On Minimizing Turns in Robot Coverage Path Planning

Stanislav Bochkarev Stephen L. Smith

Abstract— **In this paper we study sweep coverage path planning, in which a robot must cover all points in a workspace with its footprint. In many coverage applications, including cleaning and monitoring, it is beneficial to use coverage paths with minimal robot turns. To address this, we provide an efficient method to compute the minimum altitude of a non-convex polygonal region, which captures the number of parallel line segments, and thus turns, needed to cover the region. Then, given a non-convex polygon, we provide a method to cut the polygon into two pieces that minimizes the sum of their altitudes. Given an initial convex decomposition of a workspace, we apply this method to iteratively re-optimize and delete cuts of the decomposition. Finally, we compute a coverage path of the workspace by placing parallel line segments in each region, and then computing a tour of the segments of minimum cost. We present simulation results on several workspaces with obstacles, which demonstrate improvements in both the number of turns in the final coverage path and runtime.**

I. Introduction

Given a robot with a footprint χ and a workspace P, possibly with holes, the sweep coverage problem is that of computing a path contained in P such that traversal of χ along the path results in each point in P being covered by χ. Coverage path planning sees applications in demining [1], surveillance [2], search and rescue [3], target localization [4], floor sweeping [5], [6], agriculture [7], painting [8], polishing [9] and others. The evaluation of the coverage path differs between these applications. For example, energy efficiency is important for floor sweeping robots operating on constrained energy budget, whereas uniformity of the coverage is important for painting applications.

Despite differences in evaluation, for many robot types, turns have undesirable effects on the path quality [10], [11]. For example, turns for aerial vehicles with vision based acquisition systems complicate the scene acquisition process [12]. For painting applications, turns over the paint surface introduce travel length differences for parts of the nozzle, which disturbs the uniformity of the paint.

The optimal coverage path generation for convex work spaces is tractable and can be solved efficiently using sweeping [13] or spiralling motions. For general workspaces, the coverage problem is NP-hard [14]. However, there are many approximate approaches to coverage. One class of approaches is an *exact* convex decomposition on the workspace, in which the workspace is partitioned into convex regions. A coverage path consists of a tour of each region, with local coverage of a convex region performed with either sweeping or spiralling motion. Commonly-used methods for decomposition include Boustrophedon [15] and trapezoidal decomposition [16]. Huang [10] proposed a dynamic programming approach for the decomposition that minimizes the number of turns in the overall coverage path. However, the runtime is exponential, and the cost associated with transitions between these regions is assumed to be negligible. Exact convex decomposition for polygons with holes is NP-hard [17], which leads to a drawback. The decomposition of complex workspaces may contain many regions, resulting in large total transition costs between regions.

Another class of approaches is an *approximate* convex decomposition [18], in which convex regions may overlap, and the union of regions approximates the workspace. It's common to represent each of the overlapping regions as a coverage footprint (Figure 1(a)). Local coverage of each region is ensured by visiting the region's center. Complete coverage becomes the problem of planning a tour that visits each center with minimum cost [14]. A drawback of this method is that the overlap between regions reduces the efficiency of the overall coverage path. Also, the number of regions grows with the area of the workspace, making the problem of computing tours of the centers intractable for large spaces.

In this work we seek to compute coverage paths with minimum turns. Our key idea is to perform a line decomposition: an approximate decomposition of the workspace into regions that represent the area swept by the footprint traversing a straight line (Figure 1(b)). The process starts with any convex decomposition. We propose a method of re-evaluating cuts in this convex decomposition with the objective of lowering the number of turns in the path. For each polygon in the final decomposition, a minimal line decomposition is performed. The complete coverage path is then generated by computing a tour of lines with minimal cost. Unlike existing approaches, the tour is not forced to complete local coverage of each polygon before proceeding to the next.

Contributions: Our key contributions are threefold. First, we provide a method for computing the minimum altitude of a non-convex polygon, which captures the number of parallel line segments, and thus turns, needed for coverage. Second, given an initial convex decomposition of a workspace, we propose a method to iteratively re-optimize and delete cuts of the decomposition in order to optimize the altitude of the polygon on each side of the cut. Third, we compute a coverage path of the workspace by placing parallel line segments in each region, and then computing a minimum-length tour of the corresponding approximate convex decom-

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (sbochkar@uwaterloo.ca; stephen.smith@uwaterloo.ca).
This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

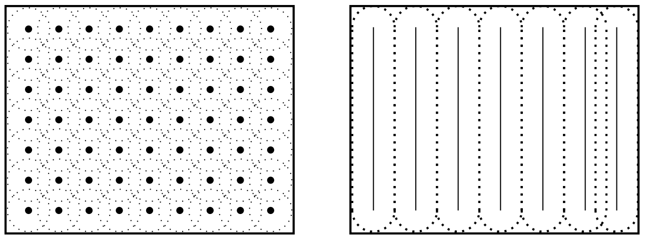
(a) (b)

Fig. 1: (a) Coverage footprint at a point. (b) Coverage footprint over a line.

position. The tour is computed by generating and solving a Generalized Traveling Salesman Problem (GTSP) instance using existing solvers. Proofs are abbreviated to sketches due to space constraints and will appear elsewhere in full.

II. Preliminaries

A. Geometric Preliminaries

A *simple polygon* is a non-intersecting chain of straight line segments forming a closed loop and specified as a set of points, i.e., $Z = \{v_i \in \mathbb{R}^2 | i = 1, \ldots, n\}$. Note that since the chain is a circuit, any $v_i \in Z$ has two adjacent line segments. A *boundary of a simple polygon* is a set of points, ∂Z, along a line connecting any two adjacent vertices of a chain. A *clockwise simple polygon* is a simple polygon where vertices are specified in a clockwise order. We associate these types of simple polygons with holes in the workspace. A *counter-clockwise simple polygon* is a simple polygon where vertices are specified in a counter-clockwise order. This type of simple polygons are associated with the external boundary of the workspace. Finally, a *polygon* is a set of clockwise and counter-clockwise simple polygons. For clarity, we will refer to simple polygons as chains and reserve the term polygon for a set of chains, i.e., $P = \{Z_0, \ldots, Z_m\}$. Z_0 is a counter-clockwise chain (i.e., the boundary) and all subsequent Z_i are clockwise chains (i.e., holes). A *boundary of a polygon* is the set of points defined as $\partial P = \bigcup_{i=1}^{M} \partial Z_i$ where $Z_i \in P$.

A *reflex vertex* is a vertex in one of the chains of P that has an internal angle of more than π. As such, a polygon is *convex* if and only if there are no reflex vertices. Conversely, a polygon is non-convex if and only if it contains at least one reflex vertex. Note that the presence of a hole guarantees at least one reflex vertex.

B. Generalized Traveling Salesman Problem Preliminaries

Suppose a complete graph $G = (V, E, w)$ is given where V is a set of vertices, E is a set of edges, and w is a set of edge weights. Suppose V is partitioned into pairwise disjoint sets $\{V_1, \ldots, V_p\}$ where $V_i \subset V$. The Generalized Traveling Salesman Problem (GTSP) is the problem of computing a tour that visits exactly one vertex from each V_i such that the length of the tour is minimized. The TSP is a special case of the GTSP where $|V_i| = 1$ for each i and is NP-hard.

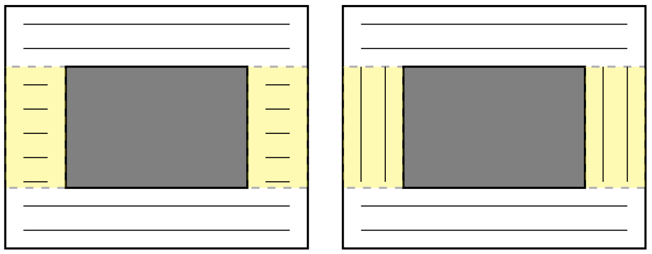
Fig. 2: Polygon regions where lines can be reoriented.

III. Problem Definition and Approach

Given a polygon P possibly with holes, our goal in this paper is to compute a coverage path with the additional objective of minimizing the number of turns. We assume that the map of the workspace is polygonal and noise free, and the localization problem is solved. We deal with kinematic robot models with $v_{\max}$ as the maximum velocity. In Section VI, we utilize Dubins' car model for our simulations. We assume robot's coverage footprint is a circle of radius r. Our approach to the coverage problem consists of two main subproblems, described in the following two subsections.

A. Minimum Turn Decomposition

We focus on the specific type of paths called polygonal sweeping paths, which can be segmented into two types of segments: straight and transition segments. For a given polygonal path, define a set of straight segments in the path as R and set of transition segments as T. Transition segments are associated with turns as they represent a path from one straight segment to another. Observe that minimizing $|R|$ leads to minimizing $|T|$. Note that the traversal of the coverage footprint along a straight segment $R_i \in R$ generates a straight segment footprint γ_i. See Figure 1(b). Note that the union of all γ_i is approximately equal to the original polygon with areas near the boundaries missing. With these observations, our first problem can be formulated as follows:

Problem III.1 (Min Line Set Decomposition)**.** *Given a polygon P, compute a set $\Gamma = \{\gamma_i | i = 1, \ldots, f\}$ such that f is minimized and $\bigcup_{i=1}^{f} \gamma_i$ approximately decomposes P.*

The solution to this problem is not trivial. One possible solution is to arrange all $\gamma_i \in \Gamma$ to be parallel to each other throughout the entire polygon. This solution is not necessarily optimal but can act as a starting point in an optimization. A more sophisticated approach is to decompose P into polygons $P_0, P_1, \ldots, P_k$, where for each P_i, the associated Γ_i consists of parallel segments. The segments in each Γ_i should be oriented to minimize $|\Gamma_i|$. For example, the polygon in Figure 2 is decomposed into four regions. Lines in the two regions on the sides of the polygon are reoriented to reduce the number of lines in those regions. For the optimal orientation, we define the θ altitude of a polygon.

Minimum Altitude: For *convex* polygons, the altitude in direction θ is defined as the minimum distance between two parallel lines angled at $\theta + \frac{\pi}{2}$ with respect to the x-axis

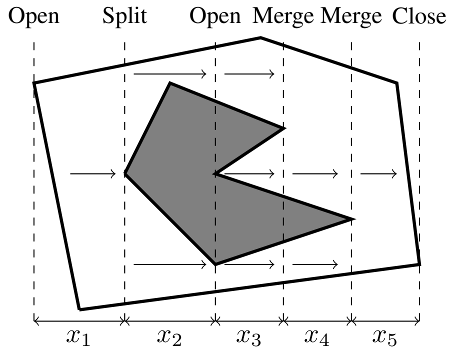

Fig. 3: Process of measuring altitude with θ equal to 0.

such that all vertices of the convex polygon are contained between these two lines [10]. This notion can be extended to general polygons. The method of obtaining θ altitude for a general polygon is shown in Algorithm 1. Figure 3 provides an example of this procedure. In the example, the altitude is $x_1 + 2x_2 + 3x_3 + 2x_4 + x_5$. Note that if r is the radius of the footprint of the robot and α is the altitude of a polygon along θ then total number of lines required for complete coverage in direction θ is

$$n = \left\lfloor \frac{\alpha}{2r} \right\rfloor. \tag{1}$$

Thus, the altitude provides the cardinality of set Γ_i when all $\gamma_j \in \Gamma_i$ are oriented at $\theta + \frac{\pi}{2}$ with respect to x-axis.

Algorithm 1 get_general_altitude(P, θ)

```
Input: Polygon $P = \{Z_0, Z_1, \ldots\}$, direction $\theta$.
 1: counter $\leftarrow$ 0, $\alpha \leftarrow 0$
 2: Rotate $P$ by $-\theta$ to align direction with $x$-axis.
 3: Sort all vertices in $P$ by their $x$-coordinates
 4: for each $v_i$ in the sorted list :
 5:     $\alpha \leftarrow \alpha +$ counter $\times (x_{v_i} - x_{v_{i-1}})$
 6:     if both vertices adjacent to $v_i$ are on the right :
 7:         Increment counter by 1
 8:     else if both vertices adjacent to $v_i$ are on the left :
 9:         Decrement counter by 1
10: return $\alpha$
```

Given a polygon P, we are interested in finding the minimum altitude $\alpha^*(P)$. Note, however, that the altitude can be measured with respect to an infinite number of directions θ. The following result addresses this issue.

Proposition III.1 (Minimum Altitude Directions, [10])**.** *Given a polygon P, the direction of minimum altitude is orthogonal to one of the edges of the polygon.*

By Proposition III.1, minimum altitude can be computed by performing Algorithm 1 on each direction associated with the edge of the polygon. The runtime of this approach is $O(n^2 \log n)$ in number of vertices of a polygon.

With the notion of altitude for general polygons, the Problem III.1 can be restated as follows.

Problem III.2 (Min Altitude Decomposition)**.** *Given a polygon P, find k and $P_1, P_2, \ldots, P_k$ such that $\sum_{i=1}^{k} \alpha^*(P_i)$ is minimized where $\bigcup_{i=1}^{k} P_i = P$.*

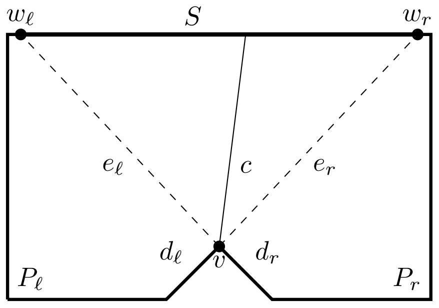

Fig. 4: Decomposing cut originating at a reflex vertex.

B. Minimum Cost Tour

Once a set Γ is generated, the problem becomes that of choosing an order and direction to cover each $\gamma_i \in \Gamma$, i.e., a tour of Γ. Observe that each γ_i can be traversed with two directions. Hence, the second subproblem can be defined as follows:

Problem III.3 (Minimum Cost Tour)**.** *Given a polygon P and a set of lines Γ, generate a matrix M of transition costs between any pair of elements of Γ. Find a tour of Γ such that cost of the tour is minimized.*

IV. Min Altitude Polygon Decomposition for Coverage Planning

This section covers our solution to Problem III.2. Section IV-A introduces decomposition related terminology that is needed to present our decomposition method.

A. Decomposition Related Preliminaries

Definition IV.1 (Cone of bisection)**.** *Suppose a non-convex polygon P containing a reflex vertex v is given. Let $d_\ell = \{v_k, v\}$ and $d_r = \{v, v_l\}$ be the two adjacent edges of v. The* cone of bisection *at v is defined by two line segments, $e_\ell = \{v, w_\ell\}$ and $e_r = \{v, w_r\}$ that are parallel to d_r and d_ℓ respectively with $w_\ell, w_r \in \partial P \setminus \{v\}$. See Figure 4.*

Definition IV.2 (Cut space)**.** *Consider a non-convex polygon P with reflex vertex v, along with the cone of bisection. The cut space is a set $S \subset \partial P$ of all points on ∂P between w_ℓ and w_r visible from v.*

The cut space S can be represented by its straight line segments $S_1, \ldots, S_L$ where $\bigcup_{i=1}^{L} S_i \subseteq S$.

Definition IV.3 (Decomposing Cut)**.** *Given a non-convex polygon P with a reflex vertex v, a decomposing cut is a straight line segment $e = \{v, w\}$ within the cone of bisection of v where v, w belong to the same chain. See Figure 4.*

B. Min Altitude Decomposition

This section outlines the approach to Problem III.2. The overall steps of the process are described in Algorithm 2.

Algorithm 2 accepts a polygon as the input. On Line 1, an initial decomposition is generated using any convex decomposition technique. The rest of the algorithm attempts to re-optimize this decomposition in order to decrease the

Algorithm 2 min_alt_decomposition(P)

```
Input: Polygon $P = \{Z_0, Z_1, \ldots\}$
1: $D \leftarrow$ any convex decomposition of $P$
2: repeat
3:     Re-optimize a cut from $D$
4:     Update cost of the decomposition
5: until stopping criteria is met
```

overall altitude. Re-optimization steps on Lines 3 and 4 are performed until the cost of the decomposition no longer decreases. Part of the re-optimization step is the removal of existing cuts. This is achieved by maintaining a list of reflex vertices from the original polygon. For each vertex in this list, the algorithm looks for incident edges that were not present in the original polygon. These edges are removed, forming a combined polygon in the process. An optimal decomposing cut from this reflex vertex is attempted. The procedure for making optimal decomposing cuts is introduced in Section IV-C.

C. Optimal Decomposing Cut

The re-optimization step operates on a polygon with a reflex vertex. This polygon is formed by removing previous cuts that were made by a convex decomposition. Our algorithm will either generate a new cut or no cut depending on what is optimal with respect to the altitude of the polygon. We do this by searching for optimal decomposing cuts. The algorithm for doing that is shown in Algorithm 3. The procedure operates on a single chain and a specified reflex vertex in that chain. On Line 2, a cut space is generated for the reflex vertex, which provides a set of potential cuts. Two polygons, P_ℓ and P_r, are formed by initializing the cut at the first point of the cut space on Line 3. Two sets of altitude directions are generated on Lines 4-5 based on the two polygons. The main loop on Lines 6-15 locates candidates for a cut for each combination of directions and each straight segment of the cut space. Lines 9-10 locate special points on S_i called transition points for each altitude direction. These points are points on S_i that yield cuts minimizing the sum of altitudes. Transition points are found by locating a point of intersection of S_i and a line h_i parallel to the edge e_i passing through v (left of Figure 5). Before moving on to the next straight segment of the cut space, P_l is modified to include S_i in its boundary. P_r is modified to exclude S_i from its boundary. The straight segment S_i may also add a new altitude direction θ, which has to be accounted for in Θ_ℓ and Θ_r. These operations are carried out on Lines 13-15. The algorithm returns the cut that has the lowest cost.

D. Proof of Correctness

This section establishes the correctness of Algorithm 3.

Claim IV.1. *Given a polygon P and a reflex vertex v, Algorithm 3 returns a decomposing cut forming two new polygons P_ℓ and P_r that minimize $\alpha^*(P_\ell) + \alpha^*(P_r)$.*

Due to space constraints, we provide only a sketch of the proof, which will appear in full elsewhere. The proof

Algorithm 3 find_optimal_cut(P, v)

```
Input: Polygon $P = \{Z_0\}$, reflex vertex $v$
1: $A_\ell \leftarrow \emptyset,\ A_r \leftarrow \emptyset,\ U \leftarrow \emptyset$
2: Find cut space $S$ for $v$
3: $P_\ell, P_r \leftarrow$ polygons after the cut at first endpoint of $S$
4: $\Theta_\ell \leftarrow$set of directions $\theta$ orthogonal to edges of $P_\ell$
5: $\Theta_r \leftarrow$set of directions $\theta$ orthogonal to edges of $P_r$
6: for each $S_i \in S$ :
7:     for each $\mathrm{dir}_\ell \in \Theta_\ell$ :
8:         for each $\mathrm{dir}_r \in \Theta_r$ :
9:             $d_\ell \leftarrow$ transition point on $S_i$ w.r.t $\mathrm{dir}_\ell$
10:            $d_r \leftarrow$ transition point on $S_i$ w.r.t $\mathrm{dir}_r$
11:            Add tuple $(d_\ell, \mathrm{dir}_\ell, \mathrm{dir}_r)$ to $U$
12:            Add tuple $(d_r, \mathrm{dir}_\ell, \mathrm{dir}_r)$ to $U$
13:    Modify $P_\ell$ by adding $S_i$ to its boundary
14:    Modify $P_r$ by removing $S_i$ from its boundary
15:    Modify $\Theta_\ell, \Theta_r$ if new $\theta$ was introduced by $S_i$
16: Compute costs for all tuples $(u, \mathrm{dir}_\ell, \mathrm{dir}_r) \in U$
17: return Lowest cost element from $U$
```

consists of several steps. First, the existence of a minimum is shown. Next, transition points are introduced, which contain a minimum. Finally, the minimum is found by evaluating all transition points.

We are interested in studying how a choice of a cut affects the altitudes of P_ℓ and P_r. Let us parameterize the orientation of a cut with respect to d, where d is the normalized distance along the cut space from w_ℓ. Let w_ℓ be a point on the cut space when $d = 0$ corresponding to the cut on edge e_ℓ (Figure 4). Define $\alpha_{\ell,i}(d)$ and $\alpha_{r,i}(d)$ as functions that measure the altitude orthogonal to edge e_i in P_ℓ and P_r respectively.

Note, all $\alpha_{\ell,i}(d)$ and $\alpha_{r,i}(d)$ are semi-continuous functions of d, with discontinuities occurring only when the cut sweeps past a pinch vertex in the cut space as shown on the right of Figure 5. Utilizing the extension of the extreme value theorem, Lemma IV.1 is established.

Lemma IV.1. *There exists a $d \in [0, 1]$ that minimizes $\alpha_{\ell,i}(d) + \alpha_{r,j}(d)$.*

The next part of the proof shows that transition points are actually points that minimize the altitude functions. The following Lemma formalizes this. Let d_ℓ^{trans} and d_r^{trans} refer to transition points for P_ℓ and P_r respectively.

Lemma IV.2. *For a given S_i, altitudes $\alpha_{\ell,i}(d)$ and $\alpha_{r,j}(d)$, a minimizer of $\alpha_{\ell,i}(d) + \alpha_{r,j}(d)$ is $d \in \{d_\ell^{\text{trans}}, d_r^{\text{trans}}\}$.*

The proof of this lemma follows from the linearity of altitude functions. With our parameterization, altitude functions are shown to be piece-wise linear functions of d. The minima of the sum of such functions can be shown to occur at transition points. With Proposition III.1 and Lemma IV.2, we find the optimal cut by computing the sum for each $S_i \in S$ and all combinations of altitude directions and pick the transition point with the minimum. Algorithm 3 performs this exact procedure.

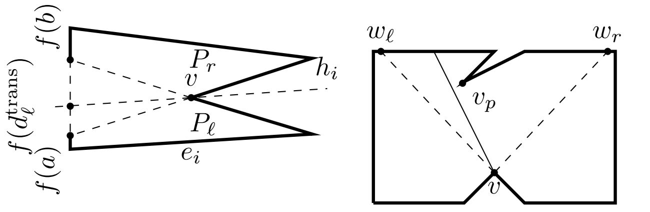

Fig. 5: Left: Transition point. Right: Pinch vertex at v_p.

V. GTSP Tour Generation

Given a polygon P, Algorithm 2 is called on P, resulting in a set D of polygons. For each $P_i \in D$, we generate the minimum set Γ_i of straight segment footprints. Note, each $\gamma_j \in \Gamma_i$ has two possible traversal directions. We refer to the choice of this direction as γ_j^1 or γ_j^2. Define $\text{cost}(\gamma_i^m, \gamma_j^l)$ as the robot transition cost from region γ_i in m direction to region γ_j in l direction where $m, l \in \{1, 2\}$ and $i, j \in \{1, \ldots, k\}, i \neq j$. This transition cost takes the dynamics of the robot into account. Computing this cost for each pair γ_i^m, γ_j^l, a complete graph $G = (V, E, w)$ is constructed where each vertex $v \in V$ represents γ_i^m for some $m \in \{1, 2\}$ and $i = \{1, \ldots, k\}$, each edge $e = \{v, z\}$ in E represents transition between vertices v and z, and weights represent robot transition costs between vertices. Furthermore, V is partitioned into sets $\{\gamma_i^1, \gamma_i^2\}$ for $i = \{1, \ldots, k\}$. The graph G and the partition define the GTSP instance, and a GTSP tour on this graph gives a coverage path for the robot.

VI. Simulations

Our algorithm was implemented in Python. We utilized several computational geometry libraries including Shapely [1] and Visilibity [19]. The heuristic solver GLKH [20] was used as a GTSP solver. Transition costs between segments were computed assuming a Dubins' vehicle model.

Our method was compared to two other approaches. The first method relies on an approximate point decomposition of the workspace from [14]. Each point is assigned eight headings. These headings represent possible traversal directions for that point. Finally, the GTSP tour of minimum cost is computed as in [21]. In the second method, the coverage path is computed but the re-optimization procedure is not performed on the initial decomposition. The initial decomposition is generated with a Python library for greedy convex decomposition, based on [22]. We tested all approaches on four different workspaces of similar size but various complexity.

Performance figures of all three methods are shown in Table I. We compare four aspects of the three approaches: the size of the GTSP instance, the execution time from start to finish, the total length of the tour, and the area covered by the coverage path as percentage of the total area of the polygon. In all cases, our method reduced the number of turns in the final coverage path. This is shown in the reduction in GTSP size, which is equivalent to the reduction in number of straight line segments. Naturally, since our method uses greedy decomposition as a starting point, it runs more slowly then the pure greedy approach. On average, our method is about three times slower than the greedy approach. However, it is faster then the point decomposition by a factor of 100.

Notably, the overall path lengths are shorter with the greedy decomposition compared to our method. However, this is attributed to the greedy decomposition producing many regions whose area is small relative to the size of the footprint. These regions are typically narrow, which makes it difficult to place lines for complete area coverage. Decompositions that produce excessive number of such regions result in a significant portion of the total area uncovered. Our method produces fewer regions and leaves less area uncovered. From Table I, our method leaves on average half of the uncovered area left by the greedy approach. Note that the point decomposition in some cases covers less area than the competing algorithms. However, this occurrence is due to the limitation of our implementation of the approximate decomposition, which does not place any points at a distance of less than r from the boundary, leading to uncovered areas near polygon boundaries. This has the effect of reducing the covered area and the path length as well.

Figure 6 shows the resultant paths for three different approaches. The line segments are orange and Dubins transition costs are green. Note that green segments do not necessarily indicate the path of the robot but rather show the cost associated with the transition. The red lines highlight shared edges of adjacent polygons in the decomposition. The first row of figures show the point decomposition. The second row of figures show the greedy decomposition technique. And the last row shows our re-optimization technique. Figure 6 demonstrates a reduction in the number of polygons with our method compared to the greedy decomposition. Inefficiencies in the GTSP tour occur due to the very large size of the GTSP instance, which poses a challenge for the heuristic GLKH solver.

VII. Conclusion

In this paper we proposed a new approach for robot coverage that utilizes an approximate convex decomposition of the environment into straight line segments. Our key contribution is an algorithm that seeks to minimize the number of lines by repeatedly performing optimal decomposing cuts in the environment. For future work, we are interested in extending this approach to workload partitioning for multiple robot coverage.

References

[1] E. U. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *The International journal of robotics research*, vol. 22, no. 7-8, pp. 441–466, 2003.

[2] A. Ahmadzadeh, J. Keller, G. Pappas, A. Jadbabaie, and V. Kumar, "An optimization-based approach to time-critical cooperative surveillance and coverage with UAVs," in *Experimental Robotics*. Springer, 2008, pp. 491–500.

[1]The Shapely library is available at `https://github.com/Toblerity/Shapely`

	Point Decomposition				Greedy Decomposition				Min Alt Decomposition			
	Size	Time	Length	Area	Size	Time	Length	Area	Size	Time	Length	Area
Shape 1	7144	2h	312.6	87.8%	96	**2s**	**231.0**	91.6%	**78**	2s	242.3	**95.1%**
Shape 2	7288	2h	306.3	86.9%	86	**2s**	**216.8**	84.6%	**80**	7s	228.9	**91.7%**
Shape 3	7464	2h	349.3	88.3%	136	**6s**	**226.9**	79.8%	**92**	15s	228.1	**89.1%**
Shape 4	6736	2h	312.2	85.1%	116	**3s**	**226.5**	88.1%	**88**	13s	234.4	**93.5%**

TABLE I: Decomposition methods comparison for multiple test shapes.

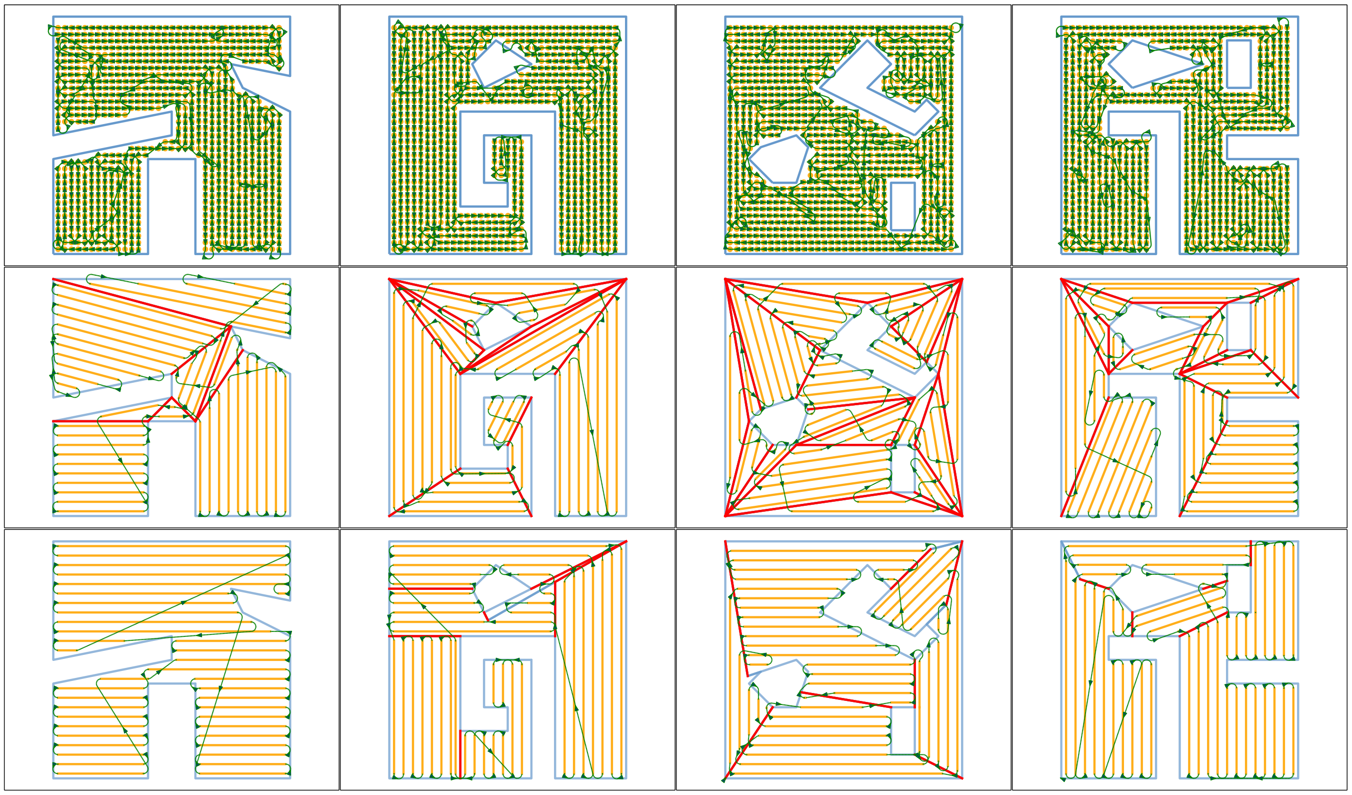

Fig. 6: First row: point decomposition. Second row: greedy optimization. Third row: min altitude decomposition.

[3] L. Lin and M. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 709–714.

[4] L. M. Miller and T. D. Murphey, "Optimal planning for target localization and coverage using range sensing," in *Int Conf on Automation Science and Eng*, 2015, pp. 501–508.

[5] J. Hess, M. Beinhofer, and W. Burgard, "A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots," in *IEEE Int Conf on Robotics and Automation*, 2014, pp. 5600–5605.

[6] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, "BA*: an online complete coverage algorithm for cleaning robots," *Applied Intelligence*, vol. 39, no. 2, pp. 217–235, 2013.

[7] I. Hameed, D. Bochtis, and C. A. G. Sørensen, "An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas," *International Journal of Advanced Robotic Systems*, vol. 10, no. 231, pp. 1–9, 2013.

[8] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Uniform coverage of automotive surface patches," *The International Journal of Robotics Research*, vol. 24, no. 11, pp. 883–898, 2005.

[9] M. Rososhansky, F. J. Xi, and Y. Li, "Coverage based tool path planning for automated polishing using contact stress theory," in *Int Conf on Automation Science and Eng*, 2010, pp. 592–597.

[10] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2001, pp. 27–32.

[11] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Efficient complete coverage of a known arbitrary environment with applications to aerial operations," *Autonomous Robots*, vol. 36, no. 4, pp. 365–381, 2014.

[12] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-based road-following using a small autonomous aircraft," in *IEEE Aerospace Conference*, vol. 5, 2004, pp. 3006–3015.

[13] I. Maza and A. Ollero, "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Distributed Autonomous Robotic Systems*, 2007, pp. 221–230.

[14] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1, pp. 25–50, 2000.

[15] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*. Springer, 1998, pp. 203–209.

[16] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651–668, 2009.

[17] A. Lingas, "The power of non-rectilinear holes," in *Automata, Languages and Programming*. Springer, 1982, pp. 369–383.

[18] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258 – 1276, 2013.

[19] K. J. Obermeyer and Contributors, "The VisiLibity library," `http://www.VisiLibity.org`, 2008, r-1.

[20] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[21] J. Le Ny, E. Feron, and E. Frazzoli, "On the Dubins traveling salesman problem," *IEEE Trans Automatic Ctrl*, vol. 57, no. 1, pp. 265–270, 2012.

[22] J. Fernández, B. Tóth, L. Cánovas, and B. Pelegrín, "A practical algorithm for decomposing polygonal domains into convex polygons by diagonals," *Top*, vol. 16, no. 2, pp. 367–387, 2008.