12-2017

A Complete Coverage Algorithm for 3D Structural Inspection using an Autonomous Unmanned Aerial Vehicle

Venkat Ramana Reddy Garlapati

A Complete Coverage Algorithm for 3D Structural Inspection using an Autonomous Unmanned Aerial Vehicle

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements for the Degree

Masters of Science

University of Nebraska at Omaha

by

Venkat Ramana Reddy Garlapati

Dec 2017

Supervisory Committee:

Prithviraj (Raj) Dasgupta

Stanley Wileman

Robin Gandhi

A Complete Coverage Algorithm for 3D Structural Inspection using an Autonomous Unmanned Aerial Vehicle

Venkat Ramana Reddy Garlapati, MS

University of Nebraska, 2017

Advisor: Dr. Prithviraj(Raj) Dasgupta

This thesis presents a novel algorithm for complete coverage of three-dimensional structures to address the problem of autonomous structural inspection using an Unmanned Aerial Vehicle (UAV). The proposed approach uses a technique of cellular decomposition based on Morse decomposition to decompose the 3D target structure into 2D coverable faces that are subsequently connected using a graph-based representation. We then use graph traversal techniques such as the Traveling Salesman Problem (TSP) to generate a flight coverage path through the decomposed faces for a UAV to completely cover the target structure, while reducing the coverage time and distance. To test the validity of our proposed approach, we have performed a series of experiments using a simulated AscTec Firefly UAV in different environments with 3D structures of different sizes and geometries, within the Robot Operating System (ROS) Gazebo simulator. Our results show that our approach guarantees complete coverage of the target structure. Comparison of our coverage strategy with other strategies shows that our proposed TSP-based coverage strategy performs up to 50% better in reducing the flight path with an average of 30% fewer turns and 12% less coverage duration than a largest-area-first approach.

Acknowledgements

I wish to express my deep respect and indebtedness to my advisor, Dr. Raj Dasgupta, for his valuable guidance, patience, encouragement and support in every phase of my research. I would like to thank my supervisory committee members Dr. Stanley Wileman and Dr. Robin Gandhi for their valuable support. I would also like to express my gratitude to the CMANTIC lab members for their continuous support and their willingness to help me in every possible way. I extend my gratitude to the Dept. of Computer Science for the assistantship and Office of Research and Creative Activity (ORCA) for GRACA which tremendously supported my research. Finally, I would like to thank my parents, my other family members and friends for their inspiration and love throughout the course of this dissertation.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In this thesis, we investigate techniques that will enable unmanned aerial vehicles (UAVs) to autonomously inspect a three dimensional (3D) structure. UAVs equipped with their GPS-based navigation capabilities are currently used extensively in various 2D complete coverage applications, including flood and wildfire monitoring [2], agricultural surveying [3], and traffic management [4]. However, their usage in 3D coverage problems has been fairly restricted. There are several challenges that need to be addressed in 3D UAV coverage such as developing an efficient 3D coverage algorithm that guides the UAVs trajectory along complex structures such as buildings, towers, and bridges, maneuvering the UAV autonomously in small spaces close to the structures being inspected, localizing the UAV, and avoiding collisions with obstacles along the structure. Similarly, applications that require 3D coverage for inspection/surveying purposes such as, building inspection, bridge inspection, cell phone tower inspection, and gas pipelines surveillance. To address issues, we need to develop accurate control units and efficient path planning algorithms which enable full coverage of a target 3D structure. The main research question that we plan to investigate in this thesis is the following; How can we enable autonomous inspection of a 3D structure by a UAV by developing a novel 3-D structural inspection algorithm that guarantees complete inspection of the structure, while reducing the inspection time

and distance? To address this research question, this thesis makes the following research contributions:

- A 3D decomposition algorithm that decomposes the enveloped target structure into 2D faces and sub-faces, such that the union of the surfaces of the faces and sub-faces completely covers the exposed surface of the target structure.
- A graph-based representation to connect the 2D faces and sub-faces.
- A TSP-based coverage tour to visit each face at least once.
- The comparison of TSP-based coverage traversal with respect to Largest Area First greedy approach to evaluate the performance of our approach.

Structural inspection is an instance of the Coverage Path Planning (CPP) problem in a 3-D environment. Path planning is one of the extensively studied problems and an important task in the field of robotics. It deals with finding a continuous collision-free path between a start point or start configuration and a goal point or goal configuration. The prerequisite for the robot is to know its current location and map of the environment to find the location of goal and stationary obstacles. Path planning in a static environment is easy as compared to the dynamic environment where the obstacles are not stationary. The geometry of robot and obstacles is represented in a 2-D or 3-D workspace, while the path of the robot is represented in configuration space. For example, if the robot is a point translating in 2-D plane, configuration is represented using two parameters (x, y). For the flying robot (UAV) which can translate and rotate in 3-D workspace, configuration is represented using 6 parameters: Euler angles (α, β, γ) for rotation and (x, y, z) for translation.

Coverage Path Planning is the task of determining a path such that it passes over all points of an area of interest simultaneously avoiding obstacles in the environment. During the initial research on CPP, one of the works [5] examined the basic requirements a robot must address to perform complete coverage operation. They are: robot must move through

all the points in the target area, robot must not overlap the paths, simple motion patterns such as sweeping or circular motions should be used to cover the target area, robot must maintain optimal coverage path in terms of length and coverage time, and the robot should avoid the obstacles. These coverage requirements were initially stated for the ground robot moving in 2-D environment but they are equally applicable to 3-D environment. All of these requirements are however not possible to address depending on the coverage dimension and complexity of the target environment.

Some of the classic 2-D methods include Trapezoidal decomposition [6] [7] which is an exact cellular decomposition technique that handles only planar and polygonal environments. Boustrophedon decomposition [8] which is similar to Trapezoidal but generates fewer cells, hence shorter coverage paths are obtained. Morse-based cellular decomposition is an approach based on critical points of Morse function [9] which has advantage of handling non-polygonal obstacles. Spanning Tree Coverage [1] is an online approach which allows robot to subdivide the workspace into a grid map and follow a systematic spiral path.

By reviewing the recent works [10] [8] [9] [1] [11] [12] [13]in the field of path planning and coverage path planning, it is clear that most works concentrate only on 2-D environment, limiting the behaviors of the robots to one plane or in some cases considering the height as a constant to achieve coverage in 2.5-D environment. With high availability of low cost UAVs, coverage path planning applications are extended to 3-D environments which tend to be unstructured and has uncertain factors such as the UAVs should be able to fly in occluded spaces, avoid flying close to the surfaces, and should handle different levels of air pressure between and around building like structures. Since 2D CPP algorithm is a variant of the travelling salesman algorithm, the complexity of the algorithm is proven to be NP-hard. Similarly, the complexity of 3D CPP increases exponentially with the number of coverable surfaces of the environment. To plan a collision free path that completely covers the target environment, a robot should be equipped with appropriate sensors to

perceive the environment and suitable control algorithms to maneuver it. Model of the environment has to be decomposed into free space cells for the robot to cover. In the context of 3D structures, the target structure can be viewed as multiple polyhedrons adjacent with each other, creating a complex model to decompose. From the optimization view point, since the robot must pass over all points in the workspace, CPP problem is related to the Traveling salesman problem (TSP) where instead of visiting each city, an agent must visit a neighborhood of each city. In this case instead of visiting each free space cell i.e., going to a particular point in the cell, an agent must completely cover the cell in sweeping or circular pattern to inspect the structure completely. Hence finding a 3D coverage path plan is an NP-hard problem and there exist no common solutions. So, the aforementioned basic requirements by Cao et al for the CPP problem are prioritized based on the objectives of coverage.

To address the above mentioned problem, we propose an approach that takes the bounding coordinates of the target structure and decomposes it into non-overlapping rectangular 2D cells. To facilitate this, we considered a 2D virtual plane which is moved across the target structure to observe the changes in structure. The plane continuity changes at events when the plane splits, expand, contract or ends. These events track the changes in the environment and are used as the basis for cellular decomposition. The decomposed 2D cells are then modeled as a weighted graph with cells representing the graph vertices and the boundary between adjacent cells as edges with the edge weight corresponding to the distance between the centroids of two adjacent cells.The Traveling Salesman algorithm is applied to the final graph to determine the shortest possible route connecting the free cells such that each cell is visited exactly once and the route terminates at the starting cell. To avoid collisions of the robot with the structure, we do not construct a coverage path directly on the surfaces of the 2D cells, but the coverage path is planned in an offset surface from which the UAV will inspect the structure. That is, the path is planned on a virtual surface that wraps the target structure at a fixed offset distance.

To verify our approach, we tested this inspection path planning algorithm initially in simulation, with challenging structures placed in various environment layouts. Our results show that the proposed approach guarantees complete coverage of the target structures. Our TSP-based coverage approach performed up to 50% better in reducing the flight path and 12% less coverage duration than a largest-area-first approach.

The rest of this document has the following structure. In Chapter 2 we discuss the related works on this topic, Chapter 3 presents our proposed approach to decompose the 3D target structure into coverable surfaces, then in Chapter 4 we present how the coverable surfaces are represented in a graph data structure and the tour is planned. Then in Chapter 5 we discuss the experiments we performed to validate this approach and the results of the various experiments. And finally in Chapter 6 we summarize our work, discuss the conclusions we can draw, and provide a discussion on the future work for this topic.

Chapter 2

Related Work

In this chapter, we survey the related work on robotic CPP. We start with the general path planning problem, that is, finding the path between start and goal locations while avoiding obstacles. Then, we focus on CPP approaches in a 2-D environment using ground robots. Finally, we extend the study to CPP approaches in 3-D environments mainly using underwater and aerial robots.

2.1 Path Planning

Path planning for mobile robots is a task to find a collision-free route, from a specified start location to a goal location by avoiding obstacles and satisfying certain criteria like minimizing the coverage time, battery, and length of the coverage path. Path planning methods are classified based on the environment the robot is placed in as, static with stationary obstacles or dynamic with both stationary and mobile obstacles. For static environments, the classic path planning approaches include Visibility Graph [14] where the robot connects visible vertices of polyhedron in a graph, Cell Decomposition [15] which divides the free space into cells and an optimal path is designed through these free space cells, and Potential Field-based methods [16] that use potential field in the configuration space to solve the path planning problem. These approaches have several

limitations, such as higher time complexity in high dimensions, and getting trapped in local optima, which make them inefficient. To overcome the drawbacks of the above mentioned methods, sampling based algorithms such as Rapidly-exploring Random Tree (RRT) [17], and Probabilistic Road Map (PRM) [18] were developed. In these methods, the robots need prior information about the workspace where they operate. Then the algorithm samples the environment into set of nodes to search randomly to find an optimal path. Other optimal search algorithms such as Dijkstras [19], A* [20] and D* [21] have been developed with major advantage of high speed implementation. For a dynamic environment, sensor-based path planning approaches like Genetic Algorithms [22], Neural Networks [23], and Simulated Annealing [24] are best suited.

The above mentioned approaches are designed to build an optimal path between two desired points. These approaches cannot be extended to complete coverage applications as the main objective of the coverage application is to cover each and every point on the target environment. For sensor based approaches, usually the sensor range is small compared to the size of the environment which makes these approaches inefficient and not applicable to extend to coverage path planning applications.

2.2 Coverage Path Planning

Coverage path planning determines a path that guarantees that an agent will pass over every point in a given environment. The robotic applications for this task include lawn mowers [5], painter robots [25], vacuum cleaning robots [26] , demining robots[27], inspection of complex underwater structures [28], and Bridge Inspection, just to name a few.

To solve the coverage problem in a structured manner, a coverage path planning algorithm is used to determine a set of waypoints in the environment that the robot should travel to so that it is able to cover every portion of the free space of the environment using its sensor. Researchers have proposed several algorithms to solve the robot coverage

problem. The CPP problems are classified as either off-line or on-line. Choset has originally proposed this classification in his survey [29]. Off-line algorithms rely on an already known environment with stationary obstacles. In contrast, in on-line algorithms the prior information of the full environment is not known and the robots utilize sensors such as sonic and visible or infrared light-based sensors to cover the target space while avoiding obstacles in real time.

The CPP problem is related to the Traveling salesman problem [30] but the agent must visit a neighborhood of each city, instead of visiting all cities each one at least once. However, in CPP the agent must pass over all points in the target area in contrast to visiting all the neighborhoods. Two other related problems to CPP are the art gallery problem and the watchman route problem. The art gallery problem originates from the real world problem to find the minimum number of guards needed to station so that they can observe the entire gallery [31]. The watchman route problem finds the shortest route from a given point back to itself so that every point is visible in the given environment from at least one point in the route [32]. The lawnmower problem which does not consider obstacles and finds a path to cut all the grass of a given region, is also proven to be NP-hard. All these are NP-hard, hence the computational time required to solve these problems increases drastically with the dimension of the problem.

One of the simplest exact cellular decomposition techniques which can yield a complete coverage path is the trapezoidal decomposition [6], which handles only planar, polygonal spaces. This method can be classified as off-line. To form the decomposition, at each vertex v_{i}, draw two segments, one called upper vertical extension and the other called lower vertical extension. These extensions start at the vertex and terminate when they first intersect an edge of the polygon that lies immediately above or below v_{i}, respectively. The upper and lower vertical extensions divide the free space into trapezoid shaped cells as shown in the Figure 2.1. Hence, simple back and forth motions can be used to cover each cell. Complete coverage is guaranteed by finding an exhaustive walk through the adjacency

graph associated to the decomposition. A drawback of trapezoidal decomposition is

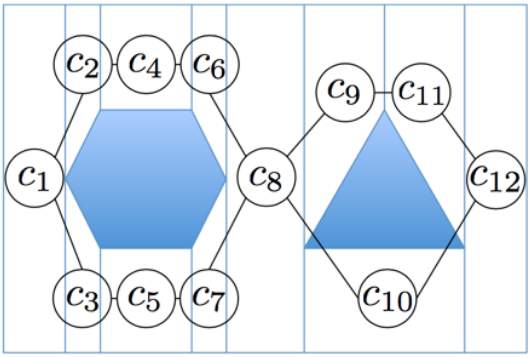
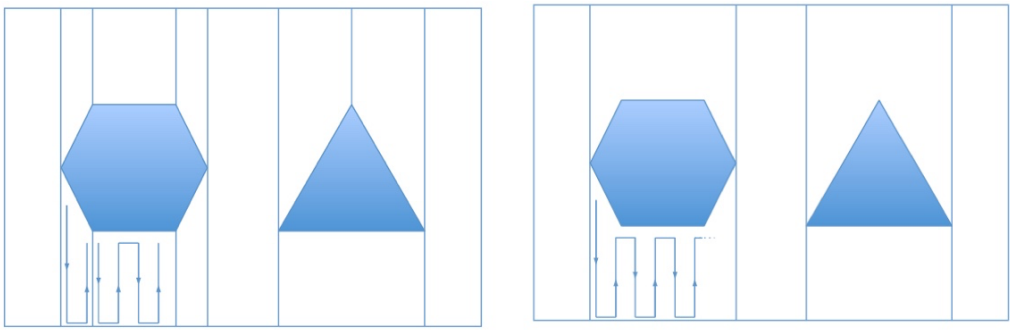


Figure 2.1: Trapezoidal decomposition of an example workspace with its corresponding adjacency graph

that it generates many cells that, intuitively, can be merged together to form bigger cells. To overcome this limitation Choset et al [33] proposed one of the earliest and most successful techniques to solve the robot coverage problem in a two-dimensional (2-D), planar environment using a technique called Boustrophedon Cellular Decomposition (BCD). It is similar to trapezoidal decomposition but effectively reduces the number of cells. Therefore, shorter coverage paths are obtained. In this method, the environment is dynamically divided into polygon-shaped cells by the robot as it covers the environment; each cell is then covered using back-and-forth sweeping motions using a seed spreader algorithm [25]. Later Acar et al., [34] generalized the BCD by proposing a novel cellular decomposition approach based on critical points of Morse functions. The Morse-based decomposition has the advantage of handling non-polygonal obstacles. In the BCD, a vertical slice, defined in terms of the Morse function h(x,y) = x, is swept from left to right in the workspace. The slice is parameterized by λ, which fixes its location in the target space. Increasing the value of the λ, the slice sweeps from left to right through the workspace. The slices connectivity changes depends on its intersection with the obstacles. These connectivity changes are marked as critical points and are used to decompose the environment into cells. The decomposed cells are connected through its adjacency graph

and an Eulerian tour is found to cover all the cells at least once in a sweeping or circular pattern to achieve complete coverage. A key point of Morse decompositions is that by choosing different Morse functions to define the slice that is swept through the space, different decomposition and coverage path patterns can be generated. A limitation of the Morse decomposition method is that it cannot handle rectilinear environments. This is because it is not possible to determine critical points in those environments which correspond to a change in the topology of the space [9]. The main objective of our approach is to overcome the limitations of Morse decomposition and create a novel 3D complete coverage path planning algorithm for rectilinear environments.



(a) Trapezoidal decomposition (b) Boustrophedon decomposition

Figure 2.2: A decomposition with less cells allows for shorter coverage paths.

An online approach for covering a continuous planar area with fixed sensor footprint attached to a mobile robot is proposed by Gabriel and Rimon [1]. The algorithm, called Spanning Tree Covering (STC), subdivides the work-area into disjoint cells (grid map) corresponding to the square-shaped tool, then follows a systematic spiral path. This path is generated by following a spanning tree of the grid map that the robot incrementally built using its onboard sensors. But these methods have limitations of not being able to guarantee complete coverage, owing to leaving some unoccupied and partially unoccupied cells uncovered due to sensor noise, robot localization error or randomness of the approach. The backtracking Spiral Algorithm [35], an extension to Spiral Spanning Tree Algorithm enables the mobile robot to cover every unoccupied cell and also all partially occupied

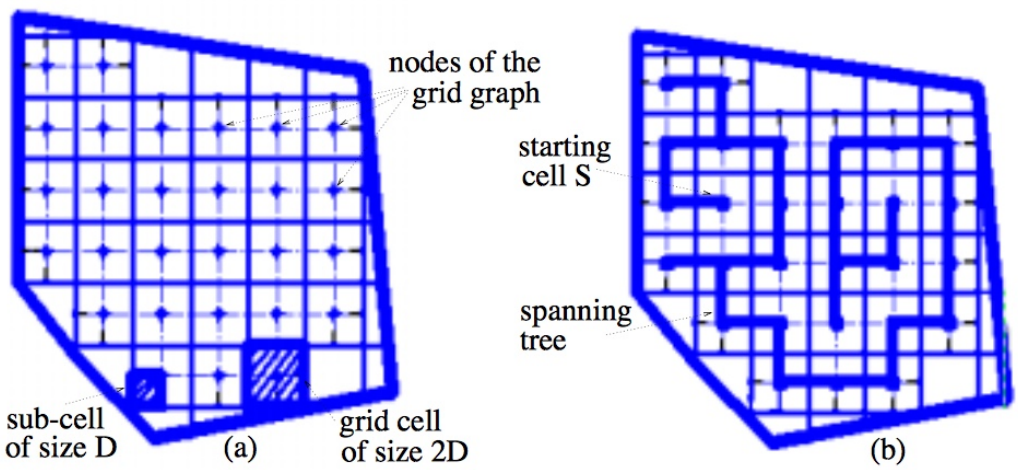


Figure 2.3: Spanning Tree Coverage [1]

cells. The idea in this extension is that the partially occupied cells are part of the external ring of the systematic spiral path and they are covered using a wall-following algorithm. Both STC and BSA are validated in simulation. Another on-line complete coverage path planning solution is proposed by Choi et al [36]. Their proposed algorithm uses a high-resolution grid map representation to reduce the number of turns on the coverage path.

In [28] Englot and Hover contributed a framework for analyzing the probabilistic completeness of a sampling-based coverage path planning algorithm and identified quantitative bounds on the probability of obtaining a feasible solution. They have developed an off-line, sampling-based coverage algorithm to achieve complete sensor coverage of complex 3D structures. The planning is performed in two steps. First, a graph of feasible paths for the robot is constructed using random sampling until the set of nodes of the graph allows complete coverage of the structure. Then, a minimum cost closed walk along the graph which fully covers the structure is searched in the graph. Their target application is autonomous ship hull inspection, in which the robot must cover the in-water part of the hull surface using a sensor such as sonar. The sensory data collected is later used to construct an accurate 3D model where anomalies in the hull surface can be sought. The limitation of their implementation is the size of underwater vehicle, which does not always fit into the spaces between the component structures at the stern. Most of the techniques

proposed in this paper are relevant to underwater vehicles and not directly transferable to aerial vehicles with more complex kinematic constraints in 3-D environments.

Recently researchers have also used UAVs for unmanned search and surveillance applications using multiple UAVs. Maza and Ollero proposed a technique to use multiple UAVs to solve a problem of cooperative searching a given area to detect objects of interest [37]. Their algorithms divide the area depending on the robots relative capabilities and initial locations. Partitioned areas are assigned to UAVs which cover the area in a zig-zag pattern. Here the aerial robots are heterogeneous each having different capabilities in terms of sensors and battery life. This approach is validated in simulation with three UAVs to search an area defined by a convex polygon with seven edges with no obstacles. The results show that the partitioned areas are being assigned to UAVs according to their relative capabilities. Each UAV finds the optimal sweep direction and does search operation in zig-zag motion. Another approach, based on the Cognitive-based Adaptive Optimization algorithm to solve the problem of deploying multiple UAVs to perform surveillance was proposed by Renzaglia et al [38]. They have mentioned that this approach addresses two main objectives a) maximize the area covered by robot and b) for every point in the terrain, the closest robot is as close as possible to that point. This paper mainly discusses deploying multiple flying robots at specific points over the terrain so that each robot would hover over its respective part of terrain and continuously monitor the surface. Though the terrain is complex, the robots does not cover them closely using its 3-D maneuver capabilities.

An approach for inspection of 3D surfaces that combines geometric processing with sampling-based motion planning was proposed in [39]. The objective of this paper is to compute a set of waypoints whose joint visibility ratio is at least $\alpha(0 < \alpha < 1)$ and a dynamically-feasible and collision-free trajectory that enables the aerial vehicle to reach all the waypoints. The waypoints are first generated by using random sampling or approximations of the medial axis via skeletonizing algorithms. This approach also seeks to minimize the number of the waypoints by applying visibility filtering mechanisms based

on a computation of a hitting set via Monte-Carlo search over an axis aligned bounding box obstacle tree. and the overall distance traveled by the aerial vehicle. After generating the waypoints, a multi-goal motion planning approach is applied to compute a collision free trajectory. This approach achieves visibility of the 3D structures and is suitable for surveillance applications but not suitable for complete coverage applications like inspecting structures for cracks or painting robots where the aerial robot is required to cover the complete structure closely at an offset distance.

Andreas Bircher et al presented a new algorithm for 3D coverage path planning for structural inspection operations using aerial robots in [40]. In this paper, the 3D structure to be inspected is represented as triangular mesh or a voxel-based octomap and is embedded into a bounded environment that may contain obstacle regions. It is assumed that for each triangle in the mesh, there exists an admissible viewpoint configuration from which the triangle is visible. An alternating two-step optimization algorithm is used at every iteration to find a new and improved set of viewpoints that together provide full coverage with decreased path cost. The algorithm does not focus on minimizing the number of viewpoints, therefore selects one admissible viewpoint for every triangle in the mesh of the structure to be inspected. As a TSP solver is employed to compute the best tour, with each viewpoint as a vertex in the graph, the complexity of this approach is very high compared to our approach where we consider each face of the structure as a vertex in the graph to compute TSP.

In [41], Breitenmoser, Metzger, Siegwart, and Rus proposed a solution to the problem of covering a non-planar surface in 3D space using a group of robots. They have designed two distributed coverage control methods that both divide the area into cells (i.e., homogeneous triangle mesh) using Centroidal Voronoi Tessellation and gives optimal initial locations for the robots on the surface in 3D space. In the first method, the coverage control algorithm minimizes the cost function to compute the shortest path by using the Lloyd algorithm in decentralized fashion. The second method called the local cell exchange

algorithm is based on Euclidean distances and minimizes the cost function to approach optimal robot configuration by locally exchanging mesh cells between Voronoi regions. Both these methods are implemented on ground robots to cover a non-planar surface in 3D space.

A new coverage scheme which studied complete 3-dimensional coverage with 2.5-dimensional features using a sensor attached to an UAV optimizing the time in trajectory planning was presented by Cheng et al., [42]. The authors assumed that the sensor with a conical field of view is installed on the bottom of the UAV and is able to rotate in 3 degrees of freedom around a fixed point. Next, 2.5-D urban features are approximated for the coverage surfaces using hemispherical and cylindrical primitives. This method simplifies the model to achieve complete coverage of 3-D urban buildings of different sizes. With this simplification it may not be possible to cover the buildings which are close to one another because of occlusions. Also this approach might not be able to extend to other complex structures like bridges, and telephone towers

Our proposed approach to solve structural inspection using 3D cellular decomposition is different from the previously mentioned techniques. Most of the approaches address the inspection problems by considering 1-dimension (that is, height or depth) as constant. These approaches are suitable for surveillance applications such as flood and wildfire monitoring, agricultural surveying but are not applicable to inspecting complex structures such as bridges and buildings. Some of the approaches use viewpoint sampling to achieve inspection of complex 3D structures. These techniques do not generate the trajectory to closely monitor the surfaces of the structure, as their objective is to achieve visibility of the structure. Also, computing a tour to visit each viewpoint leads to a high complexity algorithm. Our approach generates the trajectory that makes the UAV constantly maintain an offset distance close to the structure which is best suited for inspecting bridges, buildings, and other complex 3D structures. Our algorithm is also less complex compared to other 3D inspection algorithms as we plan a tour to connect 2D surfaces instead of

planning a tour to connect a large number of viewpoints.

Chapter 3

3D Inspection Structure Decomposition to Coverable Surfaces

3.1 Environment

We consider a quadrotor UAV deployed within an environment with target structures that need to be inspected. The UAV is capable of localizing itself within the environment [43] and uses its camera sensor to inspect the structures. The environment contains one or more structures at distinct locations that are to be inspected completely by the UAV. Our approach assumes that the 3D structures are enveloped by rectilinear surfaces at an offset distance from the target structure as shown in Figure 3.1. Every offset surface of the target structure is perpendicular to its adjacent offset surfaces. Therefore, the complete bounding structure takes the form of rectangular prisms overlapping with each other or protruding over other adjacent rectangular prisms. The offset enveloped structure of the target structure is shown with solid lines in Figure 3.1

Our objective in this thesis is to generate a collection of waypoints that the UAV could navigate through to perform complete coverage of the surfaces of the structure from a fixed offset. This problem is divided into four steps: i) given the 3D coordinates of the

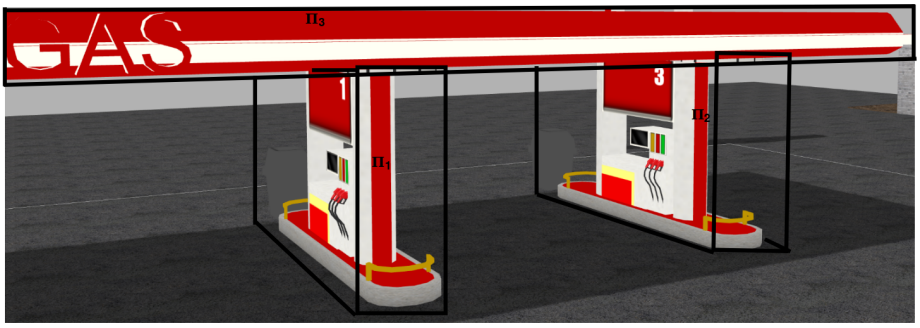


Figure 3.1: A sample target structure of a gas station that needs to be inspected. The solid black lines represent the envelpe around the structure

structure's extremities, determine the exposed surfaces or faces of the structure that need to be covered, ii) determine the adjacency constraints between the faces calculated in step i), iii) represent the faces and adjacencies between them using a graph-based representation from the adjacency information, and iv) use the graph to determine a coverage flight path for the UAV that completely covers the target structure while reducing certain metrics including the coverage time, number of orientation changes of the UAV, and total distance travelled by the UAV. We describe the first two steps in the following sections of this chapter and the latter two in the next chapter.

3.2 3D Cellular Decomposition

A target structure is represented as a set of 3D coordinate points $\{x_i, y_i, z_i\}$ representing its extremities. Each point is offset by a fixed distance d; the set of coordinate points for the enveloped target structure is given by $\{ (x_i \pm d), (y_i \pm d), (z_i \pm d) \}$. The entire enveloped target structure is composed of one or more rectangular prisms. Each prism, π_j, is represented as $\pi_j = \{ (x_j^{min}, y_j^{min}, z_j^{min}) , (x_j^{max}, y_j^{max}, z_j^{max}) \}$ where $(x_j^{min}, y_j^{min}, z_j^{min})$ and $(x_j^{max}, y_j^{max}, z_j^{max})$ are the minimum and maximum coordinates of prism π_j. Hence, the enveloped target structure is represented by a collection of rectangular prisms given by $\Pi = \{\pi_1, \pi_2, \ldots\}$. The coverage surfaces of the target structure are formed by the exposed portions of these faces and the UAV's path is planned on the offset, virtual surfaces on the

enveloped target structure corresponding to each of the original structure's surfaces.

The goal of our 3D decomposition algorithm is to decompose the enveloped target structure into faces, also called cells. For this, we extend 2D Boustrophedon cellular decomposition(BCD) to handle 3D surfaces. The input to the algorithm is the set of extreme coordinate points of the rectangular prisms bounding the enveloped target structure. The algorithm first converts each rectangular prism of the envelped target structure into a set of 2D cells. For this, a virtual 2D vertical plane $\mathcal{P}$, which is initialized with the minimum 2D coordinates $(y^{\text{min}}, z^{\text{min}})$ and maximum 2D coordinates $(y^{\text{max}}, z^{\text{max}})$ of the enveloped structure as described earlier. $\mathcal{P}$ is moved across the enveloped structure as shown in Figure 3.2.

$$y^{\text{min}} = \min_{j=1,...|\Pi|} ({y_{\text{j}}}^{\text{min}})$$

$$y^{\text{max}} = \min_{j=1,...|\Pi|} ({y_{\text{j}}}^{\text{max}})$$

$$z^{\text{min}} = \min_{j=1,...|\Pi|} ({z_{\text{j}}}^{\text{min}})$$

$$z^{\text{max}} = \max_{j=1,...|\Pi|} ({z_{\text{j}}}^{\text{max}})$$

As $\mathcal{P}$ moves through the enveloped structure, the connectivity changes on $\mathcal{P}$ are categorized into four events, as described below:

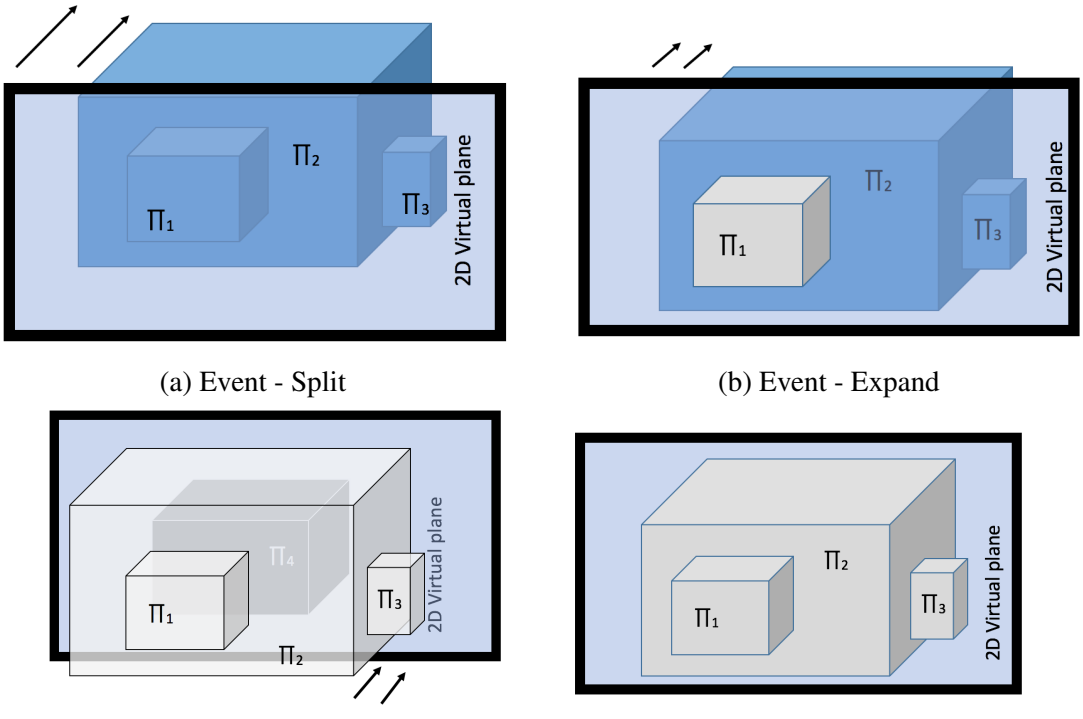
- $Split$, when $\mathcal{P}$ encounters the 2D faces of one or multiple prisms, rectangular holes are formed on $\mathcal{P}$. For example, as shown in Figure 3.2(a), a split event occurs on $\mathcal{P}$ when it encounters the left most surface of rectangular prism π_1

- $Expand$, when $\mathcal{P}$ meets the end of smaller prism which is protruding out of larger prism and encounters the starting face of the latter larger prism. For example, as shown in Figure 3.2(b), an expand event happens when $\mathcal{P}$ encounters the leftmost face of π_2 after sweeping through π_1. Another expand event happens when, while sweeping through π_2, $\mathcal{P}$ encounters the leftmost face of π_3

- *Contract*, when $\mathcal{P}$ meets the end of larger prism and encounters the starting face of smaller prism. For example, as shown in Figure 3.2(c), a contract event happens when $\mathcal{P}$ encounters the leftmost face of prism π_4 after sweeping through π_1, π_2, and π_3.

- *End*, when $\mathcal{P}$ meets the end of previous prism and does not encounter new prism. For example, as shown in Figure 3.2(d), an end event happens when $\mathcal{P}$ completes sweeping all prisms π_1, π_2, and π_3 and does not encounter a new prism.

These four events are used to identify the connectivity changes on the enveloped target structure and decompose it into individual prisms. Note that each prism can have six faces, but some of these faces might be overlapping completely or partially with a face of an adjacent prism. These faces are inaccessible and should not be further considered for coverage. We describe a technique to remove the inaccessible portions of faces while keeping the accessible portions suitable for coverage in the next section.



(a) Event - Split

(b) Event - Expand

(c) Event - Contract

(d) Event - End

Figure 3.2: A Virtual Plane sweeping through the target structure.

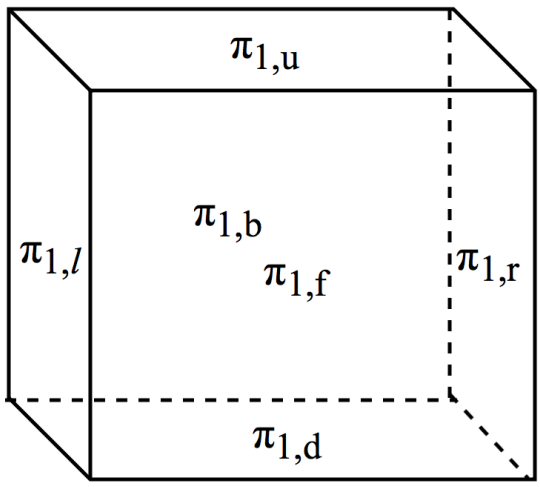


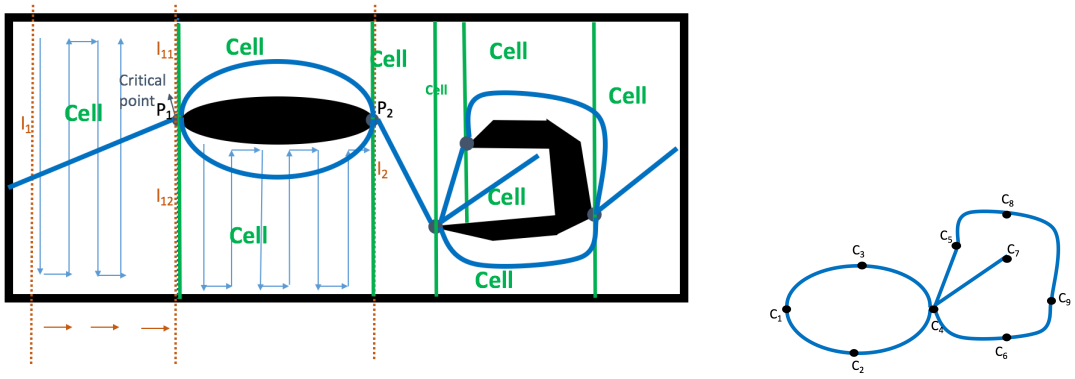
Figure 3.3: Rectangular prism showing six faces. The label of each face is shown at its center

3.3 Cell Overlaps and 2D BCD

Consequently, the enveloped structure is decomposed into 2D cells and these decomposed cells are stored in an adjacency graph. As shown in Figure 3.5, when two prisms are connected, faces can overlap with each other leading to certain portions of a face becoming inaccessible for coverage. This leads to connectivity changes in the UAV's flight path. To address this, we do 2D Boustrophedon Cellular Decomposition on an overlapped face, further dividing this face into sub-faces.

We refer to the six faces of a rectangular prism as front, left, back, right, up, and down respectively whose front is the first-encounterd face of a prism while decomposing the structure. For example, the faces of π_1, are denoted by $\pi_{1,\mathrm{f}}$, $\pi_{1,\mathrm{l}}$, $\pi_{1,\mathrm{b}}$, $\pi_{1,\mathrm{r}}$, $\pi_{1,\mathrm{u}}$, and $\pi_{1,\mathrm{d}}$ as shown in Figure 3.3.

Consider the overlap between the faces of π_1 and π_2 shown in Figure 3.5 π_1 is protruding from π_2, and $\pi_{1,\mathrm{b}}$ is overlapped over $\pi_{2,\mathrm{f}}$. In this case, $\pi_{1,\mathrm{b}}$ is completely overlapped by a part of $\pi_{2,\mathrm{f}}$ and cannot be covered. Correspondingly $\pi_{2,\mathrm{f}}$ has a rectangular portion that is overlapped by $\pi_{1,\mathrm{b}}$, equal to the area of $\pi_{1,\mathrm{b}}$, which also cannot be covered. These regions are considered as overlapped cells, and marked in black in Figure 3.5.



(a) Vertical line sweeping through environment

(b) Decomposed 2D cells connected in Reeb graph

Figure 3.4: 2D Boustrophedon Decomposition with cells and tour

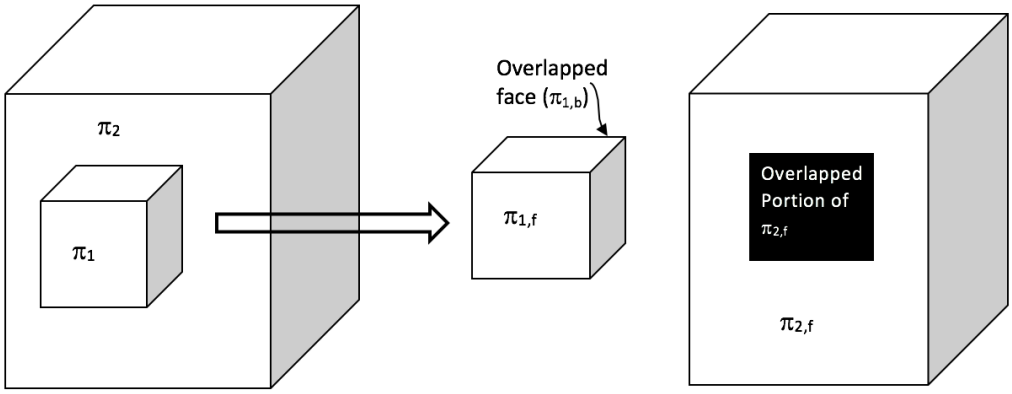


Figure 3.5: Prism1 creates an overlapped portion on Prism2.F face

Therefore, it is treated as an obstacle while decomposing the face $\pi_{2,f}$.

The BCD algorithm is applied on a face with overlapped portions to divide its free space into rectangular cells. The union of such cells will cover the free space of the face. As mentioned in Section 2.2, in BCD, the free space in the environment is partitioned into cells. The input to the BCD algorithm is a grid-based bitmap representation of the environment where obstacles are represented by 1-bits and free space by 0-bits. A virtual vertical line l is moved through the map of the environment along the x-axis. When l encounters an obstacle (e.g. at point P_1 in Figure 3.4(a)), its connectivity changes as it splits into two segments l_{11} and l_{12}. The location at which l's connectivity changes is called a *critical point*. Similarly, while moving l to the right when two line segments merge

(e.g. at point P_2), there is, once again, a change in connectivity of l, and this location is again recorded as a *critical point*. The vertical lines passing through the critical points and their endpoints on the boundary of the environment or obstacles form the boundaries of the decomposed cells. The union of these cells cover the free space in the environment. The decomposed cells and the boundaries between adjacent cells are stored respectively as vertices and edges of a Reeb graph (Figure 3.4(b)) denoted by $G_{\mathrm{r}} = \{V_{\mathrm{r}}, E_{\mathrm{r}}\}$where V_{r} is a non-empty set of vertices that correspond to the critical points and E_{r} is a set of edges. The pattern in which the edges meet at vertices reflect the changes in connectivity of the virtual verticle line.

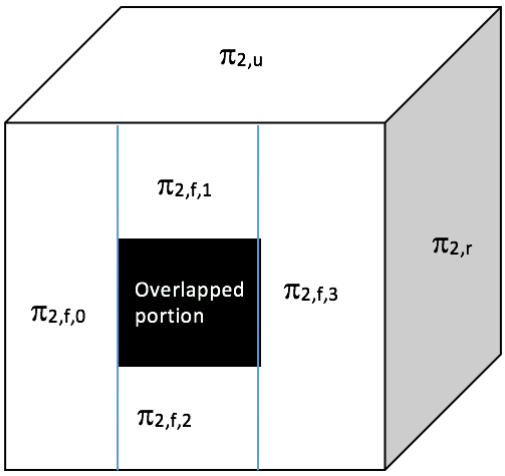


Figure 3.6: 2D Boustrophedon decomposition on prism2.F

In the Figure 3.6, after applying the 2D Boustrophedon cellular decomposition on the face $\pi_{2,\mathrm{f}}$ the decomposed sub-faces are labelled as $\pi_{2,\mathrm{f},0}$, $\pi_{2,\mathrm{f},1}$, $\pi_{2,\mathrm{f},2}$, and $\pi_{2,\mathrm{f},3}$. These sub-faces are also added into the graph to plan a tour for completely covering all the faces and sub-faces. In the next chapter we discuss how 2D faces and sub-faces are stored in a graph, how to remove the overlapped cells, and how to find an exhaustive path to cover every face and sub-face at least once.

Data: $Cuboid[\,]$ derived from (x, y, z) cordinates.
Result: $cells[\,]$ containing decomposed cells from environment.

```
1  begin
2      z ⟵ min_z(Cuboid[])
3      cells ⟵ [ ]
4      current_cells ⟵ [ ]
5      G ⟵ empty graph
6      while z < max_z do
7          if z intersects cuboid then
8              current_cells[] ⟵ face_of_cuboid
9          end
10         foreach face ∈ current_cells do
11             if cuboid of face ends then
12                 cells.add(face)
13                 cells.add(face.create_faces())
14                 current_cells.delete(face)
15                 r = new Node(face, area)
16                 add_node(G, face)
17                 opened_face_nodes = new Node(opened_faces[], area)
18                 adjacent(G, r, opened_face_nodes, ⊥ edgeWeight)
19             end
20         end
21         increment z
22     end
23     foreach cell ∈ cells[] do
24         if cell has overlap then
25             boustrophedon_cells[] = cell.boustrophedon_decompose()
26             b_nodes[] = new Node(boustrophedon_cells[], area)
27             if node1 is adj to node2, ∀node1, node2 ∈ b_nodes[] then
28                 newEdge(node1, node2, bousWeight)
29             end
30             foreach node ∈ overlapped_cubiod do
31                 if node shares edge with node_b ∈ b_nodes[] then
32                     newEdge(node, node_b, ⊥ edgeWeight)
33                 end
34             end
35         end
36     end
37 end
```

Algorithm 1: 3D Cell Decomposition

Chapter 4

Graph Representation of Decomposed Surfaces and Traversal

In the previous chapter, we proposed a technique to decompose a 3D enveloped target structure into 2D faces and sub-faces by extending 2D BCD to handle 3D surfaces. In this chapter, we propose an approach to connect the decomposed faces and sub-faces in a graph-based respresentation to determine a coverage tour for the UAV that completely covers the enveloped target structure. We do this by connecting the faces of each prism in a graph, based on their adjacency and then establish connection between the prisms in a graph through the 2D sub-faces that are decomposed because of an overlap.

4.1 Map Decomposed Surfaces to Vertices

As discussed in Chapter 3, the enveloped target structure is bounded by the rectangular prisms. When the virtual plane sweeps through the bounding workspace, it decomposes the prisms into 2D surfaces called faces. Each prism has six faces and each face is connected to four other faces of the same prism through shared edges. We map these decomposed faces of each prism to the vertices in a graph data structure, to yield one cyclic graph for each prism.

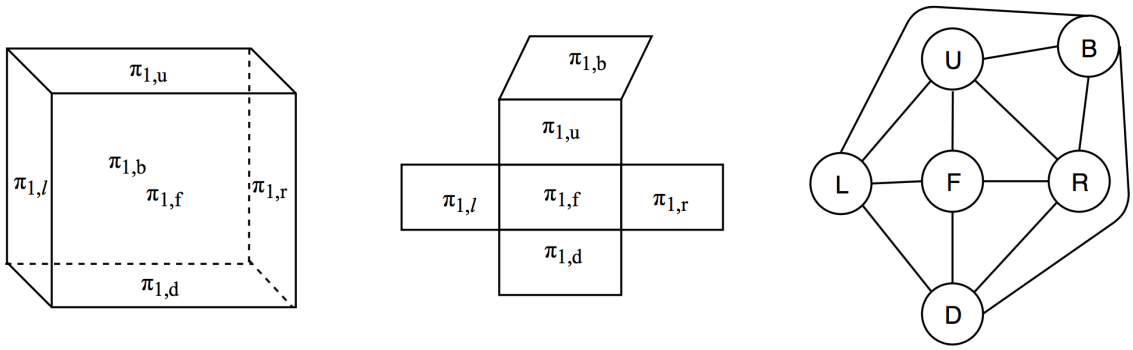


Figure 4.1: A Prism decomposed into 6 faces

As shown in Figure 4.1, the faces of each prism are mapped to the vertices in an undirected graph data structure. Two faces u and v in an undirected graph G are adjacent in G if u and v share same edge e of G. Such an edge e is called incident with faces u and v and e is said to connect the faces u and v. The degree of a vertex in an undirected graph is the number of edges incident with it. In this case, each face is connected to four other faces therefore the degree of each vertex is four. For example, the front face F is connected to faces left L, up U, right R, and down D. The back face B is opposite to the front face F, hence it is not connected to F. Because the graph underlying the face connectivities in the target structure is sparse, we use an adjacency list to represent it.

Vertex	Adjacent Vertices
F	L, U, R, D
L	F, U, D, B
U	B, L, F, R
R	D, U, B, F
D	F, L, B, R
B	R, U, L, D

Table 4.1: Table to show adjacency list representation

The target structure is composed of one or more prisms and each prism can be mapped to an undirected graph connecting its faces. Each prism is mapped to an isomorphic graph as they have the same structure when we ignore the identities of their vertices. These isomorphic graphs have one-to-one correspondence between vertices that preserves the

adjacency relationship. In the next sub-sections, we discuss how these isomorphic graphs representing prisms are connected to each other to form a final graph of accessible faces of the enveloped target structure

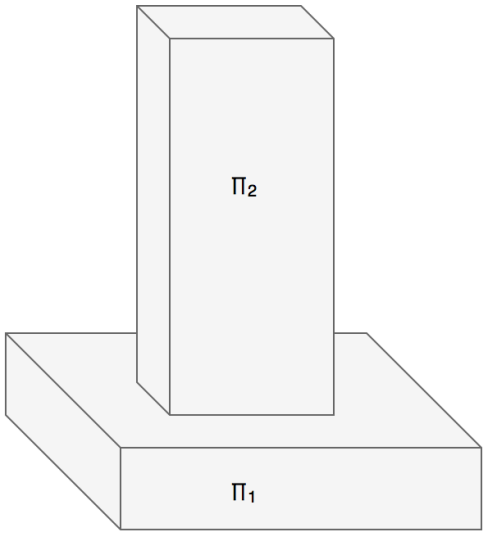
4.2 Identify Overlapped Vertices

Recall that enveloped target structure can be comprised of multiple prisms when the target structure has protrusions. If the enveloped target structure corresponds to a single rectangular prism, then the corresponding coverage graph is a simple graph connecting each vertex (face) to four adjacent vertices with six vertices and twelve edges. But in case of more complex structures, where the enveloped target structure is composed of multiple prisms, we have to connect the graphs corresponding to each prism to its adjacent prisms at the appropriate adjacent faces while eliminating overlapped faces, if any. For example, consider the target structure shown in Figure 4.2. In this case, prism π_2 is protruding from prism π_1 that is, the prisms are connected. We need to establish the connection between two isomorphic graphs to represent the enveloped target structure as a single graph.

We can simply establish an edge between the π_2's down face $\pi_{2,\mathrm{d}}$ and π_1's up face $\pi_{1,\mathrm{u}}$ to connect two graphs but it is evident that $\pi_{2,\mathrm{d}}$ and a portion of $\pi_{1,\mathrm{u}}$ are not accessible. Hence, the vertices representing these faces should be removed from the final coverage graph.

As shown in the Figure 4.3(a), $\pi_{1,\mathrm{u}}$ face has an overlapped portion which is of area equal to the area of $\pi_{2,\mathrm{d}}$ face. As discussed in Section 3.3, we decompose $\pi_{1,\mathrm{u}}$ into non-overlapping sub-faces. The face $\pi_{1,\mathrm{u}}$ is decomposed into four sub-faces and labelled them as $\pi_{1,\mathrm{u},0}$, $\pi_{1,\mathrm{u},1}$, $\pi_{1,\mathrm{u},2}$, and $\pi_{1,\mathrm{u},3}$ as shown in Figure 4.3(a). The BCD algorithm also stores the decomposed sub-faces and boundaries between adjacent sub-faces as vertices and edges respectively in a reeb graph (Figure 4.3(b)). In the next section, we describe how reeb graph is connected to graphs representing π_1 and π_2.

(a) Target Structure in Real World



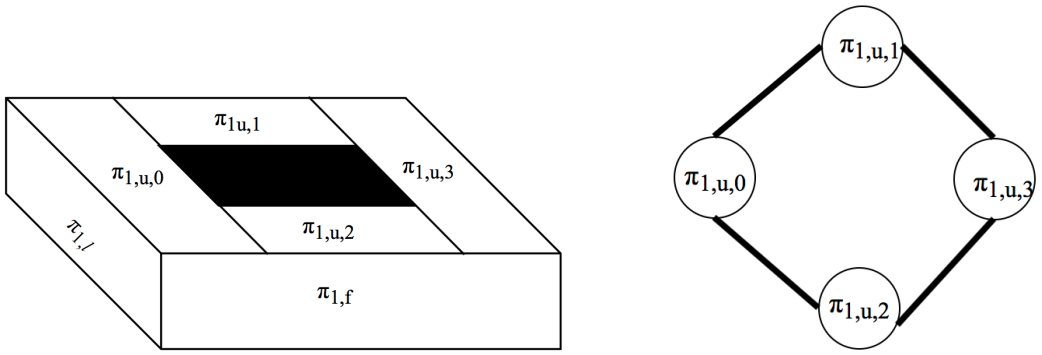
(b) Representing Target structure with Prisms

Figure 4.2: A Sample Real world target structure.

4.3 Connecting Prisms in Final Graph

As shown in Figure 4.2(b), π_1 and π_2 are adjacent to each other. These two prisms intersect on $\pi_{1,u}$ where π_2 is protruded from π_1. As discussed in the section 4.2, the face $\pi_{1,u}$ is decomposed into sub-faces and a reeb graph is generated connecting these sub-faces.

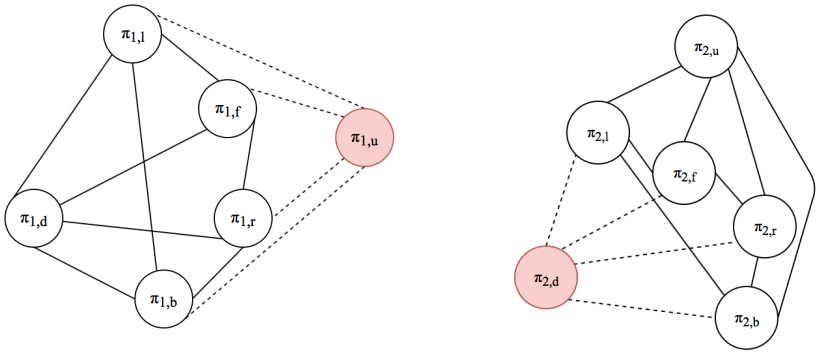
In our sample enveloped target structure 4.2(b), $\pi_{2,d}$ face is completely overlapped and it is inaccessible. Therefore, we do not include $\pi_{2,d}$ in the graph representation of π_2 as shown in Figure 4.4(b). In case of $\pi_{1,u}$ face, we decomposed it into 2D sub-faces as it has a portion of area which is inaccessible. Hence, we remove the vertex corresponding to the face $\pi_{1,u}$ as it does not exist as shown in Figure 4.4(a). After removing the vertices corresponding to inaccessible faces from their respective graphs, in this case we have a graph with three connected components where two components represent π_1 and π_2, and one more component represents the reeb graph connecting sub-faces as shown in Figure 4.5.



(a) 2D Decomposition of $\pi_{1,u}$ face and black portion represents uncoverable area

(b) Reeb graph connecting sub-faces

Figure 4.3: 2D Decomposition into sub-faces and connecting them in graph

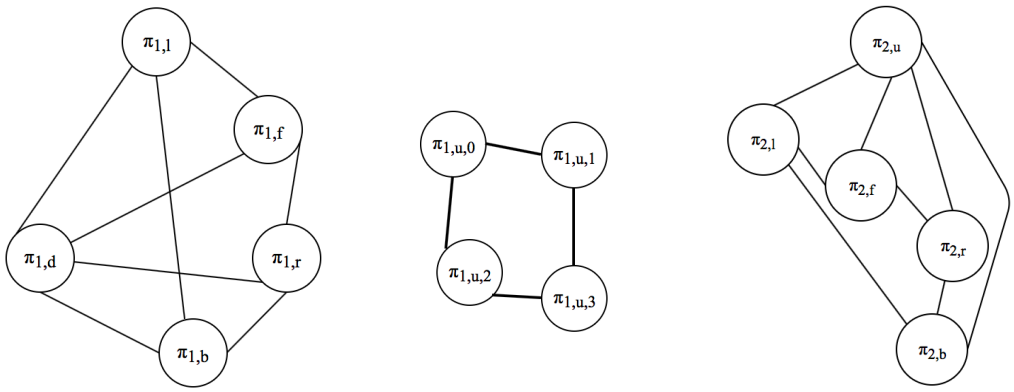


(a) Graph representation of π_1

(b) Graph representation of π_2

Figure 4.4: Graph showing removed edges to delete inaccessible faces

In order to cover sub-faces, they should be included in the coverage graph. Therefore we need to connect the reeb graph to the graphs representing π_1 and π_2. Each sub-face from the reeb graph also shares its boundaries with faces of its own prism, or an adjacent prism, or both. For example, the sub-face $\pi_{1,u,0}$ shares its boundaries with three faces of π_1, that are $\pi_{1,f}$, $\pi_{1,l}$, and $\pi_{1,b}$, one face of π_2, that is $\pi_{2,l}$, and two sub-faces $\pi_{1,u,1}$ and $\pi_{1,u,2}$ that are adjacent to it in the reeb graph. Similarly, every sub-face shares at least two of its boundaries with two different prisms. Therefore we connect π_1 and π_2 through the sub-faces. Hence, the reeb graph representing connectivity of sub-faces, acts as a bridge between the graphs representing π_1 and π_2. Based on this criteria, we construct the coverage graph by connecting graphs representing π_1, sub-faces, and π_2



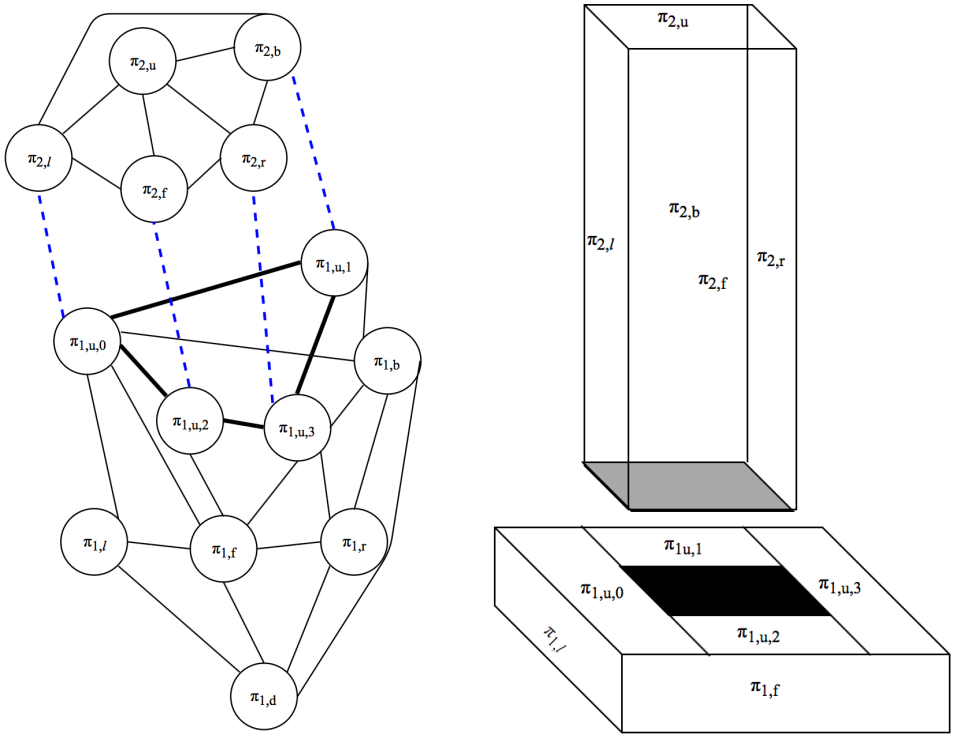
(a) Graph representation of π_1 (b) Reeb graph connecting sub-faces (c) Graph representation of π_2

Figure 4.5: Graph showing connected components representing π_1, reeb graph, and π_2

. After constructing all the edges to connect π_1 and π_2 through the sub-faces, the final coverage graph built is shown in Figure 4.6(a)

4.4 Constructing Tour

In the previous section, we have constructed an undirected graph $G = \{V, E\}$, where V represents the set of decomposed faces and E represents the set of edges or boundaries that are shared between adjacent faces. The edge weight connecting two faces is the distance between them. If $V = \{v_1, v_2, ..., v_n\}$ represents vertices or faces in the graph, and then weight c_{ij} is the distance between the center of the face v_i and face v_j. The final graph is an undirected graph in which the edges are symmetric that is c_{ij}=c_{ji}. Now we have the final undirected weighted graph and need to construct a tour such that the UAV covers all faces using the shortest path available and covering each face exactly once. As discussed in section 1, Complete Coverage Path Planning problem is an instance of Traveling salesman problem that requires constructing tour through the graph vertices- a well known NP-hard problem. Our approach is to find the shortest tour connecting all the faces on the final graph using TSP. For number faces n, the number of paths that must be explored to find the shortest one are $(n-1)!$. Thus this problem grows exponentially with the number of



(a) Final graph representing the target structure 4.2(a)

(b) Enveloped target structure after 3D decomposition

Figure 4.6: Final graph represending decomposed enveloped target structure

faces. Since we are decomposing the environment into faces instead of finding numerous viewpoints proposed in other approaches, we have optimized the complexity of TSP by reducing the number vertices in the graph.

In order to use TSP, the final graph should have the following properties: It should be a complete graph. That is, for all the vertices in the graph, there should be an edge connecting every pair of vertices. It should have a Hamilton circuit, a circuit or cycle that connects every vertex in the graph. TSP is defined as the problem of finding an optimal Hamilton circuit in a complete graph. Since our final graph is not a complete graph, there is no guarantee that a Hamilton circuit exists in the final graph. In our final graph, an edge exists only between the faces that are physically connected in the real world to minimize the number of changes in orientation for a UAV to move to the next face. Adjacent faces in the final tour connect to each other in two ways. One way is when these faces co-exist

side-by-side in the same surface as $\pi_{1,u,0}$ and $\pi_{1,u,2}$ shown in Figure 4.3(a). Another way is when one face is perpendicular to the other face as $\pi_{1,u,2}$ and $\pi_{1,f}$. Therefore the number of turns to reach the next face in the UAV trajectory is reduced to 0 or 1 in more than 90% of the cases.

Below are some of the techniques to solve TSP to find an optimal or near optimal solution.

- Backtracking, in which initially a current best tour is computed greedily and then systematically all possible tours are generated by rejecting bad tours (that is, when the tour longer than the current best tour) to find an optimal solution.

- Brute force method, where the total number of possible tours are computed and shortest tour is selected as final tour. This technique gives an optimal solution, but it is not efficient.

- Greedy approach, in which the nearest vertex is always visited and then returning to the starting vertex when all the vertices have been visited once. This approach is also called the Nearest Neighbor Method.

- Simulated Annealing, in which a near optimal solution is found by always accepting tours with reduced length and accepting tour that increased from previous tour, only with some probability. This technique avoids falling into a local minima and provides near optimal solution within short time.

We have constructed two weighted graphs for each target structure, one graph $\mathcal{G}_{\text{dist}}$, with edge weights equal to the distance between the faces, and another graph $\mathcal{G}_{\text{area}}$, with edge weights equal to the area of the adjacent face. Using these two graphs, we find two flight coverage paths. One approach is to find the shortest tour connecting all the faces on graph $\mathcal{G}_{\text{dist}}$ using TSP. This approach minimizes the face to face distance by choosing the nearest adjacent face. It also reduces the number of turns of the UAV because the next nearest

face to the current face is always physically adjacent on the enveloped target structure. Hence, in order for the UAV to go to next face, it can take a maximum of one turn.Another approach is greedy. Hence, it finds flight coverage path using $\mathcal{G}_{\text{area}}$ that maximizes the area of coverage. That is, it chooses the largest area first. So our goal is to solve an optimization problem where we have to find one solution with minimizing distance of tour and another with maximizing area.

One more important optimization criteria is to choose a coverage pattern for the UAV to completely cover each face of the enveloped target structure. We used Lawnmower(zig-zag) pattern to cover the surface of each face. The width between the zig-zag lanes should be equal to the UAV's sensor footprint w , which is given as an input to the algorithm so that it can be varied based on the sensors that the UAV uses. Now the UAV can cover the face in back and forth motion along the lanes perpendicular to the sweep direction. The time to cover each face is sum of time to travel along the lanes and the time taken to hover and start at the end of each lane before moving to the next lane. Hence, choosing a consistent sweep direction to cover all faces is not efficient as it would result in large number of waypoints as shown in Figure4.8. The UAV makes a stop at each waypoint and then starts to reach next waypoint. Lawnmower pattern's efficiency greatly depends on the number waypoints, so it is important to find the optimal sweep direction for each face. We chose a sweep direction that is perpendicular to the longest edge of the face as shown in Figure4.7. Covering faces with right sweep direction has decreased large number of waypoints per face.

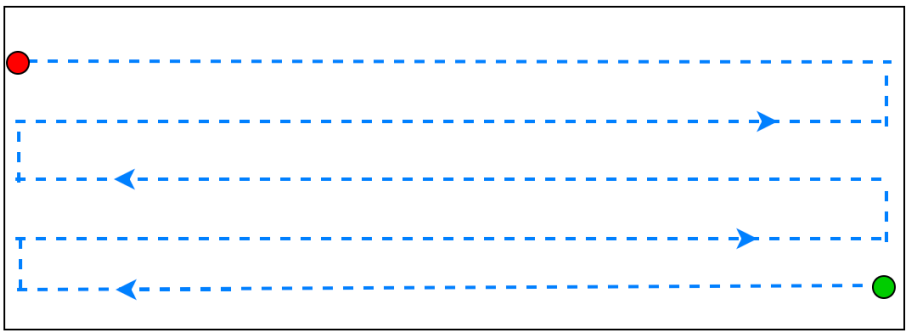


Figure 4.7: Zig-zag motion with sweep direction along the longest edge

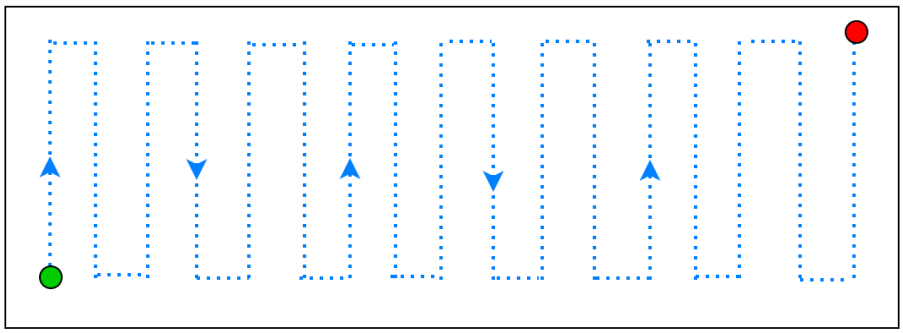


Figure 4.8: Zig-zag motion with large number of turns to cover same area

We verify our 3D structural decomposition approach by comparing the TSP-based coverage traversal with respect to Largest Area First(LAF) greedy approach. In the next chapter, we discuss the performance of TSP and Greedy approaches with respect to our 3D structural inspection algorithm.

Chapter 5

Experimental Results

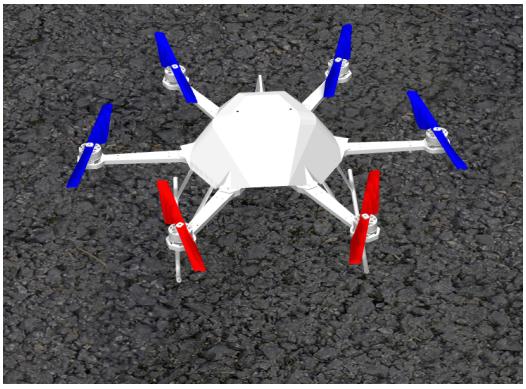
To validate the suitability of our proposed 3D cellular decomposition and coverage approaches for inspection problems, we performed a series of experiments within a simulation environment with a variety of target sturctures. For each experiment, we recorded the target structure's percentage of surface covered, and the performance of our algorithm for different algorithm parameters including the velocity of the UAV and the time spent at each waypoint to analyze the sensory data. The following sections describe how the experiments were constructed and performed, and summarize the main experimental results.

5.1 Simulated Experiments

5.1.1 Setup

The simulated experiments were conducted on the RotoS Simulator using an accurate model of autonomous UAV called the AscTec Firefly Hexacopter. RotorS is an open source simulator based on Robotic Operating System (ROS) and Gazebo simulator. It supports advanced physics engines that enable accurate simulations for modeling UAVs, sensors, actuators and the environment. It provides a 3D view of the environment and also

allows the user to build custom models in the environment. All the components in RotorS for the Firefly UAV were designed to be analogous to its real world counterpart. This will enable use of the same algorithm, code, parameters, controllers and state estimators in the simulation as well as on the physical UAV. The simulator was run on Intel i7 8-core CPU running at 3.2 GHz machine using Ubuntu 14.04 and ROS Indigo. The UAV is equipped with inertial measurement unit(IMU), generic odometry sensor, acceleration sensor, gyroscope, camera, barometric pressure sensor, and a GPS sensor. The IMU, acceleration, gyroscope, and odometry sensors together allow the robot to know its current location and orientation inside the environment. The barometric pressure sensor provides the UAV with altitude information. The data from all the sensors is fused into an Extended Kalman Filter (EKF) [44] to get the state (3D location) and pose(orientation) estimates of the UAV.



(a) A simulated Firefly UAV in RotorS Gazebo Simulator

(b) Real Firefly UAV

Figure 5.1: AscTec Firefly UAV.

The simulated and actual Firefly UAVs are shown in Figure 5.1(a). The environment is 50 meters x 50 meters and contained five different target structures given in Table 5.1. The environment layout including some of the sample target structures like houses, gas-station, store, and tower can be seen in Figure 5.2. The UAV starts at the center of the environment (at coordinates (0,0,0)) before starting to inspect the target structures. We also set the UAV's sensor foot print to 1 meter such that the UAV covers the enveloped target

structure in zig-zag pattern with a distance of 1 meter between the lanes.

Figure 5.2: Environment layout in RotorS Gazebo Simulator

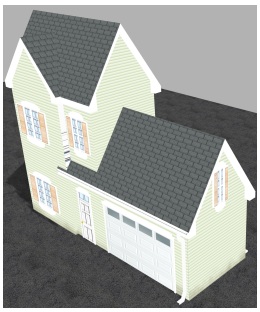
The 3D decomposition approach is verified by comparing the traversals planned on the coverage graph. We computed the TSP traversal that minimizes the distance between faces and compared it with a Largest Area First and Nearest Neighbor First traversal approaches. The 3D decomposition technique is the same across the three traversal approaches. Consequently, the faces in the graph-based representation remain same in all the tours. For each structure, the following steps are performed:

- The 3D decomposition algorithm decomposes the enveloped target structure into 2D faces and sub-faces, connect them in a coverage graph-based representation.
- A tour is planned to visit each face at least once.
- Waypoints are generated following a zig-zag pattern to cover the surfaces of all faces.
- These waypoints are given as an input to the UAV so that it completely covers all the surfaces of the enveloped target structure by traversing those waypoints.

The input to the TSP and nearest neighbor traversal algorithms is a weighted graph where each edge's weight is proportional to the distance between the faces connected by these edges. For the largest area first greedy traversal technique we use a weighted graph with face weights, where face weight is proportional to area the face. From the experiments we collected the following data:

Target Structure	Total Area	No. of Faces
House1	218.09	11
House3	347.30	19
Store	411.59	12
Gas Station	510.37	19
Name Board	132	6

Table 5.1: Details of target structures

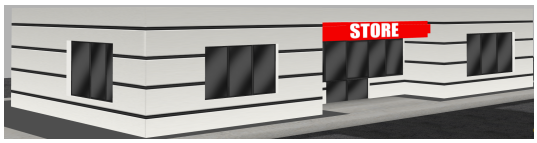


(a) House1

(b) House3

(c) Name Board

(d) Gas Station



(e) Store

Figure 5.3: Target Structures.

- Environment parameters
  - Number of faces
  - Average face area
- Algorithm performance metrics
  - Total Distance covered
  - Total Time taken
  - Total number of turns

Each metric reveals a different aspect of the performance of our proposed approach. The number of faces and average face area are both metrics that are used to capture the

complexity of the structure. Average face area is inversely proportional to the complexity of the structure i.e., the structures with low average face area is more complex compared to the structures with high average face area. The distance traveled and coverage time are both conventional metrics used for measuring the performance of robotic coverage algorithms. The distance traveled during coverage is a good indication of the energy used, because the most energy intensive task for the UAV is to spin its rotors. The time taken for coverage also provides an indication of energy required to completely cover the target structure but, unlike distance traveled, the time taken also gives us a measure of number of stops the UAV has made to take turns or change its orientation. The total number of faces is also a metric that is used to quantify the algorithm's performance. The energy consumed and time taken for coverage are proportional to the total number of turns.

5.1.2 Results

Figure 5.4 shows the trends in distance covered by the UAV as the number of faces increases for each enveloped target structure. In the TSP approach, the algorithm selects the closest face by minimizing the total distance of the tour. Hence Figure 5.4 shows no pattern in distance traveled to cover each face . But in the greedy approach the algorithm selects the face with the largest area first. Hence Figures 5.4(a)-(d) clearly shows that the distance traveled to cover each face gradually decreases as the number of faces increases. With the greedy approach, on average, 50% of the enveloped target structure is covered just by visiting less than one-third of the total number of faces.

In Figure 5.4, we also show the total distance traversed to cover all the faces for both approaches. The greedy approach takes an average distance that is 10% more than the total distance traversed by using TSP approach. This is because, in the TSP approach, the next face is selected such that it is adjacent to the current face, while in the greedy approach, the next face is not always adjacent to the current face. While using the latter, the UAV travels more to move from one face to other. We also observed that, in both the approaches, the

enveloped target structures of House3 and Gas Station have an equal number of faces (20), but, the total distance traveled to cover House3 is less compared to Gas Station. This shows that, House3 is a more complex and compact structure as compared to Gas Station.

We compare our three graph traversal approaches in terms of the tour length to visit each face at least once for five enveloped target structures. For the narrow and small structures like House1 where opposite surfaces are separated by a small distance (i.e., the distance between front and back surfaces is small), the TSP's tour length is 10% lower than the Nearest Neighbor and 20% lower than the Largest Area First approaches. For bigger structures like House3, Gas Station, and Store, the tour length computed by TSP and Nearest Neighbor differs by a small amount and they produce on average a 50% shorter tour as compared to the Largest Area First approach.

Figure 5.6 shows the time metric, in terms of time taken to cover one square-meter of area vs average face area. For example, for enveloped target structure House3, the UAV took 1.34 seconds to cover one square-meter of area. This plot shows the effect of the average face area on the duration of coverage. The target structures with a low average face area require more time for coverage, as compared to the target structures with a high average face area. We observed that the greedy approach requires 12% more time to cover every square meter of area when compared to the TSP approach.

As the UAV covers the area of an enveloped target structure by traveling in a zig-zag motion, we measured the distance that the UAV needs to travel to cover one square-meter of area, as shown in Figure 5.7. The structures with large average face area require less distance to cover one square meter of area because the ratio of the distance traveled for coverage to the distance traveled between faces is high for these structures. Our results show that largest-area-first approach wastes on average 10% of the total distance traveled because it makes repetitive visits to already covered faces while reaching the next uncovered face.

For a rotorcraft UAV, in order to change its orientation (i.e., to take a turn), it needs

to hover and adjust its orientation toward the specified direction. Therefore, the number of turns is proportional to time as well as to energy. Figure 5.8 shows the comparison between TSP and Greedy approaches in terms of the number of turns the UAV has to take in order to cover the enveloped target structures. Using TSP, the number of turns for each structure is not greater than the number of faces that the target structure has. This is because TSP selects the next face which is either on the same surface or on the surface perpendicular to the current surface, thereby reducing the number of turns. Greedy approach takes an average of 30% more turns as compared to TSP. This increases its energy consumption and also its total time taken to cover the enveloped target structure

Figure 5.9 shows covered area of each face as time increases. For example, in Figure 5.9(a), considering Greedy approach, the UAV took 0 to 32 seconds to cover an area of 30 square-meters, and 32 to 77 seconds to cover an area of 41.25 square-meters. These areas correspond to individual faces of the House1 target structure. The UAV has covered two faces within 77 seconds and from 77^{th} second to 85^{th} second, during which period the UAV's effort is not utilized for coverage purpose. Instead, it is wasted on travel to reach the next face. In the greedy approach, the next face is not guaranteed to be adjacent to the current face. Hence, the UAV has to cross many faces to reach the next face to cover. In Figure 5.9, every dip in the plot to zero is an indication that the UAV's effort is wasted during that period of time. In contrast, in TSP, the next face is guaranteed to be adjacent to the current face. Hence, there is no waste of time during the coverage.

Target Structure	Total Area	No. of Faces	Avg. Area, σ
House1	218.09	11	24.2, 10.2
House3	347.30	19	17.36, 12.1
Store	411.59	12	31.66, 19.63
Gas Station	510.37	19	25.59, 24.8 5
Name Board	132.5	5	26.5, 11.8

Table 5.2: Total area, number of faces, average face area, standard deviation for each target structure)

Target Structure	Total Distance	Total Time	Distance b/w faces	Number of Turns
House1	287.14	3:58	26.87	10
House3	486.11	7:45	57.43	12
Store	493.12	5:40	55.08	11
Gas Station	621.56	6:53	68.75	12
Name Board	158	1:40	12.18	5

Table 5.3: Metrics - TSP

Target Structure	Total Distance	Total Time	Distance b/w faces	Number of Turns
House1	294.14	4:15	35	13
House3	534.11	8:42	132.74	16
Store	557.12	6:40	121.51	12
Gas Station	683.56	8:26	143.08	25
Name Board	162	1:48	15.43	7

Table 5.4: Metrics - Largest Area First

Table 5.2 shows total area, number of faces, average area, and standard deviation of area for each target structure used in our experiments. The target structures (House3) with small total area and a larger number of faces are more complex compared to the structures (Store) with large total area and smaller number of faces. Table 5.3 and Table 5.4 shows the results from the simulation experiments using the TSP and the largest-area-first approaches respectively. The time is measured in minutes and tour length is the sum of distances between faces in final tours. Both tables show that for each target structure, TSP-based coverage perform better than largest-area-first approach.

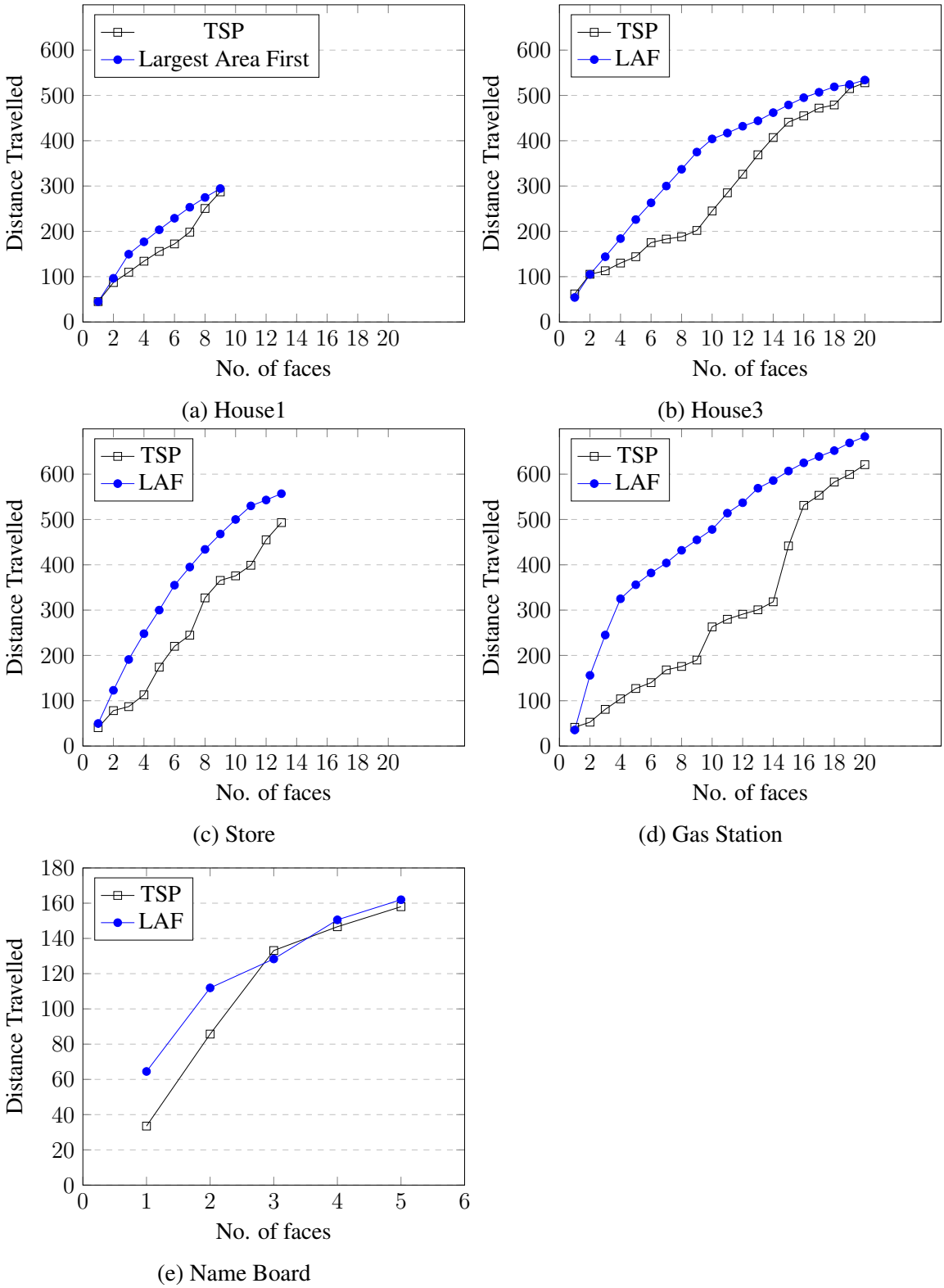


Figure 5.4: Distance Trends as Number of faces increase using Greedy Approach

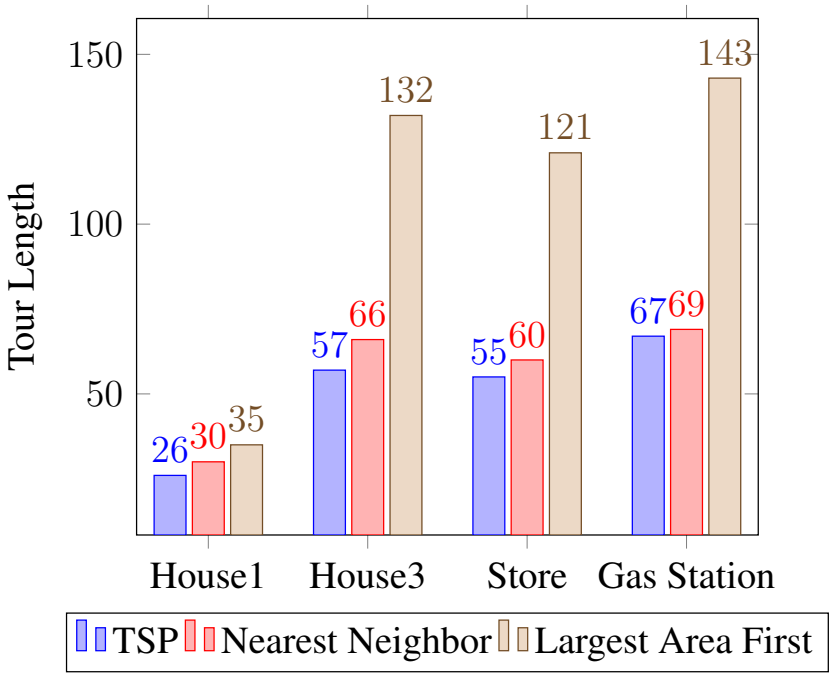


Figure 5.5: TourLength to visit all faces calculated for three traversal algorithms

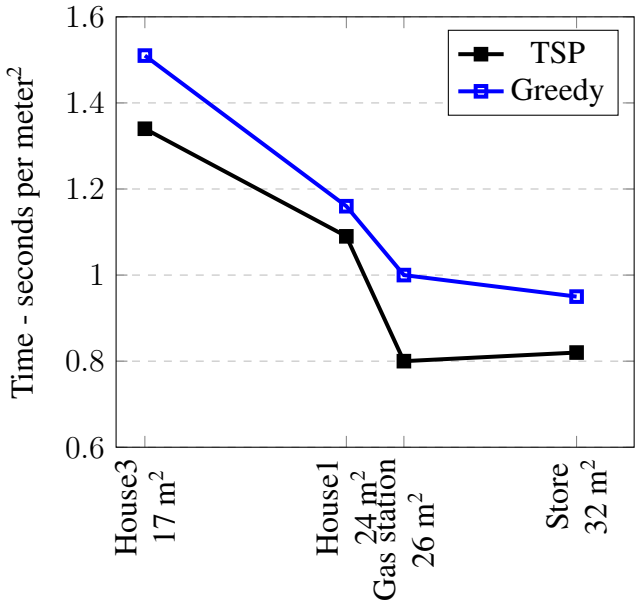


Figure 5.6: Total Time normalized over total area vs Average Face Area

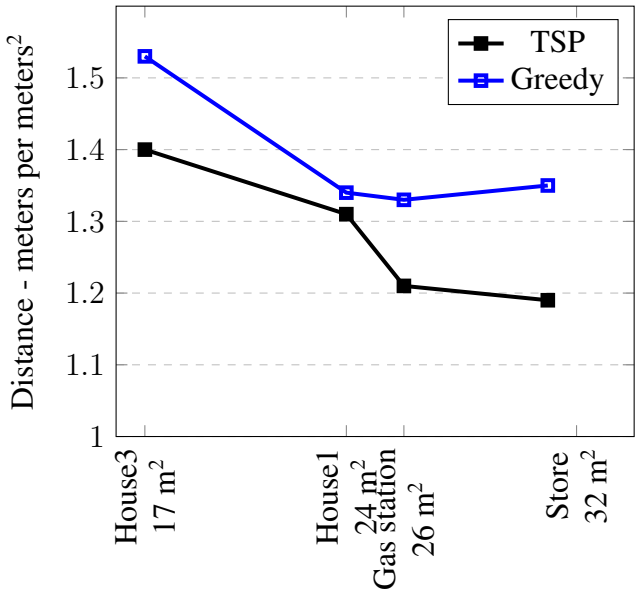


Figure 5.7: Total Distance normalized over total area vs Average Face Area

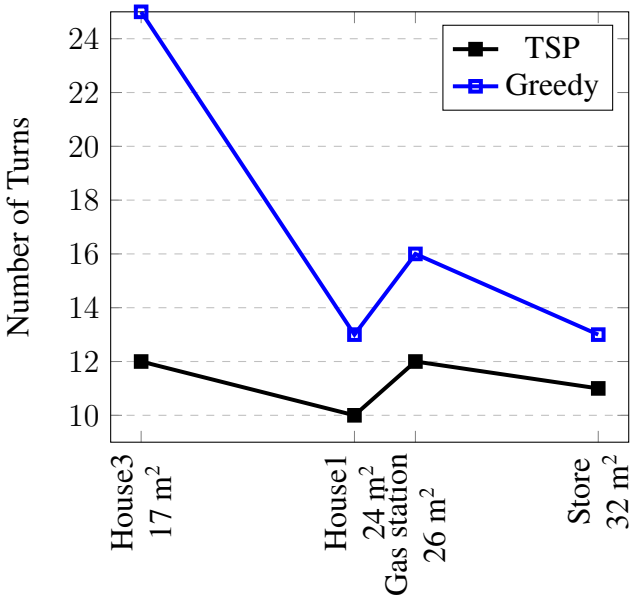
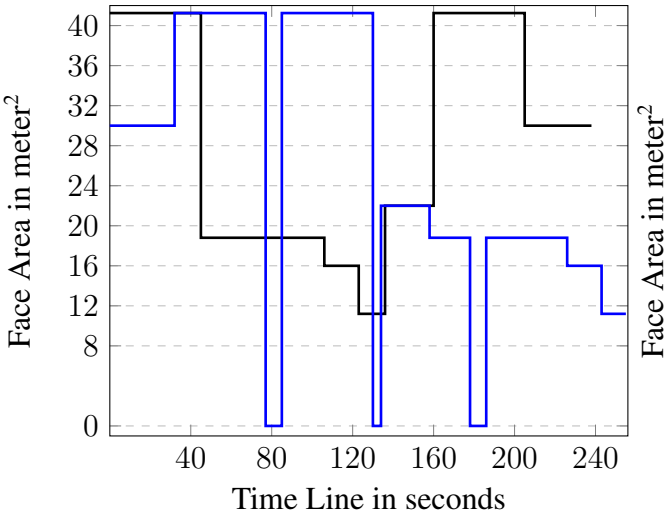
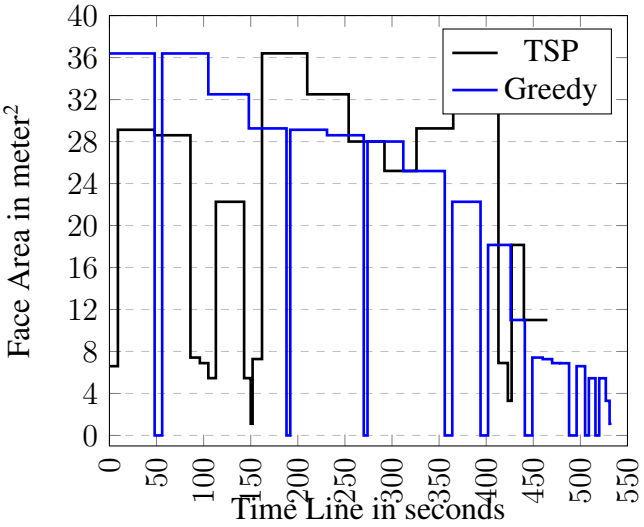


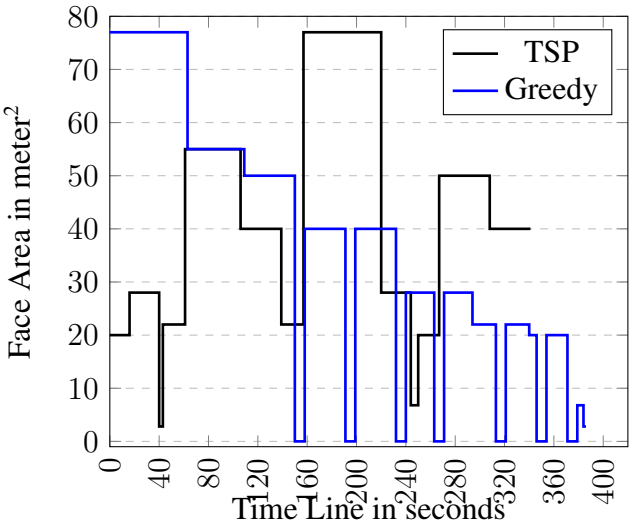
Figure 5.8: Number of Turns vs Average Face Area



(a) House1



(b) House 3



(c) Store

Figure 5.9: Coverage Area vs Time Line in seconds

Chapter 6

Conclusion and Future Work

In this thesis, we introduced the 3D structural inspection problem where a 3D complete coverage flight path is planned for a UAV to completely cover a 3D target structure. As a solution to this problem, we proposed a new 3D cellular decomposition algorithm by extending Boustrophedon Cellular Decomposition (BCD) to handle 3D structures. The 3D decomposition algorithm scans the enveloped target structure to decompose it into 2D coverable faces. From the adjacency information, we connected the decomposed faces in a graph-based representation. Finally, the traveling salesman problem (TSP), which is an instance of complete coverage path planning algorithm, is used to find a tour to completely cover each face. The UAV covers each face in a zig-zag pattern to completely cover entire structure.

6.1 Lessons Learned

We tested our proposed approach using a simulated UAV. The simulation was done using a simulator that provides an accurate simulation of the real-world dynamics of UAV and can directly extend the application to its real-world UAV, AscTec Firefly. The proposed algorithm was thoroughly evaluated to test its capability to handle complex 3D structures. Our approach guarantees 100% coverage of a target structure. When compared with a

greedy largest area first complete coverage approach, the TSP approach performed 50% better in reducing the flight path length, with an average of 12% less total distance traveled by the UAV. This means that using TSP, our 3D structural inspection algorithm reduced the total time taken for completely covering the enveloped target structure, which reduces the battery consumption of the UAV. With TSP, we have reduced repeated coverage (i.e., the UAV traveling over an already covered face to reach the next face). In contrast, the Largest Area First approach increases repeated coverage by upto 50% . We also observed that TSP decreases the number of changes in orientation for a UAV by to an average of 30% as compared with the greedy approach. This is also a factor that affects the total time and energy spent by the UAV for coverage. We learned that the selection of the coverage pattern and the coverage direction on each face play a major role in decreasing the total time of coverage as well as the number of waypoints generated to cover the surfaces of all faces. For example, if we choose coverage direction perpendicular to shortest edge on each face for the Gas Station target structure, the UAV has 243 waypoints as compared to only 157 waypoints by choosing the coverage direction perpendicular to longest edge. This approach saves 30% of the total coverage duration.

6.2 Future Work

As future work, we would like to look into the following topics to understand the 3D decomposition and structural inspection problem more effectively:

- Three-dimensional path planning for UAVs to avoid obstacles in a complex dynamic environment. Our approach handles the obstacles in a static environment as they are known a priori But avoiding obstacles dynamically during the flight is a challenging problem. To plan a collision free path in dynamic environments, we need to develop techniques to use data from range sensors like laser range finders and detect the objects around the UAV in real time, then plan a path through the feasible regions

while minimizing the deviation from the initally planned tragectory.

- Multi-UAV coordination to inspect large structures. Some of the target structures can be large and complex with more faces. In such cases, due to the limitations of the battery on the UAVs, it is not possible for one UAV to completely cover the structure. Hence, multiple UAVs need to be deployed such that they can collectively cover an entire target structure in parallel. To do this, the decomposed faces are shared among the UAVs based on their average area such that all the UAVs would cover equal portions of the structure in parallel, decreasing the total time of coverage. The UAVs would also need to communicate and take into account the trajectory of other UAVs to avoid collision during the flight while traveling from face to face.

- Structural Inspection in GPS denied environments: UAVs are currently used extensively in outdoor environments, but their use in indoor applications have been fairly restricted, owing mainly to the difficulty to maneuver them in smaller indoor spaces and the inability to use GPS.Nevertheless, there are many indoor applications where UAVs could provide a safe, reliable and resilient means to perform operations that are dangerous for humans such as surveillance inside chemical plants, inventory scanning in cold storages etc. Our proposed approach is applicable to decompose the mentioned environments but the challenge is to localize the UAV. One approach to solve this problem is using AprilTag markers. These markers are easily detected with the help of camera sensor of UAV and the AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera. Hence, UAVs can be localized in the indoor environments by placing these markers over the coverable surfaces.

- On-board infra-red and thermal sensors to develop applications for pipeline and bridge inspection, leak detection, building efficiency etc. In future, we can extend our approach to develop a complete system by attaching infra-red and thermal sensors

to aid in inspection of structures. These sensors can detect poor insulation in the buildings by identifying areas of higher temperature. The data from these sensors can be processed in real time or sent to engineers for further analysis.

- Extend complete coverage path planning algorithm to handle curved and convex surfaces: In our approach, we assumed that 3D structures are enveloped by rectilinear surfaces at an offset distance from the target structure. In some cases, the target structures can have curved surfaces which would result in inspecting the structure at a distance greater than the specified offset distance that might result in poor sensor data. Our future direction is to develop complete coverage algorithms that are applicable for structures having curved surfaces.

In conclusion, we proposed a new approach of 3D cellular decomposition to solve inspection problems. We have compared our TSP-based coverage strategy with other strategies. With our approach, we have achieved 100% coverage of target structures with reduced repeated coverage. Our approach performed up to 50% better in reducing the flight path and 12% less coverage duration than a largest-area-first approach.

Bibliography

[1] Y. Gabriely and E. Rimon. Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 954–960 vol.1.

[2] C. Phan and H. H. T. Liu. A cooperative UAV/UGV platform for wildfire detection and fighting. In *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing*, pages 494–498.

[3] Yanbo Huang, Steven J. Thomson, W. Clint Hoffmann, Yubin Lan, and Bradley K. Fritz. Development and prospect of unmanned aerial vehicle technologies for agricultural production management. 6(3):1–10.

[4] Suman Srinivasan, Haniph Latchman, John Shea, Tan Wong, and Janice McNair. Airborne traffic surveillance systems: Video surveillance of highway traffic. In *Proceedings of the ACM 2Nd International Workshop on Video Surveillance & Sensor Networks*, VSSN '04, pages 131–135. ACM.

[5] Zuo Llang Cao, Yuyu Huang, and Ernest L. Hall. Region filling operations with random obstacle avoidance for mobile robots. 5(2):87–102.

[6] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers.

[7] Howie M. Choset. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

[8] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz. Exact cellular decompositions in terms of critical points of morse functions. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2270–2277 vol.3.

[9] Milnor John. Morse theory. (AM-51), volume 51.

[10] P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar. Exact cellular decomposition of closed orientable surfaces embedded in rfr;3. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 1, pages 699–704 vol.1.

[11] Israel A. Wagner, Yaniv Altshuler, Vladimir Yanovski, and Alfred M. Bruckstein. Cooperative cleaners: A study in ant robotics. 27(1):127–151.

[12] Z. J. Butler, A. A. Rizzi, and R. L. Hollis. Contact sensor-based coverage of rectilinear environments. In *Proceedings of the 1999 IEEE International Symposium on Intelligent Control Intelligent Systems and Semiotics (Cat. No.99CH37014)*, pages 266–271.

[13] A. Zelinsky, R. A. Jarvis, J. C. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *In Proceedings of International Conference on Advanced Robotics*, pages 533–538.

[14] H. Kaluer, M. Brezak, and I. Petrovi. A visibility graph based method for path planning in dynamic environments. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 717–721.

[15] Jean-Claude Latombe. Approximate cell decomposition. In *Robot Motion Planning*, The Springer International Series in Engineering and Computer Science, pages 248–294. Springer, Boston, MA. DOI: 10.1007/978-1-4615-4022-9_6.

[16] C. W. Warren. Global path planning using artificial potential fields. In *1989 International Conference on Robotics and Automation Proceedings*, pages 316–321 vol.1.

[17] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 113–120 vol.1.

[18] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. 12(4):566–580.

[19] E. W. Dijkstra. A note on two problems in connexion with graphs. 1(1):269–271.

[20] W. Zeng and R. L. Church. Finding shortest paths on real road networks: the case for a*. 23(4):531–543.

[21] Anthony Stentz. The focussed d* algorithm for real-time replanning. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659.

[22] Yanrong Hu and S. X. Yang. A knowledge based genetic algorithm for path planning of a mobile robot. In *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, volume 5, pages 4350–4355 Vol.5.

[23] Il-Kyun Jung, Ki-Bum Hong, Suk-Kyo Hong, and Soon Chan Hong. Path planning of mobile robot using neural network. In *Proceedings of the IEEE International Symposium on Industrial Electronics, 1999. ISIE '99*, volume 3, pages 979–983 vol.3.

[24] H. Miao and Y. C. Tian. Robot path planning in dynamic environments using a simulated annealing based approach. In *Robotics and Vision 2008 10th International Conference on Control, Automation*, pages 1253–1258.

[25] Prasad N. Atkar, Aaron Greenfield, David C. Conner, Howie Choset, and Alfred A. Rizzi. Uniform coverage of automotive surface patches. 24(11):883–898.

[26] F. Yasutomi, M. Yamada, and K. Tsukamoto. Cleaning robot control. In *1988 IEEE International Conference on Robotics and Automation Proceedings*, pages 1839–1841 vol.3.

[27] Douglas W. Gage. Randomized search strategies with imperfect sensors. In *In Proceedings of SPIE Mobile Robots VIII*, pages 270–279.

[28] Brendan Englot and Franz S. Hover. Sampling-based coverage path planning for inspection of complex structures. In *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, ICAPS'12, pages 29–37. AAAI Press.

[29] Howie Choset. Coverage for robotics a survey of recent results. 31(1):113–126.

[30] Esther M. Arkin and Refael Hassin. Approximation algorithms for the geometric covering salesman problem. 55(3):197–218.

[31] T. C. Shermer. Recent results in art galleries [geometry]. 80(9):1384–1399.

[32] Fajie Li and Reinhard Klette. An approximate algorithm for solving the watchman route problem. In *Robot Vision*, Lecture Notes in Computer Science, pages 189–206. Springer, Berlin, Heidelberg.

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

[34] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. 21(4):331–344.

[35] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara. BSA: A complete coverage algorithm. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2040–2044.

[36] Y. H. Choi, T. K. Lee, S. H. Baek, and S. Y. Oh. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5788–5793.

[37] Ivan Maza and Anibal Ollero. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems*, volume 6, pages 221–230. DOI: 10.1007/978-4-431-35873-2_22.

[38] A. Renzaglia, L. Doitsidis, A. Martinelli, and E. B. Kosmatopoulos. Multi-robot 3d coverage of unknown terrains. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 2046–2051.

[39] Stefan Edelkamp, Baris Can Secim, and Erion Plaku. Surface inspection via hitting sets and multi-goal motion planning. In *Towards Autonomous Robotic Systems*, Lecture Notes in Computer Science, pages 134–149. Springer, Cham.

[40] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart. Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6423–6430.

[41] A. Breitenmoser, J. C. Metzger, R. Siegwart, and D. Rus. Distributed coverage control on surfaces in 3d space. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5569–5576.

[42] Peng Cheng, J. Keller, and V. Kumar. Time-optimal UAV trajectory planning for 3d urban structure coverage. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2750–2757.

[43] M. Hirose, Y. Xiao, Z. Zuo, V. R. Kamat, D. Zekkos, and J. Lynch. Implementation of UAV localization methods for a mobile post-earthquake monitoring system. In *2015 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS) Proceedings*, pages 66–71.

[44] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, August 2005. Google-Books-ID: k_yOQgAACAAJ.