Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)

Multi-Robot Connected Fermat Spiral Coverage

Jingtao Tang, Hang Ma

Simon Fraser University
{jingtao_tang, hangma}@sfu.ca

Abstract

We introduce the Multi-Robot Connected Fermat Spiral (MCFS), a novel algorithmic framework for Multi-Robot Coverage Path Planning (MCPP) that adapts Connected Fermat Spiral (CFS) from the computer graphics community to multi-robot coordination for the first time. MCFS uniquely enables the orchestration of multiple robots to generate coverage paths that contour around arbitrarily shaped obstacles, a feature that is notably lacking in traditional methods. Our framework not only enhances area coverage and optimizes task performance, particularly in terms of makespan, for workspaces rich in irregular obstacles but also addresses the challenges of path continuity and curvature critical for non-holonomic robots by generating smooth paths without decomposing the workspace. MCFS solves MCPP by constructing a graph of isolines and transforming MCPP into a combinatorial optimization problem, aiming to minimize the makespan while covering all vertices. Our contributions include developing a unified CFS version for scalable and adaptable MCPP, extending it to MCPP with novel optimization techniques for cost reduction and path continuity and smoothness, and demonstrating through extensive experiments that MCFS outperforms existing MCPP methods in makespan, path curvature, coverage ratio, and overlapping ratio. Our research marks a significant step in MCPP, showcasing the fusion of computer graphics and automated planning principles to advance the capabilities of multi-robot systems in complex environments. Our code is publicly available at https://github.com/reso1/MCFS.

Introduction

In the evolving landscape of multi-robot systems, the efficiency and effectiveness of Multi-Robot Coverage Path Planning (MCPP) (Almadhoun et al. 2019) remain pivotal in a myriad of applications, ranging from environmental monitoring (Collins et al. 2021) to search-and-rescue operations (Song et al. 2022) in complex workspaces. Traditional methodologies, such as cellular decomposition (Latombe and Latombe 1991; Acar et al. 2002) and grid-based methods (Gabriely and Rimon 2001; Hazon and Kaminka 2005), have laid a solid foundation for understanding and navigating the challenges inherent in these tasks. However, as the complexity of environments and the demand for more efficient coverage increase, there is a growing need for innovative strategies that can adeptly handle workspaces rich in irregular obstacles with both high precision and adaptability.

This paper introduces a novel algorithmic framework, called Multi-Robot Connected Fermat Spiral (MCFS), which revolutionizes MCPP by building upon the principles of Connected Fermat Spiral (CFS) (Zhao et al. 2016) from the computer graphics community. This represents the first application of leveraging CFS to solve MCPP challenges in automated planning and robotics, showcasing a unique interdisciplinary fusion. MCFS stands out for its unique ability to coordinate the robots in generating contour-like coverage paths, elegantly adapting to the intricacies of arbitrary-shaped obstacles—a characteristic not typically addressed by traditional methods. Its contouring ability also enhances task efficiency in both time and operation cost (e.g., energy) by balancing the path costs across multiple robots, as indicated by the makespan (Zheng et al. 2010).

Besides task efficiency, a key challenge in MCPP is managing the deceleration and sharp turns required by nonholonomic robots. Traditional methods (Lu et al. 2023; Vandermeulen, Groß, and Kolling 2019), often focused on minimizing path turns, are restricted to rectilinear workspaces and rely on decomposing the area into rectangles. This approach is less effective in arbitrary-shaped environments. On the contrary, the essence of our MCFS framework lies in its global coverage strategy, conceptualizing the paths as a series of interconnected spirals that seamlessly integrate the movements of multiple robots. This strategy results in smooth covering paths without the need for decomposition, inherently accounting for path curvature—a vital factor for efficient robotic navigation.

Drawing inspiration from the original application of CFS in additive manufacturing (Gibson et al. 2021), our MCFS framework innovatively adapts CFS to tackle the MCPP problem, which generates continuous and smooth coverage paths by converting a set of equidistant contour-parallel isolines into connected Fermat spirals. MCFS first constructs a graph of isolines, associating each vertex with an isoline and connecting it to associated vertices of adjacent isolines. It then reduces the MCPP problem to Min-Max Rooted Tree Cover (MMRTC), a combinatorial optimization problem that finds a forest of trees to cover all vertices of the

graph while minimizing the makespan. Our framework is versatile, allowing coverage paths to start from arbitrary starting points as required in MCPP, and optimizes the distribution of the coverage of both multiple whole isolines and segments of an isoline among multiple robots, showcasing an innovative approach to effectively managing the makespan, curvature, and path continuity for each robot.

We conclude **our key contributions** as follows: (1) We propose a unified version of CFS that standardizes the stitching of adjacent isolines, allowing for customized priorities in selecting stitching points and providing scalability and ease of adaptation to MCPP by enabling coverage paths to start from any given initial robot positions. (2) We demonstrate how our MCFS extends this unified version of CFS to MCPP and effectively solves the corresponding MMRTC problem. (3) We introduce two optimization techniques: one that adds edges between non-adjacent but connectable pairs of isolines to expand the solution space and another that refines the MMRTC solution for balanced path costs and reduced overlap in multi-robot coverage. (4) We present extensive experimental results validating the superiority of our MCFS over state-of-the-art MCPP methods in metrics of makespan, path curvature, coverage ratio, and overlapping ratio, showcasing its effectiveness in diverse coverage scenarios.

Related Work

We categorize existing Single-Robot Coverage Path Planning (CPP) and MCPP methods into grid-based, cellular decomposition, and global methods. We refer interested readers to (Tomaszewski 2020) for a more detailed taxonomy.

Grid-Based Methods: Grid-based coverage methods abstract workspaces into square grids (Hazon and Kaminka 2005; Kapoutsis, Chatzichristofis, and Kosmatopoulos 2017; Tang, Sun, and Zhang 2021), allowing for the application of various graph algorithms. One prominent method, Spanning Tree Coverage (STC) (Gabriely and Rimon 2001), constructs a minimum spanning tree and then generates circumnavigating paths on the tree to cover the workspace. STC-based MCPP methods (Hazon and Kaminka 2005; Tang and Ma 2023, 2024a) work by finding a set of trees that jointly visit all vertices and assigning each robot a path that circumnavigates a tree. While convenient, the complexity of optimally solving grid-based MCPP grows exponentially in the workspace size and the number of robots.

Cellular Decomposition Methods: These methods decompose the workspace into sub-regions by detecting geometric critical points, such as trapezoid (Latombe and Latombe 1991) and Morse (Acar et al. 2002) decomposition. CPP methods generate zigzag paths in these subregions for coverage (Choset 2000; Wong and MacDonald 2003), and MCPP methods connect and assign these subregions, filled with zigzag paths, to robots for cooperative coverage (Rekleitis et al. 2008; Mannadiar and Rekleitis 2010; Karapetyan et al. 2017). Additionally, some research optimizes the direction of the zigzag paths for single robots (Oksanen and Visala 2009; Bochkarev and Smith 2016). Although efficient, these methods are less suitable for obstacle-rich or nonrectilinear workspaces due to their reliance on geometric partitioning.

Global Methods: Global CPP methods directly generate paths to cover the workspace without decomposing it. They fall into two types: the first type generates separate paths that contour around obstacles (Yang et al. 2002), and the second type generates a closed path, including Spiral Path (Ren, Sun, and Guo 2009) and CFS that are notable for their continuous and smooth paths. CFS paths are especially convenient as their entry and exit points are adjacent, facilitating the integration of multiple paths. A recent paper has built a CFS path based on an exact geodesic distance field to cover a terrain surface (Wu et al. 2019). However, to our knowledge, there are no global methods yet developed for MCPP.

Connected Fermat Spiral (CFS)

In this section, we present our unified version of CFS, an adaptation of the original CFS concept. The original CFS employs a two-phase process to transform a set of equidistant isolines into a closed path that covers an input polygon workspace. It utilizes a graph structure, where vertices represent individual isolines and edges connect vertices whose respective isolines have adjacent segments. Initially, the original CFS identifies a set of "pockets"—connected components on the spanning tree of the graph. The first phase transforms the isolines within each pocket into a *Fermat spiral* (Lockwood 1967), and the second phase stitches these isolated Fermat spirals to construct the final, connected Fermat spiral by traversing the pockets using the graph edges. For more details of the original CFS, see Appendix A in the full version of this paper (Tang and Ma 2024b).

Our unified version of CFS modifies the graph construction of isolines and consolidates the original two-phase process into a singular, cohesive operation for the CPP problem. The primary modification in our approach lies in the stitching phase. Rather than explicitly identifying pockets and then stitching the resulting isolated Fermat spirals, our method integrates a unified process that simultaneously addresses both the conversion of isolines within a pocket into Fermat spirals and the interconnection of these spirals. This integrated process is applied to every stitchable pair of isolines, effectively merging the conversion and stitching phases. By traversing a rooted spanning tree of the graph, the same connected Fermat spiral as the original CFS is obtained. The advantage of our unified CFS approach is twofold. Firstly, it enhances scalability, facilitating the incorporation of diverse utilities within the framework. Secondly, it simplifies the extension of CFS to MCPP.

Constructing Isolines and the Isograph

We describe our approach for generating layered isolines from a given polygon workspace to be covered and building the isograph. The polygon is enclosed by its boundary, consisting a set of interior boundary polylines that represent obstacles and an exterior boundary polyline.

Generating Layered Isolines: The procedure starts by uniformly sampling a 2D mesh grid of points within the polygon. A distance field is built for these points, representing their shortest distance to the polygon boundary (encompassing both the interior obstacle boundary polylines and the exterior boundary polyline). We denote the distance between

Algorithm 1: Unified Version of CFS

Input: isograph G, entry point $\mathbf{p}_0$

1 $r \leftarrow$ the isovertex of G containing $\mathbf{p}_0$
2 $\pi \leftarrow I_r(\mathbf{p}_0)$, $U \leftarrow \emptyset$
3 **for** $(u, v) \in$ DFS traversal edges of G from r **do**
4 remove any $(\mathbf{p}, \mathbf{q})$ from $O_{u,v}$ where $\mathbf{p} \in U$ or $\mathbf{q} \in U$
5 $(\mathbf{p}, \mathbf{q}) \leftarrow f(O_{u,v})$ ▷ by any stitching tuple selector f
6 stitch $I_v(\mathbf{p})$ into π by stitching $\mathbf{p}$ to $\mathbf{q}$ and $\mathcal{B}(\mathbf{p})$ to $\mathcal{B}(\mathbf{q})$
7 $U \leftarrow U \cup \{\mathbf{p}, \mathbf{q}\}$
8 **return** π

isolines at adjacent layers as l, and the largest distance to the polygon boundary among all points as l_{max}. We then use the *Marching Squares* algorithm (Maple 2003) to generate layered isolines for each layer $i = 1, 2, ..., \lfloor l_{max}/l \rfloor$. This ensures that the distance between each point in the layer-i isoline and the polygon boundary is $l \times i$. The last step resamples equidistant points along each isoline, maintaining a consistent distance of l between adjacent points.

Building the Isograph: We define *isograph* of the layered isolines as an undirected graph $G = (V, E)$, where V is the set of *isovertices*, each associated with a unique isoline. For ease of reference, we let I_v and L_v denote the isoline associated with any $v \in V$ and its respective layer. Similar to the original CFS, we define a *connecting segment set* $O_{u\to v}$ for any pair of isovertices $u, v \in V$ in adjacent layers (i.e., $|L_u - L_v| = 1$) as:

$$O_{u\to v} = \{\mathbf{p} \in I_u \,|\, \forall z \in V, d(\mathbf{p}, I_v) < d(\mathbf{p}, I_z) \wedge L_z = L_v\} \quad (1)$$

where $d(\mathbf{p}, I)$ denotes the distance between point $\mathbf{p}$ and isoline I. Unlike the original CFS which directly constructs an undirected edge (u, v) if $O_{u\to v}$ is nonempty, we also consider $O_{v\to u}$ for edge construction. This consideration provides flexibility in traversing the isograph in any order and from any root isovertex in the CFS context. It also avoids adding edges (u, v) where the respective isolines I_u and I_v are separated by multiple isolines, as such pairs may be unsuitable for stitching in the CPP context (see Fig. 2 for the case study). Therefore, we define a set $O_{u,v}$ of *stitching tuples* for any $u, v \in V$ in adjacent layers as:

$$O_{u,v} = \{(\mathbf{p}, \mathbf{q}) \in O_{u\to v} \times O_{v\to u} \,|\, \mathbf{p} = \mathcal{C}_u(\mathbf{q}) \wedge \mathbf{q} = \mathcal{C}_v(\mathbf{p})\} \quad (2)$$

where $\mathcal{C}_u(\mathbf{p})$ denotes the nearest point along isoline I_u to point $\mathbf{p}$. Subsequently, an undirected edge (u, v) is formed for any $u, v \in V$ in adjacent layers with a nonempty $O_{u,v}$. Each $(\mathbf{p}, \mathbf{q}) \in O_{u,v}$ serves as a candidate stitching tuple to connect isolines I_u and I_v by stitching $\mathbf{p}$ to $\mathbf{q}$ and $\mathcal{B}_u(\mathbf{p})$ to $\mathcal{B}_v(\mathbf{q})$, where $\mathcal{B}_u(\mathbf{p})$ denotes the point preceding $\mathbf{p}$ along isoline I_u in counterclockwise order. Fig. 1 shows how four isolines are connected via the squares as the stitching points.

Although the original CFS assigns a weight of $|O_{u\to v}|$ to each edge to retain a low-curvature path when determining the isograph traversal order for connecting isolated Fermat spirals, we currently leave the edge weight definition application-specific and will explicitly address this objective for every stitching operation in the stitching tuple selector.

Unifying the CFS Algorithm

We detail our unified version of CFS in Alg. 1, which takes as input an isograph G and an entry point $\mathbf{p}_0$. The algorithm

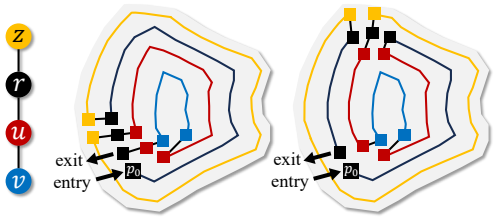

Figure 1: The unified version of CFS on a workspace (grey region). Colored squares represent the stitching tuples. From left to right: The input isograph, the path resulting from the CFS selector, and the path resulting from the MCS selector.

starts by identifying the isovertex r containing $\mathbf{p}_0$ [Line 1] as the root for a depth-first search (DFS) traversal of G. It then initializes the CFS path π to be constructed and the set U to record the points already used to stitch the isolines [Line 2]. The main loop then iterates over the DFS edges [Line 3] and stitches the corresponding pair of isolines for each edge [Lines 5-6]. Specifically, a stitching tuple $(\mathbf{p}, \mathbf{q})$ is selected via any selector [Line 5]. For any isovertex $v \in V$, we use $I_v(\mathbf{p})$ to denote the counterclockwise path along isoline I_v starting at $\mathbf{p}$ and ending at $\mathcal{B}_v(\mathbf{p})$. This path segment is then stitched into π using the selected stitching tuple [Line 6]. The set U is updated to include these newly selected stitching tuples [Line 7]. Following the iterations over all DFS edges, the final path π is constructed to stitch together all isolines and completely cover the given polygon.

Stitching Tuple Selector

We now propose three stitching tuple selectors, each designed to select an appropriate stitching tuple o from a given set $O_{u,v}$ for connecting isolines I_u and I_v. Fig. 1 demonstrates an example of these selectors.

Random Selector: The random selector f_{rnd} randomly selects a stitching tuple from the set $O_{u,v}$.

Connected Fermat Spiral (CFS) Selector: The CFS selector f_{cfs} aligns our unified version of CFS with the original CFS. It attempts to select a stitching tuple from $O_{u,v}$ for $(u, v) \in E$ that is adjacent to the previously selected stitching tuple of $(r, u) \in E$ or $(r, v) \in E$. Either (r, u) or (r, v), with its stitching tuple already selected by f_{cfs}, will be visited before (u, v) in the DFS traversal (Line 3). Assuming that (r, u) is visited first with the selected stitching tuple $(\mathbf{p}', \mathbf{q}') \in O_{r,u}$, f_{cfs} then checks for $o = (\mathbf{p}, \mathbf{q})$ in $O_{u,v}$ where $\mathcal{B}(\mathbf{p}) = \mathbf{q}'$. If such a tuple exists, it is selected for (u, v); otherwise, the first tuple in $O_{u,v}$ is selected.

Minimum Curvature Stitching (MCS) Selector: The MCS selector f_{mcs} iterates through $O_{u,v}$ to identify the stitching tuple $o = (\mathbf{p}, \mathbf{q})$ that minimizes the curvature difference $\Delta\kappa(o)$ before and after stitching, defined as:

$$\Delta\kappa(o) = \sum\nolimits_{\mathbf{p}\in o} [\kappa_\pi(\mathbf{p}) - \kappa_{I_u}(\mathbf{p})] \quad (3)$$

where $\kappa_\pi(\mathbf{p})$ and $\kappa_{I_u}(\mathbf{p})$ denote the curvatures at any point $\mathbf{p}$ on the new stitched path π using o and on the original isoline I_u, respectively. Formally, the MCS selector is defined as $f_{\text{mcs}}(O_{u,v}) = \arg\max_{o\in O_{u,v}} \Delta\kappa(o)$.

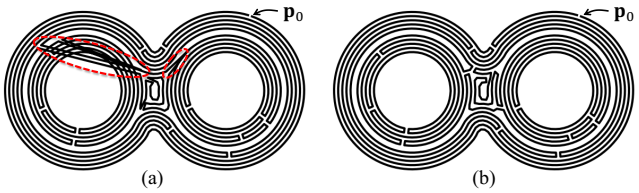

Figure 2: The CFS paths for (a) the original unidirectional $O_{u\to v}$ in Zhao et al. (2016), with the artifacts outlined in red dashed circles, and (b) our bidirectional $O_{u,v}$.

Case Study: Unified vs Original CFS

We discuss the necessity of modification in the construction of the isograph edge set of our unified version of CFS in the CPP context. Unlike the original CFS that uses a unidirectional $O_{u\to v}$ in Eqn. (1) for edge set construction and always starts traversal from the lowest-layer isovertices, our unified CFS defines a more versatile bidirectional $O_{u,v}$ (Eqn. (2)). This modification addresses the requirement in CPP (and MCPP) for starting a coverage path from an arbitrary given point $\mathbf{p}_0$, as accommodated by Alg. 1. Our unified CFS starts the graph traversal from isovertex r, whose respective isoline contains $\mathbf{p}_0$, without the restriction of r being the lowest-layer isovertex. Consequently, valid stitching tuples may not exist for edge construction if only single-directional tuples from layer i to layer $i+1$ are considered as in the original CFS. Moreover, an isovertex u with a local innermost isoline may find a nonempty $O_{u\to v}$ for any isovertex v with $L_v = L_u + 1$, recognizing (u, v) as an edge, which potentially introduces path overlapping. Fig. 2-(a) exemplifies such cases where some local innermost isolines are stitched to the isolines at adjacent layers yet separated by other isolines, a scenario effectively managed in our unified CFS (Fig. 2-(b)) but problematic in using the original CFS definitions.

Multi-Robot CFS Coverage

In this section, we present our MCFS framework for solving MCPP. MCFS computes multiple trees from an input isograph, each corresponding to a different robot, and then applies CFS on each tree to compute individual coverage paths. First, we detail the CFS-based formulation of MCPP and introduce its reduction to Min-Max Rooted Tree Cover (MMRTC) (Even et al. 2004; Tang and Ma 2023). We then present two optimization techniques, isograph augmentation and MMRTC solution refinement, aiming to further enhance the MCPP solution.

Problem Formulation

We present our problem formulation of MCPP that facilitates the extension of CFS. The problem of MCPP is to find a set $\Pi = \{\pi_i\}_{i\in I}$ of coverage paths for a set I of robots that minimizes the makespan (i.e., the maximum path cost). Following the existing literature (Zheng et al. 2010; Tang, Sun, and Zhang 2021), we assume that each robot starts and ends at a given position, corresponding to a pair of adjacent entry and exit points in the CFS context. Formally, the objective of MCPP is minimizing the makespan τ, represented as:

$$\min_{\Pi} \tau = \min_{\Pi=\{\pi_i\}_{i\in I}} \max\{c(\pi_1), c(\pi_2), ..., c(\pi_{|I|})\}. \quad (4)$$

When using CFS to generate each coverage path in Π, the path length is linear in $|\pi|$ and therefore the cost of any path π can be evaluated as $c(\pi) = |\pi|$, since each isoline in CFS contains equidistant points (as detailed in the last section). For an isograph $G = (V, E)$, each $v \in V$ is assigned a weight $w_v = |I_v|$, representing the number or points in isoline I_v. Consequently, the cost of any tree $T \subseteq G$ is $c(T) = \sum_{v\in V(T)} w_v$. The MMRTC problem parallels MCPP in its aim of finding a makespan-minimizing set of rooted trees, where each graph vertex is covered by at least one tree. Given a graph $G = (V, E)$ and a set $R = \{r_i\}_{i\in I} \subseteq V$ of root isovertices for robots, the objective of MMRTC is defined as:

$$\min_{\mathcal{T}=\{T_i\}_{i\in I}} \max\{c(T_1), c(T_2), ..., c(T_{|I|})\} \quad (5)$$

where each $T_i \in \mathcal{T}$ is a tree rooted at r_i and $c(T_i)$ is its tree cost. Let $V(T)$ and $E(T)$ denote the vertex set and edge set of any tree T, respectively. The solution set $\mathcal{T}$ must satisfy $v \in \bigcup_{i\in I} V(T_i)$ to ensure the coverage of all $v \in V$. Since the CFS stitches each isoline I_v of $v \in V(T_i)$ to construct the coverage path $\pi_i \in \Pi$, we have $c(\pi_i) = |\pi_i| = \sum_{v\in V(T_i)} |I_v| = c(T_i)$. Therefore, for any isograph G and the set R of root isovertices for robots, the objective values in Eqn. (4) and Eqn. (5) are identical under CFS, effectively reducing MCPP to MMRTC.

We employ the Mixed Integer Programming (MIP) model proposed in (Tang and Ma 2023) to solve MMRTC optimally. The optimal set of trees obtained is then used to produce coverage paths by applying our unified CFS (Alg. 1) on each tree. Figs. 4-(a) and (b) illustrate a 2-tree MMRTC instance and its corresponding solution. For more details about the MIP model, see Appendix B in the full version of this paper (Tang and Ma 2024b).

Optimization: Isograph Augmentation

Recall that the isograph building process considers each edge only for two isolines in adjacent layers. This process, while efficient, often results in a sparse graph structure in the isograph and thus an undesirable MMRTC solution where certain isovertices are repetitively covered by multiple trees. One common example of such repetition appears for a *cut isovertex*, defined as a vertex whose removal increases the number of connected components in the graph. Such repetitions become more common as the number of trees (robots) increases or when tree roots are clustered, thereby leading to increased makespan and reducing the overall quality of MCPP solutions. To mitigate this issue, we propose to augment the sparse isograph with additional edges connecting isovertices in nonadjacent layers. This augmentation aims to reduce the sparsity of the isograph and allow MMRTC trees to explore new routes for joint coverage, thereby reducing repetitions and balancing tree costs.

The augmentation of an isograph $G = (V, E)$ operates by adding a set $E^{\#}$ of augmented edges, defined as:

$$E^{\#} = \{(u, v) \,|\, \forall u, v \in V, 2 \le d_G(u, v) \le \delta\} \quad (6)$$

where $d_G(\cdot, \cdot)$ denotes the graph distance between any two

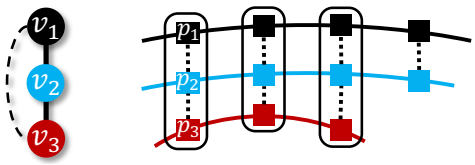

Figure 3: Left: The augmented isograph with original edges (solid lines) and an augmented edge (dashed line). Right: Three sequences of stitching tuples (black boxes) for O_{v_1,v_3}.

isovertices in G, and δ is a hyperparameter that sets the augmentation level. For the edges in $E^{\#}$, stitching tuples are constructed differently from those edges in the original isograph edge set E. Without loss of generality, we consider an edge $(v_1, v_{k+1}) \in E^{\#}$ and its shortest path $(v_1, v_2, ..., v_{k+1})$ in the original G (i.e., each segment (v_i, v_{i+1}) is part of E and k is the graph distance between v_1 and v_{k+1}). The set $O_{v_1,v_{k+1}}$ comprises all pairs of $\mathbf{p}_1$ on the isoline of v_1 and $\mathbf{p}_{k+1}$ on the isoline of v_{k+1} that can be feasibly connected, forming valid stitching tuples $(\mathbf{p}_1, \mathbf{p}_{k+1})$. Such points are connectable iff they form a sequence of consecutive stitching tuples $(\mathbf{p}_1, \mathbf{p}_2) \in O_{v_1,v_2}, \ldots, (\mathbf{p}_k, \mathbf{p}_{k+1}) \in O_{v_k,v_{k+1}}$, which ensures that the straight-line segment between the pair does not intersect more than $k-1$ isoline(s) or any obstacles within the workspace. Fig. 3 demonstrates an example of the adding procedure of an augmented edge (v_1, v_3) with three valid stitching tuples in O_{v_1,v_3} and how p_1 and p_3 can be connected via p_2. Given that the distance between adjacent isolines is set as l previously, we assign a weight $w_e = l \times k$ to each $e = (u, v) \in E^{\#}$ with a layer difference of k (i.e., $|L_u - L_v| = k$), which approximates the additional path cost incurred by any tree containing e. The cost of any tree T is thus updated to $c(T) = \sum_{v \in V(T)} w_v + \sum_{e \in E(T)} w_e$ in the MMRTC solving. Once the augmented edge set $E^{\#}$ and the corresponding stitching tuple sets O are constructed, the original isograph G is updated by setting $E = E \cup E^{\#}$, and the same MMRTC model is solved on the augmented G.

Optimization: MMRTC Solution Refinement

Despite that isograph augmentation reduces isovertex repetitions in the optimal MMRTC solution, two bottlenecks persist in achieving a better MCPP solution. The first bottleneck results from certain isovertex repetitions that remain unresolved by augmentation alone, notably when multiple robots share the same root isovertex or multiple trees use the same vertex. The second bottleneck arises from the limitation of an optimal MMRTC solution in balancing tree costs when the traversing costs of the isolines vary significantly. To tackle the above two bottlenecks, we propose the MMRTC solution refinement process (Alg. 2) that leverages two functions PAIRWISEISOVERTICESSPLITTING (PIS) and ADDIMPROVINGREPETITION (AIR): PIS disperses the coverage of the isoline of an isovertex with repetitions among multiple robots, while AIR introduces *improving repetition* by selectively adding an isovertex from a higher-cost tree to a lower-cost tree. Both PIS and AIR are crucial in refining the MMRTC solution: PIS directly addresses the issue of isovertex repetitions, while AIR strategically adjusts coverage load distribution to balance costs among the trees, enhancing the overall MCPP solution.

Algorithm 2: MMRTC Solution Refinement

Input: isograph $G = (V, E)$, optimal MMRTC solution $\mathcal{T}$
1 optimized solution $\mathcal{T}^* \leftarrow \mathcal{T}$, set of used isovertices $U \leftarrow \emptyset$
2 $M \leftarrow$ set of all repeatedly visited isovertices in $\mathcal{T}$
3 call ADDIMPROVINGREPETITION($\mathcal{T}, M, U$) if $M = \emptyset$
4 max-heapify M ordered by the number of occurrences
5 **while** $M \neq \emptyset$ **do**
6 $u \leftarrow M.pop()$
7 $\mathcal{T}_u \leftarrow$ set of all trees containing u in current solution $\mathcal{T}$
8 **for** $(u, v) \in \{(u, v) \in E \mid v \notin U\}$ **do**
9 $h, \mathcal{T}_u \leftarrow$ PAIRWISEISOVERTICESSPLITTING($\mathcal{T}_u, u, v$)
10 set h^* to h and $\mathcal{T}_u^*$ to $\mathcal{T}_u$ if $h < h^*$
11 use $\mathcal{T}_u^*$ to update $\mathcal{T}$, $U \leftarrow U \cup \{u, v\}$, $M \leftarrow M / \{v\}$
12 set $\mathcal{T}^*$ to $\mathcal{T}$ if its evaluated makespan is smaller
13 call ADDIMPROVINGREPETITION($\mathcal{T}, M, U$) if $M = \emptyset$
14 **return** $\mathcal{T}^*$
15 **Function** ADDIMPROVINGREPETITION($\mathcal{T}, M, U$):
16 $P \leftarrow$ set of leaf isovertices (i.e., with a degree of 1) $u \notin U$ in the highest-cost tree in $\mathcal{T}$ that are not from PIS splitting
17 $T, u \leftarrow$ lowest-cost $T \in \mathcal{T}$ and any $u \in P$ such that u is not in T but a neighbor of some $v \notin U$ in T
18 add u and edge (u, v) to T, $M \leftarrow M \cup \{u\}$
19 **Function** PAIRWISEISOVERTICESSPLITTING($\mathcal{T}_u, u, v$):
20 $h^* \leftarrow +\infty, \mathcal{T}_u^* \leftarrow \mathcal{T}_u$
21 **for** $\mathbf{o} = (o_1, \ldots, o_{|\mathcal{T}_u|}) \in O_{u,v}^{|\mathcal{T}_u|}$ **do**
22 $\mathcal{T}_u' \leftarrow$ a copy of $\mathcal{T}_u$
23 split u, v into $|\mathcal{T}_u|$ new isovertices z's by stitching I_u, I_v via $\mathbf{o}$, each assigned to a $T \in \mathcal{T}_u'$ ▷ see Figs. 4-(b)(c)(d)
24 **for** $T \in \mathcal{T}_u$ **do**
25 **if** $v \in T$ **then**
26 replace each edge $(u, \cdot)$ or $(v, \cdot)$ (except edge (u, v)) with $(z, \cdot)$ in T and remove u, v from T
27 **else**
28 replace each $(u, \cdot)$ with $(z, \cdot)$ in T and remove u from T
29 mark each edge $(z, \cdot)$ as nonadjacent if $O_{z,\cdot} = \emptyset$
30 $h \leftarrow$ sum of the standard deviation of the tree costs in $\mathcal{T}_u'$ and the distance corresponding to any nonadjacent edge
31 set h^* to h and $\mathcal{T}_u^*$ to $\mathcal{T}_u'$ if $h < h^*$
32 **return** $h^*, \mathcal{T}_u^*$

Pseudocode: The MMRTC solution refinement process outlined in Alg. 2 [Lines 1-13] iterates through all isovertices with repetitions in decreasing order of their number of occurrences across different trees. For each such isovertex u, the process aims to find the best way to optimize the set $\mathcal{T}_u$ of trees containing u within the MMRTC solution $\mathcal{T}$. To do so, the process calls PIS to evaluate splitting u with each neighbor v not used for PIS before and updates $\mathcal{T}$ to incorporate the optimized tree set $\mathcal{T}_u^*$ that yields the smallest h-value [Lines 5-11], with a subsequent update to the current best solution $\mathcal{T}^*$ if $\mathcal{T}$ is better [Line 12]. The process also calls AIR to potentially add an improving repetition to an empty M [Lines 3 and 13]. As every iteration records isovertices used for PIS in U [Line 11] and AIR only adds unused isovertices to M [Lines 16-17], Alg. 2 terminates after at most $|V|/2$ iterations since two new isovertices are added to U on Line 11 in each iteration.

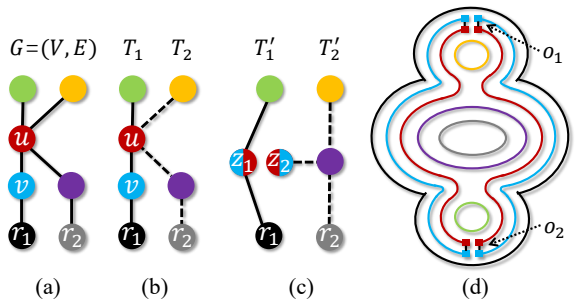

Figure 4: Pairwise isovertices splitting from u, v into a, b at stitching tuples o_1, o_2. (a) Isograph G. (b)(c) Two trees of G (in dashed and solid lines, respectively) before and after the splitting. (d) The layered isolines, each corresponding to the isovertex in the same color, and their post-split segments.

AIR ([Lines 15-18]) identifies one leaf isovertex, unused for PIS before and not resulting from PIS splitting, from the highest-cost tree [Line 16] and adds it as an improved repetition to the lowest-cost neighboring tree [Line 17-18], allowing for redistributing the tree costs.

PIS ([Lines 19-32]) takes as input not only u with repetitions but also its neighbor v [Line 19], essential for forming a closed loop from two isoline segments (as shown in red and blue in Fig. 4-(d)), and splits them into $|\mathcal{T}_u|$ new isovertices. Each new isovertex z corresponds to a closed loop and is then integrated into its designated tree $T \in \mathcal{T}_u$ [Lines 24-29]. To heuristically select the best way of cost-balancing splitting, PIS evaluates each possible mapping $\mathbf{o}$ from the stitching tuples in $O_{u,v}$ to the trees in $\mathcal{T}_u$ (through the $|\mathcal{T}_u|$-th Cartesian power of $O_{u,v}$) by computing the h-value for its resultant tree set $\mathcal{T}'_u$ [Lines 21-30]. This includes: (1) Obtaining the stitching tuple set $O_{z,\cdot}$ for each new edge $(z, \cdot)$ by encompassing all valid stitching tuples on its assigned closed loop. (2) Incorporating the distance between isolines I_z and I_x into the h-value [Line 30] if an edge (z, x) marked as nonadjacent (i.e., $O_{z,x} = \emptyset$) is used in the optimized solution, necessitating an additional shortest path to route between I_z and I_x. Fig. 4 demonstrates how an isovertex u, contained in two trees, split into two new isovertices via PIS.

Case Study: MMRTC Solution Optimizations

We give a concrete example to better illustrate how the two aforementioned optimizations of isograph augmentation (Aug) and solution refinement (Ref) improve an MMRTC solution obtained from the original MIP model. As shown in Fig. 5, we use the instance *char-P* of Fig. 6, where the four trees are rooted in the same isovertex. The original MMRTC solution in the first row demonstrates four isovertices (filled in colors) with repetitions, yielding highly unbalanced costs among trees. With Aug (δ is set to 4) in the second row, the sparsity of the isograph G decreases and thereby provides more routing options starting at the root, making the solution less isovertex repetitions and more cost balanced. With both Aug and Ref in the third row, the solution is further improved by deduplicating all isovertices with repetitions and dynamically adjusting the costs between the trees.

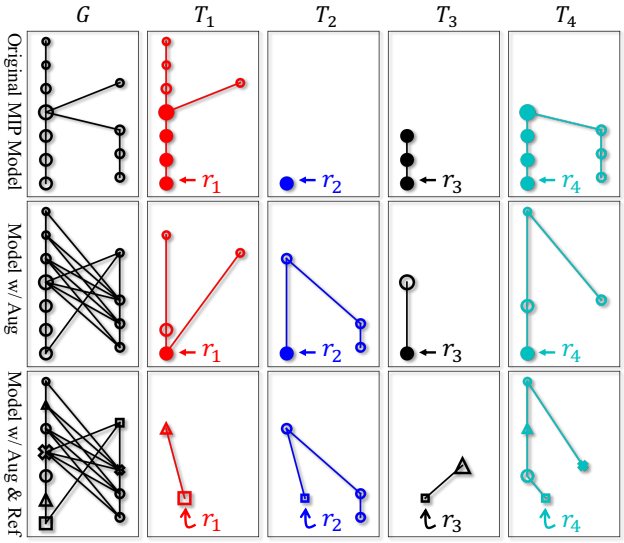

Figure 5: Three MMRTC solutions $\mathcal{T}=\{T_i\}_{i=1}^4$ on isograph G depicted in three rows. The weight of an isovertex corresponds to its marker size. An isovertex filled with color is covered by multiple trees. In the third row, PIS split the isovertices in the same marker (except circles) in G into new isovertices in the same marker and assigned to trees in $\mathcal{T}$.

Selectors	***char-I***	***char-C***	***char-A***	***char-P***	***char-S***	***2-torus***	***office***
random	2.824	0.924	1.228	2.095	1.084	1.070	12.93
CFS	1.306	0.747	0.848	1.724	0.887	0.819	11.77
MCS	1.269	0.782	0.874	1.277	0.960	0.969	8.289

Table 1: Curvature comparison between stitching tuple selectors in the unified version of CFS for single-robot CPP.

Empirical Evaluation

This section presents our experimental results on a 3.49 GHz Apple® M2 CPU laptop with 16GB RAM.

Setup: The MMRTC MIP model for MCFS is solved using the Gurobi solver (Gurobi Optimization, LLC 2023) with a runtime limit of 30 minutes and an MST-based initial solution for warm start-up (Tang and Ma 2023). Whenever MCFS is equipped with isograph augmentation, the hyper-parameter δ is set to $\min\{|I|, 4\}$, where $|I|$ is the number of robots for the MCPP instance, balancing between the MMRTC model complexity and the solution quality.

Instances: As existing MCPP benchmarks like (Tang and Ma 2023) are tailored for grid-based methods on 2D grid maps, we use a more diverse set of workspaces to design MCPP instances displayed in Fig. 6, ranging from fully non-rectilinear (*2-torus*) to mostly rectilinear (*office*) ones. The distance l between adjacent isolines in all instances is 0.1, which is also the cover diameter of the robots. The number of robots ($|I|$) in the instances ranges from 2 to 9. In *char-I* and *char-P*, two robots and four robots share the same root isovertex, respectively. In *2-torus*, three pairs of robots share three root isovertices, respectively. In all other instances, robots start from different root isovertices.

Metrics: In addition to the makespan τ, we report the fol-

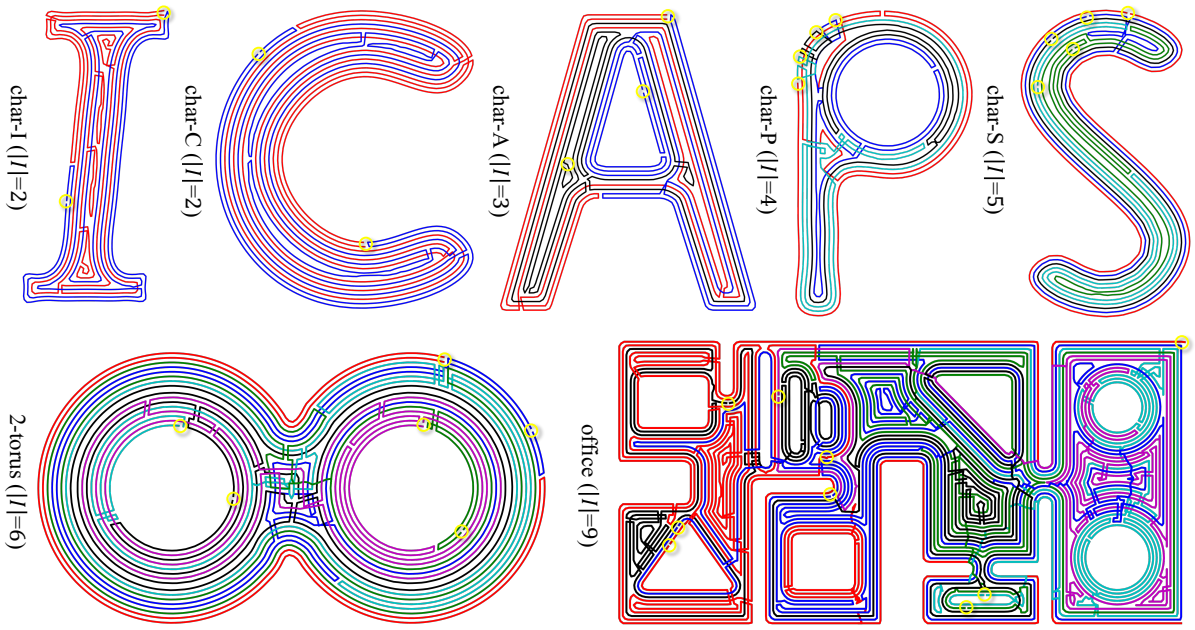

Figure 6: Coverage paths from MCFS. Different paths are in different colors. Yellow circles are root positions.

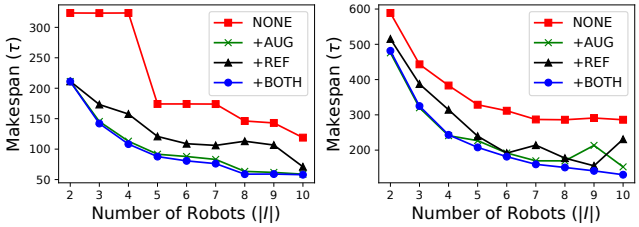

Figure 7: MCFS makespan comparisons on instances *2-torus* (left) and *office* (right) with different number of robots.

lowing metrics to evaluate an MCPP method and its solution: (1) Curvature: Average curvature of all paths (smaller values indicate smoother paths). (2) Coverage: Ratio between the covered area and the total workspace. (3) Overlapping: Ratio between the repeatedly covered area and the total workspace area. (4) Runtime: Total runtime of the method, including the MIP model solving time (when applicable).

Stitching Tuple Selectors: Tab. 1 compares curvature among the random, CFS, and MCS stitching tuple selectors. Both CFS and MCS selectors outperform the random selector, with average reductions of 24.6% and 27.9%, respectively. For less complex workspaces such as *2-torus* that can be filled with smooth isolines, the CFS selector with staircase-like stitching paths outperforms the MCS selector since the MCS selector struggles to distinguish small curvature differences. However, for complex workspaces like *office*, the MCS selector significantly excels by strategically selecting sharp corner points as stitching tuples, thereby substantially reducing the curvature. Based on these findings, the MCS selector will be used in the MCFS framework for the remainder of our experiments.

Ablation Study: To validate the effectiveness of isograph augmentation (Aug) and MMRTC solution refinement (Ref) for MCFS, Tab. 2 reports results for four MCFS variants: using only the original MMRTC solution, with Aug, with Ref, and combining both (labeled **`NONE`**, **`AUG`**, **`REF`**, and **`BOTH`**, respectively). Compared to NONE, REF and AUG reduce the makespan by an average of 29.7% and 36.0%, respectively. For *char-I*, *char-P*, *2-torus*, and *office*, this reduction is attributed to decreased overlapping ratio, particularly where the robot root positions are identical or adjacent. BOTH further enhances this effect in more complex instances for more complex instances like *2-torus* and *office*, doubling the reduction in the overlapping ratio, resulting in a greater makespan reduction. For *char-C*, *char-A*, *char-S* where overlapping ratios of NONE are already low, the makespan reduction of REF results from the iterative cost-balancing procedure, whereas the makespan reduction of AUG results from a larger MMRTC solution space via the augmented edges. Although both REF and AUG require a longer runtime, this increase in runtime is less pronounced for complex instances where the MMRTC MIP model solving dominates. Overall, BOTH yields the largest average makespan reduction of 43.6% compared to NONE, combining the strengths of both REF and AUG in makespan minimization at the cost of slightly longer runtime. Fig. 7 further shows the evolving performance of four MCFS variants in two instances with increasing numbers of robots. It indicates that both optimizations are crucial with more robots as each robot needs to cover fewer isolines, providing a more robust MMRTC solution improvements and thereby the makespan reductions. Aug consistently aids in reducing makespan by expanding the MMRTC solution space, though it increases the complexity and runtime of the resulting MIP model, and Ref effectively redistribute the costs of the imbalanced MM-

	Method	*char-I*	*char-C*	*char-A*	*char-P*	*char-S*	*2-torus*	*office*
	robots	2	2	3	4	5	6	9
Makespan (τ)	TMC	99.94	136.3	87.75	75.19	62.51	133.7	154.0
	TMSTC*	91.33	117.9	84.35	50.63	56.41	113.9	238.1
MCFS	NONE	132.3	179.8	75.4	106.8	50.46	174.2	291.0
MCFS	+REF	**69.74**	125.7	63.44	52.86	50.46	108.9	213.4
MCFS	+AUG	85.37	106.3	63.14	48.23	46.26	87.86	155.5
MCFS	+BOTH	70.75	**105.0**	**63.14**	**35.13**	**36.04**	**80.73**	**141.2**
Curvature	TMC	2.541	3.433	7.482	6.115	5.011	3.341	8.459
	TMSTC*	2.476	1.801	2.655	2.869	2.259	1.335	2.117
MCFS	NONE	1.129	0.776	**0.950**	0.970	1.050	1.299	1.192
MCFS	+REF	2.512	0.842	0.981	1.184	1.050	1.357	1.737
MCFS	+AUG	**0.972**	**0.758**	1.047	**0.828**	**0.787**	1.070	**1.087**
MCFS	+BOTH	1.026	0.795	1.047	1.428	1.068	**1.064**	1.352
Coverage	TMC	86.8%	87.6%	88.4%	88.0%	85.8%	91.5%	89.2%
	TMSTC*	90.6%	92.4%	**91.0%**	**90.2%**	91.2%	93.7%	**91.3%**
MCFS	NONE	91.1%	92.4%	89.5%	89.4%	91.9%	**94.6%**	91.2%
MCFS	+REF	91.1%	92.4%	89.4%	89.4%	91.9%	94.5%	91.1%
MCFS	+AUG	**91.1%**	**92.5%**	89.4%	89.4%	**91.9%**	94.5%	91.1%
MCFS	+BOTH	91.0%	92.4%	89.4%	89.4%	91.8%	94.5%	91.1%
Overlapping	TMC	8.76%	7.76%	**5.59%**	7.89%	18.8%	15.8%	15.3%
	TMSTC*	8.12%	6.25%	9.37%	13.1%	16.5%	15.5%	17.1%
MCFS	NONE	82.6%	5.46%	5.91%	**62.5%**	6.79%	86.6%	50.2%
MCFS	+REF	**6.50%**	**5.44%**	5.92%	7.95%	**6.79%**	25.0%	24.0%
MCFS	+AUG	22.4%	6.41%	6.75%	22.2%	7.48%	20.0%	24.5%
MCFS	+BOTH	7.27%	6.25%	6.63%	10.8%	7.41%	**9.62%**	**13.1%**
Runtime	TMC	0.25s	1.26s	0.97s	0.33s	76.0s	30.4m	31.2m
	TMSTC*	1.21s	1.78s	1.77s	1.02s	2.70s	8.22s	**27.9s**
MCFS	NONE	**0.24s**	**0.38s**	**0.44s**	**0.29s**	**0.31s**	**1.57s**	30.1m
MCFS	+REF	8.59s	11.7s	8.60s	5.08s	0.60s	39.8s	33.1m
MCFS	+AUG	0.34s	0.60s	0.85s	0.46s	0.60s	13.9m	30.2m
MCFS	+BOTH	7.13s	12.5s	20.0s	7.89s	15.6s	15.2m	37.5m

Table 2: Solution quality for different MCPP algorithms.

RTC trees through isovertex splitting. Specifically, for the *2-torus* instance, Aug plays a pivotal role, whereas Ref contributes only marginal improvements; for the *office* instance, either Aug or Ref individually contributes significantly to the solution improvement, while the combined use of both demonstrates a more robust enhancement in the solution.

Comparison: We compare MCFS (+BOTH) with two state-of-the-art grid-based MCPP methods, TMC (Vandermeulen, Groß, and Kolling 2019) and TMSTC* (Lu et al. 2023), that minimize path turns. To adapt TMC and TMSTC* to the non-rectilinear workspaces in our instances, we use overlay grids to approximate the workspaces, followed by shortest pathfinding for robot return to root positions post-coverage. Note that the reported coverage and overlapping ratios for TMC and TMSTC* are approximations due to the workspace approximation and small intersection of their coverage paths with obstacles, whereas the values for MCFS are exact. In Tab. 2, while the average coverage ratios of TMC, TMSTC*, and MCFS are comparably close (with a 3.51% variance), MCFS demonstrates an average makespan reduction of 32.0% and 27.9%, curvature reduction of 75.7% and 47.8%, and overlapping ratio reduction of 13.6% and 20.9% compared to TMC and TMSTC*, respectively. Both MCFS and TMC require longer runtime due to solving MIP models for MMRTC and MTSP, respectively, especially in instances with larger isographs or more robots (e.g., *office*).

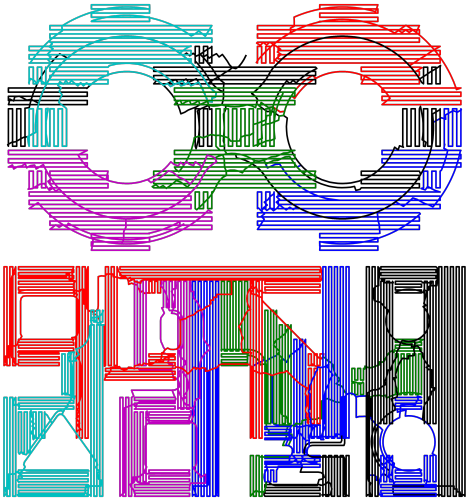

Figure 8: TMC MCPP solutions for *2-torus* and *office*.

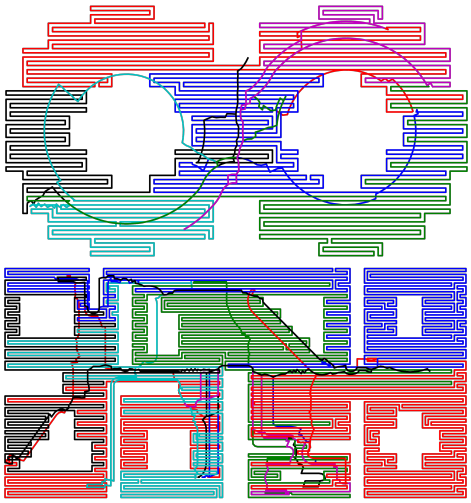

Figure 9: TMSTC* MCPP solutions for *2-torus* and *office*.

Fig. 8 and Fig. 9 visualize the coverage paths via TMC and TMSTC*, respectively. These paths exhibit a back-and-forth boustrophedon pattern, leading to high curvature and imperfect coverage around complex obstacles. In contrast, MCFS notably excels in generating smooth paths that efficiently contour around arbitrarily shaped obstacles, a clear visual advantage over the other methods as shown in Fig. 6.

Conclusions

We proposed the MCFS framework, an innovative approach that blends principles from computer graphics and automated planning to tackle the challenges of covering arbitrarily shaped workspaces in complex MCPP tasks. Future work includes improving isoline quality to further boost the coverage ratio, incorporating kinodynamic constraints into the generation and stitching procedures of the isolines, and developing heuristics to accelerate the PIS function and the MMRTC solving for large numbers of robots or isolines.

Acknowledgements

This work was supported by the NSERC under grant number RGPIN2020-06540 and a CFI JELF award.

References

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

Almadhoun, R.; Taha, T.; Seneviratne, L.; and Zweiri, Y. 2019. A survey on multi-robot coverage path planning for model reconstruction and mapping. *SN Applied Sciences*, 1: 1–24.

Bochkarev, S.; and Smith, S. L. 2016. On minimizing turns in robot coverage path planning. In *CASE*, 1237–1242.

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

Collins, L.; Ghassemi, P.; Esfahani, E. T.; Doermann, D.; Dantu, K.; and Chowdhury, S. 2021. Scalable coverage path planning of multi-robot teams for monitoring non-convex areas. In *ICRA*, 7393–7399.

Even, G.; Garg, N.; Könemann, J.; Ravi, R.; and Sinha, A. 2004. Min–max tree covers of graphs. *Operations Research Letters*, 32(4): 309–315.

Gabriely, Y.; and Rimon, E. 2001. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31: 77–98.

Gibson, I.; Rosen, D. W.; Stucker, B.; Khorasani, M.; Rosen, D.; Stucker, B.; and Khorasani, M. 2021. *Additive manufacturing technologies*, volume 17. Springer.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

Hazon, N.; and Kaminka, G. A. 2005. Redundancy, efficiency and robustness in multi-robot coverage. In *ICRA*, 735–741.

Kapoutsis, A. C.; Chatzichristofis, S. A.; and Kosmatopoulos, E. B. 2017. DARP: divide areas algorithm for optimal multi-robot coverage path planning. *Journal of Intelligent & Robotic Systems*, 86: 663–680.

Karapetyan, N.; Benson, K.; McKinney, C.; Taslakian, P.; and Rekleitis, I. 2017. Efficient multi-robot coverage of a known environment. In *IROS*, 1846–1852.

Latombe, J.-C.; and Latombe, J.-C. 1991. Exact cell decomposition. *Robot Motion Planning*, 200–247.

Lockwood, E. H. 1967. *A book of curves*. Cambridge University Press.

Lu, J.; Zeng, B.; Tang, J.; Lam, T. L.; and Wen, J. 2023. TMSTC*: A Path Planning Algorithm for Minimizing Turns in Multi-robot Coverage. *IEEE Robotics and Automation Letters*, 8(8): 5275–5282.

Mannadiar, R.; and Rekleitis, I. 2010. Optimal coverage of a known arbitrary environment. In *ICRA*, 5525–5530.

Maple, C. 2003. Geometric design and space planning using the marching squares and marching cube algorithms. In *2003 international conference on geometric modeling and graphics, 2003. Proceedings*, 90–95. IEEE.

Oksanen, T.; and Visala, A. 2009. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8): 651–668.

Rekleitis, I.; New, A. P.; Rankin, E. S.; and Choset, H. 2008. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52: 109–142.

Ren, F.; Sun, Y.; and Guo, D. 2009. Combined reparameterization-based spiral toolpath generation for five-axis sculptured surface machining. *International Journal of Advanced Manufacturing Technology*, 40: 760–768.

Song, H.; Yu, J.; Qiu, J.; Sun, Z.; Lang, K.; Luo, Q.; Shen, Y.; and Wang, Y. 2022. Multi-UAV Disaster Environment Coverage Planning with Limited-Endurance. In *ICRA*, 10760–10766.

Tang, J.; and Ma, H. 2023. Mixed Integer Programming for Time-Optimal Multi-Robot Coverage Path Planning with Heuristics. *IEEE Robotics and Automation Letters*, 8(10): 6491–6498.

Tang, J.; and Ma, H. 2024a. Large-Scale Multi-Robot Coverage Path Planning via Local Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17567–17574.

Tang, J.; and Ma, H. 2024b. Multi-Robot Connected Fermat Spiral Coverage. arXiv:2403.13311.

Tang, J.; Sun, C.; and Zhang, X. 2021. MSTC*: Multi-robot Coverage Path Planning under Physical Constrain. In *ICRA*, 2518–2524.

Tomaszewski, C. K. 2020. *Constraint-Based Coverage Path Planning: A Novel Approach to Achieving Energy-Efficient Coverage*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Vandermeulen, I.; Groß, R.; and Kolling, A. 2019. Turn-minimizing multirobot coverage. In *ICRA*, 1014–1020.

Wong, S. C.; and MacDonald, B. A. 2003. A topological coverage algorithm for mobile robots. In *IROS*, 1685–1690.

Wu, C.; Dai, C.; Gong, X.; Liu, Y.-J.; Wang, J.; Gu, X. D.; and Wang, C. C. 2019. Energy-efficient coverage path planning for general terrain surfaces. *IEEE Robotics and Automation Letters*, 4(3): 2584–2591.

Yang, Y.; Loh, H. T.; Fuh, J.; and Wang, Y. 2002. Equidistant path generation for improving scanning efficiency in layered manufacturing. *Rapid Prototyping Journal*, 8(1): 30–37.

Zhao, H.; Gu, F.; Huang, Q.-X.; Garcia, J.; Chen, Y.; Tu, C.; Benes, B.; Zhang, H.; Cohen-Or, D.; and Chen, B. 2016. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics*, 35(4): 1–10.

Zheng, X.; Koenig, S.; Kempe, D.; and Jain, S. 2010. Multirobot forest coverage for weighted and unweighted terrain. *IEEE Transactions on Robotics*, 26(6): 1018–1031.