} - e_k^{(\ell)}\|_2 \quad (5)$$

The full planning problem is posed as a joint optimization over both the choice of candidate paths and the visitation order of partitions:

$$\min_{\substack{\sigma \in S_N \\ \mathbf{j} \in \prod_{i=1}^N \{1, \dots, M_i\}}} \sum_{t=1}^N \tilde{c}_{\sigma_t}^{(j_{\sigma_t})} + \sum_{t=1}^{N-1} d(x_{\sigma_t}^{(j_{\sigma_t})}, e_{\sigma_{t+1}}^{(j_{\sigma_{t+1}})}), \quad (6)$$

where $\sigma = (\sigma_1, \dots, \sigma_N)$ is a permutation of $\{1, \dots, N\}$ and $\mathbf{j} = (j_1, \dots, j_N)$ with $j_i \in \{1, \dots, M_i\}$ selects a candidate for partition P_i , and t indexes the visitation position along σ (i.e., σ_t is the t -th visited partition). This formulation jointly determines which candidate path to use for each partition and in what order to visit the partitions, so as to minimize the total cost. The first term accounts for the coverage cost within each partition, while the second captures the transition cost between consecutive partitions along the selected tour.

This problem is combinatorially complex, as it requires evaluating all possible permutations of partition visitations along with all combinations of candidate paths; a brute-force approach would yield a time complexity of $O(N! \cdot M^N)$, with $M = \max_i M_i \leq 8$. Instead, for small to moderate sets of partitions, we recommend a dynamic programming (DP) formulation that mirrors the classical Held-Karp algorithm for the Traveling Salesman Problem (TSP)[15].

The core idea is to build up partial solutions incrementally. We define a DP state $F(S, i, j)$ as the minimum cost of visiting a subset $S \subseteq \{1, \dots, N\}$, ending at partition P_i , and using candidate $j \in \{1, \dots, M_i\}$ for that final partition. Each state represents a partial tour that visits exactly the elements of S and terminates with a specific path variant of a specific region. The recurrence relation considers all ways to reach (i, j) by extending a smaller subproblem: for each previously visited partition $P_k \in S \setminus \{i\}$ and candidate $\ell \in \{1, \dots, M_k\}$, we compute the total cost as the sum of (i)

the optimal cost to reach (k, ℓ) , (ii) the local cost $\tilde{c}_i^{(j)}$, and (iii) the connector cost from the exit point of (k, ℓ) to the entry point of (i, j) . Among all such predecessors, we select the minimum. The base case initializes the cost of visiting a single partition with a single candidate path, and the final solution is obtained by evaluating all complete tours over the full set $\{1, \dots, N\}$. We refer the reader to [16], [15] for further implementation details.

C. Discussion on Time Complexity

As discussed in Section IV, each feasibility test (Criterion 1) examines $O(n + m)$ candidate sweep lines, each requiring an intersection query against ∂P . Let E denote the total number of unit grid edges on ∂P (including hole boundaries) for the current rectilinear polygon. Under a conservative model in which each sweep-line intersection query costs $O(E)$, one feasibility test costs $O((n + m)E)$, which is $O(E^2)$ in the grid-traced setting since $n + m \leq E$. Over recursive cuts, this cost is applied to each subpolygon, hence $\sum_i O((n_i + m_i)E_i)$, equivalently $\sum_i O(E_i^2)$. This is $O(E^2)$ for balanced splits and degrades to $O(E^3)$ in the worst case under highly unbalanced recursion. The merge stage then performs repeated pairwise merge attempts with feasibility re-evaluation of adjacent candidate unions. If \tilde{N} is an upper bound on the number of subpolygons present in any merge pass, then each pass requires $O(\tilde{N}^2)$ pairwise checks, and repeated passes yield a conservative worst-case merge cost of $O(\tilde{N}^3 E^2)$.

The complexity depends on the number of unit grid edges on the boundary, so increasing the grid resolution generally increases runtime (through more sweep lines and more boundary edges to process). If the physical shape is preserved and only the scale changes, the final decomposition remains the same. However in practice, grid resolution is often tied to the upstream segmentation/vision pipeline, which can change the extracted polygon shape (e.g., boundary detail, narrow passages, or holes). In that case, the input geometry itself is different, so the decomposition result may also change.

The global optimizer is a Held-Karp dynamic program over the final (post-merge) partitions. Note that since the number of candidates per partition is bounded $M \leq 8$, the overall time complexity of this method is $O(N^2 \cdot 2^N)$, which is significantly more efficient than the brute-force approach, and when paired with our low-partition-count decomposition, it remains tractable for the vast majority of polygonal regions encountered in practice. For large partition counts, we recommend a Nearest-Neighbor-style [16] heuristic that incrementally connects partition-candidate pairs to minimize cumulative sweep and connector cost.

V. EVALUATION

A. Experimental Setup & Baselines

We evaluate our method on a benchmark of 13 irregular polygonal ROIs with varied scale, aspect ratio, concavity structure, and internal holes, designed to resemble segmented landcover features (e.g., woodlands, water bodies, and clustered buildings) with geometrical complexities that pose

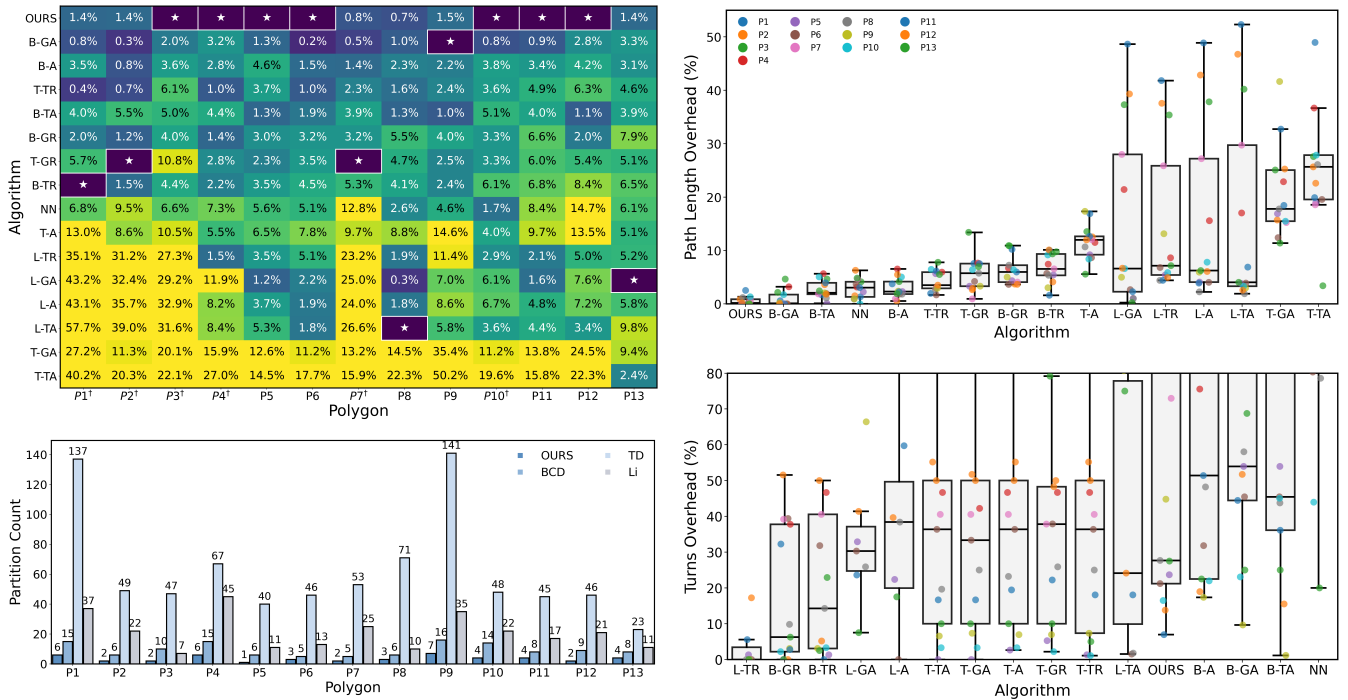


Fig. 3: Performance overview across 13 benchmark polygons. **Top-left:** Traversal-time overhead heatmap (%), sorted by average overhead (darker is better); * marks the best-performing algorithm per polygon. Polygons marked with † contain interior holes, Li cannot handle holes, so we fill them before planning with Li. **Bottom-left:** Partition counts per polygon for each decomposition method (OURS, BCD, TD, Li). **Top-right:** Path length overhead distributions; **Bottom-right:** Turn overhead distributions. Box plots are sorted by mean overhead. Algorithms include OURS (proposed), BCD (boustrophedon), T (trapezoidal), L (Li et al.), GA (genetic algorithm), ACO (ant colony optimization), TA (Threshold Accepting), TR (Torres et al.), GR (Google OR-Tools), and NN (nearest neighbor).

nontrivial challenges to modern CPP pipelines. We construct 15 CPP pipelines by pairing one decomposition method with one global optimizer (i.e., the pairwise combinations considered in this study). The decomposition module outputs a partition set, and the global optimizer determines the global coverage trajectory over those partitions, including candidate sweep selection and visitation order.

We use three decomposition baselines: Trapezoidal Decomposition (TD) [3], Boustrophedon Cellular Decomposition (BCD) [4], and Li’s vertex-edge minimum-width decomposition [6]. TD and BCD are implemented using Fields2Cover [17], and Li’s method is implemented using MathWorks `coverageDecomposition` [18]. Li’s method, inspired by Huang’s minimal-altitude criterion, is a key baseline because it captures the central trade-off in width-based decomposition, namely reduced within-partition turn cost at the expense of increased partition count (i.e., overfragmentation).

For global routing/selection, we use Torres et al.’s heuristic [10] (MathWorks `uavCoveragePlanner` [19]), Google OR-Tools [20] (the default TSP backend used in Fields2Cover), and three reimplemented metaheuristics following the partition-order/sweep-candidate encoding of Jimenez et al. [11] with respective hyperparameters listed in Table II: GA, ACO, and Threshold Accepting (TA). As an additional non-decomposition baseline, we include a nearest-neighbor (NN) planner that greedily connects raw sweep waypoints from the original track set.

Each pipeline outputs a stitched waypoint trajectory that guarantees full coverage. To estimate travel time under realistic motion constraints, we use Ruckig [21], [22] for offline time parameterization of the stitched waypoint trajectory (i.e., waypoint-to-waypoint line segments). Under our holonomic UAV model (Sec. III), omnidirectional planar motion eliminates the need for heading reorientation and curvature-constrained travel between consecutive segments. Hence, traversal time is determined by the translational motion along each segment together with the deceleration/acceleration needed at transitions between consecutive segments. Our Ruckig-based model captures this behavior directly by time-parameterizing each segment under bounded velocity, acceleration, and jerk. The resulting trajectory traversal time serves as our primary performance metric. We additionally report total path length, partition count per decomposition strategy, and number of turns. All metrics are reported as relative overheads with respect to the best value achieved for each polygon, computed as $(M - M_{\min})/M_{\min}$. This normalization enables fair comparison across polygons with different scales and geometric complexity.

B. Analysis & Discussion of Results

The experimental results in Figure 3 and Table I suggest that the decomposition strategy is the primary driver of end-to-end performance. In general, methods that minimize the number of partitions achieve lower time-completion and path-length overheads, with more consistent performance across different polygon shapes. For instance, our proposed

TABLE I: Algorithm Performance Summary (Sorted via Lowest Traversal-Time Overhead)

Alg.	μ_T^\downarrow (%)	μ_L^\downarrow (%)	μ_K^\downarrow (%)	Win $^\uparrow$	Top3 $^\uparrow$
OURS	0.56	0.56	63.88	7	11
BCD-GA	1.32	0.97	69.27	1	9
BCD-ACO	2.85	3.00	67.17	0	1
TD-TR	2.97	4.33	38.15	0	4
BCD-TA	3.26	2.76	82.38	0	3
BCD-GR	3.63	6.36	17.26	0	2
TD-GR	4.00	5.58	37.95	2	2
BCD-TR	4.29	6.74	26.10	1	1
NN	7.06	2.82	160.40	0	1
TD-ACO	9.02	11.61	37.86	0	0
LI-TR	11.95	15.49	1.85	0	0
LI-GA	12.91	15.40	27.86	1	4
LI-ACO	14.18	16.35	35.57	0	0
LI-TA	15.20	16.57	63.29	1	1
TD-GA	16.95	20.83	37.53	0	0
TD-TA	22.33	27.19	37.21	0	1

Notation μ_T : mean traversal-time overhead relative to the per-polygon best; μ_L : mean path-length overhead; μ_K : mean turns overhead; **Win**: number of first-place finishes; **Top3**: number of top-three finishes. Arrows: \uparrow higher is better, \downarrow lower is better.

method produces the fewest partitions across all polygons, typically in the single digits or low teens (Bar plot in Fig. 3). This directly contributes to its strong performance, achieving a mean path-length overhead of 0.56%, seven wins, and eleven top-three finishes.

This trend extends to other decomposition strategies. BCD produces more partitions than our method but avoids excessive over-fragmentation seen with TD and Li. When paired with GA, BCD becomes the second-best performer, with 1.32% mean overhead, and nine top-three finishes (Table I). In contrast, pipelines based on TD and Li show much weaker performance. The lowest seven ranks in both path-length and execution-time overhead are occupied by TD- or Li-based configurations.

These results suggest that optimizer choice is typically secondary to decomposition quality. Within each decomposition family, changing the optimizer shifts performance by only 2 to 3 percentage points. For example, BCD ranges from 1.32% with GA to 4.29% with Torres. Likewise, Li spans 11.95% to 15.20% across optimizers, while TD has more extreme variance, reaching up to 22.33% in its worst configuration (Table I). These patterns suggest the optimizer’s limited ability to mitigate poor decomposition. Once decomposition defines the number and geometry of regions, sequencing can only offer marginal gains, but they cannot compensate for the overhead imposed by excessive partition counts or poorly aligned partitions.

Figure 3 (turn-overhead box plot) highlights a key limitation of decomposition of width-driven decomposition strategies. While such approaches have been shown to lower turning costs (as discussed in Section II), their utility diminishes in the holonomic setting. Specifically, width-based decomposition often increases the total number of partitions, creating a competing objective between within-region turn cost and global path compactness. This trade-off is partic-

TABLE II: Metaheuristics Parameter Settings

Algorithm	Parameter(s)	Value(s)
TA	$\lambda_{\text{turns}}, K, p_{\text{choice}}$	0.15, 20000, 0.45
	t_0, t_1	2.0, 0.02
ACO	$\lambda_{\text{turns}}, K, w_{\text{elite}}, m$	0.15, 200, 2, $\max(10, 2 \cdot P)$
	$\alpha_{\text{order}}, \alpha_{\text{choice}}, \beta, \rho, q$	1.0, 1.0, 2.0, 0.10, 1.0
	$\tau_0, \tau_{\text{min}}, \tau_{\text{max}}$	1.0, $10^{-6}, 10^6$
GA	$\lambda_{\text{turns}}, N, G, f_{\text{elite}}, k_{\text{tourn}}$	0.15, 450, 350, 0.05, 4
	$p_{\text{mut,order}}, p_{\text{mut,choice}}$	0.30, 0.40

Notation K iterations; P number of partitions; $m = \max(10, 2 \cdot P)$ ants; $\alpha_{\text{order}}, \alpha_{\text{choice}}$ pheromone exponents; β heuristic exponent; ρ evaporation rate; q deposit scale; w_{elite} elite deposit weight; $\tau_0, \tau_{\text{min}}, \tau_{\text{max}}$ pheromone levels; N GA population size; G generations; f_{elite} elite fraction; k_{tourn} tournament size; $p_{\text{mut,order}}, p_{\text{mut,choice}}$ mutation probabilities; λ_{turns} turns weight.

ularly unfavorable for holonomic UAVs, which can reorient in place with minimal time or energy penalty. The resulting proliferation of narrow partitions leads to longer inter-region connectors and more frequent entry-exit transitions, which inflate global trajectory length and execution time. Moreover, excessive partition counts exacerbate planning complexity. Optimizers that rely on combinatorial reasoning—such as metaheuristics or TSP-style solvers—scale poorly with partition number. As partition count grows, these optimizers either become intractable or must resort to early convergence and local minima, degrading overall performance. Consequently, decomposition strategies that emphasize width minimization at the expense of region count tend to underperform in our context. This effect is evident in the relatively high overheads of width-driven methods like Li, compared to OURS and BCD, which favor structural simplicity and lower partition counts.

The Nearest-Neighbor baseline (7.06% time overhead) further confirms that structured decomposition remains essential even for holonomic vehicles, as greedy point-to-point traversal over raw sweep waypoints produces disorganized and redundant coverage sequences.

These results show that effective coverage planning over irregular polygons hinges on balancing geometric flexibility with partition simplicity. By deferring sweep direction selection and replacing width-based splitting with a cumulative gap severity metric, our method preserves larger, more sweep-efficient regions. This reduces both path length and execution time without incurring the penalties of over-fragmentation.

VI. CONCLUSION

This work introduced a decomposition and global optimization strategy tailored to the challenges of holonomic UAV-based CPP in highly irregular environments, such as those found in emergency response missions. We depart from conventional width-based approaches that dominate the CPP literature, and instead we propose a viable alternative built on a uniaxial feasibility criterion and a cut selection process guided by cumulative gap severity. The goal is to distribute clusters of concavity across fewer, structurally sweepable subregions to minimize partition count while preserving

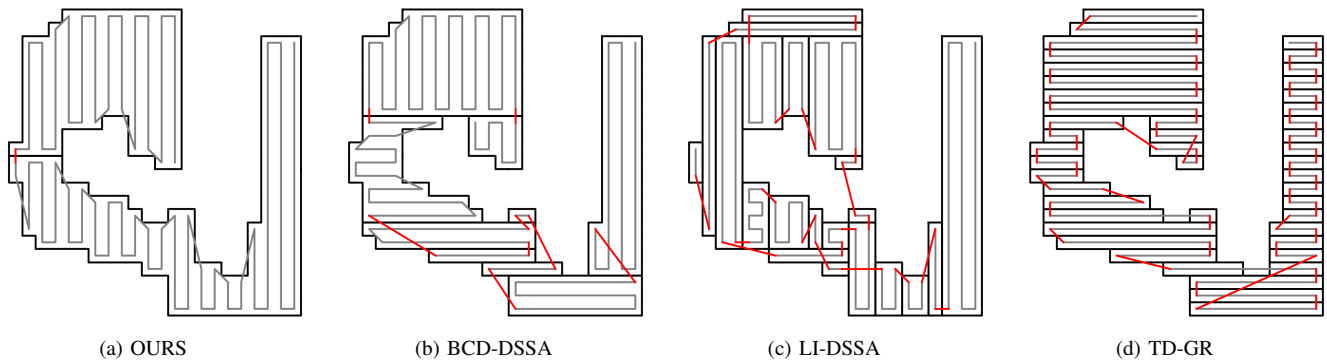


Fig. 4: Final stitched coverage on polygon P12 for Ours and three top-performing baselines, one from each decomposition family. Red lines indicate inter-partition connectors.

parallel-track feasibility. Furthermore, we presented a global optimizer that enumerates sweep candidates for each partition and jointly selects region paths and transition connectors to minimize path length, turn count, and connector cost.

We benchmarked our method against fifteen CPP pipelines comprising established decompositions and state-of-the-art optimizers. Generally, results show that decomposition quality is the dominant factor influencing performance. Methods producing fewer, larger partitions yield shorter execution times and lower path overheads. This advantage stems from reduced inter-partition transitions and fewer axis discontinuities. Our method achieved the lowest mean time and path overheads while demonstrating the most consistent performance across a diverse set of complex polygons. In emergency response applications, the compounding effect of small inefficiencies is far from negligible. Real-world missions often require coverage of dozens or hundreds of disjoint ROIs, where even modest overheads accumulate into significant delays that can impact mission outcomes. Code and benchmark data are publicly available in [16].

REFERENCES

- [1] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 1. IEEE, 2001, pp. 27–32.
- [2] M. Höffmann, S. Patel, and C. Büskens, "Optimal guidance track generation for precision agriculture: A review of coverage path planning techniques," *Journal of Field Robotics*, vol. 41, no. 3, pp. 823–844, 2024.
- [3] R. Seidel, "A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons," *Computational Geometry*, vol. 1, no. 1, pp. 51–64, 1991. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0925772191900124>
- [4] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.
- [5] R. Bähmann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," in *Field and Service Robotics: Results of the 12th International Conference*. Springer, 2021, pp. 277–290.
- [6] Y. Li, H. Chen, M. Joo Er, and X. Wang, "Coverage path planning for uavs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, 2011, special Issue on Development of Autonomous Unmanned Aerial Vehicles. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957415810001893>
- [7] X. Yu and J. Y. Hung, "Coverage path planning based on a multiple sweep line decomposition," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015, pp. 004 052–004 058.
- [8] L. D. Nielsen, I. Sung, and P. Nielsen, "Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges," *Sensors*, vol. 19, no. 19, p. 4165, 2019.
- [9] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning," in *2016 IEEE international conference on automation science and engineering (CASE)*. IEEE, 2016, pp. 1237–1242.
- [10] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert Systems with Applications*, vol. 55, pp. 441–451, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417416300306>
- [11] P. A. Jimenez, B. Shirinzadeh, A. Nicholson, and G. Alici, "Optimal area covering using genetic algorithms," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*. IEEE, 2007, pp. 1–5.
- [12] J.-H. Wei and J.-S. Liu, "Generating minimax-curvature and shorter n3-spline path using multi-objective variable-length genetic algorithm," in *2010 International Conference on Networking, Sensing and Control (ICNSC)*, 2010, pp. 319–324.
- [13] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.
- [14] A. Rom and I. Kelman, "Search without rescue? evaluating the international search and rescue response to earthquake disasters," *BMJ Global Health*, vol. 5, no. 7, p. e002398, 2020.
- [15] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied mathematics*, vol. 10, no. 1, pp. 196–210, 1962.
- [16] T. Alarcon, "Coverage-path-planning-pugs: Reference implementation for coverage path planning via uniaxial-feasible, gap-severity guided decomposition," <https://github.com/TonyAlarcon/Coverage-Path-Planning-PUGS>, 2026, gitHub repository, MIT License, accessed 2026-02-23.
- [17] G. Mier and contributors, "Fields2cover: Open-source coverage path planning library," <https://fields2cover.github.io/>, 2025, accessed: 2025-09-11.
- [18] MathWorks, "coveredecomposition (matlab uav toolbox) — decompose coverage regions into subregions using vertex-edge decomposition," MATLAB Documentation, UAV Toolbox, 2025, online; accessed 2025-09-XX. [Online]. Available: <https://www.mathworks.com/help/uav/ref/coveredecomposition.html>
- [19] —, "uavcoverageplanner (uav toolbox)," <https://www.mathworks.com/help/uav/ref/uavcoverageplanner.html>, 2025, accessed: 2025-09-11.
- [20] Google, "Or-tools," <https://developers.google.com/optimization>, 2025, accessed: 2025-09-13.
- [21] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," 2021. [Online]. Available: <https://arxiv.org/abs/2105.04830>
- [22] —, "Ruckig: Online trajectory generation library," <https://pypi.org/project/ruckig/>, 2025, accessed: 2025-09-11.