Coverage Path Planning and Room Segmentation in Indoor Environments using the Constriction Decomposition Method

by

Stanley Brown

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2017

Abstract

The task of complete coverage path planning in complex 2D environments is a classic NP-Hard problem that has been an active research topic for well over 30 years. A common approach to solving coverage problem in such environments is to partition, or segment, the target environment into a set of cells that have some property that allows any given cell to be covered in an optimal or near optimal manner. If each cell of an environment is visited and covered by some agent, then the entire environment is said to be covered.

This work proposes a novel segmentation method, called the Constriction Decomposition Method (CDM), that works by locating constriction points in indoor, 2D environment and then partitioning the environment based on the constriction points. When the CDM is applied to 2D maps of office or laboratory environments, the CDM produces a segmentation that closely resembles a room based decomposition. Once the environment has been decomposed into regions, this work demonstrates that each room can be covered using a simple coverage path planning algorithm that exploits the fact that the resulting cells do not contain any constriction points. The lack of constriction points in each region means that each region, or room, can be completely covered using a series of contour following paths followed by a series of back and forth motions. Once a set of coverage paths are produced for each cell, a tour between all path is found using a heuristic Traveling Salesman Problem (TSP) solver.

The proposed segmentation and coverage path planning methods are tested on a set of 15 indoor environments that are derived from a set of floor plans corresponding to five office and seven laboratory environments. The quality of the segmentation produced by the CDM is directly compared to existing methods on a qualitative and quantitative basis using a series of metrics proposed by other authors. The set of coverage paths for each environment are compared to existing work based on the ratio between the total path length and the ratio between the inter-sector path and the total coverage path length. Based on these metrics it is demonstrated that the CDM and the CDM coverage path planner produces both superior segmentations and coverage plans in 2D indoor environments.

Acknowledgements

I would like to thank my supervisor, Dr. Steven Waslander for the wonderful opportunity to work under him and all the guidance and support he has provided over these past two years. I would also like to thank all of the members of the WAVElab, you are all wonderful, hard working people and it will be hard to find such a fun, yet productive and creative environment again. I would also like to thank Laura for all her help in editing this thesis and for generally putting up with me while I wrote it.

Dedication

I would like to dedicate this thesis to my parents who have always supported me no matter what crazy goal I choose to pursue.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Due to the growing availability of low-cost, low-power computation resources, the dream of having a service robot cook, clean and tidy up your home or office is quickly becoming a reality. Combined with incredible improvements in sensor technology, mapping and localization algorithms, it is already possible to build and market commercial scale, autonomous floor cleaning robots. However, this technology only allows the robot to localize itself and follow provided paths. If an entire area must be covered, such as in the task of floor cleaning, an efficient Complete Coverage Path (CCP) is required.

The applications of optimal CCP algorithms extend to ground, aerial and even underwater robotics. Such uses include agricultural applications such as plowing, seeding, fertilizing and crop harvesting, environmental applications such as coral reef inspection, aerial surveys, pipeline inspection, civil applications such as building and bridge inspection and even humanitarian efforts such as mine clearing [18], [27], [22], [21], [29].

Complete Coverage Path Planning (CCPP) is the task of determining an optimal path that allows a robot to cover the entirety of free space in an environment that has a minimal path length. It is most closely related to the Covering Salesman Problem, a variant of the Travelling Salesman Problem [4]. In the Travelling Salesman Problem, an agent is provided with a list of buyers along with their respective distances from one another. The task of the agent is to visit each with the shortest total path. In the Covering Salesman problem [3] each of buyers the salesman wishes to meet are willing to move some distance w. This means, rather then visiting each buyer directly, the salesman instead come within a distance w of all buyers in the network.

In the work of [4], the Covering Salesman Problem is proven to be NP-Hard, as one would expect since the TSP itself is NP-Hard. If one considers a polygonal environment where all buyers are contained within a simple polygon and a circular or square agent with size w, the task of mowing a lawn or cleaning a floor can be directly related to the Covering Salesman Problem. In [3], Arkin describes the lawn mowing problem as the coverage problem where an agent is allowed to move outside of the boundaries, and the pocket milling problem, where the agent is not allowed to cross the boundary. For indoor environments, it is assumed that walls are not passable by the agent, and therefore the task of floor cleaning is analogous to the pocket milling problem which is also proven to be NP-Hard for polygonal environments [3].

Using the same classification as Choset [11], CCP algorithms designed for robotic applications can usually be classified as either complete or heuristic. If a CCP is classified as complete, it can be proven that the algorithm will cover the entirety of the free space, otherwise the algorithm is classified as heuristic. Coverage methods may also be classified as online or off-line depending on whether or not the method requires a known map of the environment. Online methods adapt and can re-plan in real-time, but generally utilize a heuristic method to generate coverage paths. Examples of online methods are found in [13], [26], and [34]. Conversely, offline methods use a known map of the environment to determine a coverage plan. For indoor environments, such as offices, laboratories and warehouses, the assumption of a known map generally holds. Further, for most cases where CCP is applied to floor cleaning in indoor environments, the map can generally be assumed to be static. Exceptions to this assumption include objects such as furniture, shelving or people. For this reason, the focus of this work is on offline methods.

A common approach to a NP-Hard problem such as the CCP is to reduce the problem into a set of smaller problems that can be solved individually and then combined into an approximate solution. In the case of working in a 2D polygonal environment, the environment is partitioned into smaller sub-polygons. The decomposition method tends to fall into one of two categories, exact and inexact decompositions [11]. Exact decompositions typically break the environment down into smaller sub-sections, or cells, that have some kind of topological characteristic that allows for a coverage path to be generated based on the topological characteristics of the resulting cells. Examples of this include the trapezoidal decomposition [18], the Boustrophedon decomposition [10], the Morse Cell decomposition [12], and the Greedy convex polygon decomposition [13].

Inexact decompositions break the environment down into a set of regular sized cells using a square or hexagonal grid [18]. Generally, these cells are at least as small as the robot or agent performing the coverage such that if a robot visits a cell the area represented by the cell is considered covered. Example methods that utilize an inexact decomposition include the Wave-front, Spiral Spanning Tree Coverage (STC) [17] and the work of [26] which demonstrates a grid based method for use on robots with limited sensing range.

The major drawback of inexact and grid based methods is the direct correlation between how well the environment is represented and the resolution of the map. As demonstrated in Figure 1.1, curved surfaces and any portions of the environment a that run off-axis of the grid are poorly represented. There is also an ambiguity as to whether or not a cell is occupied in areas where only a small portion of the cell is occupied by an object. Marking a cell that is only partially occupied will result in an area not being covered, but marking it unoccupied may lead to a collision. As a result of poor approximation and the ambiguity of of occupied cells, some areas of the floor will be uncovered during the execution of a grid based coverage plan. Increasing the resolution of the map helps to reduce this error, but the memory requirement of storing the grid information grows at a rate of n^2 with increasing resolution and any searches performed on the grid will also share a similar increase in run time performance.

While not directly addressed in the CPP literature, a closely related area of research is Room or Map segmentation. Room segmentation can be viewed as an exact decomposition method, as it seeks to break the environment down into rooms. While most exact decomposition methods break the environment down into cells based the desired coverage pattern used to cover the cells [10], [12], [18], [13], room segmentation methods seeks to decompose the environment in the same way that a human would.

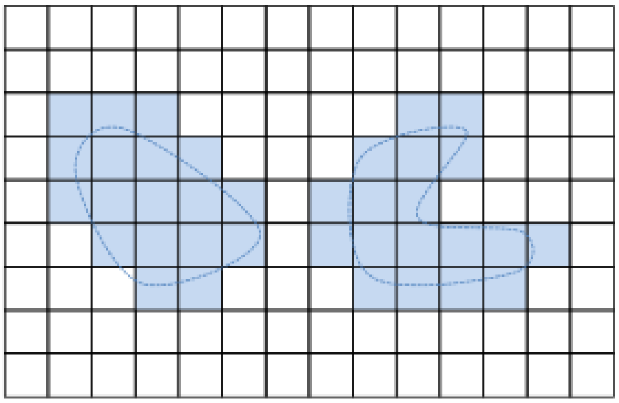


Figure 1.1: An example of a grid based decomposition. The curved, off axis obstacle edges cannot be well represented at the grid resolution, resulting in a under-representation of the free space [18].

The advantage of breaking an environment into rooms rather than just arbitrary polygons is that it not only serves to decompose the environment, but it also simplifies task management and provides an intuitive way for a human to interact with the robot. If a robot enters a room, and finishes its task completely before moving to the next room, a typical user will be able to immediately understand and interpret the robot's behaviour. Further, if one wishes for a room not to be cleaned, removing the room from the list of areas to be cleaned is quite simple from the user side. Additionally, such a method does not require the whole map to be re-segmented, unlike the methods listed in [10], [13].

1.1 Contributions and Approach

In this work, I make several contributions. First I develop and present a novel, exact decomposition method based on constriction points in the environment. I term this decomposition the Constriction Decomposition Method, or CDM. When applied to indoor environments such as offices or commercial spaces, this novel decomposition method partitions the environment into a set of cells that closely resembles a room based decomposition by inserting edges at doorways and other tight space in the environment. Secondly, by creating a segmentation based on the constriction points in the environment, I demonstrate that the resulting cells can always be covered by a series of contour following paths that will have minimal overlap between passes. Using this assumption, I develop a coverage path planning algorithm that produces a series of coverage paths that take advantage of the unique characteristics of the cells produced by the CDM. I guarantee complete coverage of the entire environment by ensuring each cell is covered by my coverage path planning algorithm and by ensuring each cell is visited during a complete coverage operation. The visitation ordering of each cell is found by transforming the cell visitation ordering problem into a Hamiltonian Tour problem and solving it using the Lin-Kernighan heuristic [28] implemented by [20]. Final contribution is evaluation against multiple existing methods in both decomposition and path planning for comparison.

The remainder of this thesis proceeds as follows. Chapter 2 presents background information and an outline of the formulation of the CCP problem for indoor coverage path planning in polygonal environments. Chapter 3 presents the derivation and the development of both the CDM and the CDM coverage path planner. Chapter 4 presents the results of applying the CDM to a set of laboratory based environments and compares both the segmentation results and complete coverage path results to existing work. Lastly, Chapter 5 provides a conclusion and a discussion of further work that could improve the results and robustness of the CDM segmentation and possible improvements to the coverage path planning results.

Chapter 2

Background

2.1 2D Coverage Planning in Planer Environments

In the following work, it is assumed that a noise-free and accurate map of the operating environment is provided. Further, any curves in the environment are assumed to be accurately represented by a series of straight line segments. 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 $ei \in \{v_i, v_j : i \neq j\}$. The set of all edges $E_f = \delta F$ assumes that the environment and all holes are defined by polygons. By inserting additional edges into the environment, $E' \subset F$, with vertices on B, F may be partitioned into a set of m cell, $C = \{c_1, \ldots, c_m\}$ where each cell is defined a region bounded by subset of edges from $E_{c_i} \subset E' \cup' E_f$ such that cell c_i is bounded completely by E_{c_i}.

An effector, agent, or robot, with a coverage width, w, 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 coverage path of each cell, $p_i \in c_i$, and shortest length inter-path segments in the free space, $\rho_{i,i+1} \subset F$, as follows:

$$P = [p_1, \rho_{1,2}, p_2, \ldots, \rho_{m-1,m}, p_m] \tag{2.1}$$

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

Lastly, it is assumed 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 [14], [32] and [24].

2.2 Related Work

2.2.1 Coverage Path Planning

One of the most common approaches to solving the CPP problem in 2D, polygonal environments is to employ a divide and conquer strategy. In complex environments it is often infeasible and computationally intractable to determine an optimal coverage path for an entire environment in one step due to the NP-Hardness of the problem [3]. Rather, many authors employ a decomposition step where they partition the environment into a set of cells, calculate an optimal or near optimal path for each cell and then concatenate them using the approaches described in 2.1. Methods that employ the partitioning approach are referred to as exact cellular decomposition methods in the literature.

One of the earliest exact cellular decomposition methods developed for coverage path planning is the trapezoidal decomposition method. The trapezoidal decomposition method works by partitioning a polygonal environment into cells by inserting a set of edges up and down from each obstacle vertex to the bounding polygon [25]. 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 to the inserted edges. While the trapezoidal decomposition method is shown to be algorithmically complete, it typically generates a number large of cells that can be merged and covered using the same coverage pattern. This idea is formalized in the derivation of the Boustrophedon Decomposition [10].

The word boustrophedon comes from ancient Greek and literally means "the way of the ox" [10]. 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, hereafter referred to as boustrophedon paths. An example of these motions is provided in Figure 2.1. Noting that a cell only needs to be created at the start and end of an obstacle, rather then every vertex, [10] modifies the trapezoidal decomposition to monitor for intersections along the line, opening new cells when a new obstacle is encountered, and closing cells once an obstacle has been passed over by the scan line. Similar to the trapezoidal method, the resulting set of cells can be covered via a series of boustrophedon paths that are aligned with the scan line used to create them and by ensuring all cells are visited and covered, the method is shown to be complete [10].

In later work, Choset [12] 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 [12] 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.

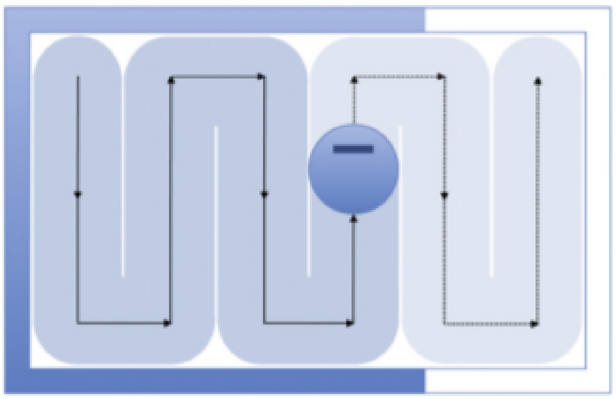


Figure 2.1: A circular robot performing coverage of a rectilinear environment using a boustrophedon coverage pattern [18].

This idea was studied in the work of Oksanen et al. [29] 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 and 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 selected 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 repeated 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. [29] uses an online recursive algorithm to find a set of paths that also minimizes some cost function. In this algorithm, unlike the Boustrophedon and trapezoidal decompositions, both straight lines and arcs are allowed.

To prevent an infeasibly large search space from being considered, the algorithm only considers curved segments that are either parallel to 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.

2.2.2 Room Decomposition

The topic of room segmentation is a closely related to the exact cellular decomposition methods used in coverage path planning. Both room segmentation and exact cellular decomposition algorithms attempt to partition an environment into a set of C cells based on some kind of end use. For coverage path planning, the cells are created in such a way that a near-optimal CCP can be generated for each individual cell. On the other hand, room segmentation algorithms seek a more general solution, where cells are often portioned in such a way that some tasks, such as robot scheduling, warehouse management, navigation, and route planning can be done quickly and efficiently. In most literature the end goal is to segment the environment into rooms, especially when the algorithms are applied to indoor environments.

The majority of literature pertaining to the room segmentation field typically falls into two categories: fully automatic and interactive segmentation. Fully automatic room segmentation methods act directly on a provided map while interactive methods require some degree of human input [7]. In this work, we focus solely on automatic segmentation methods.

One of the earliest examples of applying segmentation algorithms for simplifying indoor path planning is in the work of Thrun and Burken [31], [30]. In [30] an algorithm is derived that breaks an environment, represented as an occupancy grid, down into a set of so-called topological regions based on the location of *critical* nodes in a generalized Voronoi diagram. A critical node in a generalized Voronoi diagram is defined as a node whos neighbours all have a further distance to the boundary of the free space than the critical node. As such, critical nodes indicate contritions in the environment and by inserting an additional edge through this node, between the two closest points on the boundary of the environment, a segmentation is produced, Figure 2.2.

The major drawback of this method is related to the end goal of simplifying indoor path planning. As such, it has a tendency to produce a large number of cells in open areas that may not correspond to rooms leading to the over-segmentation of the environment. For many tasks, over-segmentation is an undesirable trait and in the work of Wurm et al

[33], the authors present a method for improving the detection of critical nodes such that the partitioning edges are more likely to be inserted at doorways. In the implementation presented in [7], a set of merging steps are added to the Voronoi segmentation algorithm that iteratively merges smaller cells into larger ones based on a set of heuristic methods that are tuned by the end user based on the environment and desired results.

In the work of [6], a morphological segmentation is presented. The algorithm operates on grid map where each grid cell is initially labelled as accessible or inaccessible. The original map is then copied into a second map that is iteratively modified by growing inaccessible regions inward using a morphological dilatation operator with a single pixel width. Before applying each dilation, the copied map is checked to see if the number of disconnected regions have increased in the current step in comparison to the previous dilation step. If so, any newly separated regions are given a unique label, copied back to the original map and labelled as inaccessible, if the total size of the new region is above a user-defined threshold. The dilatation process continues until all cells are labelled as inaccessible. Next, the regions now labelled in the original map are grown outwards using a brushfire propagation that proceeds one pixel at a time [6]. This process repeats until all cells are labelled. This process is highlighted in Figure 2.4, with the original map shown in Sub-figure 2.4a, the state of the copied map after a number of dilation steps shown in 2.4b, the labelled regions after completing the dilation are shown in Figure 2.4c and the final labelling after applying the brushfire propagation is shown in Figure 2.4d.

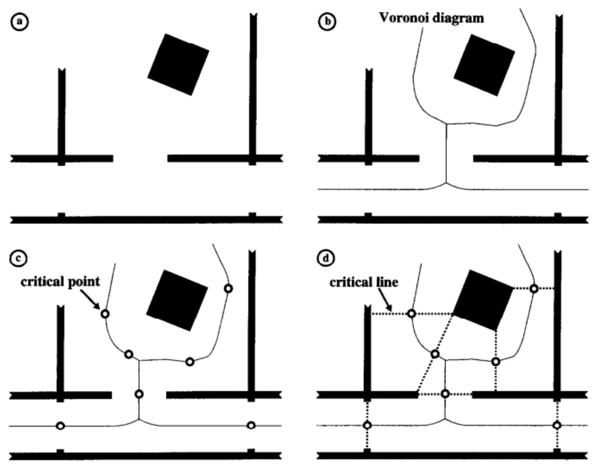
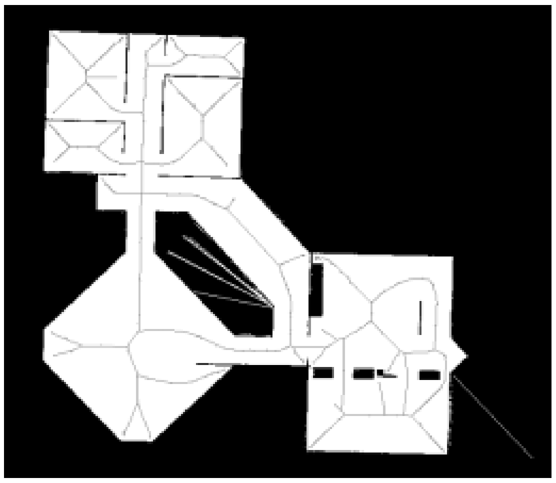
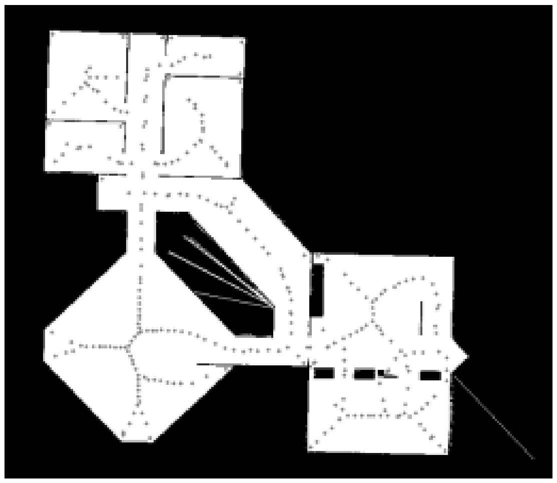
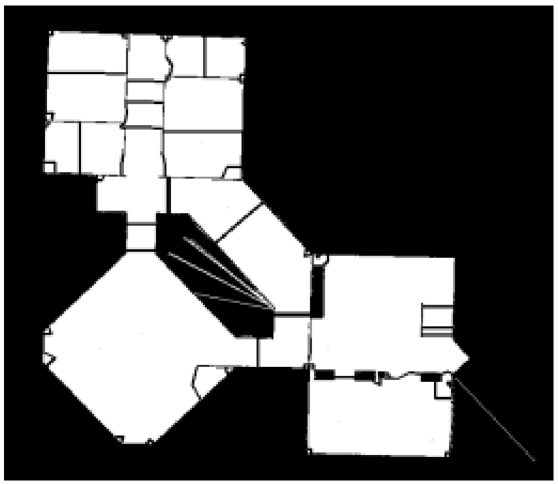


Figure 2.2: Detection of critical points and insertion of critical lines in a Voronoi Diagram. [31]

Advantages of this method, as claimed by the author, include its algorithmic simplicity and its high computational speed [7]. However, for the task of floor cleaning, this algorithm has a tendency to break large cells down into multiple sub-cells leading to over segmentation. Another drawback of this method is the requirement of having to prescribe high and low thresholds for room sizes, as it requires end user tuning depending on the environments it is applied to.


(a) Environment and its generalized Voronoi diagram


(b) Critical Points in the generalized Voronoi diagram


(c) Segmentation after merging several cells

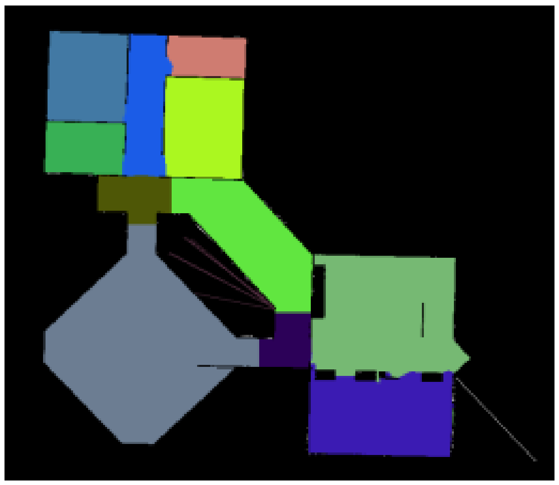
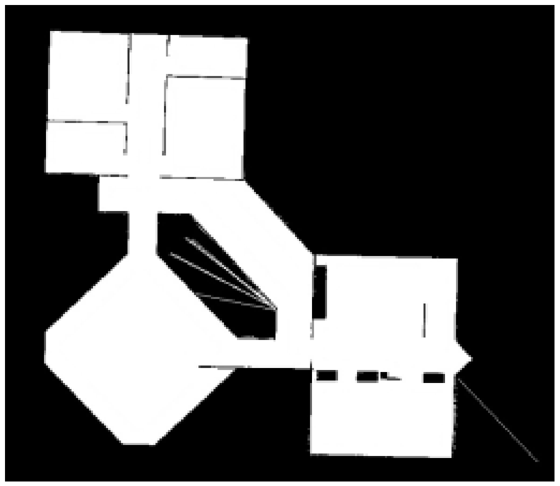
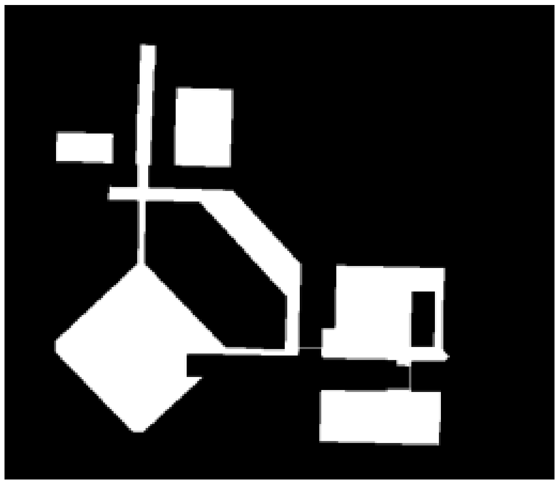

(d) Final Labeling

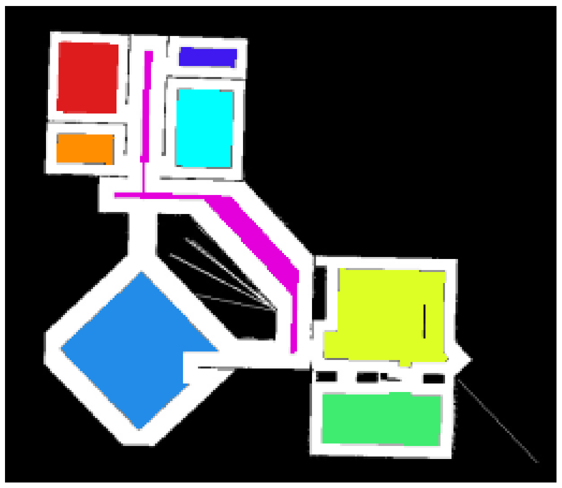
Figure 2.3: Voronoi segmentation example [7]



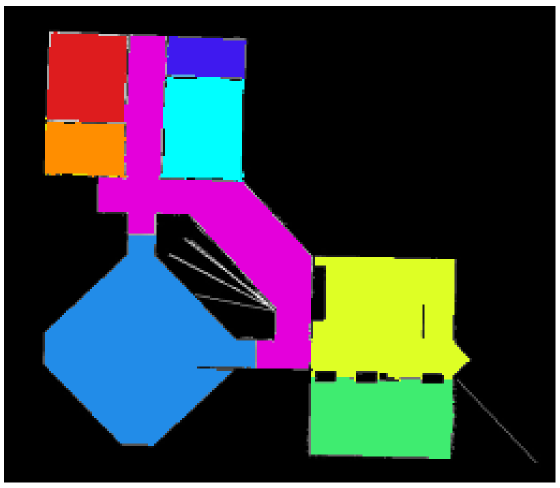
(a) Original Map



(b) Copied Environment after dilatation



(c) Initial Labelling



(d) Final Labeling

Figure 2.4: Morphological Segmentation algorithm. (a) shows the original map, (b) shows the current map after several dilatation steps, (c) is the resulting segmentation map and (d) is the final segmentation after after applying the brushfire propagation step [7]

.

2.3 The Straight Skeleton of a Polygon

For all of the environments explored in this work, the free space of the environment may be represented as a polygon, F. For every such 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. [2], a refinement [16] of which is available in the Computational Geometry Algorithms Library (CGAL) [9].

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 [2]. 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 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 [2], [19]. 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 2.5 for an example polygonal environment along with the skeleton arcs, edge and split nodes. In Figure 2.5, 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 time that each node in $S(F)$ was created corresponds to the number of times an agent can

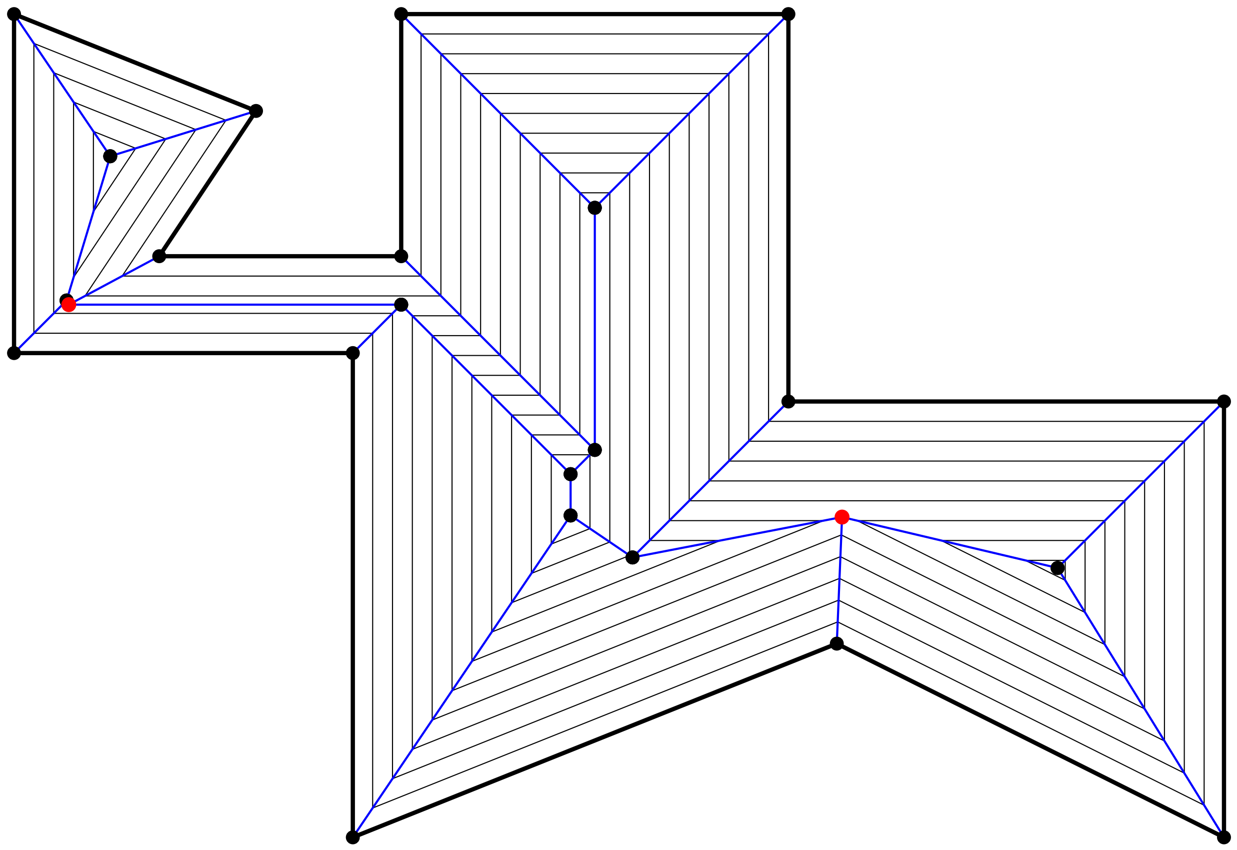


Figure 2.5: An example straight skeleton in a non-convex polygon along with offset polygons located at $t = \{1, 2, ..., 10\}$. Skeleton arcs are highlighted with thin blue lines, skeleton nodes are denoted by black dots and nodes that correspond to split events are circled in red.

circle the free space of an environment without overlapping previous paths.

In this work, the information-rich nodes of $S(F)$ are exploited to determine how to segment the free space F into a set of cells, C. The time of creation of node n is denoted as t_n and letting N denote the set of skeleton nodes $N = n \in S(F)|t_n > 0$. If a node has a creation time of zero, it is referred to as a contour node. Each node $n \in N$ also has an event associated with its creation denoted $\tau \in \nu^e, \nu^s$ where ν^e denotes an edge event and ν^s denotes a split event. Further, given that the skeleton is a graph-like structure, each node also has a set of neighboring nodes $M \subset N$ connected by an edge in $S(F)$. Therefore given any node in $S(F)$, it is possible to query the type, creation time and of its neighbours.

2.3.1 Relation to the Generalized Voronoi Graph

While similar in structure, the generalized Voronoi diagram and the straight skeleton of a simple polygon with holes are not equivalent [19]. The straight skeleton, as discussed above is created by offsetting the line of polygon in a self parallel manner into free space and the nodes and edges of the straight skeleton are located where wave fronts either collide or shrink to zero. Conversely, nodes and edges of a Voronoi diagram are equidistant from all nodes and edges of the original polygon. This also means the Voronoi diagram is made up of curves rather than only straight line segments [19], as shown in Figure 2.6. For the task of room segmentation, as observed in Figures 2.2 and noted in the work of [7] and [33] for even relatively simple environments, the generalized Voronoi diagram can have a large number of critical points that may not be located at doorways, leading to over-segmentation. For a more complete discussion of the mathematical intricacies of the generalized Voronoi diagram and Straight Skeleton, the reader is directed to the work of Held et al. [19].

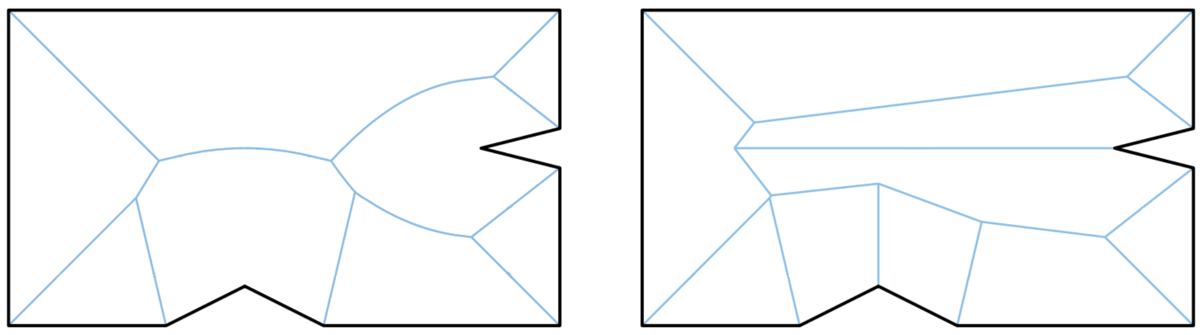


Figure 2.6: The Generalized Voronoi Diagram (left) and Straight Skeleton (right) of an example polygon. All edges in the Voronoi Diagram are equidistant from 2 or more points along the boundary while the straight skeleton, all edges are located along the angular bisectors of the boundary vertices. The Voronoi Diagram is made up of arcs while the Straight Skeleton, as suggested by its name, only contains straight line segments.

Chapter 3

Methodology

By exploiting the information contained within the straight skeleton, a novel decomposition is proposed. The method works by searching the straight skeleton of a simple polygon that represents the entire environment for its split nodes. These split nodes are of the greatest interest to this work as they encode the constriction points of an environment. Whenever a split event occurs, the resulting polygon of the wave front 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).

3.1 The Constriction Decomposition Method (CDM)

As in Section 2.3, let the set of skeleton nodes be represented as N. Each of these nodes have a time of creation t_i and event type $\tau \in \{\tau^s, \tau^e\}$, where s and e denote the node was created in a split or edge event respectively. Each node $n_i \in N$ also has a set of neighboring nodes, M_i, connected by an edge in the straight skeleton graph.

The CDM proceeds by performing an exhaustive walk of all skeleton nodes, $N \in S(F)$, searching for split nodes. Whenever a split node is found, a new edge is inserted between the vertex v_i, $v_i \in V$ that causes the split node to occur and the edge $e_{j,k} \in E$ that is impacted by the angular bisector of v_i, such that the length of the new edge is minimized.

To do this, a new vertex , v_*, is inserted at the closest point between the segment $e_{j,k}$ and vertex v_i, splitting $e_{j,k}$ into two edges, $e_{j,*} \cup e_{*,k} = e_{j,k}$. This also adds v_* to the set of polygon vertices, $V = V \cup v_*$. Next, a new edge, $e_{*,i}$ between v_* and v_i is created, and added to the set of cell boundaries, E'. In the case where the nearest point on the edge is equal to one of its vertices, e, is inserted between the closest vertex, $\{v_j, v_k\} \in e_{j,k}$ and v_i.

The insertion of the edge e^* 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 $F' = F \setminus c_i$. The process of removing and inserting new edges 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 highlighted in Figure 3.1.

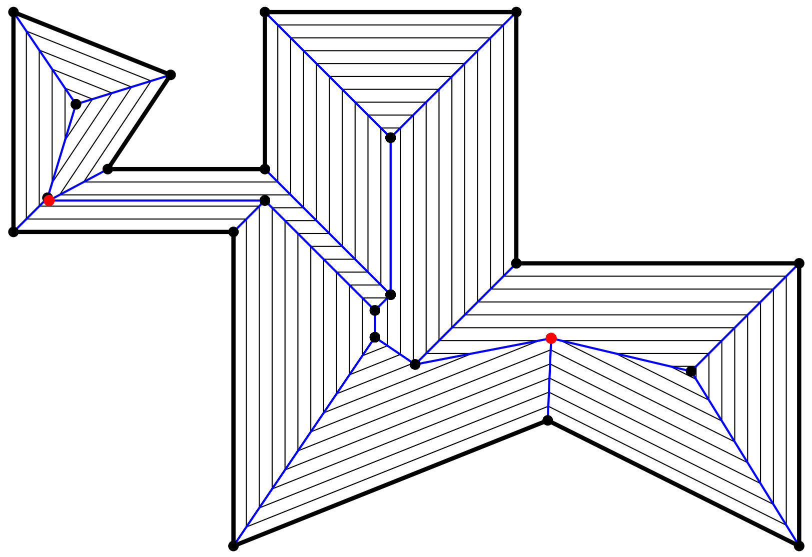
Let $\psi : N \to \tau$ be a function that takes a skeleton node, $n \in S(F)$ and returns its event type, either split or edge. Next, let η be defined as a function that takes a vertex, $v_i \in V$, and an edge, $e_{j,k} \in E$, and returns a virtual vertex, v_* on the edge, $e_{n,m}$ such that the distance between v_i and v_* is minimized, $\eta(e_{j_k}, v_i) \to v^*$. Then the CDM process can be described as follows in Algorithm 1.

Algorithm 1 Constriction Decomposition Method

```
**Skeleton Nodes**: $N$
**Initialize** $E'$ **as**: $E' = E$
**for all** $n_i \in N$ **do**
  **if** $\psi(n_i) = \tau^s$ **then**
    $v_* = \eta(v_i, e_{n,m})$
    $E' = E' \cup e_{*,i}$
  **end if**
**end for**
```

3.1.1 Determining Skeleton Node Type

Depending on the way the algorithm used to generate the straight skeleton is implemented, the skeleton type, τ may not be recorded. This is the case of the CGAL implementation of Felkel's algorithm for generating a straight skeleton [16], [9]. Rather than recording the skeleton type, the CGAL straight skeleton function instead records the "collapse time" of each skeleton node which is equal to the time, t, the node was created during the contour shrinking process detailed in 2.3. By examining the collapse time of each node, the node type can be determined using the following definition of the function $\psi(n_i)$.


(a) Original Environment

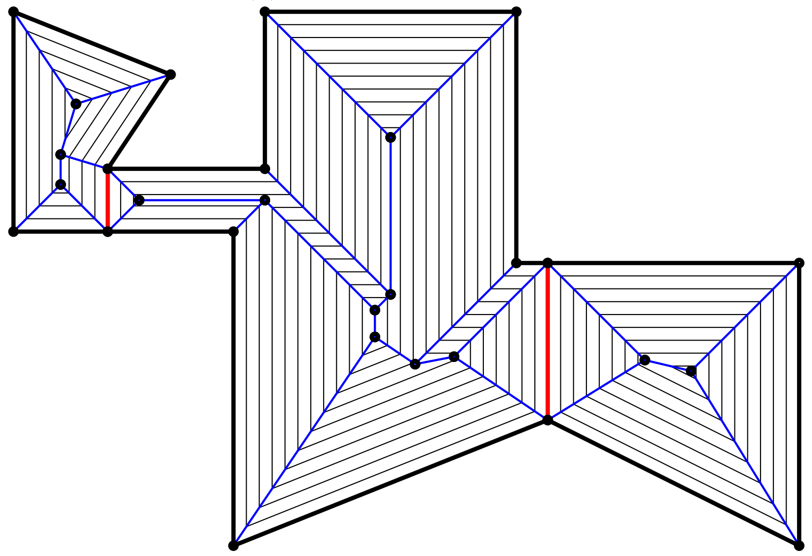
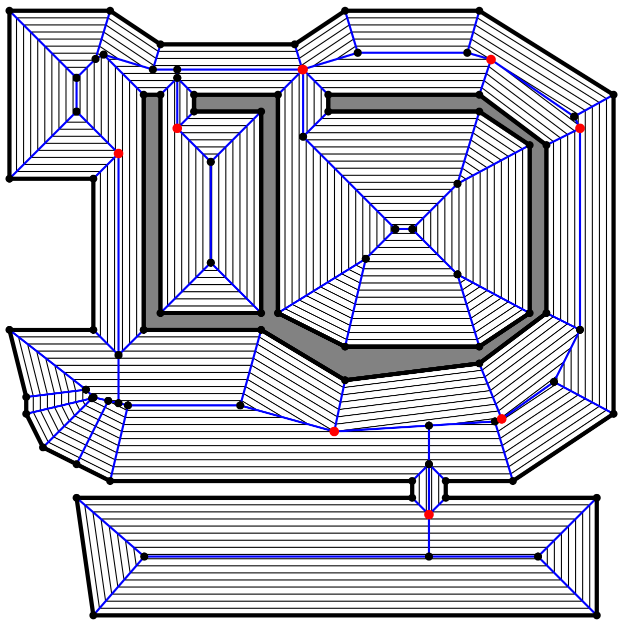

(b) Decomposed by the CDM

Figure 3.1: The set of cells created by Constriction Decomposition Method (CDM) for the environment depicted in Figure 2.5, along with the corresponding set of cell straight skeletons. (a) shows the original environment and its corresponding straight skeleton while (b) shows the resulting decomposition and the corresponding set of straight skeletons. In (a), red circles indicate the split nodes and in (b) red lines indicate the new edges inserted as part of the decomposition. In (b), the resulting set of straight skeletons contain no split nodes as indicated by the corresponding set of contours shown for each cell.


(a) Original

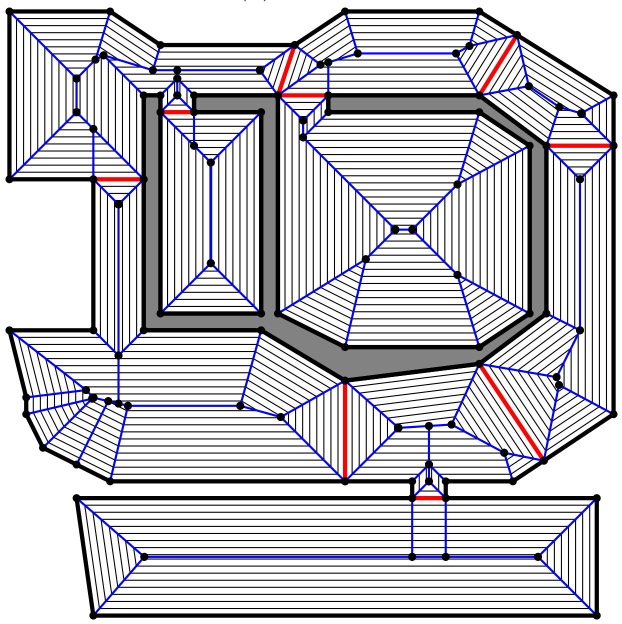

(b) Decomposed by the CDM

Figure 3.2: The set of cells created by the CDM when applied to a polygonal environment with holes. (a) shows the original environment with split nodes highlighted in red while (b) shows the resulting segmentation, with the new cell edges highlighted in red. As in 3.2, the resulting set of cells contain no split nodes in their straight skeletons.

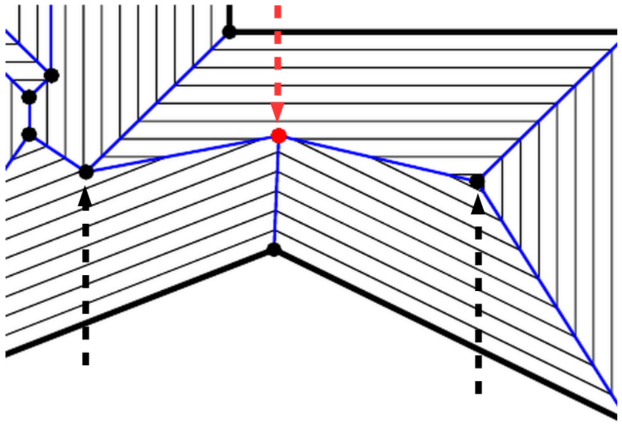


Figure 3.3: A split node, highlighted in red, has a lower collapse time, t, than both of its neighbors, highlighted with black arrows. In this case the split node has a collapse time of $t_n = 5.9$, the left neighbor has a collapse time of $t_{m_1} = 7.54$ and the right neighbor has a collapse time of $t_{m_2} = 8.3$.

As detailed in Section 2.3, split nodes $\psi(n) = \tau^s$ are located at the point where two wavefronts collide during the contour offsetting process and split the wavefront at that time into two separate polygons. As the contouring process continues, the separate polygons continue to shrink until all edges of all polygons shrink to zero, leaving a set of edge nodes, $\psi(n_i) = \tau^k$. This means that any split node must be attached to at least two edge nodes and further, any edge nodes connected to a spit node must both have a higher collapse time than the split node as highlighted in Figure 3.3.

Therefor if a split node exists, it must be connected to at least two edge nodes, and these edge nodes must have a creation time, t, greater than the split node ν_{split}. Therefore all nodes, $n \in N$ can be labeled as a split or edge node by comparing the collapse time of a given node n to all of its neighbors $M_n \subset N$, where M_n denotes the neighbors of node n. If all neighbors $m \in M_n$ of n have a greater collapse time than n, then n must be a split node, $\psi(n) = \tau_s$, otherwise n is an edge node $\psi(n) = \tau_e$. If we let $\omega : n \rightarrow \mathbb{R}$ be defined as a function that takes a skeleton node and returns its collapse time, t, Algorithm 2 can be used to label all nodes $n \in N$ as split or edge.

Algorithm 2 starts by assuming each node, $n \in N$, is a split node, $\psi(n) = \tau_s$, and labels it as such. Next Algorithm 2 compares the collapse time of n to every neighboring node $m \in M_n$. If any of n's neighbors, $m \in M_n$ have a collapse time lower than n, then n is re-labeled as an edge node.

Algorithm 2 Label Split nodes in $S(F)$

```
Skeleton Nodes: $N$
for all $n \in N$ do
    $\psi(n) = \tau_s$
    for all $m \in M_n$ do
        if $\omega(n) \geq \omega(m)$ then
            $\psi(n) = \tau_e$
        end if
    end for
end for
```

3.2 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 spiraling, contour following paths. These paths will appear identical to the offset polygons produced as part of the shrinking process discussed in Section 2.3 and shown in Figure 3.1.

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, which in turn will lead 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.

Instead, we propose a coverage method for non-convex cells which first performs contour following starting from the border of the cell and spiraling in until the remaining free space forms a convex shape, at which point a series of boustrophedon paths results in complete coverage. This method is referred to as CDM coverage pattern, and has several advantages over boustrophedon paths and spiraling coverage paths. Firstly, it does not lead to the creation of additional cells as boustrophedon paths do, and secondly, in the case of agents with non-holonomic constraints, it provides a headland to perform the turning operations if required [29]. By covering the last portion of free space in a cell after spiraling inward, the problem of sharp cornering near the middle of the cell is also avoided. Furthermore, for many robotic applications the boundary of the free space is often treated with additional care due to the greater risk of collision in the case of poor localization. An example of paths produced by the CDM coverage planner when applied to a test environment is shown in Figure 3.4.

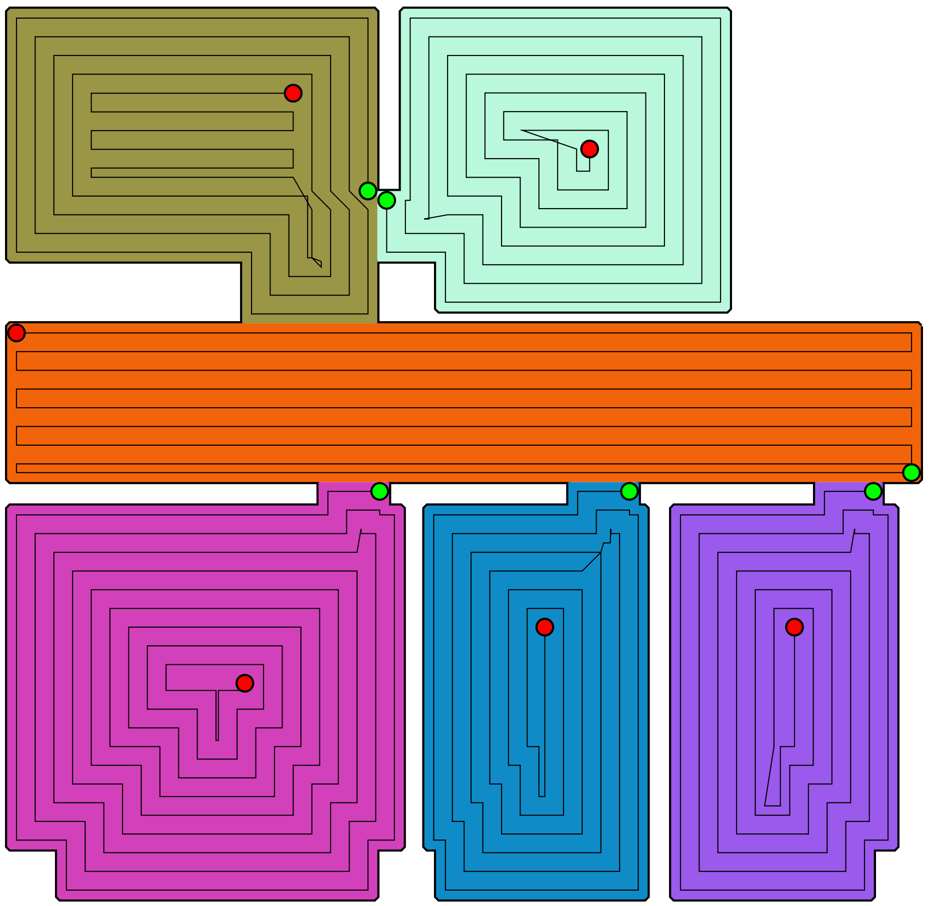


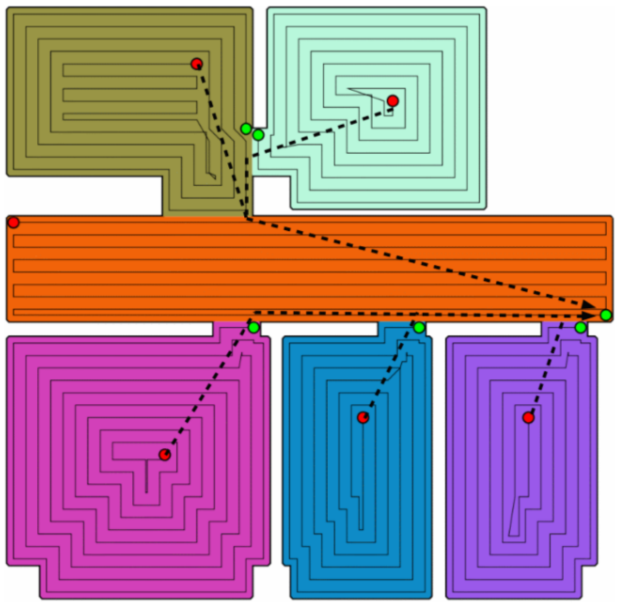
Figure 3.4: Coverage paths produced in an example environment using the CDM coverage path algorithm. The green and red circle indicate the start and end of the paths respectively, while the background colors indicate the cells produced by the CDM.

3.3 Cell Visitation Order

The last step of any cellular decomposition method is to connect the coverage paths in such as way that the inter-cellular travel is minimized. One issue that arises when attempting to find the optimal cell visitation order, is that the location of where the agent performs coverage in a cell produced by the CDM can have a dramatic effect on both the coverage path length and the end point. In this work it was found that for the environments explored in this work, the inter-cellular travel is usually less than 15 percent of the coverage path length. Therefore to reduce the complexity of the cell-ordering problem, it is assumed that all coverage paths start at the entrance to the cell that corresponds to the shortest coverage path for that cell.

Once a set of coverage paths has been created, one for each cell, the optimal visitation order must be determined. By creating a weighted, undirected graph between the end point of each coverage path in each cell and the start points of all other coverage paths as shown in 3.5, the cell visitation problem can be treated as the graph based Traveling Salesman Problem and solved using a number of TSP approximation algorithms that are readily available.

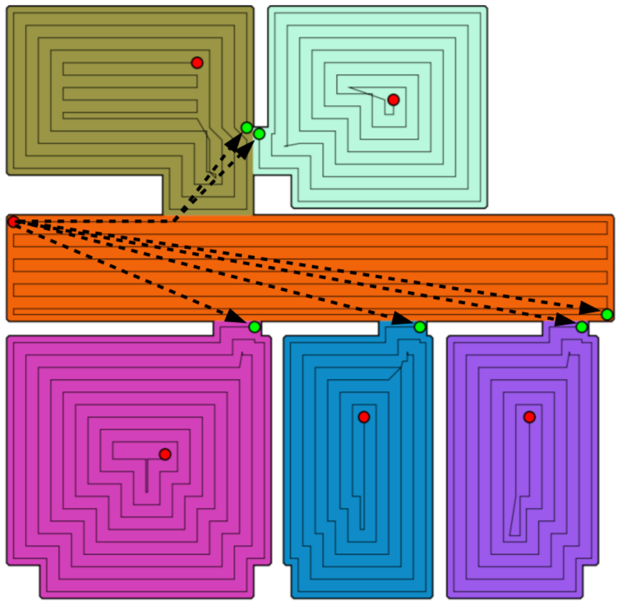
In this work, an adjacency matrix is used to represent an adjacency graph between each cell, where the weights listed in the graph correspond to the distance between two points which are found using Dijkstra's Algorithm and a probabilistic road map (PRM) [25]. The optimal visitation order is found using the LinKernighan heuristic (LKH) [28] TSP approximation algorithm implemented by [20]. This implementation directly accepts adjacency matrices as a problem input. The output from the LKH TSP solver is then used to directly connect the start and end points of all paths and produce a complete coverage tour as shown in Figure 3.6.



(a)

		To					
		1	**2**	**3**	**4**	**5**	**6**
From	**1**	-	9.5	12.1	9.1	10.0	9.8
	2	8.9	-	12.7	5.5	8.9	7.4
	3	9.4	11.8	-	11.4	7.0	12.4
	4	6.7	3.7	10.7	-	8.6	3.5
	5	7.2	9.5	8.6	9.2	-	3.7
	6	7.8	7.5	12.1	6.7	9.7	-

(b)



(c)

		To					
		1	**2**	**3**	**4**	**5**	**6**
From	**1**	-	9.5	12.1	9.1	10.0	9.8
	2	8.9	-	12.7	5.5	8.9	7.4
	3	9.4	11.8	-	11.4	7.0	12.4
	4	6.7	3.7	10.7	-	8.6	3.5
	5	7.2	9.5	8.6	9.2	-	3.7
	6	7.8	7.5	12.1	6.7	9.7	-

(d)

Figure 3.5: Adjacency graph and matrix for the the inter-cellular distances. (a) highlights the inter-cellular paths between the end points of all coverage paths from cells 1-5 going to the start point of cell 6's coverage. (b) shows the corresponding row and distances for (a). (c) highlights the paths between the end point of cell 6's coverage path to the start points of coverage paths 1-5 along with the corresponding distances highlighted in row 6 of the adjacency matrix as shown in (d).

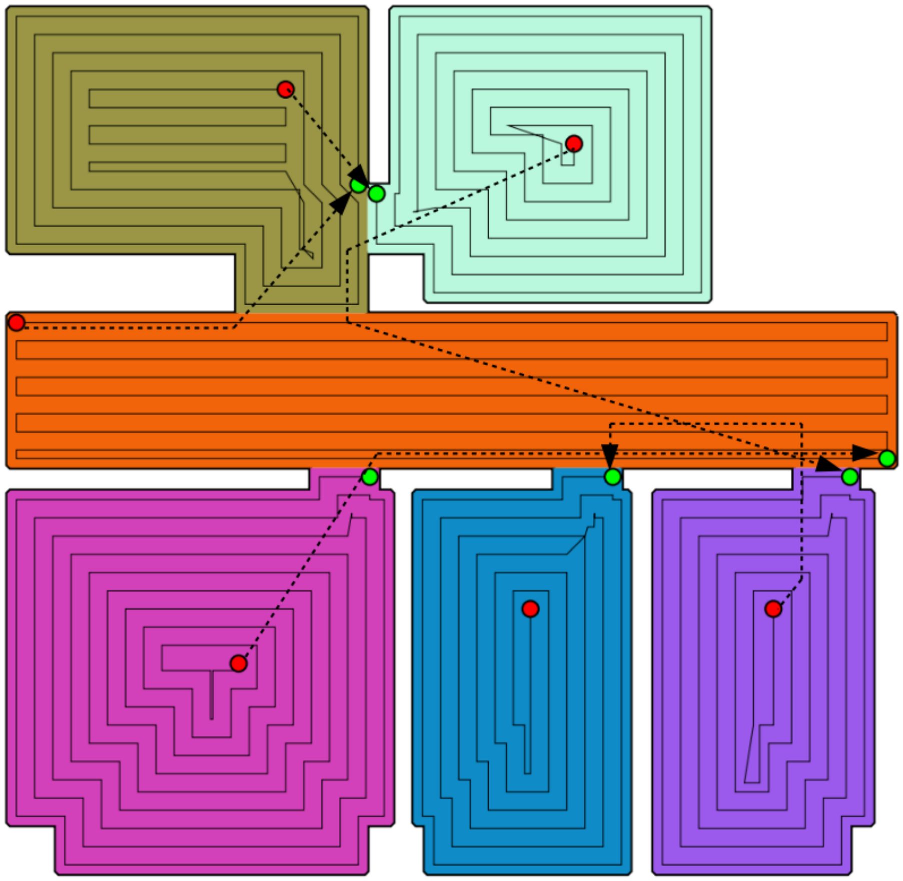


Figure 3.6: A complete coverage tour using the output from the LKH solver and the adjacent matrix input highlighted in 3.5. Solid lines indicate coverage paths where the robot is carrying out a task such as cleaning and the dashed lines indicate inter-cellular travel where the robot is assumed to be doing no work. As before, green and red circles indicate the start and end points of a coverage path respectively. In this example, the robot starts at the cell in the bottom left and finishes in the cell located at the bottom middle of the figure.

Chapter 4

Experimental Results

The CDM segmentation algorithm is implemented in C++ and heavily relies on the Computational Geometry Algorithms Library (CGAL). The straight skeletons for all examples were generated using CGAL's implementation of Felkel's method [16, 9]. 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 [16]. The environments themselves are represented using the 2D-Arrangements library and any operations, such as edge insertion, are performed using methods from the 2D-Arrangements library. The path generation algorithm described in Section 3.2 relies on the polygon offsetting library to produce spiraling paths.

The segmentation results of the CDM algorithm are compared to the morphological and Voronoi segmentation algorithms described in [7] along with the Boustrophedon algorithm [10]. The Boustrophedon decomposition algorithm, BD, also uses CGAL to perform mathematical operations on 2-D polygons. The morphological and Voronoi segmentation algorithms are implemented in a software package provided by [7] and run on the same environments the CDM is tested on. The package provided by [7] also calculates a set of quantitative metrics to evaluate and compare the numerical properties of the resulting segmentations.

The CDM coverage path planning algorithm outlined in Section 3.2 is implemented and applied to the room segmentation results produced by the CDM. The room segmentation results are only compared to the paths generated using the BD and path generation function outlined in [10]. Grid based methods such as the Spiral STC were not considered as they do not result in an exact cellular decomposition and do not operate in the polygonal environments studied in this work.

4.1 Room Segmentation Results

The CDM algorithm is first tested on a set of simple, ideal, polygonal environments shown in Figure 4.1. These test environments include a convex, a non-convex and a non-convex environment with a non-convex hole. In all three test environments in Figure 4.1, the CDM correctly identifies the split nodes in the environment and produces a segmentation. When the CDM is applied to a concave polygon, as shown in Figure 4.1c, the segmentation is simply equal to the original polygon because the straight skeleton of any convex environment will not contain any split nodes due to the lack of reflex vertices [2] [16].

Next the CDM was tested on a set of environments with increasing complexity that are selected from a set of environments provided by the work [7]. The test set of environments are provided as a set of JPEG images with varying quality. Some of the images appear to be produced from floor plans of various offices, while others are produced from the results of various 2D laser-SLAM algorithms. The varying image quality meant that only a subset of the maps were usable in this work. From the 20 environments provided by [7], only 10 were selected, 5 of the office environments and 5 of the laboratory environments. Since the environments are provided as a set of images, they are treated as occupancy grids as is done in [7]. To apply the CDM to these environments the occupancy grid maps are transformed to a polygonal representation of the environment. This process is detailed below.

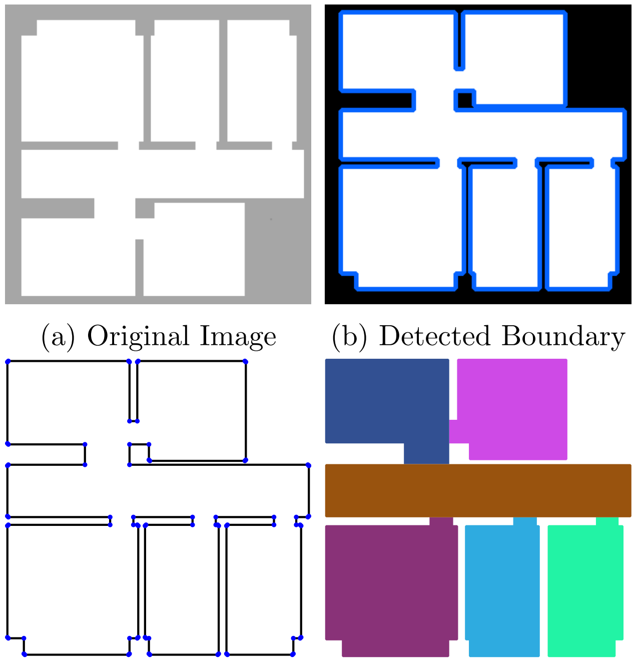
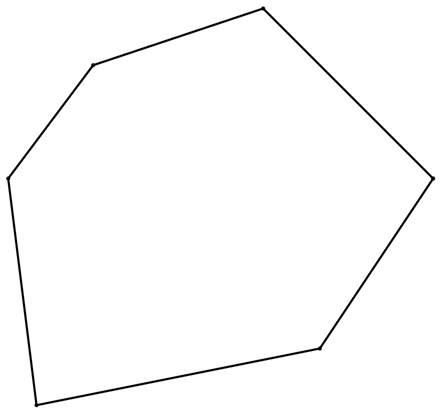
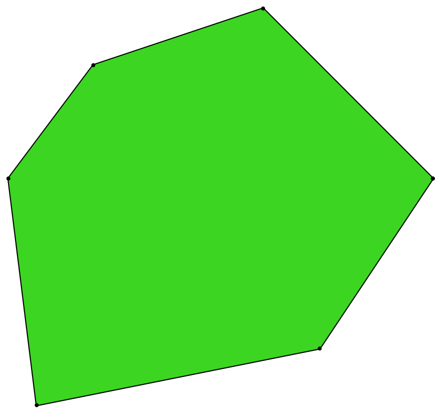
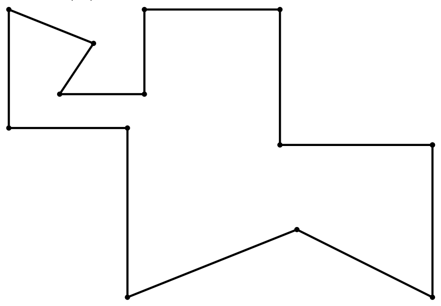

(a) Original Image (b) Detected Boundary
(c) Extracted Contours (d) CDM Segmentation

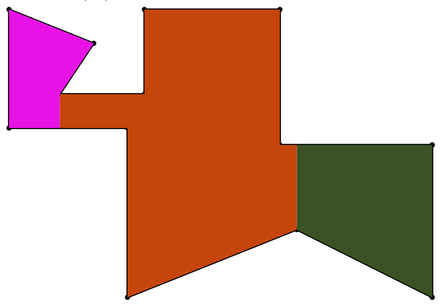
Figure 4.2: Converting a grid map (a) to a polygonal representation for the CDM shown in (c). (b) shows the contours detected using the contour function in OpenCV and (d) highlights the resulting segmentation using the CDM on (c).

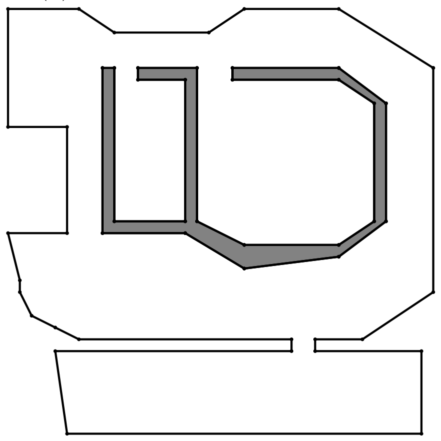
The Open Computer Vision Library (OpenCV) is an open source package that contains a multitude of useful functions for extracting information from images. In this case, the maps provided by [7] are converted to a set of binary images using the thresholding function in OpenCV and contours are extracted using the OpenCV contouring function using the simple approximation setting.


(a) Convex Environment


(b) Convex Segmented


(c) Non-convex Enviornment


(d) Non-convex segmentated


(e) Non-convex with a hole

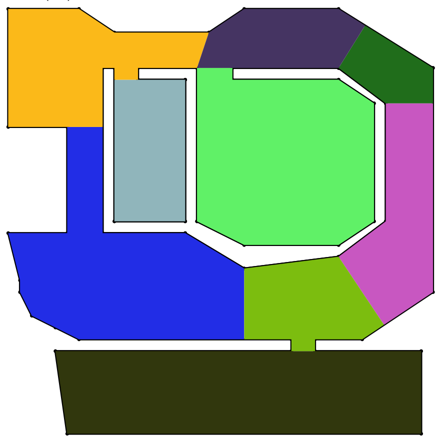
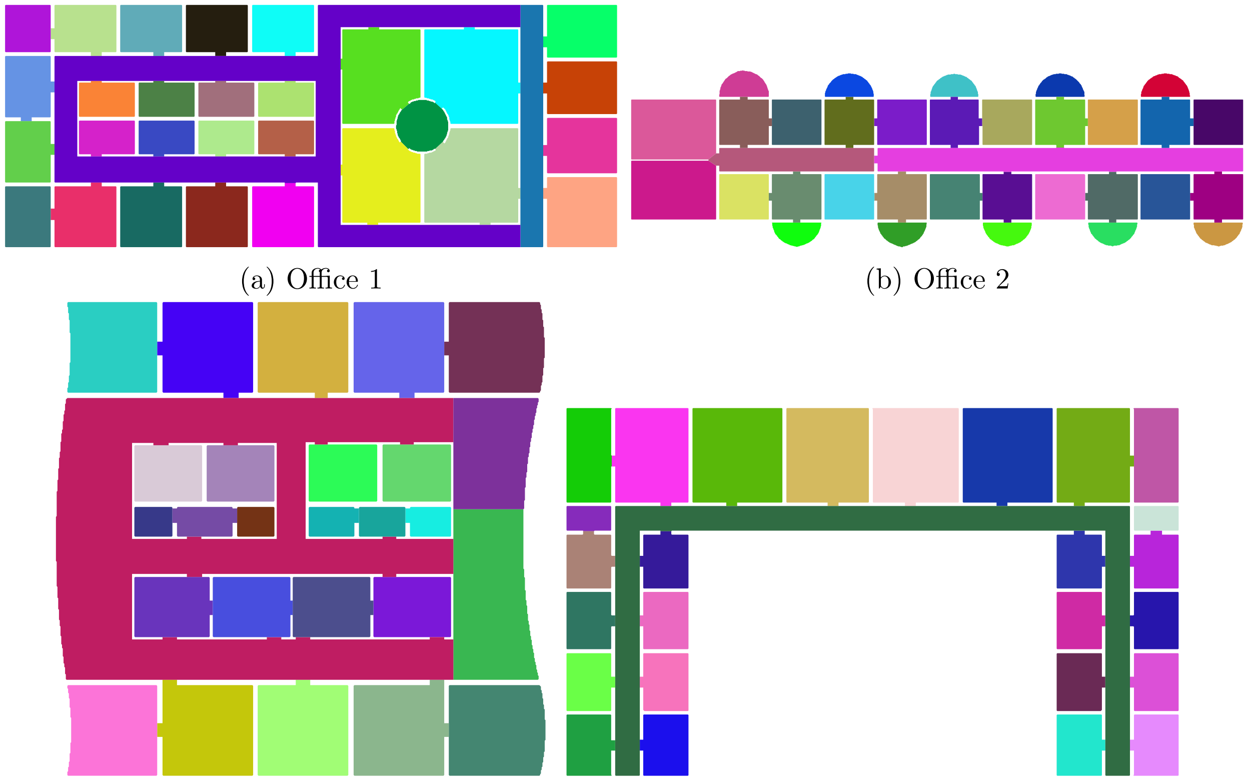

(f) Non-Convex with a hole segmented

Figure 4.1: The CDM applied to a test set environments. Figures 4.1c and 4.1d demonstrate that for a convex environment, the CDM does not insert any additional edges as expected. Figures 4.1c, 4.1d and 4.1e, 4.1f highlight the CDM applied to a convex and a non-convex environment with holes. In all figures, the CDM correctly identifies constriction points in the environment and inserts a set of additional edges.



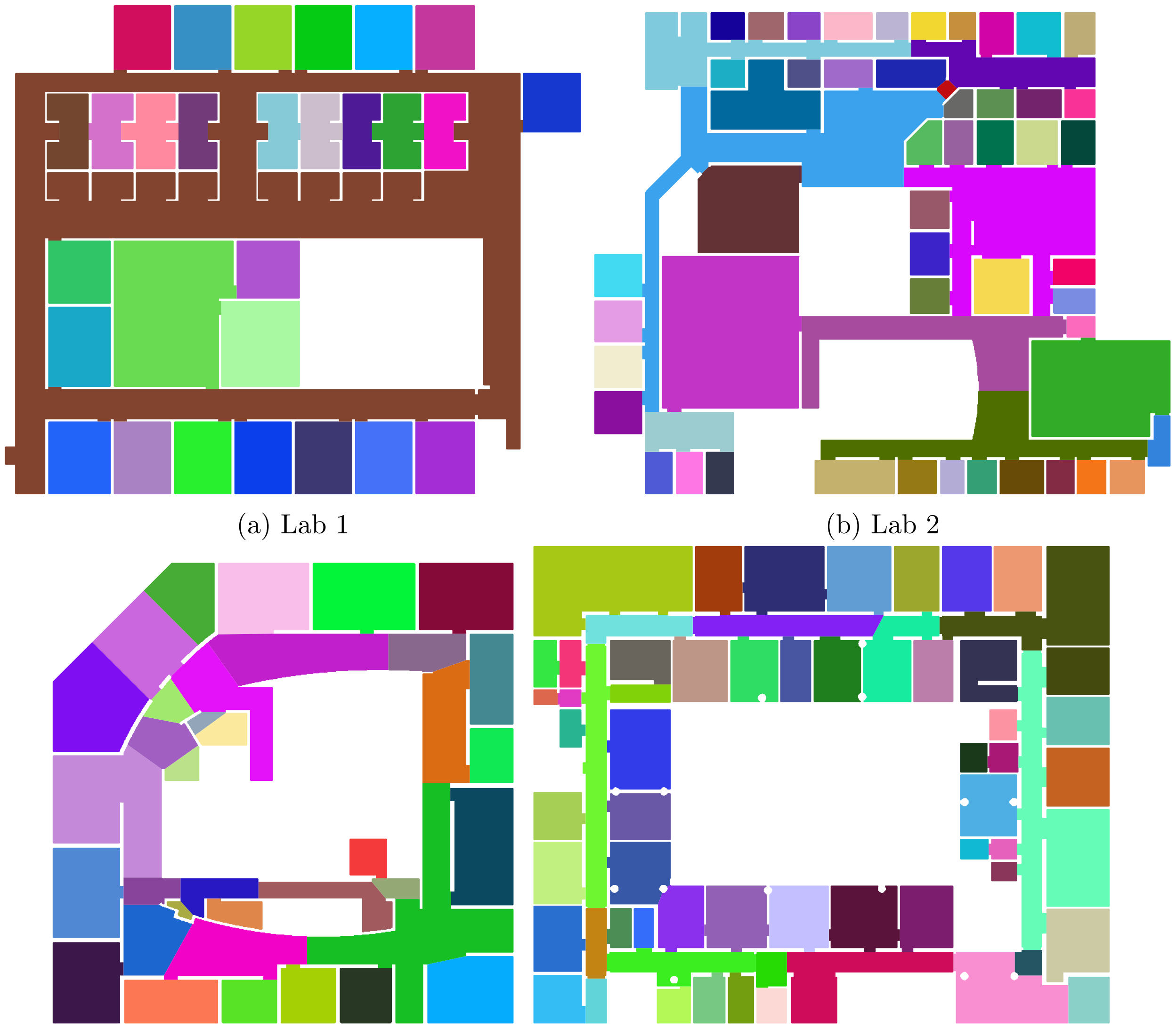
(a) Office 1 (b) Office 2

(c) Office 3 (d) Office 4

Figure 4.3: Office environment segmented with the CDM

The simple approximation setting removes any points that lay in a straight line along a contour. This process is demonstrated in Figure 4.2 along with the resulting segmentation.

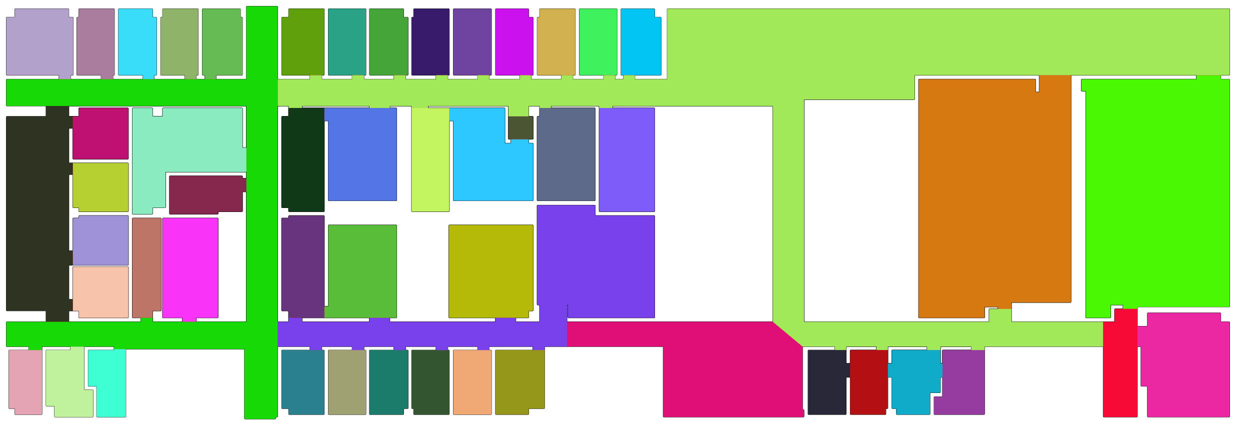
The CDM was tested on a set of twelve environments, ten of which are used in the work of [7] and two that are generated from floor plans of two buildings at the University of Waterloo, Ontario. The two floor plans are from the Engineering 5 (E5) building's 3rd floor and the Environmental 3 (ENV-3) building's second floor and are referred to as the UW environments. The resulting segmentations are highlighted in Figures 4.3, 4.4 and 4.5 and demonstrate the CDM's results when applied to four of the tested office environments, four of the lab environments and the two UW environments respectively.



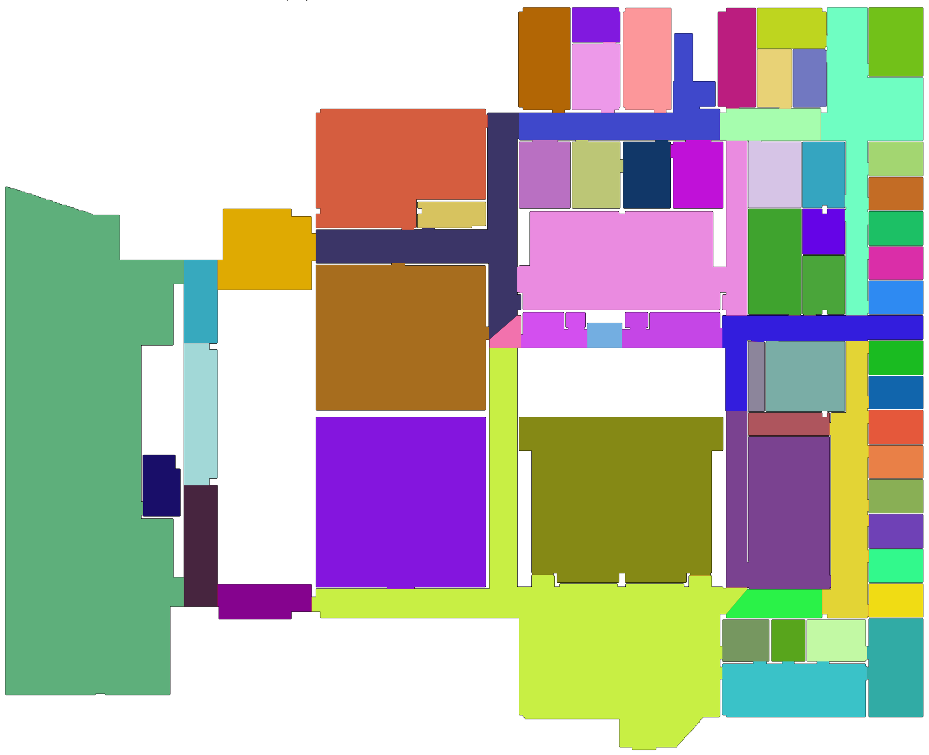
(a) Lab 1 (b) Lab 2

(c) Lab 3 (d) Lab 4

Figure 4.4: Lab environments segmented with the CDM



(a) Engineering 5 - 3rd Floor



(b) Environment 5

Figure 4.5: Segmentation result of applying the CDM to UW environments

4.1.1 Evaluation of Room Segmentation

To evaluate the overall quality of the CDM's segmentation, the environments highlighted in Figures 4.3, 4.4 and 4.5 were also segmented using the BD [10], Morphological and Voronoi segmentation methods implemented by [7]. A side by side, visual comparison of the segmentations produced by running the CDM, BD, Morphological and Voronoi segmentation algorithms when applied to the office, laboratory and floor plan environments are presented in Figures 4.6, 4.7, and 4.8 respectively.

Qualitative Evaluation

Several general patterns are observed in Figures 4.6, 4.7 and 4.8. First the CDM segments the maps into more defined rooms and hallways in comparison to the Morphological and Voronoi methods. In all cases, the BD is inferior to all other tested methods as it greatly over-segments the environment and tends to make cells that span across two rooms and a hallway if door ways are aligned, as observed in Figure 4.6 in Offices 1 and 2.

Overall, the Voronoi segmentation has the most similar results to the CDM, but has two undesirable tendencies. Firstly, the Voronoi segmentation tends to have rooms that spill out of the doorway and into hallways. This is most noticeable in Figure 4.6, Office 1 and 2 and Figure 4.7 Lab 1, 2 and 3. Secondly, in all environments, the Voronoi segmentation method over-segments hallways and other long, narrow passages. This tendency can be considered both a negative and a positive depending on the final application. Over segmentation in environments like Figure 4.6, can greatly simplify tasks such as path planning and region based searching in comparison to the CDM. For example in the Office 1 environment, the Voronoi segmentation method segments the main hallway into several regions while the CDM treats the hallway as a single, long and highly non-convex shape. For the use of coverage path planning however, over-segmentation is generally considered a negative trait as over-segmentation generally results in greater inter cellular travel [10].

The Morphological segmentation method [6] yields identical results to the CDM on the Offices 2, 4 and 5 environments (Office 5 is not shown here), but tends to under-segment in Offices 1 and 3 and all laboratory environments. In all Lab environments, Figure 4.7, there are at least four cells that contain partial portions of several rooms, or several rooms a large portion of hallway, or an extremely large portion of the map such as Lab 1 where 5 rooms and a hallway are grouped into one cell.

Figure 4.6: Segmentation Results: Office Environments

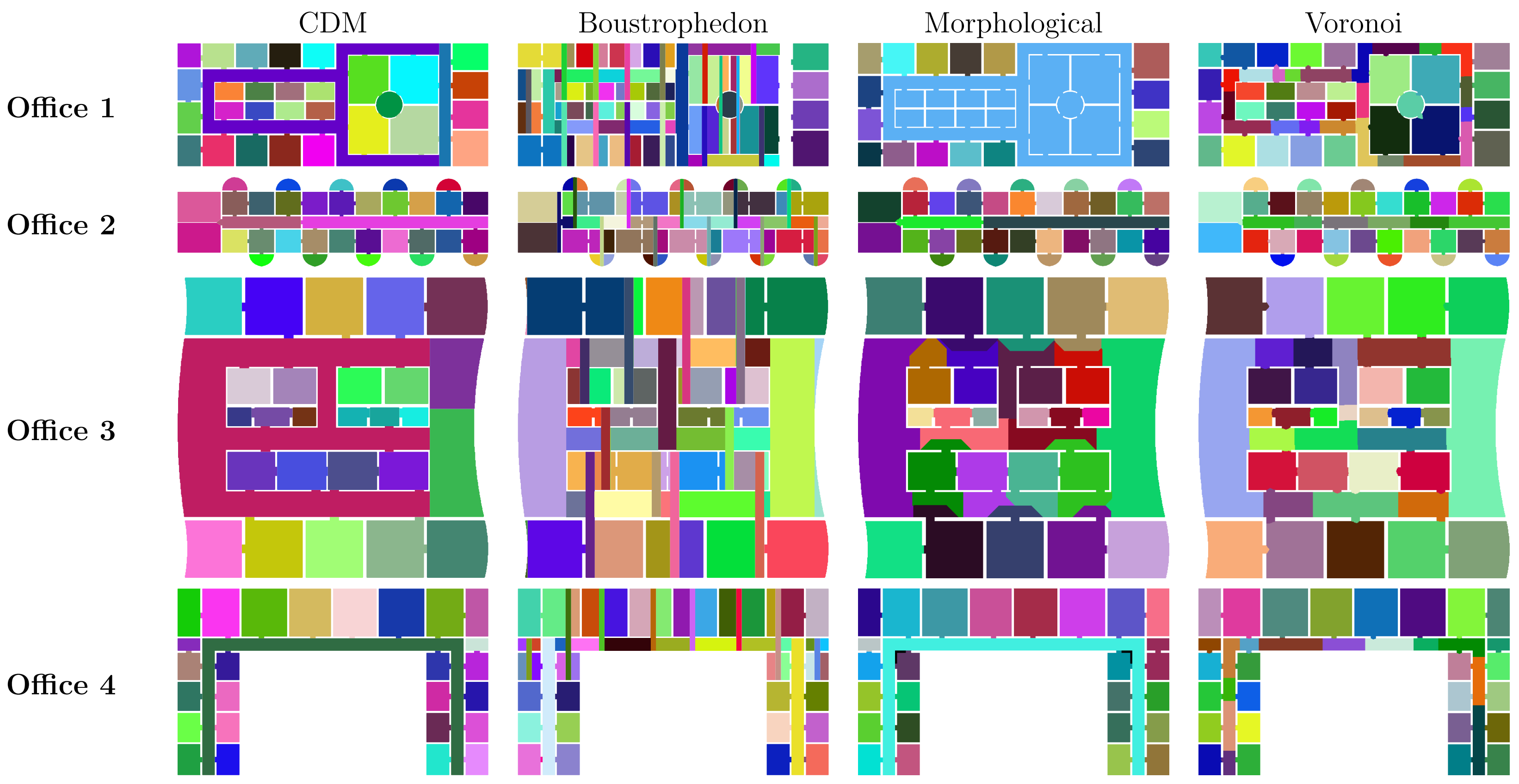


Figure 4.7: Segmentation Results: Laboratory Environments

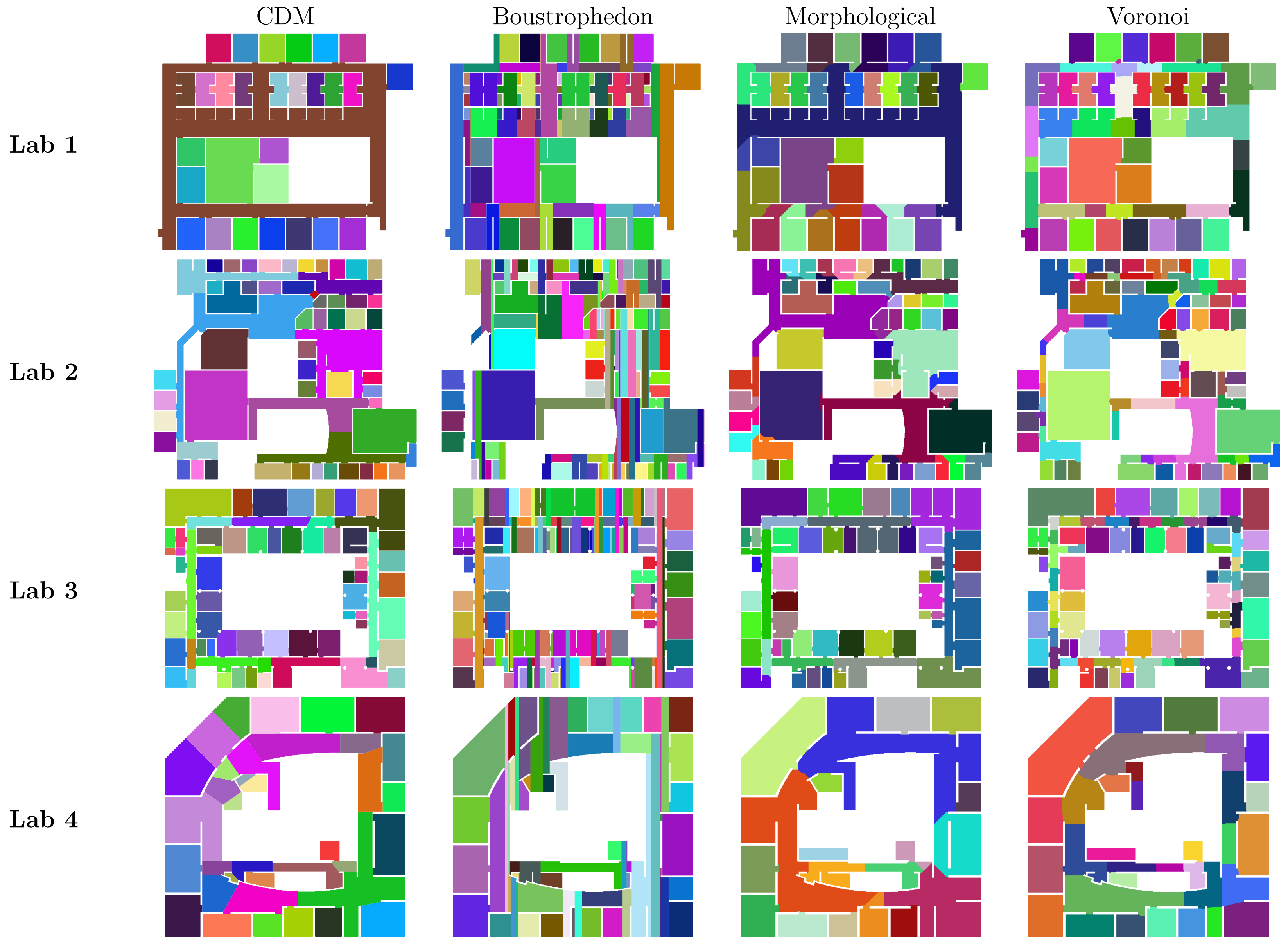
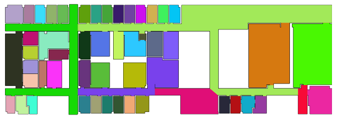
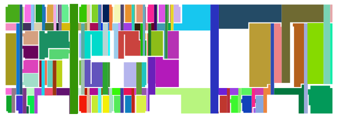
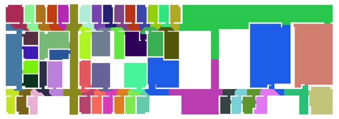
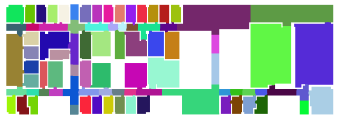
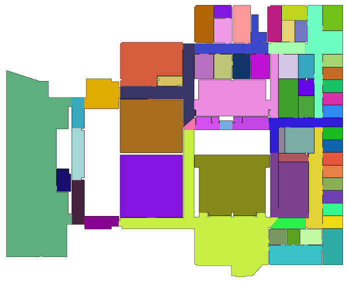
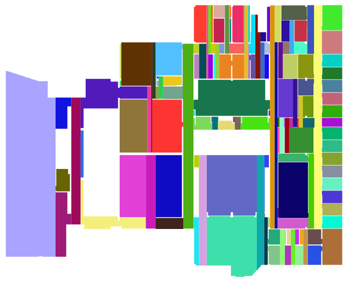
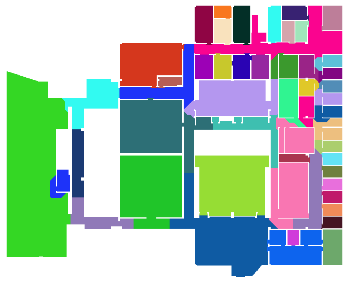
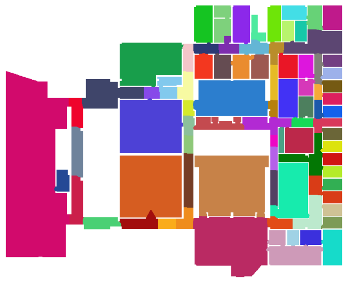


Figure 4.8: Segmentation Results: Floor Plan Environments

	CDM	Boustrophedon	Morphological	Voronoi
Engineer-ing 5				
Environ-mental 3				

Quantitative Comparison

In [6], the authors present a set of metrics to qualitatively evaluate and compare segmentation methods. Again, using software provided by [6], the metrics presented in [7] are calculated for each tested environment and summarized across all segmentation methods.

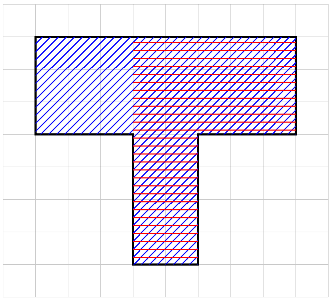
Let A_i, K_i, and B_i indicate the area, perimeter, and the area of the rotated bounding box of Cell C_i, respectively. The eigenvalues calculated using the principle component analysis, PCA, of cell C_i are denoted as $e_{1,i}$, and $e_{2,i}$. Let GT_i indicate a ground truth cell such that the area of the intersection of $GT_i \cap C_i$ is maximized and let the area of the intersection be denoted as $A_{GT_i \cap C_i}$. Using the above notation, the comparison metrics are defined as follows.

- **Cells**: The number of cells created by the segmentation method
- **Area**: The average area of each cell, A_i, in m^2
- **Perimeter**: The average perimeter of each cell, K_i, in m
- **A-Compactness**: The average of the cell area divided by its perimeter, A_i/K_i
- **B-Compactness**: The average of each cell's area divided by the area of the rotated bounding box of each cell, B_i, A_i/B_i.
- **Shape**: The quotient of the eigenvalues associated with each cell, $e_{1,i}/e_{2,i}$.
- **Recall**: The area of intersection $A_{GT_i \cap C_i}$, divided by the area of the cell, A_i.
- **Precision**: The area of intersection $A_{GT_i \cap C_i}$, divided by the area of the cell, A_{GT_i}

In general, fewer cells with relativity large areas and short perimeters are considered ideal as they indicate the cell has a large amount of open space in relation to its border. The metric, A-Compactness provides a unitless measure to quickly assess this value as it is the ratio of the area divided by the squared perimeter. The squareness and convexity of a cell can be evaluated by looking at the B-Compactness, which is the area of the cell divided by the rotated bounding box that encompasses it. Square cells will have a high B-Compactness value and cells that are long and spread out, such as the hallways, will have a low B-Compactness.

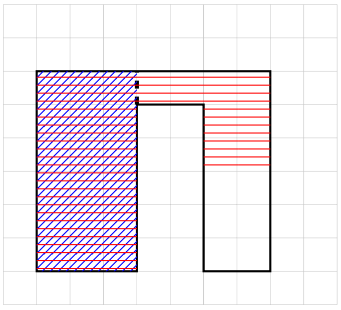
The Shape metric is also a measure of squareness, as it is the ratio of the primary axis over the secondary axis as found with the PCA of a cell. A shape value of 1 is observed in a perfectly square cell and a shape value of 4 will be observed in rectangular cell where the length is four times greater then the width.

The measurements of Recall and Precision are a way to compare the segmentation quality with a ground truth in a quantitative manner. A value of 1 for both recall and precision indicates a perfect match between the ground truth and the segmentation. Recall is high in segmentations where large cells completely overlap and low if an environment is over segmented. Precision is high when an environment is over-segmented and low in under-segmented environments. Figure 4.9 highlights two environments one with high recall and low precision, 4.9a and the other with low Recall, high Precision 4.9b.



$$A_i = 21$$
$$A_{GT} = 30$$
$$A_{C_i \cap GT_i} = 21$$
$$\textbf{Recall} = 1$$
$$\textbf{Precision} = 0.7$$

(a) High recall and low precision



$$A_i = 26$$
$$A_{GT} = 18$$
$$A_{C_i \cap GT_i} = 18$$
$$\textbf{Recall} = 0.69$$
$$\textbf{Precision} = 1$$

(b) Low recall and high precision

Figure 4.9: Examples of recall and precision metrics. Blue indicates the region of a ground truth cell GT_i and horizontal lines indicate the segmented area C_i

Each of the metrics is calculated for the five office, five lab and UW environments outlined in 4.1.1 using each segmentation method along with a set of ground truth segmentations. For the lab and office environments, the ground truth segmentations are taken directly from [6] and for the UW environments, the segmentation is based on the location of doorways in the floor plans used to create the Engineering 5 and Environmental 3 maps. The average and standard deviation of each metric for each segmentation method is presented in Table 4.1.

In Table 4.1 a strong match between a segmentation and the ground truth is indicated if both the metric's average value and standard deviation matches the ground truth's values. The importance of looking at the standard deviation as well as the area can be seen in Figures 4.3, 4.4 and 4.5, particularly in the Office 1 and 3, Lab 1, 2, and 3 and Engineering 5 and Environment 3 floor plans. In these environments, the characteristics of the resulting segmentation cells can vary dramatically depending if a cell, C_i is associated with a hallway or a room. Hallways, which tend to be narrow, long and highly non-convex, will have dramatically different Area, A-Compactness, B-Compactness, Shape and Perimeter values compared to square cell associated with a simple room. This variability is observed in standard deviation of each metric shown in Table 4.1. An environment such as Office 3, Figure 4.3c has a higher observed variation in each metric compared to Office 2, Figure 4.3c.

From Table 4.1 the CDM, on average, produces fewer and larger cells in comparison to all other segmentation methods as indicated by a relatively low A-Compactness, high Recall and average Precision metric. The high Recall value is expected as the CDM, from a room segmentation viewpoint, tends to under-segment environments, Figure 4.5b. The

Table 4.1: Average and standard deviation of segmentation characteristics averaged over all Lab, Office and Floorplan environments. Ground Truth indicates the characteristics of a map that was hand labeled

	Segmentation Method				
	CDM	BD	Morph	Voronoi	Ground Truth
Cells	**34 ± 16**	94 ± 42	36 ± 14	56 ± 21	40 ± 14
Area	31 ± 209	9.5 ± 38.2	**25.28 ± 87**	16.24 ± 46.5	24.81 ± 154
Perimeter	**26.65 ± 101**	20.17 ± 69.9	25.03 ± 87.5	16.42 ± 55.6	28.74 ± 109
A-Compactness	0.057 ± 0.09	0.033 ± 0.11	**0.042 ± 0.04**	0.051 ± 0.031	0.045 ± 0.178
B-Compactness	**0.78 ± 0.6**	0.03 ± 0.323	0.83 ± 0.503	0.92 ± 0.33	0.35 ± 0.316
Shape	**4.88 ± 39**	10.74 ± 80	4.51 ± 18	4.14 ± 14	6.05 ± 23.8
Recall	**0.86 ± 0.12**	0.58 ± 0.19	0.82 ± 0.26	0.80 ± 0.26	1.00 ± 0
Precision	0.76 ± 0.16	0.76 ± 0.10	0.78 ± 0.12	**0.85 ± 0.14**	1.00 ± 0

highest precision is observed by the Voronoi method as expected due to its tenancy to over-segment all environments, yet maintain room like shapes in the resulting segmentations. The BD scores the worst on almost all metrics, with the largest number of cells produced, smallest average areas, large Shape values due to the elongated nature of the cells and a dramatically lower Recall in comparison to all other segmentation methods.

Given that the primary goal of the CDM algorithm is to produce as few cells as possible while still being able to cover the cells with a complete coverage path planning pattern, the tendency of under-segmentation can be seen as a positive for coverage applications. If the application is to instead navigate an environment, locate an obstacle or perform the coverage operation with limited resources such that the agent cannot clean extremely large cells without refueling, the Voronoi segmentation method would be considered more effective.

4.2 Coverage Path Planning Results

The CDM coverage planning algorithm outlined in section 3.2 is implemented in C++ and applied to the lab, office and UW environments. An implementation of the boustrophedon coverage planning algorithm [10] is also applied to segmentation cells generated using the BD. The coverage path planning results for both the CDM and the BD are compared using their normalized coverage path lengths $\eta_{coverage} = \frac{l_{coverage}}{l_{opt}}$ where $l_{coverage}$ is the sum of the length of all coverage paths of all cells and l_{opt} is defined as the optimal coverage path length. For the environments analyzed in this work, it is assumed that $l_{opt} = \frac{a_F}{w}$ where a_F is the area of the free space of the environment and w is the width of the robot or agent, χ, performing the coverage operation. This formulation implicitly assumes that all regions in the environment are reachable by χ and that χ can be placed anywhere in F. With the exception of extremely simple environments such as a square, l_{opt} is likely impossible to achieve with a real robot. A value of $\eta_{coverage} = 1$ indicates a perfect coverage path.

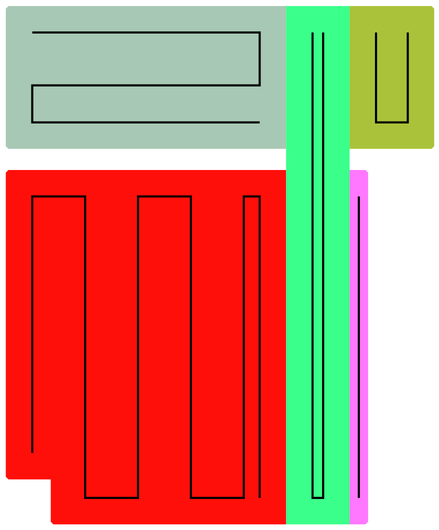


Figure 4.10: Over-segmentation of the map by the BD leads cells that require overlapping coverage passes to guarantee complete coverage of each cell.

Both the BD and the CDM coverage path planning algorithms are exact decomposition methods that require the robot performing coverage to travel along paths between the start and endpoints of the coverage patterns for each cell. During inter-cellular travel it is assumed that the robot is not performing any task, and so areas passed over during inter-cellular travel are not considered covered. The total length of a complete coverage path of the entire environment, l_{total}, is equal to the sum of the length of the inter-cellular travel, $l_{intercell}$ and $l_{coverage}$. Similar to the coverage plans, the total coverage path lengths of the BD and CDM is done using the normalized path length, $\eta_{total} = \frac{l_{coverage}+l_{intercell}}{l_{opt}}$.

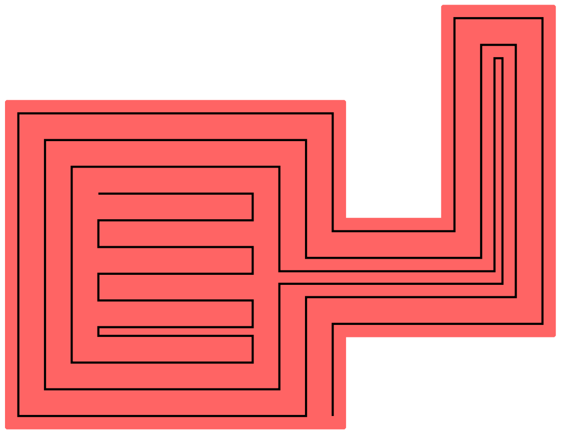
Table 4.2 highlights η_{total} and $\eta_{coverage}$ of paths generated using the CDM and BD coverage planning algorithms applied to the segmentations of each environment discussed in section 4.1. The results of 4.2 are generated using a robot with a width $w = 1.0$m. As the data in Table 4.2 indicates, the CDM produces both shorter coverage paths and total coverage paths on average as compared to the BD. This is due to the BD's tendency to

Table 4.2: The length of the coverage paths and total path length (Coverage path + Inter-cellular path length) normalized by the theoretical optimal coverage tour length each map using a robot of width 1.0m

	BD		CDM	
Map	Coverage	Total Path	Coverage	Total Path
E5	1.27	1.73	1.07	1.19
Env 3	1.17	1.41	1.10	1.25
Lab 1	1.12	1.37	1.17	1.29
Lab 2	1.12	1.35	1.12	1.21
Lab 3	1.26	1.46	1.14	1.25
Lab 4	1.26	1.39	1.14	1.38
Lab 5	1.24	1.43	1.16	1.51
Office 1	1.10	1.38	1.08	1.40
Office 2	1.16	1.51	1.13	1.28
Office 3	1.20	1.52	1.17	1.42
Office 4	1.23	1.53	1.13	1.35
Office 5	1.22	1.76	1.05	1.42
Average	**1.19**	**1.50**	**1.12**	**1.33**
Stdev	**0.06**	**0.14**	**0.04**	**0.10**

over-segment the environment as shown in Table 4.1. Thin cells that are less than a robot's width must still have a path that passes through them, even though the path may overlap with other neighboring cells as shown in Figure 4.10, leading to a non-optimal coverage path length $\eta_{coverage} > 1$. Due to over-segmentation, the BD also requires on average three times as many inter-cellular paths as compared to the CDM.

The CDM also suffers from a similar issue of redundant coverage due to the spiraling paths. If a long hallway or corridor is attached to a large room, there may be a requirement to produce a coverage plan that has redundant coverage, see Figure 4.11. The two paths shown in 4.11 highlight two robots of the same size covering the same environment. In Figure 4.11b, the robot has a width of 1.25m and must perform an additional contour following path to ensure the centre of the hallway section is covered. In Figure 4.11a, the of width of 0.93 m can cover the same environment with very little overlap during the last contour following path. The need for redundant coverage occurs when the modulus of the width of the section, $w_{section}$, and the width of the section is not a multiple of two robot widths, $2w$, is non zero, $w_{section} \bmod 2w \neq 0$. Any remainder indicates the amount of overlap on the final contour following path performed by the robot as shown in Figure


(a) Robot width = 1.25 m, $\eta_{coverage} = 1.14$

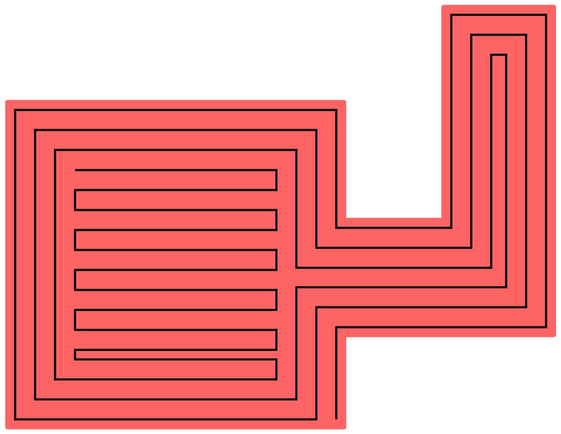

(b) Robot width = 0.93 m, $\eta_{coverage} = 1.03$

Figure 4.11: The amount of overlap on contour following paths is dependent on the robot size and the width of the hallway

4.11, with the worst case occurring when $width_{section} \bmod 2w = w$ which indicates a robot must completely backtrack over the portion of its path within that area of the cell. As the robot width increases, the more likely a significant overlap will be required to ensure complete coverage. This trend is observed in Figure 4.12.

The relation between agent widths of 0.25, 0.5, 0.75, 1.0, and 1.25 m and the average normalized and total path lengths for all environments is highlighted in Figure 4.12. As seen in 4.12 the CDM and BD both become more efficient when coverage is performed by a smaller agents as expected from Figures 4.10 and 4.11.

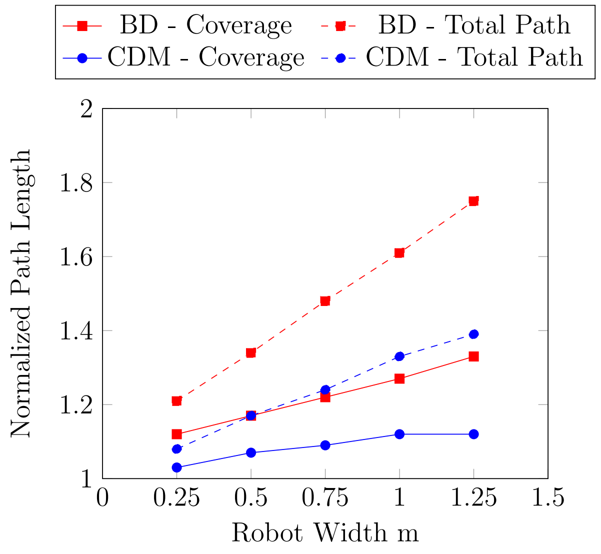


Figure 4.12: Effect of robot width on the normalized coverage and total path lengths.

Select examples of the CDM method from Table 4.2 are shown in Figures 4.13 through 4.17 with the CDM coverage paths generated for each cell. Figure 4.13 highlights a segmentation where a number of rooms near the center of the environment are grouped with the hallway. While from a room segmentation point of view this is perhaps undesirable, the CDM coverage algorithm from section 3.2 determines an efficient coverage path. Figure 4.14 highlights the largest environment tested in this work that also has the largest variety of shapes of rooms and hallways. Next, a pair of environments with curved segments that are approximated as straight lines are shown in Figures 4.15 and 4.16, demonstrating that the CDM can handle curvature if the curvature is approximated well enough. Lastly, an example of the CDM when applied to stereotypical, rectilinear office environments is shown in figure 4.17.

In all examples the CDM produces coverage paths that are within 15 % of the theoretical optimal for a robot of width 1.0 m. The inter-cellular travel is between 10 to 30 % of the total path length for a 1.0 meter robot which is a significant portion of the total travel time. In future work a modification to the coverage path generation could be made. Currently the start and end points of each path used by the TSP solver are taken from the start point that results in the best coverage path for each cell. In reality however, the coverage of a cell may start at any edge that is shared with a neighbour. Further, the CDM coverage algorithm could be modified in such a way that it always ends at a doorway into a neighbouring cell, which will in turn reduce the total travel time.

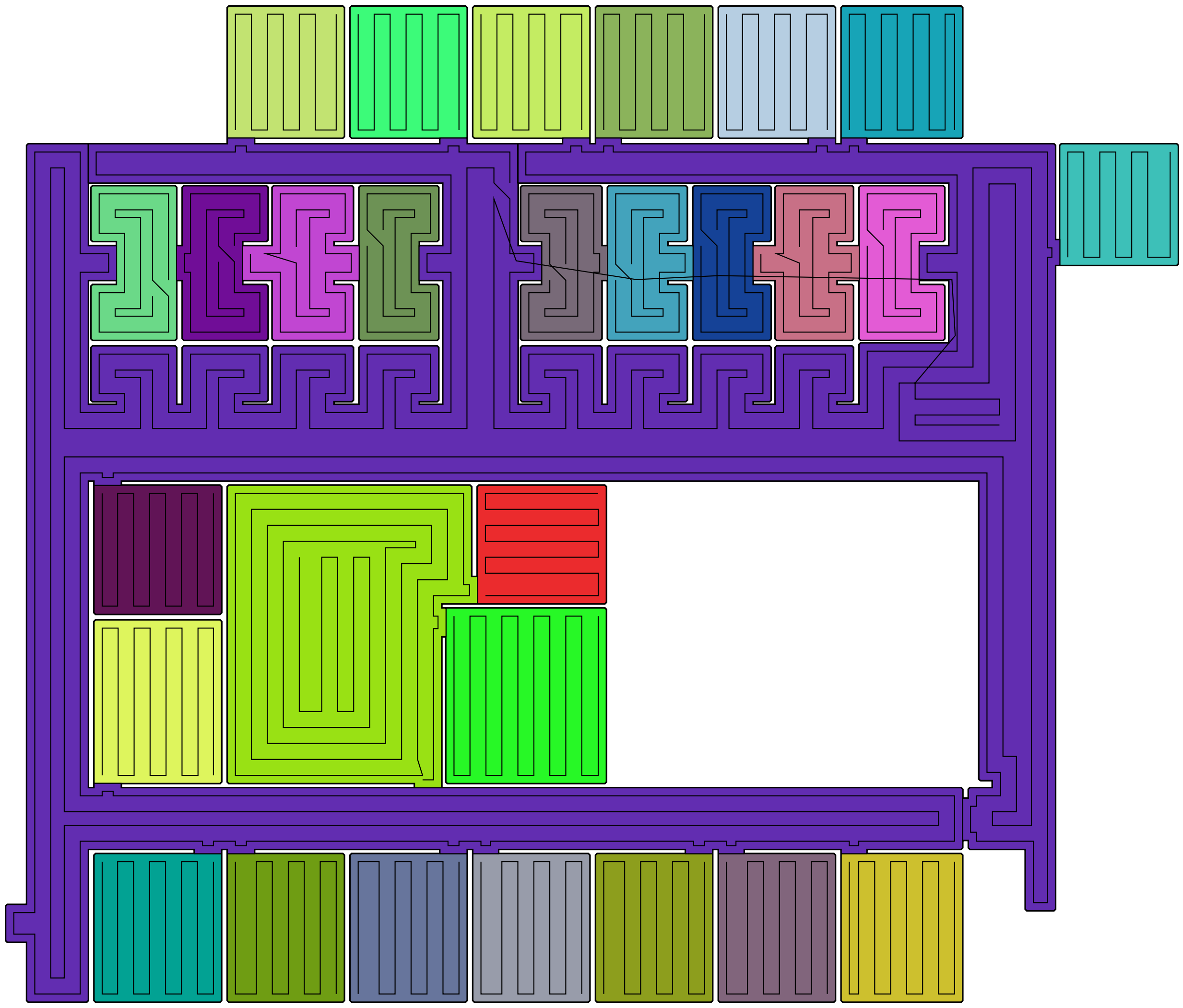


Figure 4.13: CDM Coverage paths generated for Lab 1. Despite the complex nature of the largest, non-convex cell with 8 room-like regions combined with the hallway, the CDM coverage path algorithm still determines a near optimal coverage path.

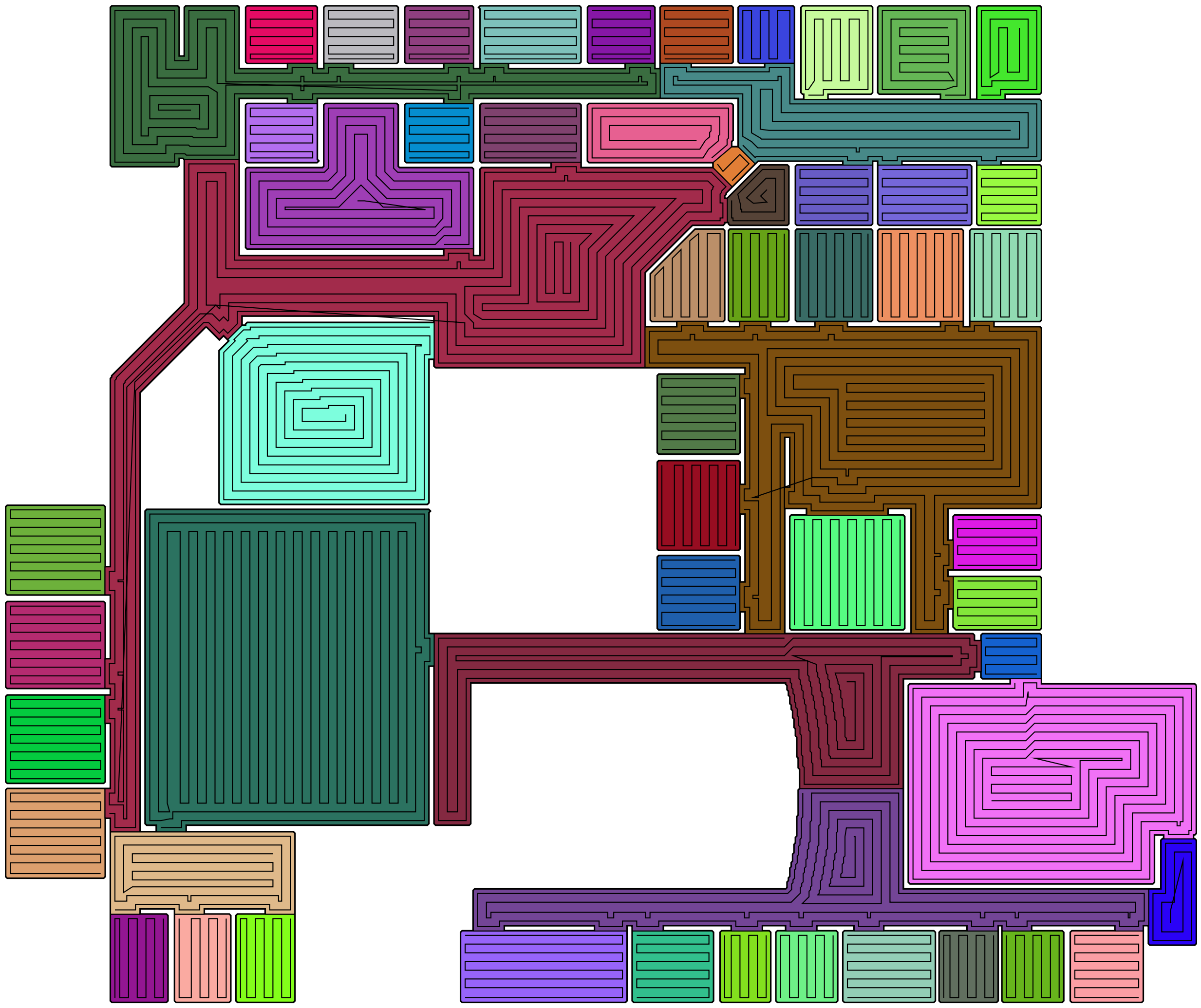


Figure 4.14: CDM Coverage paths generated for Lab 2 highlighting the largest environment explored in this work with a large variety of cell shapes.

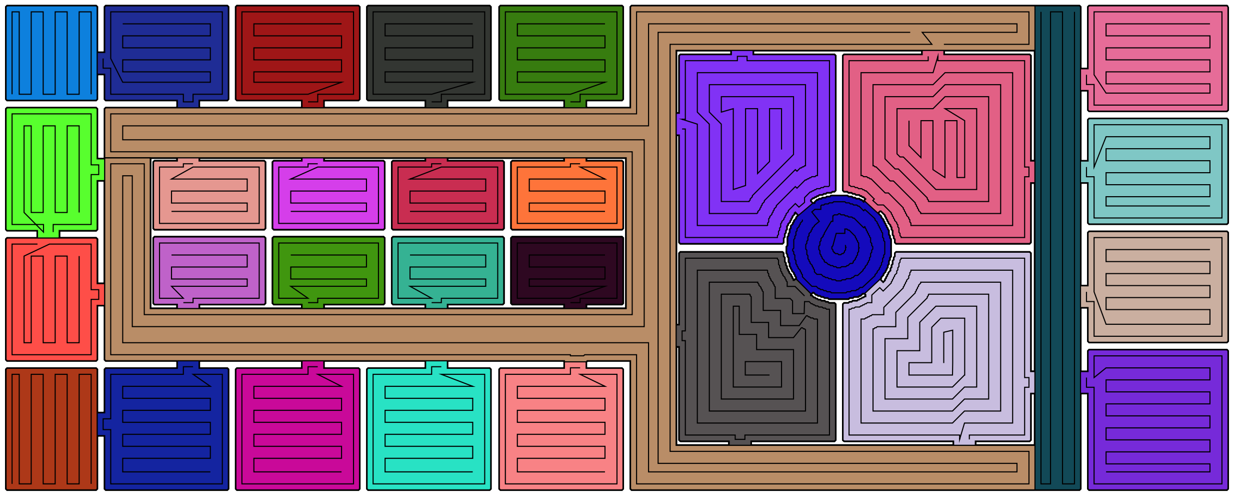


Figure 4.15: CDM Coverage paths generated for Office 1 showing coverage of a complex hallway section and a small curved region.

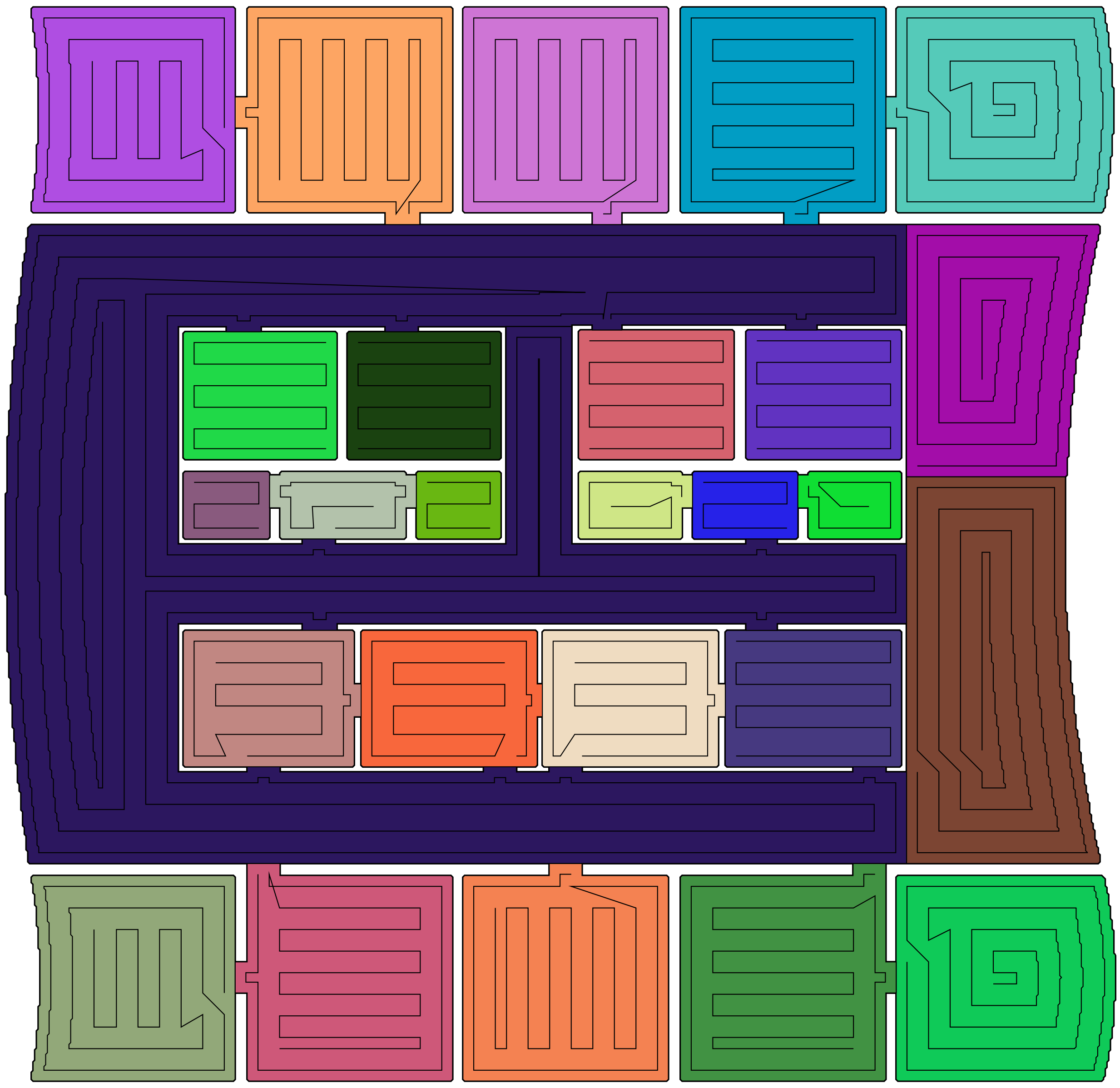


Figure 4.16: CDM Coverage paths generated for Office 2 highlighting coverage in an environment with curved outer boundaries that are approximated as a series of straight line segments.

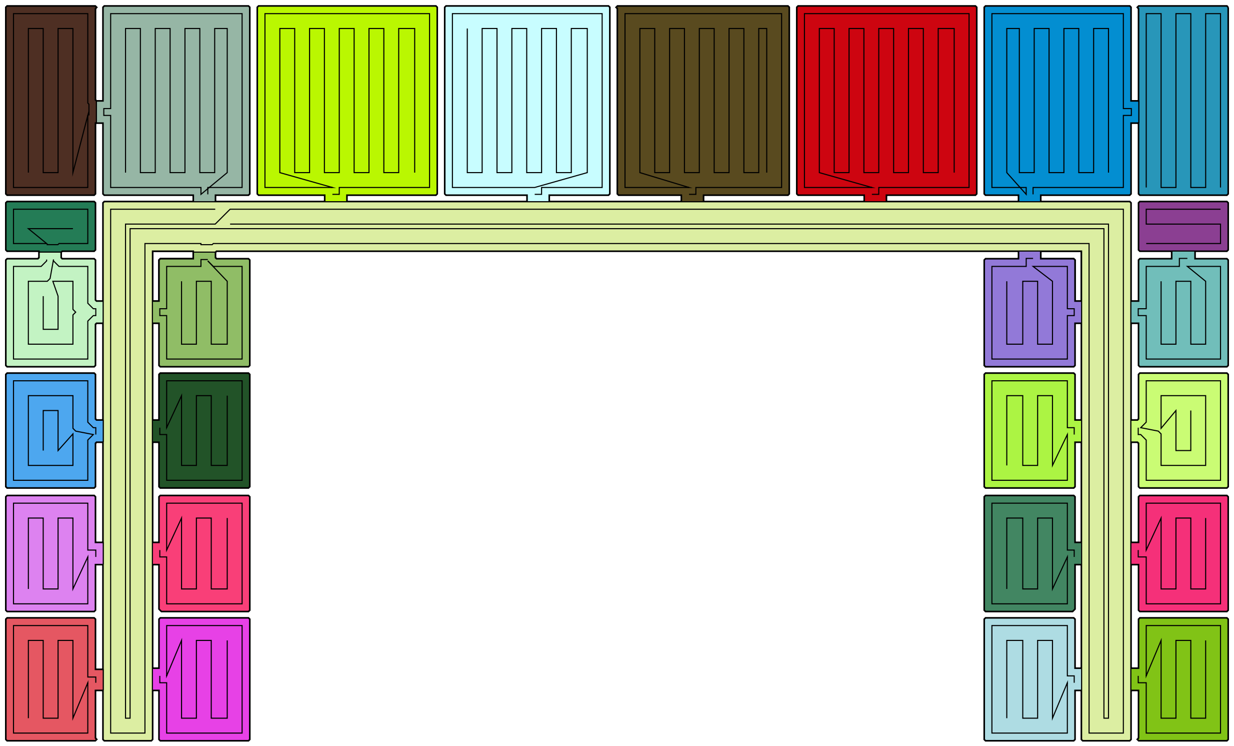


Figure 4.17: CDM Coverage paths generated for Office 4 showing coverage in a standard, rectilinear office environment.

Chapter 5

Conclusion

The demand for efficient, intuitive and accurate segmentation and coverage path planning algorithms will grow as the demand for both commercial and domestic, indoor service robots expands in the coming years. The accurate segmentation of indoor environments into rooms will allow service robots to behave and plan in a more human like manner. In particular, room segmentation, as demonstrated in this work, can be used to develop complete coverage path planning algorithms that can efficiently cover or clean a room produced by the proposed room segmentation algorithm.

This work presents a novel room decomposition algorithm that allows for an efficient, simple, yet intuitive coverage plan to be generated. Both the room segmentation algorithm (CDM) and the coverage path planner are applied to a set of 12 realistic indoor office and laboratory environments. The resulting segmentations are compared directly to the Boustrophedon [10], Morphological and Voronoi decomposition methods outlined in [7]. In the majority of the quantitative metrics, the CDM most closely represents the ground truth room segmentations provided by [7]. The segmentation provided by the CDM allows a set of coverage paths to be quickly generated that are on average no more than 1.2 times longer than the theoretical optimal path length on all environments tested. The CDM coverage path planner is compared to Boustrophedon Decomposition and cell coverage algorithm [10] and shown to have shorter total coverage paths for all environments tested.

5.1 Future Work and Possible Improvements

The algorithms presented in this work also have significant potential extensions that could improve both their robustness and efficiency. Possible improvements to the CDM include improving its robustness to noise in the map and deriving methods to handle furniture such as tables or chairs. For use on actual service robots, the CDM will need to be able to handle any noise in maps produced by any current state of the mapping or SLAM algorithms. The CDM coverage path planner might also be improved by allowing it to somehow classifying segmented cells as hallways or rooms and treating them differently during coverage planning. In particular for office environments hallways can have numerous connections between individual rooms, which often only have a single connection to the hallway. As a result, the robot performing coverage must always enter and exit hallways numerous times to reach each room, and therefore could coverer a large portion of the sections of hallways as the robot moves between rooms. A more efficient solution will be to plan and perform a coverage operation while moving between rooms through a given hallway. Further, the cell coverage method can be improved using additional cell coverage methods which follow a subset of wall edges, similar to the work of Oksanen et al. [29]. 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.

References

[1] Ercan U Acar, Howie Choset, and Ji Yeong Lee. Sensor-based coverage with extended range detectors. *IEEE Transactions on Robotics*, 22(1):189–198, 2006.

[2] Oswin Aichholzer, Franz Aurenhammer, David Alberts, and Bernd Grtner. A novel type of skeleton for polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[3] Esther M. Arkin, Sndor P. Fekete, and Joseph S.B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25 – 50, 2000.

[4] Esther M. Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197 – 218, 1994.

[5] Patrick Beeson, Nicholas K Jong, and Benjamin Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *IEEE International Conference on Robotics and Automation, 2005*, pages 4373–4379. IEEE, 2005.

[6] Richard Bormann, Joshua Hampp, and Martin Hägele. New brooms sweep clean-an autonomous robotic cleaning assistant for professional office cleaning. In *IEEE International Conference on Robotics and Automation (ICRA), 2015*, pages 4470–4477. IEEE, 2015.

[7] Richard Bormann, Florian Jordan, Wenzhe Li, Joshua Hampp, and Martin Hägele. Room segmentation: Survey, implementation, and analysis. In *IEEE International Conference on Robotics and Automation (ICRA), 2016*, pages 1019–1026. IEEE, 2016.

[8] Fernando Cacciola. A cgal implementation of the straight skeleton of a simple 2d polygon with holes. In *In Proceedings of the 2nd CGAL User Workshop*, 2004.

[9] Fernando Cacciola. 2d straight skeleton and polygon offsetting. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.7 edition, 2015.

[10] Howie Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[11] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.

[12] Howie Choset, Ercan Acar, Alfred A Rizzi, and Jonathan Luntz. Exact cellular decompositions in terms of critical points of morse functions. In *IEEE International Conference on Robotics and Automation (ICRA), 2000*, pages 2270–2277. IEEE, 2000.

[13] Arun Das, Michael Diu, Neil Mathew, Christian Scharfenberger, James Servos, Andy Wong, John S Zelek, David A Clausi, and Steven L Waslander. Mapping, planning, and sample detection strategies for autonomous exploration. *Journal of Field Robotics*, 31(1):75–106, 2014.

[14] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 1691–1696. IEEE, 2012.

[15] Elisabetta Fabrizi and Alessandro Saffiotti. Augmenting topology-based maps with geometric information. *Robotics and Autonomous Systems*, 40(23):91 – 97, 2002. Intelligent Autonomous Systems (IAS), 2002.

[16] Petr Felkel and Stepan Obdrzalek. Straight skeleton implementation. In *Proceedings of spring conference on computer graphics*, 1998.

[17] Yoav Gabriely and Elon Rimon. Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot. In *IEEE International Conference on Robotics and Automation (ICRA), 2002*, pages 954–960. IEEE, 2002.

[18] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[19] Martin Held, Stefan Huber, and Peter Palfrader. Generalized offsetting of planar structures using skeletons. *Computer-Aided Design and Applications*, 13(5):712–721, 2016.

[20] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130, 2000.

[21] Susan Hert, Sanjay Tiwari, and Vladimir Lumelsky. A terrain-covering algorithm for an AUV. *Autonomous Robots*, 3(2):91–119, 1996.

[22] Wesley H Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *IEEE International Conference on Robotics and Automation (ICRA), 2001*, volume 1, pages 27–32. IEEE, 2001.

[23] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015.

[24] Mathieu Labbé and François 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*, pages 2661–2666. IEEE, 2014.

[25] Jean-Claude Latombe. *Robot motion planning*. Kluwer Academic Publishers, 1991.

[26] Tae-Kyeong Lee, Sanghoon Baek, and Se-Young Oh. Sector-based maximal online coverage of unknown environments for cleaning robots with limited sensing. *Robotics and Autonomous Systems*, 59(10):698 – 710, 2011.

[27] Yan Li, Hai Chen, Meng Joo Er, and Xinmin Wang. Coverage path planning for uavs based on enhanced exact cellular decomposition method. *Mechatronics*, 21(5):876–885, 2011.

[28] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[29] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[30] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21 – 71, 1998.

[31] Sebastian Thrun and Arno Bü. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 944–950. AAAI Press, 1996.

[32] Michael J Tribou, Adam Harmat, David WL Wang, Inna Sharf, and Steven L Waslander. Multi-camera parallel tracking and mapping with non-overlapping fields of view. *The International Journal of Robotics Research*, page 0278364915571429, 2015.

[33] Kai M Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS), 2008*, pages 1160–1165. IEEE, 2008.

[34] Alexander Zelinsky, Ray A Jarvis, JC Byrne, and Shinichi Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *IEEE International Conference on Robotics and Automation (ICRA), 2013*, volume 13, pages 533–538, 1993.