2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

The Constriction Decomposition Method for Coverage Path Planning

Stanley Brown* and Steven L. Waslander†

Abstract— **The task of coverage path planning in 2D indoor and outdoor environments is classified as a NP-hard problem, and has been an active research topic for over 30 years. We derive a novel, exact cellular decomposition method called the Constriction Decomposition Method (CDM) and apply it to complex indoor environments. The CDM rapidly identifies constriction points in the environment and decomposes the environment into easily traversable cells by exploiting the geometric information contained in the environment's straight skeleton. Several heuristic path planning methods that find paths that completely cover each cell are explored. We apply our method on a complex indoor office-like environment and compare our results to existing methods.**

I. INTRODUCTION

Coverage path planning (CPP) is the task of determining an optimal set of paths that allow a robot or agent to transverse all free space in an environment. CPP has numerous applications for tasks that require the entirety of free space in an environment to be covered, such as crop harvesting, inspection, and mapping. The focus of this paper is the derivation of a CPP algorithm that produces coverage paths tailored to operation in complex human environments with corridors, rooms and islands, by identifying constriction points in known maps at which to divide the environment in a logical manner.

The CPP problem has been an active research topic since at least the 1980s [1] and several main strategies for approaching the problem have emerged. In many cases, the environment is broken down, either explicitly or implicitly, into cells or sub-regions to simplify the generation of coverage paths. If the cells are non-intersecting and the union of all cells fill the entire environment, the decomposition is called an exact cellular decomposition [2]. Examples of exact cellular decomposition include the Boustrophedon decomposition [3], Morse cell decomposition [2], minimum sum of angles (MSA) decomposition [4] or greedy convex polygon decomposition [5]. The Spiral Spanning Tree Coverage (Spiral-STC) method by [6] is an example of an approximate cellular decomposition, which is also simultaneously classified as a grid based method.

CPP algorithms can be classified as either offline or online methods [1]. Online CPP algorithms assume that limited or no prior knowledge of the environment exists and usually rely on a set of heuristic methods to ensure complete coverage [1]. Examples online CPP algorithms can be found in works of [5] and [7].

* M.A.Sc. Candidate, Mechanical and Mechatronics Engineering, University of Waterloo; s52brown@uwaterloo.ca.

† Assistant Professor, Mechanical and Mechatronics Engineering, University of Waterloo; stevenw@uwaterloo.ca

In this paper, we propose an offline method that generates a coverage path plan using a novel, exact cellular decomposition called the constriction decomposition method (CDM). Our method works by decomposing a pre-mapped environment into a set of sells cells based on constriction points, which are defined as areas where the narrowest point is less than the two or more neighbouring areas. For indoor environments, a hallway between two large rooms can be thought of as a constriction point and by decomposing the environment into cells based on these points, an intuitive, room-based decomposition results. It is also shown that be resulting cells can be completely covered using a set of simple contour following paths, followed by a series of back and forth paths that are aligned with the longest edge of the cell.

II. RELATED WORK

One of the earliest exact cellular decomposition methods developed is the trapezoidal decomposition method which breaks polygonal environment down into cells by inserting a set of edges up and down from each obstacle vertex to the bounding polygon [8]. Every cell that is generated in this fashion is trapezoidal, convex and can be covered in a series of back and forth motions that are parallel with the direction of the inserted edges. While the trapezoidal decomposition method is shown to be algorithmically complete, it typically generates a number of cells that could be merged and still be covered using the same coverage pattern. This idea is formalized in the derivation of the Boustrophedon Decomposition [3].

The word boustrophedon comes from ancient Greek and literally means "the way of the ox" [3]. In the Boustrophedon Decomposition, a set of cells are generated assuming that the robot covers a space in a series of back and forth motions similar to the way farmers plow or work a field. By running a scan line that is parallel to the direction of the back and forth motions the robot will take when covering an environment, and inserting an edge whenever an edge can be drawn both above and below the vertex of an obstacle, a set of cells is created. These cells can always be covered using a boustrophedon like coverage pattern and by visiting all the cells the method is shown to be complete [3].

In later work, Choset [2] demonstrated that the Boustrophedon Decomposition can be generalized by proposing a cellular decomposition based on the critical points of Morse functions. The Morse Decomposition can be generalized to any n-dimensional space [2] and also allows for different coverage patterns to be used such as spiral, circular, or diamond patterns. In the case of the Boustrophedon Decomposition, its coverage pattern can be represented by the

straight line. Similar to the Boustrophedon Decomposition, any cells generated by the Morse decomposition can be covered by a robot following the coverage pattern used to create them. In many environments however, it may be useful to select several different functions or directions when creating a cellular decomposition based on some criteria, such as maximizing the average length of the coverage swaths or minimizing the number of turns.

This idea was studied in the work of Oksanen et al. [9] which focused on the coverage of agricultural fields using tractors and other agricultural equipment using two approaches. Their first method is an exact cellular decomposition that aims to minimize a path-based cost function that starts by applying a set of trapezoidal decompositions to an environment using scan lines inclined at $0^o, 30^o, 60^o, 90^o, 120^o, 150^o$. From the resulting decompositions, the three directions that provide the lowest cost are selected, the step size in the direction, is halved and the process is repeated until the improvement per step falls below a threshold, at which point the largest cell is removed from the environment and the process is restarted on the remaining cells. The method is particularly well suited to convex fields and fields that have long straight segments.

The second method explored by Oksanen et al. [9] uses an online recursive algorithm to find a set of paths that also minimizes some cost function. In this algorithm, they do not restrict the search space to include only straight lines, but also allow curved paths. However, allowing any possible curved path results in an infeasibly large search horizon, and so the problem is constrained by restricting the possible curved segments to either follow along the outer contours of the environment or parallel to a previous swath. The paths generated from this algorithm are better able to handle a wider set of polygonal shaped fields that are non-convex, have curved edges or have large holes in them.

In this work, we explore a new cellular decomposition method that is built by splitting the environment into cells based on constriction point in the environment. Our method is focused on indoor environments which tend to have a large number of constriction points due to doorways, hallways between rooms, or aisles. Furthermore, our method does not require any driving direction or function to be asserted during the decomposition step. It is shown that the resulting cells can be covered using a simple set rules to generate coverage paths once the decomposition is completed.

III. PROBLEM FORMULATION

For coverage path planning, we assume a noise-free and accurate map of the operating environment is provided. Let $\xi \subset \mathbb{R}^2$ be a connected, bounded, polygonal environment, and let the boundary of ξ be represented by $B = \delta\xi$. Holes, or unreachable polygonal areas, in the interior of ξ are denoted by the set O and the boundary of all holes is represented by $H = \delta O$. The free space in ξ can be written as $F = \xi \setminus O$. The boundary of the free space, δF, can be represented as a set of vertices, V, and edges, E, where each edge, $e_{ij} \in E$ is defined as a line segment between two vertices $v_i, v_j, i \neq j$. The set of all edges $E = \delta F$ assumes that the environment and all holes are defined by polygons. Additional edges, $E' \subset F$, with vertices on the boundary of the free space can be added to E so as to decompose the free space into n cells, $C = \{C_1, \ldots, C_n\}$. The set of all cells is said to cover the free space, F, if $\bigcup_{i=1}^{n} C_i = F$.

An effector, agent, or robot, with a coverage area, a, is said to cover all of the free space in a cell $C_i \in C$, if it follows a path, $p_i \subset C_i$, for which the union of all coverage areas along the path, p_i, contains all points in C_i, which can be expressed as $\bigcup_P a = C_i$. Note that this definition allows for overlap of coverage areas but requires complete coverage of each cell. A complete path, P, which covers all cells in the free space, F, is formed by the concatenation of cell coverage paths, p_i, and shortest length inter-path segments in the free space, $\rho_{i,i+1} \subset F$, as follows:

$$P = [p_1, \rho_{1,2}, p_2, \ldots, \rho_{n-1,n}, p_n] \tag{1}$$

A complete path which covers the free space in an environment solves the CPP problem and is referred to as a coverage path.

Lastly, we assume that the agent performing the coverage pattern is capable of accurate localization on the order of 5-10 cm. This is currently possible for indoor environments using a variety of SLAM algorithms such as [10], [11] and [12].

IV. THE CONSTRICTION DECOMPOSITION METHOD (CDM)

There are many different approaches to generate a coverage path and most will seek to minimize either the total path length [1] or to produce paths that are feasible for an agent subject to kinematic constraints. We outline a new exact cellular decomposition method that balances the length of the paths with the difficulty of executing under kinematic constraints by creating a set of cells that can be covered using contour following motions followed by boustrophedon motions once the area closest to the boundary has been covered.

A. The Straight Skeleton of a Polygon

For every environment where the free space can be represented as a polygon, F, it is possible extract its topology using a geometric structure known as the straight skeleton $S(F)$. The concept of the straight skeleton was first proposed by Aicholzer et al. [13], a refinement [14] of which is available in the Computational Geometry Algorithms Library (CGAL) [15].

The construction of a straight skeleton, $S(F)$, can be thought of as a shrinking process applied to F, where the boundary of the free space is contracted towards its interior in a self-parallel manner [13]. Analogously, straight skeleton construction can be seen as the propagation of a wavefront from all boundary edges inward at constant speed, ω over a time t. The straight skeleton is then the collapse point graph of the shrinking environment. As the shrinking process proceeds, the boundary vertices, $v_i \in V$ of F can be thought

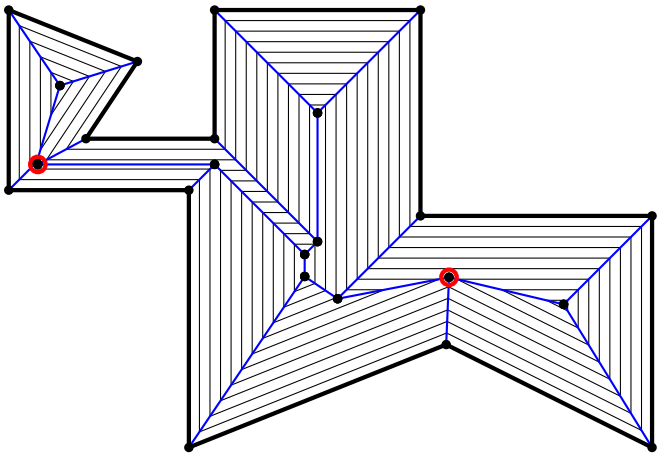

Fig. 1: An example straight skeleton in a non-convex polygon along with offset polygons located at $t = \{1, 2, ..., 10\}$. Skeleton arcs are highlighted in with thin blue lines, skeleton nodes are denoted by black dots and nodes that correspond to split events are circled in red.

to move along the angular bisector of adjacent edges, forming edges in the straight skeleton. This process continues as long as the boundary does not change topologically.

During the shrinking process, there are two possible topological changes that can occur: an edge event and a split event [13]. An edge event occurs when an edge, e_i, shrinks to a length of zero, which causes its neighbouring edges to become adjacent. A split event occurs when a reflex vertex of the wavefront collides with an edge and causes it to split into two new edges, which in turn split the wavefront into two separate polygons. The shrinking process then still continues in all polygons until all edges shrink to a length of zero. The bisector line segments traced out by the vertices of the edges during the entire process are called the skeleton edges, and the bisector arc start and end points are called the nodes of $S(F)$. A node is said to be a contour node if it corresponds to a vertex in V and a skeleton node if it does not.

The shrinking process also gives rise to a hierarchy of nested polygons, a subset of which are shown in Figure 1 for an example polygonal environment along with the skeleton arcs, edge and split nodes. In Figure 1, it can be noted that the location of a split node gives rise to an additional polygon at the corresponding time steps. For the example polygon, the first split occurs at $t = 3.6$ and second split occurs at $t = 5.8$.

The process of shrinking a polygon for some time t at speed, ω, can be related to the task of creating a set of contour following paths. If an agent with a width, w, performing a coverage operation was to follow the contours of the polygonal environment m times, the polygon introduced for the m^{th} pass is the same as the polygon created by the shrinking process detailed above at a time $t = w \cdot m$, and indeed, $\omega = w$ in this case. Therefore, the coverage problem time that each node in $S(F)$ was created corresponds to the number of times an agent can circle the free space of an environment without overlapping previous paths.

The location and time where the split nodes are created are of the greatest interest to our work as they encode the constriction points of an environment. Whenever a split event occurs, the resulting polygon of the wavefront is split into two separate polygons which means that the nodes on either side of the split node are created at a later time. Therefore, if the polygon is decomposed based on the split nodes in its straight skeleton, it is possible to create a set of cells that have no split nodes in their straight skeletons. That is, for such polygons, it will always be possible generate contour following paths that spiral inwards without crossing over a previous path, as described below. We term a decomposition based on this method the Constriction Decomposition Method (CDM).

B. The Constriction Decomposition (CDM)

Let N denote the set of all nodes, n_i, in the straight skeleton of the free space, $S(F)$, such that $n_i \in N$. Each node, n_i, is defined by its location in F, and has an associated time of creation, t_n, node type, $\tau_i = \{\tau^k, \tau^t\}$, (either skeleton or contour), event type, $\nu_i = \{\nu^s, \nu^e\}$, (split or edge) and a set of neighbouring nodes, M_i, connected by an edge in the straight skeleton graph. Given any node in the graph, its neighbours and their corresponding properties can be determined using graph traversal methods.

One method to find a set of decomposition edges E' such that F is decomposed into a set of cells C based on N is to simply search the skeleton of F, $S(F)$, for any split nodes. When a split node, n_{split}, is found, a new vertex, v_{new}, is inserted into the edge $e_{n,m}$ impacted by the angular bisector of v_i such that the length of e_{new} is minimized. In the case where the nearest point on the edge is equal to one of its vertices, e_{new}, is inserted between that vertex and v_i and no additional vertex is added to F. The insertion of the edge e_{new} results in a new cell $c_i \in F$ being created and also removes a split event from the skeleton of the remaining, non-decomposed part of $F' = F \backslash c_i$. This process is repeated until no split nodes exist and F is completely decomposed into a set of cells C. The CDM process is summarized algorithmically below and applied to the polygon in Figure 2.

Let $\theta : N \rightarrow \{\tau_i^k, \tau_i^t\}$ be function that takes a node $n_i \in N$ and returns its type, either contour or skeleton, and let $\psi : N \rightarrow \{\nu_i^s, \nu_i^e\}$ be a function that takes a node and returns its event type. Defining $\eta(v_i, e_{n,m})$ to be a function that takes a vertex, $v_i \in V$, and an edge, $e_{n,m} \in E$, and returns a virtual vertex, $v*$ on the edge, $e_{n,m}$ such that the minimum distance between v_i and v^* is minimized. Then the CDM process can be described as follows in Algorithm 1.

Algorithm 1 Constriction Decomposition Method

Skeleton Nodes: N
Boundary of Free Space: δF
for all $n_i \in N$ **do**
  if $\psi(n_i) = \nu_i^s$ **then**
    for all $n_j \in M_i$ **do**
      if $\theta(n_j) = \tau_i^t$ **then**
        $v* = \eta(v_i, e_{n,m})$
        $e^*_{new} = \{v_j, v_j\}$
        $E = E \bigcup e^*_{new}$
      end if
    end for
  end if
end for

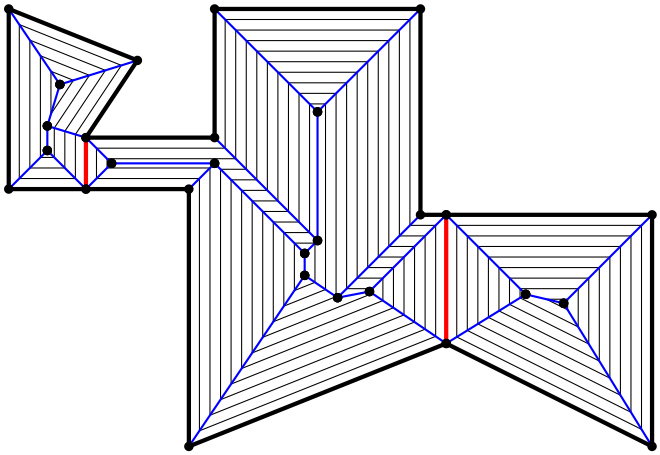

Fig. 2: The set of cells created by Constriction Decomposition Method (CDM) for the environment depicted in Figure 1, along with the corresponding set of cell straight skeletons. None of the cells created contain straight skeletons with nodes corresponding to a split event.

C. Cell Visitation Order

After completing a decomposition of an environment into cells and generating a set of paths that cover each cell, an efficient visitation order of the cells must be determined. To solve this problem, we use a similar approach to [5] by creating an augmented adjacency graph. We start first creating an weighted, undirected adjacency graph $G_a = (V_a, E_a)$, where V_s is a set of vertices located at the middle of each cell and E_s is the connections between each vertex. Every vertex in $v_{s1} \in V_s$ is connected to each other vertex in V_s and the weighting of each edge is assigned a cost based on the minimum travel distance between each cell that is found using a method such as Djkstra's Algorithm. Therefore, neighbouring cells always have a lower travel cost between each other compared to a pair of non neighbouring cells. Applying a depth or breath first search on G_s as done in [3] guarantees that all cells will be visited. However, the order in which they are visited is not guaranteed to be optimal.

The problem of finding the optimal path between all nodes in a graph is known as the Travelling Salesman Problem, which is a well known NP-hard problem. One approach to solving the cell visitation order problem is to transform it into a NP-Complete Hamiltonian Path Problem by introducing metric closure to G_s [5]. Since the graph G_s is not guaranteed to be complete (although it is fully connected), completeness can be guaranteed by adding an edge between each pair of vertices, v_i and v_j, that represents the shortest possible path between these vertices as calculated by using a shortest path algorithm such as A^* or Dijkstra's [5]. Once the additional edges are added to G_s, the problem can be solved using one of many exact or heuristic TSP solvers. An example adjacency graph for the cells created in Figure 2 is shown in Figure 3.

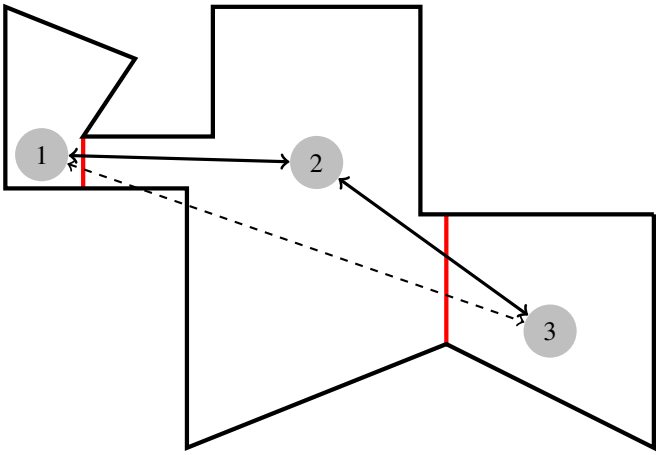

Fig. 3: An example adjacency graph for the cells produced in Figure 2. Black lines indicate direct connections between cells 1 and 2 and 2 and 3. The dashed line represents the additional edge, $e_{1,3}$ inserted into G_s its weight is equal to the sum of the weights of the edges $e_{1,2}$ and $e_{2,3}$.

D. Cell Coverage Paths

The resulting cells from the CDM always have the property that their straight skeletons do not contain any split nodes. This allows for non-convex cells produced by the CDM to be covered though a set of inward spiralling contour following paths. These paths will appear identical to the offset polygons produced as part of the shrinking process discussed in section IV-A and as shown in Figure 2. While this pattern will create a valid set of coverage paths, as the robot reaches the middle of the cell, the distance between successive turns tends to get shorter, leading to inefficient paths. Further, by ending in the middle of the cell, a robot must pass over an already covered area on its way to the next cell.

Another alternative to generating coverage paths is to take advantage of the ease with which convex cells can be covered. If a cell is convex, a series of boustrophon paths can be applied without further decomposing the cell. If a cell is non-convex, the Boustrophedon Decomposition method can be applied, but it may lead to over decomposition depending on the incline of the scan line used to generate the cells [3].

Instead, we propose a coverage method for non-convex cells which first performs contour following starting from the border of the cell and spiralling in until the remaining free space forms a convex shape, at which point a series of boustrophedon paths results in complete coverage. This method has several advantages over the previous two methods for producing coverage paths and refer to this process at the CDM coverage path approach. First, it does not lead to the creation of additional cells, and second, in the case of agents with non-holonomic constraints, it provides a headland to perform the turning operations if required [9]. By covering the last portion of free space in a cell after spiralling inward, the problem of sharp corning near the middle of the cell is also avoided. Furthermore, in many coverage operations, the area near the boundary of the free space is treated differently from the interior, due to the higher risk of a collision with obstacles.

V. Implementation Results

Our algorithm is implemented in C++ and was tested on several office-like environments. The straight skeletons for all examples were generated using the Computational Geometry Algorithms Library (CGAL)'s implementation of Felkel's method [14], [15]. This algorithm is capable of handling non-convex environments with holes with a computational complexity on the order of $O(qr + qlog(r))$ where q is the number of vertices and r is the number of reflex vertices [14]. The CGAL polygon, polygon offsetting and 2D-Arrangements libraries were also used heavily in the generation of the following examples [15].

Our results are compared to the paths created using the Boustrophedon Decomposition due to it being a special case of the Morse Cell decomposition [3] and because the CDM and BSD are both cellular decomposition methods. The trapazoidal decomposition was not used due to the large number of cells created by it, as noted in [3]. Grid based methods such as the Spiral STC were not considered as they do not result in an exact cellular decomposition.

The algorithm was tested on a total of seven environments, three of which are presented in Figures 2, 4, and 5. These environments included one simple, non-convex environments with no holes, three non-convex environments with convex and non-convex holes and three environments created from floor plans similar to Figure 5. A set of convex environments were not used as both the CDM and the Boustrophedon Decomposition yield the same results in such environments. We compared the number of cells generated with each method to the number of cells that would be produced by applying Boustrophedon Decomposition to the same environment using a vertical scan line. These results are summarized in Table I.

When applied to simple polygonal environments, the CDM method appears to, on average, result in either the same number of cells, or less then the Boustrophedon Decomposition method depending on the complexity of the environment. In more complicated, office-like environments, such as in Figure 5, the CDM not only results in a intuitive, room based

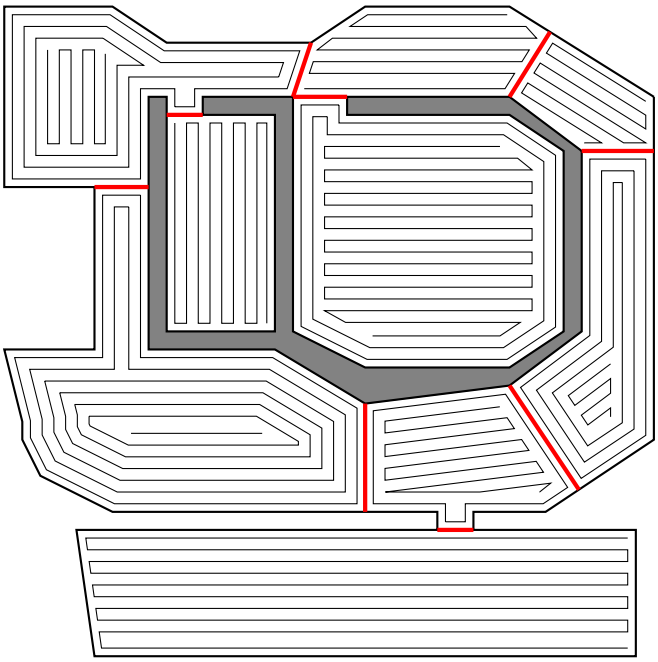

Fig. 4: Coverage paths produced in the test environment by first spiralling inwards in each cell and covering the remaining area using a set of boustrophedon paths once the remaining free space becomes convex.

decomposition as highlighted in Figure 5 where the majority of the cells created in by the CDM method correspond to rooms and hallways, but it appears to create one third of the number of cells compared tot the Boustrophedon Decomposition. Furthermore, all of the cells generated by the CDM can be completely covered using the methods outlined in section IV-D without creating any additional cells.

The run time of the CDM was benchmarked using a laptop computer with an Intel Hawsell I7-4710HQ CPU and 8 GB of RAM. The average run time for the CDM on the non-convex environments, such as Figure 1, in non-convex environments with holes such as Figure 4, and floor plan based environments such as Figure 5 was 0.2, 2.1 and 1750 milliseconds for each respective environment. In all cases the execution time is dominated by the skeleton creation process.

VI. Conclusion

A novel decomposition method is derived, implemented and tested on a set environments that are representative of a complex, indoor environments with curved sections and non-convex obstacles. Our method produces a set of paths that aim to be drivable by robots that have non-holonomic kinematic constraints. We make no claims about the overall optimality of our solution but we demonstrate that our solution guarantees full coverage by using an exact decomposition approach. In office or indoor environments, our solution generates a cellular decomposition that resembles a room-based decomposition.

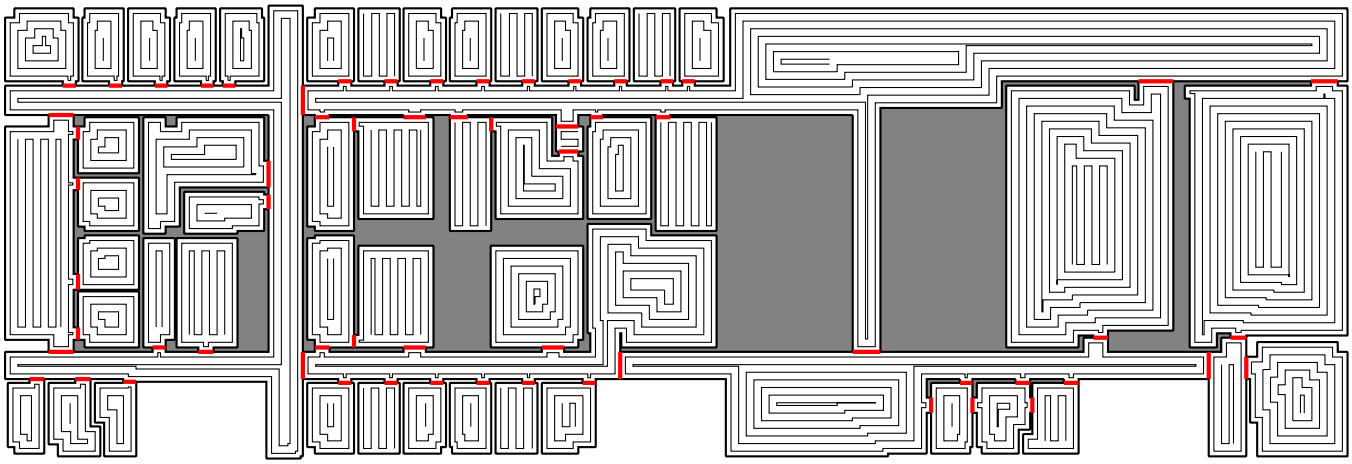

Fig. 5: Application of the CDM to the third floor of the Engineering 5 Building at the University of Waterloo. Red lines indicate additional edges introduced by the algorithm and highlights the resulting room-based decomposition. Coverage paths produced by the methodology outlined in Section IV-D are shown in with thin black lines.

TABLE I: Comparison between the number of cells created for a set of environments using the Boustrophedon Decomposition (BSD) and the Constriction Decomposition Method (CDM) in a set of four hypothetical, polygonal environments and three polygonal environments approximated from the floor plans of several building around the University of Waterloo Campus.

Environment	BSD	CDM	% Improvement
Non-convex (Fig 1)	3	3	0 %
Non-convex, with holes 1 (Fig 4)	14	9	36 %
Non-convex, with holes 2	11	3	73 %
Non-convex, with holes 3	9	5	44 %
Floor Plan 1 (Fig 5)	169	53	68 %
Floor Plan 2	158	65	58 %
Floor Plan 3	172	66	62 %

A. Improvements and Future Work

In future work, we will explore detection of hallways and doorways, and devise strategies to treat these small disruptive features separately. There might be a way to categorize cells by the number of adjacency cells and the number their contours can be transversed without creating overlapping paths. Further, we will explore additional cell coverage methods which follow a subset of wall edges, similar to the work of Oksanen et al. [9]. Such a solution will alleviate the problem of a path ending in the middle of a cell when coverage is completed in a non-convex cell. Furthermore, we intend to test our algorithm using an actual robot operating in an indoor environment similar to the one shown in Figure 5.

REFERENCES

[1] 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.

[2] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, "Exact cellular decompositions in terms of critical points of morse functions," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 2270–2277.

[3] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

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

[5] A. Das, M. Diu, N. Mathew, C. Scharfenberger, J. Servos, A. Wong, J. S. Zelek, D. A. Clausi, and S. L. Waslander, "Mapping, planning, and sample detection strategies for autonomous exploration," *Journal of Field Robotics*, vol. 31, no. 1, pp. 75–106, 2014.

[6] Y. Gabriely and E. Rimon, "Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 954–960.

[7] T.-K. Lee, S. Baek, and S.-Y. Oh, "Sector-based maximal online coverage of unknown environments for cleaning robots with limited sensing," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 698 – 710, 2011.

[8] J.-C. Latombe, *Robot motion planning*. Kluwer Academic Publishers, 1991.

[9] 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.

[10] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1691–1696.

[11] M. J. Tribou, A. Harmat, D. W. Wang, I. Sharf, and S. L. Waslander, "Multi-camera parallel tracking and mapping with non-overlapping fields of view," *The International Journal of Robotics Research*, p. 0278364915571429, 2015.

[12] M. Labbé and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2661–2666.

[13] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Grtner, "A novel type of skeleton for polygons," *Journal of Universal Computer Science*, vol. 1, no. 12, pp. 752–761, 1995.

[14] P. Felkel and S. Obdrzalek, "Straight skeleton implementation," in *Proceedings of spring conference on computer graphics*. Citeseer, 1998.

[15] F. Cacciola, "2d straight skeleton and polygon offsetting," in *CGAL User and Reference Manual*, 4th ed. CGAL Editorial Board, 2015. [Online]. Available: http://doc.cgal.org/4.7/Manual/packages.html