s

ENERGY-EFFICIENT COVERAGE PATH PLANNING IN OBSTACLE-AWARE ENVIRONMENTS USING SHAPE OPTIMIZATION

by

Umeshkumar Ghaskata

April 2026

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfilment of the requirements for the
degree of

Master of Science

Computer Science and Engineering

I grant The University at Buffalo the non-exclusive right to use this work for the University's own purposes and to make single copies of the work available to the public on a not-for-profit basis if copies are not otherwise available.

Umeshkumar Ghaskata

Copyright by

Umeshkumar Ghaskata

2026

Acknowledgments

I would like to express my deepest gratitude to prof. Karthik Dantu, my mentors Charuvahan Adhivarahan and Pranay Meshram, for their unlimited guidance, care, and support throughout my MS Thesis at the University at Buffalo. Prof. Karthik Dantu has not only shown me how to conduct groundbreaking research, but also how to be a productive member of the Academy. I would also like to thank prof. Souma Chowdhury for being on committee member and provide valuable feedback for this thesis.

I wish to thank prof. Alina Vereshchaka, Prof. Kelin luo, Prof. Shamsad Parvin for giving valuable knowledge that I could leverage during various stages of my thesis.

Last, but not least, I extend my sincere thanks to my parents and my brother for their enduring love, support, and encouragement. Their unwavering belief in me has been a pillar of strength throughout this journey. Without their sacrifices and constant support, this achievement would not have been possible.

Table of Contents

List of Tables

List of Figures

Abstract

Coverage Path Planning (CPP) is one of the important concepts used in robotics that provides autonomous robots with the ability to monitor and explore the environments around them. This concept used in many real-worlds application like search and rescue, environmental monitoring, and aerial surveying. As the problem is quite difficult especially in complex environments full of obstacles, the goal becomes to generate an optimal coverage path while maximizing the covered area, minimizing redundancy, traversal, turns, and energy consumption at the same time.

In this thesis, a new approach for energy-efficient coverage path planning in the presence of obstacles is presented based on the application of Shape optimization for optimal environment decomposition. By using a new formulation of the problem where coverage can be considered as a discrete-continuous optimization problem. After the decomposition process, an improved Lin-Kernighan-Helsgaun (LKH) algorithm-based path optimization technique is implemented to obtain better paths in terms of the waypoint sequences generated by the decomposition phase. The integration of these two parts allows obtaining not only good spatial coverage but also an efficient way to execute the coverage.

Many simulations were performed in several benchmark scenarios and maps in order to compare the performances of the new method with conventional methods, such as the square grid, the hexagonal decomposition, and the greedy deployment method. From simulation results, we have concluded that the proposed approach performs quite well regarding both coverage and overlaps. The coverage percentage obtained with the proposed method is very close to full coverage (i.e., approximately 99.8%–100%), whereas the overlap rate is minimized by 40-45%. Due to that, energy consumption values remain relatively low for the proposed method.

Chapter 1

Introduction

Coverage Path Planning (CPP) is an essential aspect of robotics engineering, which aims at creating a trajectory allowing for the full coverage of an entire environment. In addition to the need for efficient coverage, the objective of CPP is to minimize costs, overlaps, and energy usage while ensuring the safety of navigation amidst obstacles. Over time, CPP has become a crucial part of various industrial and practical scenarios such as environmental surveillance, aerial mapping, farming, infrastructure inspections, and automated vehicle navigation. [1, 2].

Whereas full coverage is usually feasible using current approaches, the difficulty here mainly centers around how to design a *good* coverage path. The criteria for determining whether a coverage path is *good* usually vary with applications, but they generally include traveling shorter distances, minimizing turning, eliminating redundant coverage, and enhancing energy efficiency. In a real-world implementation using robotics technology, a turn action adds extra costs in the form of deceleration, re-orienting, and accelerating, thereby increasing both time costs and energy expenses [3].

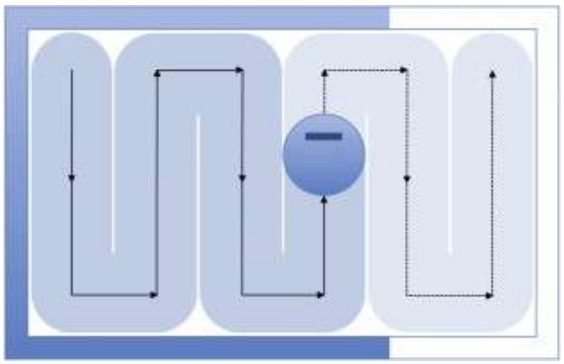


Figure 1.1: Typical zigzag path

However, in practice, there will be constraints imposed on the sensing/actuation network since there will be a maximum radius within which the nodes can sense, communicate, or visualize their environment. Hence, the problem of coverage can be stated as finding the appropriate placement for the sensors and the path along which they traverse to achieve full coverage. The majority of the current approaches use some kind of decomposition of the space in grids that leads to overlaps, inefficient patterns, and higher energy usage. Classical CPP algorithms have made certain assumptions like having a homogeneous environment and adopting some form of discretization such as square or hexagonal tessellation to make the problem easier. However, while discretization offers simplicity in computing, it also presents several drawbacks, especially when used in complex, obstacle-ridden, or non-convex environments.[1].

CPP is inherently challenging due to the presence of multiple conflicting objectives. As mentioned in [1], achieving complete coverage, minimizing overlap, avoiding obstacles, and optimizing path efficiency are mutually competing goals. Improving one aspect often leads to compromises in others, making it difficult to derive globally optimal solutions. Furthermore, the computational complexity of CPP is closely related to well-known NP-hard problems such as the Traveling Salesman Problem (TSP) and variants of coverage sequencing problems [3, 4]. As a result, deriving exact optimal solutions are often impractical for real-world applications, and most approaches rely on approximate or heuristic methods.

This problem has been tackled by most of the existing CPP algorithms through a two-step procedure: (i) Dividing the space into sub-regions, and (ii) Finding the path over such sub-regions. Although useful, the use of such decomposition techniques poses new problems, especially when using discrete decomposition techniques. For instance, discrete decomposition

often involves the use of geometric approximations and may cause redundant coverage, inefficient way-pointing, and inefficiencies in the use of energy resources.

Alternative geometric solutions have been suggested to overcome the issues of traditional grid-based approaches [3]. For instance, approaches based on circle packing and disk covering allow modeling coverage in a much more natural way by treating sensing regions as geometric objects. The key feature of such an approach is the ability to reduce the overlap between covered areas by utilizing optimal packing [3].

Inspired by the limitations of existing solutions, this paper proposes an alternative approach for the task of covering a given area with minimum possible number of units. As opposed to the traditional setting, which assumes that the number of sensors to be used is fixed and coverage radius should be optimized, our formulation is closer to geometric set covering and facility location problems that are known to be NP-hard [5].

The main novelty of the proposed approach is its ability to consider the problem from both continuous and discrete perspectives simultaneously. Namely, by integrating geometric coverage modeling with path planning, we can significantly reduce redundancy, increase the coverage efficiency, and generate energy-efficient traversal paths.

1.1 Contributions

The main contributions of this thesis are summarized as follows:

- We propose a **novel bilevel discrete–continuous optimization framework** for CPP. The upper level determines the optimal number of coverage units, while the lower level optimizes their continuous spatial placement using a Shape optimization based formulation. This integration enables accurate representation of irregular environments while reducing redundancy and overlap.

- We integrate the continuous decomposition with **discrete path optimization** using an LKH-based algorithm, enabling efficient traversal planning that minimizes both travel distance and turning cost.

- We conduct extensive simulations across multiple benchmark environments and demonstrate that the proposed method achieves **near-complete coverage (**$\approx$ 99.8–100%**)** while

outperforming square, hexagonal, and greedy deployment strategies in terms of overlap reduction and energy efficiency.

Chapter 2

Background

2.1 Terminology

In this section, we briefly study some of the key terms that will appear throughout the thesis. We provide the necessary foundation for coverage path planning, for the sensing model, and for performance measure in order to allow comparison among the various strategies of coverage in environments with obstacles.

2.1.1 Coverage Path Planning (CPP)

Coverage Path Planning (CPP) refers to the problem of generating a path or a set of paths that enable a robot or a swarm of robots to visit all points in a target region while avoiding obstacles. Unlike the traditional planning where the main objective is just getting to one specific spot, CPP though, it's goal is to cover as much of the area as possible, or at least get pretty close to covering it all.

Figure 2.1 illustrates the concept of Coverage Path Planning (CPP), adapted from [6].

2.1.2 Sensing Radius and Coverage Footprint

The sensing radius defines the maximum distance within which a robot or UAV can effectively sense or cover its environment (Figure - 2.2). It is typically modeled as a circular region centered at the robot's position.

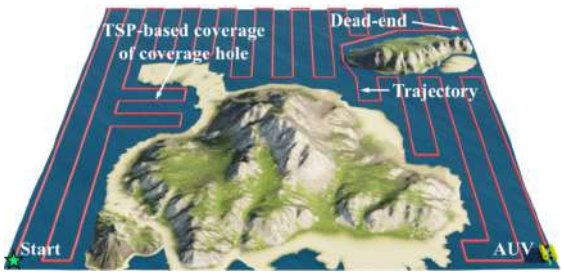


Figure 2.1: Coverage Path Planning

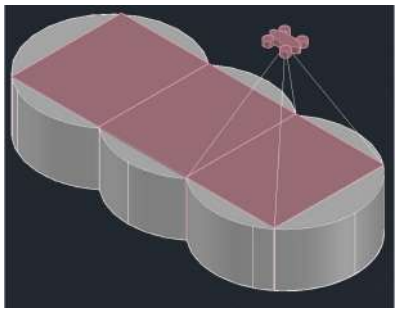


Figure 2.2: UAV Footprint

The coverage footprint refers to the area that the sensor can cover effectively from the selected location. Usually, for most coverage path planning (CPP) problems, the footprint can be considered to be a circle of radius D in two dimensions (2D) or even the projection of the sensor field of view onto the ground surface which is nothing but a 2D circle. Here in our case, the area covered will be assumed to be a circular 2D region having a radius of D if the UAV is at the center of this circle.

2.1.3 Overlap and Redundancy

Overlap occurs when multiple sensing footprints cover the same area. While some overlap may be necessary to ensure robustness and avoid coverage gaps, excessive overlap leads to redundancy and inefficient use of resources.

the Figure 2.3 is demonstrated from the paper [7]

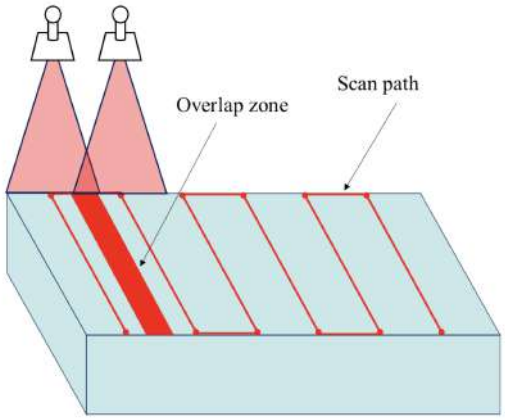


Figure 2.3: Overlap

2.1.4 Coverage Efficiency Metrics

The metrics for coverage efficiency serve as tools that assess the effectiveness of coverage strategies. The use of such metrics helps compare various methods quantitatively.

Common metrics include:

- **Coverage Ratio:** The percentage of the target area that has been successfully covered.
- **Overlap Ratio:** The amount of redundant coverage relative to the total covered area.
- **Energy Consumption:** The total energy used by the robot(s) during the coverage task.
- **Coverage Time:** The time required to complete the coverage mission.
- **Coverage per Node:** The amount of area covered per deployed robot or sensing unit.

In this thesis, we have combined these metrics into a unified performance *"score"* to evaluate trade-offs between coverage completeness, redundancy, and energy efficiency.

2.2 Routing Algorithms

Routing algorithms play a crucial role in coverage path planning (CPP) where efficient traversal of way points or regions is required. Our goal is to find the right order of the waypoint in a such a way to minimize the energy requirement. These algorithms aim to determine optimal or near-optimal paths while considering constraints such as distance, turning, and obstacle avoidance. In this section, we discuss classical and modern routing strategies relevant to CPP.

2.2.1 Travelling Salesman Problem (TSP)

The Travelling Salesman Problem (TSP) is one of the most fundamental combinatorial optimization problems in routing. It aims to find the shortest possible route that visits each node exactly once and returns to the starting point. In CPP, TSP is often used to determine the optimal visiting sequence of waypoints or regions. TSP is known to be NP-hard, making exact solutions computationally expensive for large-scale problems. As a result, approximate and heuristic methods are commonly employed in real-world applications.

2.2.2 Heuristic-Based Routing Methods

Heuristic-based routing approaches are widely used to overcome the computational complexity of exact optimization methods. These methods aim to produce near-optimal solutions with **significantly** lower computational cost.

For instance, The heuristic routing strategies can adapt dynamically to network conditions or environmental constraints. A notable approach is the use of genetic-inspired routing mechanisms, where routing decisions evolve based on local interactions and optimization criteria. Such methods enable flexible and scalable solutions, particularly in dynamic or distributed systems like mobile ad-hoc networks [8]. These heuristic approaches are especially useful in CPP scenarios.

2.2.3 Lin-Kernighan Heuristic (LKH)

The LKH algorithm is an advanced local search heuristic technique widely employed for this type of problem. In contrast to traditional heuristics, which provide fast solutions, the LKH goes further and explores alternative routes. The algorithm employs edge exchanges in the route, but the level of exploration changes according to specific conditions. This enables it to bypass possible local optima, and consequently, the solution quality improves gradually.

Due to its efficiency and scalability, LKH is considered one of the most effective heuristic methods for routing problems and has been extensively applied in coverage path planning tasks. In such applications, determining an optimal or near-optimal visiting sequence of regions significantly impacts overall path length, traversal time, and energy consumption.

Figure 2.4 and Figure 2.5 illustrate the working principle of LKH. Starting from an initial feasible tour T, the algorithm performs a sequence of edge exchanges to generate an improved tour T'. These transformations aim to reduce the total cost (e.g., distance or energy), and the process is repeated iteratively until no further improvement is possible.

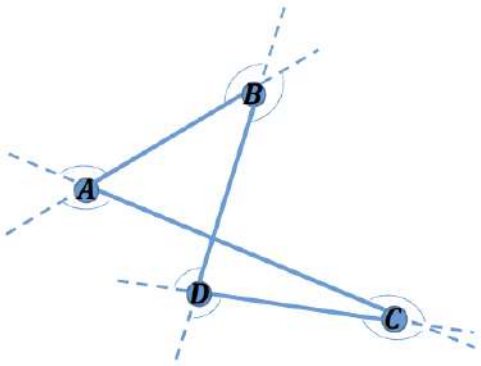


Figure 2.4: Initial feasible tour T

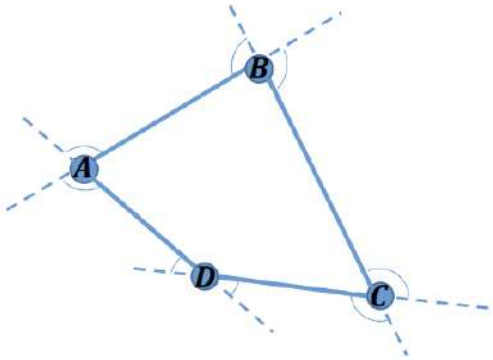


Figure 2.5: New feasible tour T′ obtained from tour T.

2.2.4 Energy-Aware Routing Strategies

Many practical scenarios are faced with such challenges where energy conservation becomes the major concern, and energy-aware routing algorithms will ensure optimization of the decision making. decision making could be considered a problem of optimization, which ensures minimizing energy consumption with respect to task requirements. For instance, routing of tasks into computational models or agents that consume minimum energy will help decrease energy consumption in the system. The mentioned above technique is highly useful for multi-UAV CPP because of energy constraints of UAVs' batteries.

2.2.5 Graph-Based Path Planning

One of the most commonly adopted techniques for path planning involves graph theory. In graph based techniques, the environment is modeled as a graph in which each node represents a state and edge represents a transition from one state to another. Techniques like A* and their extensions are frequently employed to find shortest paths. Techniques like Hybrid A* use the power of classical techniques along with system dynamics and constraints.

Recent work demonstrates that graph-based planning can be extended to handle dynamic environments by incorporating time-dependent constraints and obstacle predictions. For instance, time-indexed variants of Hybrid A* explicitly account for dynamic obstacles to generate collision-free paths in complex environments :contentReference[oaicite:2]index=2.

These methods are highly relevant for CPP in obstacle-rich environments, where both static and dynamic constraints must be considered.

2.3 Energy Consumption by UAVs

The amount of energy consumed is one of the vital considerations when designing coverage paths through unmanned aerial vehicles because it affects the operation time of the UAVs. This means that unlike in the case of mobile robots, there exist different variables affecting energy consumption in UAVs.

2.3.1 Distance-based Energy Models

Energy consumption is assumed to be directly proportional to the distance traveled in the most simple energy model. The model ensures that the shorter the path taken, the lesser the amount of energy consumed. This approach is similar to the Traveling Salesman Problem (TSP) approach. Nevertheless, this energy model fails to consider other crucial parameters of UAV dynamics.

2.3.2 Turn-aware Energy Models

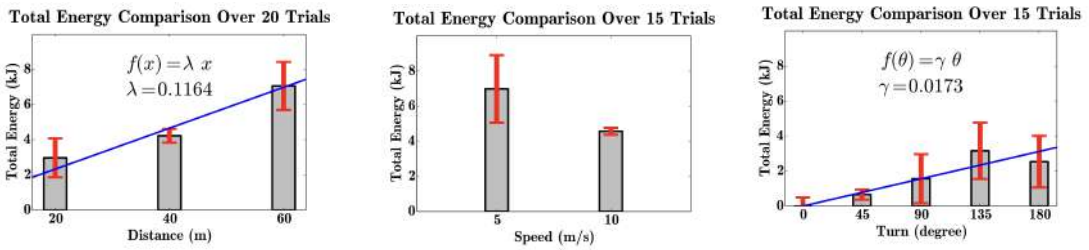
Studies have demonstrated that turning maneuvers significantly impact energy consumption in UAV systems. Experimental analysis refer [9] shows that energy increases approximately linearly with both the distance traveled and the magnitude of turn angles. Moreover, UAVs expend extra energy while turning through acceleration and deceleration processes. Thus, a turn costs more energy than linear flight; therefore, a path that avoids extra turns is considered energy effective.

2.3.3 Dynamic Energy Models (speed, acceleration)

More advanced models incorporate dynamic factors such as velocity, acceleration, and aerodynamic drag. Experimental findings by Farshad *et al.* indicate that UAV energy consumption depends not only on path length but also on speed profiles and turning behavior. In particular,

variations in velocity and frequent changes in direction increase flight time and energy usage due to additional control and stabilization requirements.

Figure 2.6 illustrates these relationships. As shown, energy usage is linearly proportional to distance, dependent on speed because of variation in flight time, and dependent on angle because of increased turns. These findings illustrate the need to consider various factors in energy-aware path planning algorithms.



(a) Energy vs. Distance (b) Energy vs. Speed (c) Energy vs. Turn Angle

Figure 2.6: Experimental analysis of UAV energy consumption under varying distance, speed, and turning conditions (adapted from [**?**]).

2.4 Types of Coverage Strategies

Coverage path planning strategies can be categorized based on several key characteristics that influence how agents operate within an environment.

Offline vs Online Coverage:

Coverage algorithms are usually categorized into two types; offline and online coverage strategies. Offline coverage strategy requires complete information of the environment to be gathered prior to deployment, which helps to develop an optimized path. On the other hand, online coverage works with the information of an environment that is partially known or even unknown. The path is developed by the robot according to the information it gathers through sensing during operation.

Single-Agent vs Multi-Agent Coverage: Coverage can be done either by one agent alone or by many agents that cooperate together. Coverage by one agent alone is relatively easy to accomplish but will take a lot of time when the area is large. Cooperation among many agents will result in a shorter completion time for the task.

Deterministic vs Probabilistic Coverage: The deterministic techniques, make use of well-defined algorithms based on structured patterns such as grids and decomposition, which are used in order to ensure total coverage of the space. The probabilistic techniques, on the other hand, make use of stochastic or sampling techniques to achieve coverage with high probability but without guarantees.

Static vs Dynamic Environments: Coverage strategies also differ based on whether the environment is static or dynamic. In static environments, obstacles and boundaries remain unchanged, allowing stable path planning. In dynamic environments, obstacles may move or new constraints may arise over time, requiring continuous re-planning and adaptation to maintain effective coverage.

2.5 Optimization in Coverage Planning

2.5.1 Discrete Optimization Approaches

Discrete optimization models the coverage problem on a discrete set of possible locations, regions, or cells in the environment. Discretization is carried out by dividing the environment into a set of nodes or cells, after which we choose an optimal subset or order that will provide complete coverage at minimal cost.

A classical formulation in this category is the **Set Cover Problem**, where the goal is to select the minimum number of sensing locations such that all points in the region of interest are covered. This problem is combinatorial and NP-hard, and is often solved using greedy approximations or integer linear programming techniques, But fundamentally it's a discrete, combinatorial optimization problem. As illustrated in Figure 2.7, a 2-covering of the region is achieved, obtained from [10].

Another important class includes **graph-based formulations**, where the environment is represented as a graph and coverage is achieved by finding optimal traversal paths. Problems such as the **Travelling Salesman Problem (TSP)** and **Vehicle Routing Problem (VRP)** are commonly used to determine efficient visiting sequences of discrete waypoints.

The primary trade-off in discretization is that increasing resolution (making the lattice or mesh finer) reduces discretization error but simultaneously increases computational complexity

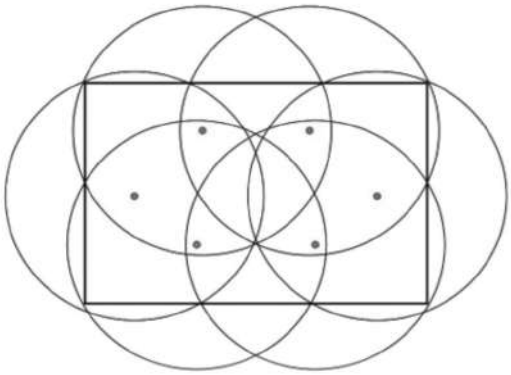


Figure 2.7: Set-Cover Problem

2.5.2 Continuous Optimization Methods

Continuous optimization methods model the coverage problem directly in a continuous domain, where decision variables such as robot positions, sensor placements, or trajectories are defined over a continuous space. These methods are particularly suitable for high-resolution coverage and environments where discretization may introduce errors.

A widely used approach in continuous coverage is **Voronoi-based partitioning** Figure 2.8, where the environment is divided into regions assigned to different agents. Each agent optimizes its position within its region to improve coverage efficiency. This leads to methods such as **Lloyd's algorithm**, which iteratively updates agent positions based on centroidal Voronoi tessellations.

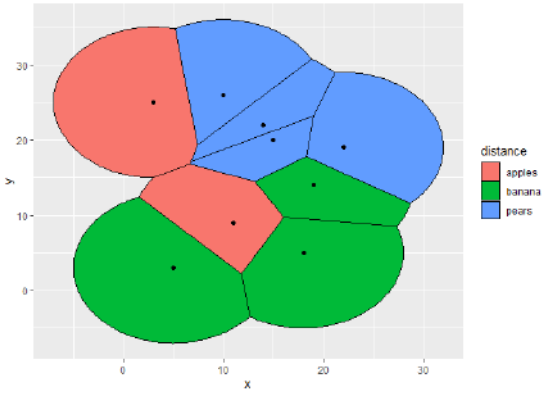


Figure 2.8: Voronoi-Based partitioning

Gradient-based optimization techniques are also commonly used, where coverage is for-

mulated as an objective function (e.g., minimizing uncovered area or maximizing sensing quality), and agents iteratively update their positions using gradient descent.

In addition, **potential field methods** guide agents toward uncovered regions while avoiding obstacles and other agents through artificial force fields, making them suitable for distributed and real-time applications. [11]

Some of the more sophisticated techniques for UAV navigation and control include **optimal control** and **trajectory optimization**, which involve optimizing the whole path of the vehicle throughout its lifetime, taking into account all the relevant dynamics, constraints, and power requirements. Continuous optimization algorithms provide a much higher degree of versatility and precision than discrete algorithms.

Chapter 3

Related Work

Existing CPP techniques can be broadly categorized based on the type of decomposition and optimization strategies employed. These include exact cellular decomposition, approximate and grid-based methods, heuristic deployment strategies, and continuous optimization approaches [3].

3.1 Decomposition Methods

Exact cellular decomposition methods, such as trapezoidal decomposition [1], boustrophedon decomposition [12], and Morse-based decomposition [13], divide the environment into non-overlapping cells. These methods construct adjacency graphs and allow efficient back-and-forth coverage within each cell.[1].

Morse-based decomposition extends exact cellular decomposition by utilizing Morse functions to partition the environment into cells defined by critical points. Within each cell, the

Figure 3.1: Cellular decomposition for $h(x) = (x_1)^2 + (x_2)^2$

structure remains consistent, enabling simple and efficient coverage strategies. This framework provides flexibility in generating different coverage patterns by varying the underlying Morse function [14].

Approximate and grid-based decomposition methods [15] discretize the environment into uniform cells shown in Figure 3.2, typically square grids, and formulate CPP as a graph traversal problem. These approaches are computationally efficient and integrate well with algorithms such as wavefront propagation and TSP-based heuristics [16]. However, they suffer from high memory requirements, discretization errors, and poor representation of irregular geometries, which can result in suboptimal solutions.

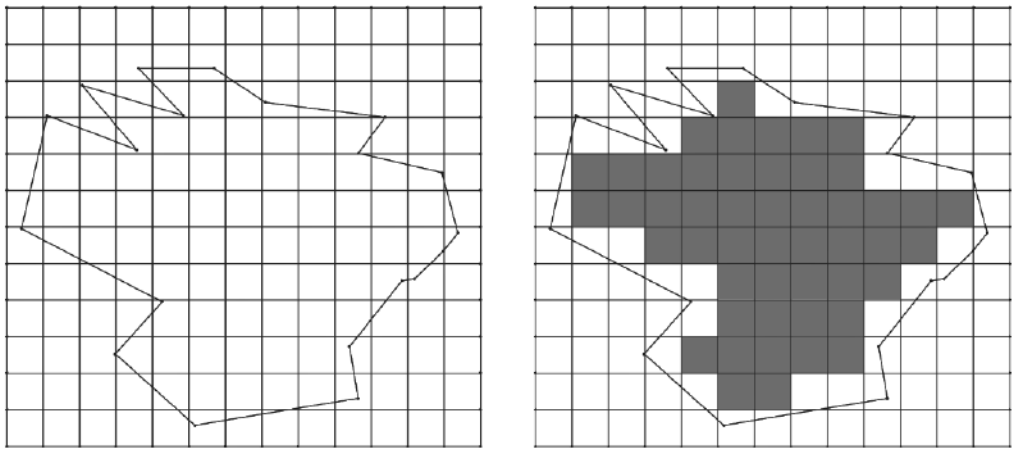


Figure 3.2: Grid Decomposition

Heuristic and greedy deployment strategies, such as incremental sensor placement [17], iteratively place agents to maximize coverage of unexplored regions. These methods rely on local information and are suitable for unknown environments. However, they are prone to local optima and may fail to achieve globally efficient coverage.

Alternative geometric decompositions, such as hexagonal tiling, improve packing efficiency compared to square grids and reduce overlap in many cases [3]. Despite these improvements, they remain fundamentally discrete approximations and inherit the limitations of discretization.

3.2 Continuous Optimization for Coverage

To overcome the limitations of discrete representations, continuous optimization techniques have been explored. In particular, the Shape optimization method proposed in [18] formulates the coverage problem as a continuous optimization problem, enabling the computation of optimal placement of coverage elements using first- and second-order derivatives. This approach is particularly effective for irregular and non-convex environments, as it reduces overlap and improves coverage efficiency. However, integrating such continuous formulations with efficient path planning remains an open challenge.

3.3 Energy-Aware Coverage Planning

Recent research has emphasized energy-efficient CPP, particularly for UAV applications where battery constraints are critical. Traditional CPP formulations primarily focus on minimizing path length, but energy consumption is also significantly influenced by turning maneuvers and motion dynamics.

For example, the UB-ANC planner proposed by Modares *et al.* incorporates both travel distance and turning costs into an energy-aware model and solves the problem using adaptations of the Lin-Kernighan heuristic [9]. Similarly, Cabreira *et al.* introduce cost functions that account for acceleration, deceleration, and flight dynamics, achieving improved energy efficiency in practical scenarios [19].

3.4 Summary and Research Gap

Discrete decomposition methods introduces a geometric approximations and end up with producing waypoints that could give redundant coverage inefficient coverage patterns. Whereas Continuous methods like Exact cellular decomposition or semi-exponential decomposition struggle with complex regions because of a high number of subregions being generated. Also Energy-aware methods ignore placement and directly optimize for path after descritize the area. To the best of our knowledge, no existing method jointly optimizes coverage placement, path planning, and energy consumption within a unified framework. Therefore, there is a clear need

for a unified framework that can optimize coverage placement, while effectively handling irregular geometries, minimizing overlap, and improving energy efficiency in realistic environments.

3.5 Proposed Approach

In this thesis, we address the limitations of existing methods by integrating continuous optimization with discrete path planning in a unified framework:

- A **Shape optimization based continuous decomposition** strategy is employed to generate an optimal distribution of coverage units over complex environments [18].
- An **LKH-based path planning algorithm** is used to compute near-optimal traversal sequences, minimizing both distance and turning cost [9].

To the best of our knowledge, this is the first work that integrates continuous coverage optimization with energy-aware CPP in a bilevel formulation. By operating directly in the continuous domain, the proposed approach accurately represents irregular environments while significantly reducing redundancy and improving overall coverage efficiency.

Chapter 4

Problem Formulation

4.1 Problem Description

Let $A \subset \mathbb{R}^2$ denote a bounded planar region representing the interest region that we want to cover by the UAV. The region may be nonconvex and may contain interior obstacles(holes). We assumed that each coverage unit provides sensing over a closed disk of fixed radius $r > 0$.

The objective of this work is:

> *Determine the minimal number of disks of radius r whose union completely covers the region A.*

More precisely, we seek both:

- The smallest integer m^* such that full coverage is achievable, and
- The corresponding disk placement configuration.

There are a few basic assumptions that form the foundation of the entire work:
The region A is a 2D space. It is also allowed to be non-convex and contain a finite number of obstacles. The boundary of this area is clearly defined and known. The environment does not change between planning and execution. No moving obstacles or environmental changes to expect. Last but not least, the cost of traversal takes into account both distance and turning.

4.2 Union of Disks

For a given integer $m \in \mathbb{N}$, define the collection of disk centers

$$x = (x_1, \ldots, x_m), \qquad x_i \in \mathbb{R}^2.$$

The union of disks is

$$\Omega_m(x) := \bigcup_{i=1}^{m} B(x_i, r),$$

where

$$B(x_i, r) = \{y \in \mathbb{R}^2 \ : \ \|y - x_i\| \leq r\}.$$

4.3 Covering Function

We can define the uncovered area function

$$G_m(x) := |A| - |A \cap \Omega_m(x)|, \tag{4.1}$$

where $|A|$ corresponding to the area of the interest region and $|A \cap \Omega_m(x)|$ corresponding to area of covered region by m number of disk.

The condition

$$G_m(x) = 0$$

is equivalent to full coverage of A and it's a non-convex function.

4.4 Inner Continuous Placement Problem

For a fixed number of disks m, we define

$$\phi(m) := \min_{x \in \mathbb{R}^{2m}} G_m(x). \tag{4.2}$$

The function $\phi(m)$ represents the minimal uncovered area achievable with m disks of fixed radius r. The function $G_m(x)$ is highly nonconvex in x and we do *not* enumerate all possible

geometric arrangements of disks (which would be combinatorial). Instead, we solve (4.2) using a second-order local optimization method.

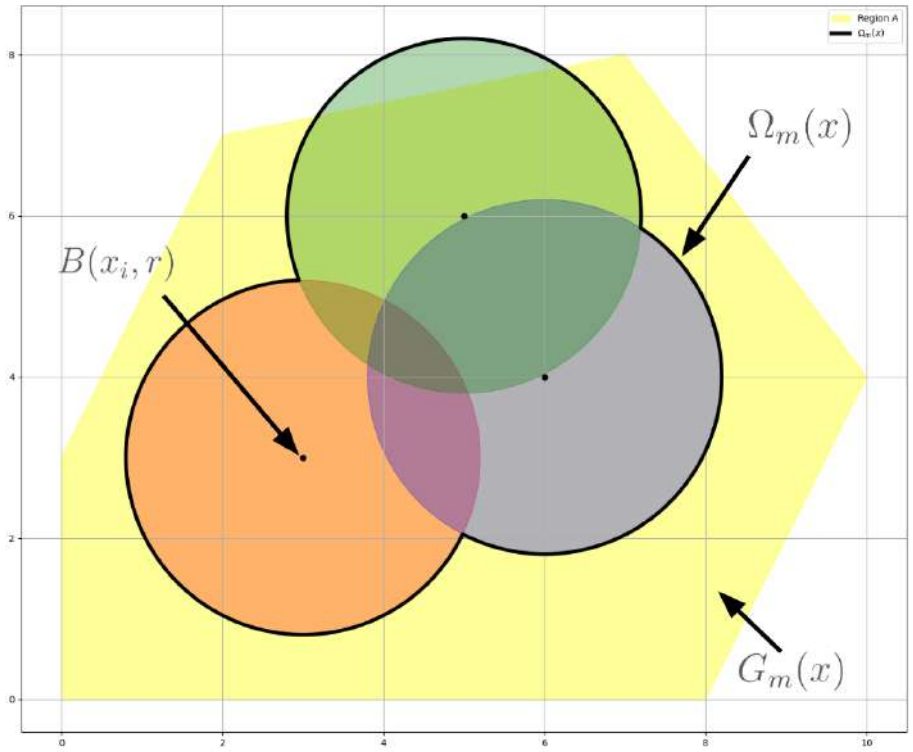


Figure 4.1: Visualization of $G_m(x)$, $B(x_i, r)$ and $\Omega_m(x)$

Figure 4.1 provides a visual illustration of the union of disks $\Omega_m(x)$ for a $m = 3$ disks. Each disk corresponds to the sensing footprint of a coverage unit, and their union defines the total covered area.

4.5 Local Refinement Approximation of $\phi(m)$

Given the nonconvexity of the coverage function, a single local optimal solution is usually not capable of providing a good solution. Instead of separately initializing random perturbations, we try to improve a solution locally by gradually imposing perturbations in an organized way.

Let $x^{(0)}$ denote an initial feasible configuration. Then, we obtain a local optimal solution using a gradient-based solver. To escape local minima, small Gaussian perturbations ($\approx 0.3r$) are applied to the centers, followed by re-optimization. The best configuration is retained.

Formally, let $\{x^{(k)}\}_{k=1}^{K}$ denote the set of refined solutions obtained after perturbation and re-optimization. We define the practical approximation

$$\tilde{\phi}(m) = \min_{k=0,\dots,K} G_m\big(x^{(k)}\big), \tag{4.3}$$

4.6 Outer Discrete Optimization Problem

The minimal disk covering number is defined as

$$m^* = \min\{m \in \mathbb{N} \ : \ \phi(m) = 0\}. \tag{4.4}$$

In practice, using tolerance $\varepsilon > 0$, we compute

$$m^* = \min\{m \in \mathbb{N} \ : \ \tilde{\phi}(m) \leq \varepsilon\}. \tag{4.5}$$

Thus, we increase m whenever

$$\tilde{\phi}(m) > \varepsilon.$$

4.7 Monotonicity Property

Feasibility is monotone with respect to m:

$$\phi(m+1) \leq \phi(m). \tag{4.6}$$

This follows since adding an disk increases the feasible space. once $\tilde{\phi}(m) \leq \varepsilon$, the search terminates.

4.8 Lower Bound Initialization

A geometric lower bound on m is given by

$$m \geq \left\lceil \frac{|A|}{\pi r^2} \right\rceil, \tag{4.7}$$

which ignores the overlaps between disks. This lower bound can also be used as a good initial value for the outer optimization, so that some computation is saved by searching around a point that is close to the optimal solution. In practice, we initialize our solver with this lower bound on m and omit unnecessary checks for smaller infeasible values.

4.9 Gradient Structure

Now we characterize the first-order and second-order structure of the coverage functional $G_m(x)$ defined in eq.(4.1).

4.9.1 Gradient

With general regularity and non-degeneracy conditions, we can see that the gradient the derivative of $G_m(x)$ with respect to the disk center x_i is only related to active boundary of disk i, i.e., the part of disk i's boundary that contributes to covering the region A[18].

Formally, the gradient is given by

$$\nabla_{x_i} G_m(x) = -\int_{A_i} \nu_i(z)\, dz, \tag{4.8}$$

where

$$A_i = \partial B(x_i, r) \cap \partial\Omega_m(x) \cap A$$

denotes the active boundary arc of disk i, and $\nu_i(z)$ is the outward unit normal vector at point z on the boundary of disk i. Geometrically, the equation indicates that the disks move in the direction that is most effective in reducing the uncovered area.

4.10 Complete Nested Optimization Formulation

The overall problem is

$$\boxed{\begin{aligned} &\min_{m\in\mathbb{N}} \quad m \\ &\text{s.t.} \quad \min_{x\in\mathbb{R}^{2m}} G_m(x) = 0. \end{aligned}} \tag{4.9}$$

This formulation separates:

- a discrete outer search over m,
- a continuous inner placement optimization over x.

This formulation results in a nested discrete-continuous optimization problem. The outer problem as a sequential search over the number of disks m while the inner problem optimizes on disk placement variables x, using gradient based local refinement procedure. Instead of performing a complete global minimization, starting from an initial feasible configuration the solution is progressively improved by re-optimizing around controlled perturbations in search to escape the local minima. Though global optimality cannot be guaranteed in the presence of nonconvexity, the convergence to locally optimal over multiple trials has been proved.

4.11 Stopping Criterion:

If

$$\frac{G(x)}{|A|} \leq \tau,$$

the solution is accepted, If uncovered area is below threshold.

Disk Insertion: Otherwise, a new disk is added in the largest uncovered region, new center computed by:

$$c_{\text{new}} = \arg\max_{p\in U} \min_i \|p - x_i\|, \quad U = A \setminus \Omega_m(x),$$

also m is incremented.

4.12 Joint Performance Metric

To evaluate the overall performance of different deployment strategies, we define a joint performance metric that combines coverage, overlap, and energy consumption into a single score.

Algorithm 1 Minimal Fixed-Radius Disk Covering with Local Refinement

Require: Region A, radius r, tolerance ε
Ensure: Disk centers x^*, minimal number of disks m^*

start with lower bound $m \leftarrow \left\lceil \frac{|A|}{\pi r^2} \right\rceil$
Initialize centers x using hexagonal lattice clipped to A
while $m \leq m_{\max}$ **do**
  $(x, G_m(x)) \leftarrow$ **LocalOptimize**(x)
  $x^* \leftarrow x, \quad G^* \leftarrow G_m(x)$
  for $k = 1$ to K **do**
    Generate perturbed centers $\tilde{x} \leftarrow x + \delta_k$
    Project $\tilde{x}$ into region A
    $(x_k, G_k) \leftarrow$ **LocalOptimize**$(\tilde{x})$
    if $G_k < G^*$ **then**
      $x^* \leftarrow x_k$
      $G^* \leftarrow G_k$
    end if
  end for
  $x \leftarrow x^*$
  if $\frac{G^*}{|A|} \leq \varepsilon$ **then**
    return x, m
  end if
  $x \leftarrow$ **AddDiskLargestGap**(x, A)
  $m \leftarrow m + 1$
end while

First, we normalized all metrics for each map to the range [0,1] and score is defined as:

t

$$\text{Score} = \frac{1 + \hat{C}}{1 + \lambda_1 \hat{O} + \lambda_2 \hat{E}} \tag{4.10}$$

where $\hat{C}$, $\hat{O}$, and $\hat{E}$ denote normalized coverage, overlap, and energy (distance cost), respectively. The weights λ_1 and λ_2 sets the relative importance of overlap and energy. In this work, we set $\lambda_1 = 0.4$ and $\lambda_2 = 0.6$, assigning slightly higher importance to energy efficiency and penalizing redundant overlap.

Chapter 5

Experimental Evaluation

5.1 Experimental Setup

We conduct experiments on various polygonal maps of different complexities (see Fig. 5.1). Due to space limitation, we display a part of the environments as an instance, and the other results also follow similar trends Each map includes obstacles and non-convex objects to challenge robustness.

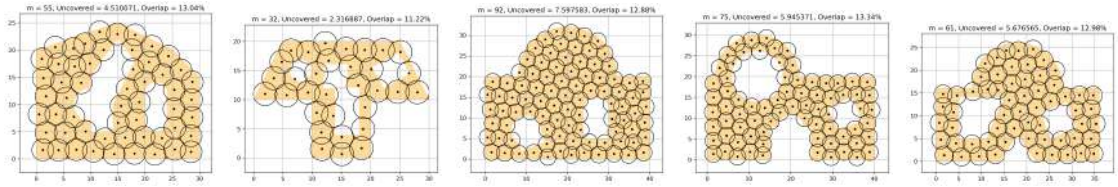
All experiments use a fixed disk radius $r = 2$ and tolerance $\tau = 0.03$. The evaluation follows a two-stage pipeline:(i) coverage generation using disk placement, and (ii) path optimization over the resulting way-points using LKH-D Heuristic.

5.2 Baseline Methods

We compare our approach with three typical deployment strategies:

5.2.1 Square Grid Deployment

In this strategy, disks are deployed on a grid over the bounding box of the region, and disks intersecting with the region are retained.



(a) (b) (c) (d) (e)

Figure 5.1: Coverage results across five maps.

5.2.2 Hexagonal Deployment

This strategy deploys disks on a hexagonal grid, which has denser packing and less overlap compared to square grids.

5.2.3 Greedy Deployment

This strategy deploys disks by a greedy strategy on an occupancy grid. In this strategy, at each iteration, frontier cells are determined by flood-fill, and the next disk center is determined by maximizing the coverage gain within the sensing radius on the sensing map, with reachability constraints to guarantee connectivity[17].

5.3 Path Optimization (LKH-D)

The problem is solved using the modified version of the Lin-Kernighan heuristic, called **LKH-D**. The algorithm starts from an initial feasible solution to the problem and iteratively tries to find improvements to the solution[20]. The difference between the standard Lin-Kernighan heuristic and the **LKH-D** is that the traversal path is a combination of the traversal distances and the turning costs. The formulation is as below..

$$C(\pi) = \alpha \sum_{i=1}^{n} d(\pi_i, \pi_{i+1}) \; + \; \beta \sum_{i=1}^{n} \theta(\pi_{i-1}, \pi_i, \pi_{i+1}) \tag{5.1}$$

We applied the same **LKH-D** heuristic to every method to compute the traversal paths over the disk centers. This is to ensure that the performance of each method is evaluated fairly without the bias of the placement method. The result of the experiment is that the better spatial distribution of disks leads to more efficient traversal paths. As shown in Fig. 5.2, the application

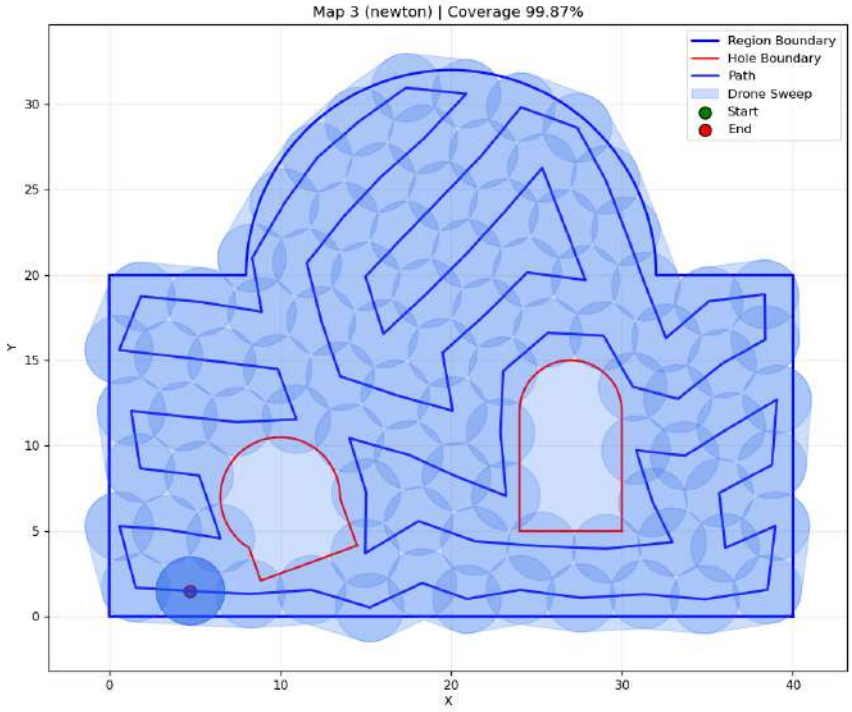


Figure 5.2: LKH-D Heuristic

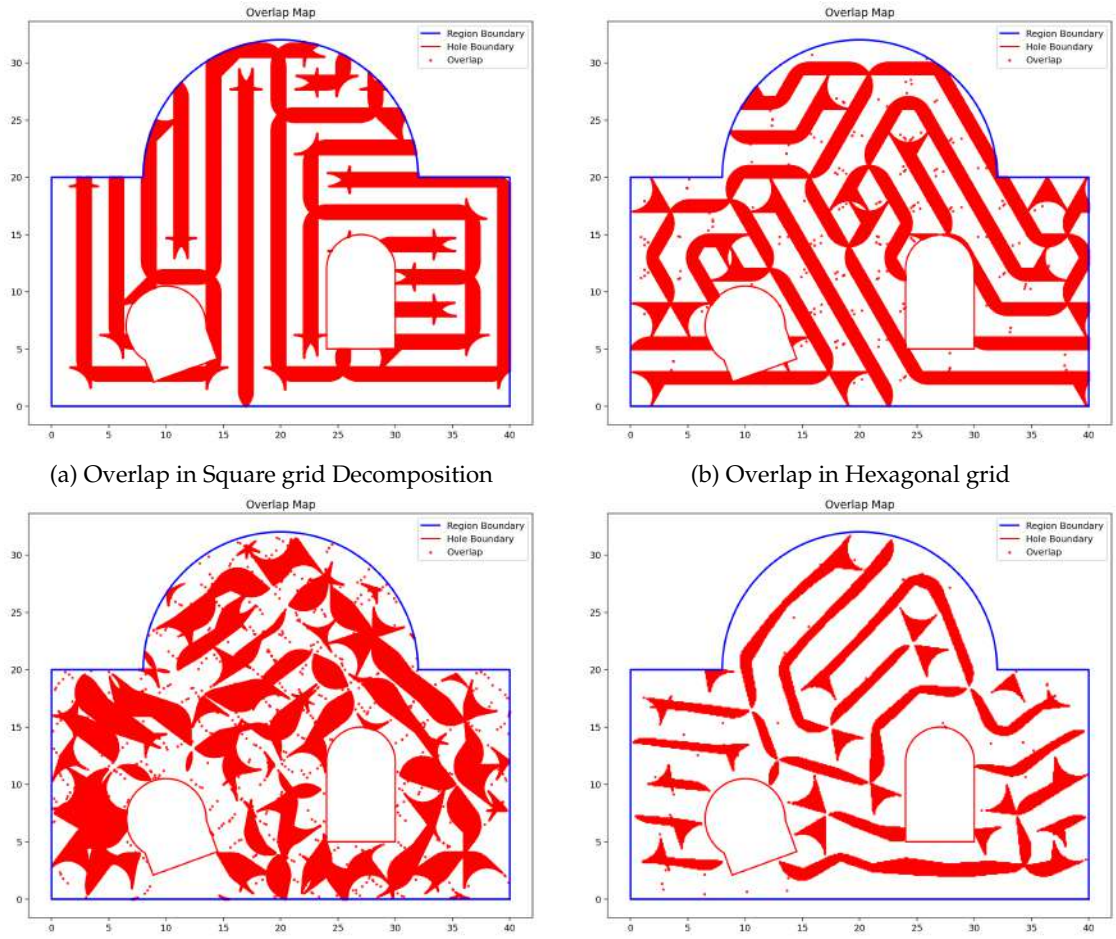
of the **LKH-D** heuristic is shown for the method applied to the map3.

5.4 Coverage Visualization

Figure 5.1 show the disk placement results in different environments. The proposed Shape optimization method works more closely to the geometric boundary and producing smoother coverage with significantly fewer overlaps.

5.5 Overlap Analysis

As seen in figure 5.3, the pattern in which the method's overlap also differs. The grid-based methods show a regular pattern due to their grid formation and the greedy deployment results in irregular clustering and overlapping. On the other hand, the Shape optimization method placed the disk along the perimeter of the regions that are not covered. This results in an even distribution of the overlapping pattern and reduces the level of redundancy. These figures show



(a) Overlap in Square grid Decomposition

(b) Overlap in Hexagonal grid

(c) Overlap in Greedy deployment

(d) Overlap in Shape optimization

Figure 5.3: Overlap distribution for four deployment strategies on Map 3: (a) Square grid, (b) Hexagonal grid, (c) Greedy deployment, and (d) Shape optimization.

the pattern for a given map-3, the results are consistent for all the environments.

5.6 Coverage vs Overlap Analysis

Figure 5.4 shows how disk count (m), coverage, and overlap interact for Map-3. Each point on the x-axis corresponds to a specific disk count, with results averaged over 10 independent trials. The error bars represent the standard deviation across these runs.

Notably, the standard deviation of $(|A| - G_m(x))$(covered region) across these runs remains very small for all values of m, which indicates low variability of our solution. That suggests the proposed method is stable and not sensitive to initialisation, and converges to a solution of

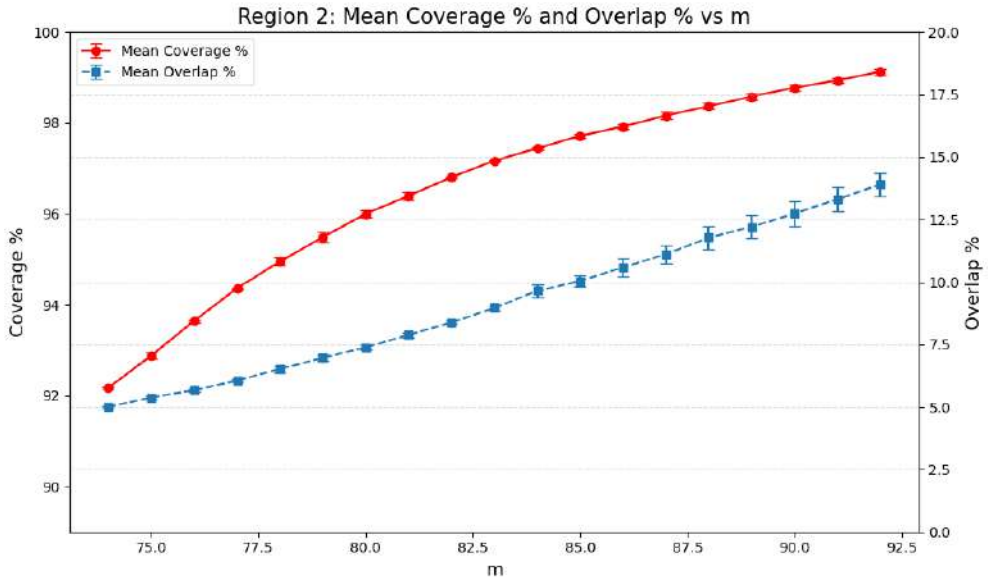


Figure 5.4: Mean coverage and overlap (%) versus number of disks m for Map 3 over 10 independent trials. Error bars denote standard deviation.

similar quality. Here we have just mentioned for Map-3, but a similar trend was seen across all maps. Results are also uploaded to the GitHub repository.

.

5.7 Energy Consumption

Figure 5.5 compares total energy cost across all methods using LKH-D planner. Averaged across all maps, the mean energy cost is 332.61 (Square), 318.63 (Hexagonal), 265.07 (Greedy), and 239.59 (Shape optimization).

5.7.1 Qualitative and Quantitative Comparison

To evaluate the effectiveness of the proposed framework, we conduct experiments on multiple benchmark environments with varying geometric complexity and obstacle configurations. Figures 5.6, 5.7, and 5.8 illustrate the qualitative behavior of different deployment strategies, while Table 5.2 provides a detailed quantitative comparison in terms of cost, coverage, overlap, path length, and turning effort.

Table 5.1: Quantitative comparison of square, hexagonal, greedy, and Shape optimization deployment strategies for five maps. Metrics include cost, coverage (%), overlap (%), path length, and turning angle. The proposed method consistently reduces overlap and traversal cost while maintaining high coverage.

Map	Method	Cost	Coverage (%)	Overlap (%)	Distance Cost	Turning (°)	Score	m
Map1	Square	285.73	**100**	34.22	283.82	**2430**	1.0191	92
	Hexagonal	291.25	99.99	31.10	288.89	3000	1.0365	78
	Greedy	229.53	98.78	29.67	224.99	5914.71	0.7073	90
	Shape optimization	**209.53**	99.86	**18.28**	**207.65**	2577.7	**1.8852**	58
Map2	Square	179.86	**100**	34.46	178.73	**1440**	1.0000	59
	Hexagonal	174.75	99.99	28.97	173.33	1800	1.0805	48
	Greedy	130.81	99.58	27.31	128.70	2796.91	0.7454	53
	Shape optimization	**121.13**	99.80	**15.58**	**119.52**	2070.93	**1.5238**	35
Map3	Square	465.73	**100**	38.03	462.55	**4050**	1.0000	148
	Hexagonal	435.09	99.99	30.73	431.41	4680	1.1850	115
	Greedy	390.08	99.06	34.49	383.88	8173.23	0.6584	150
	Shape optimization	**348.43**	99.87	**19.99**	**345.32**	4209.55	**1.8617**	95
Map4	Square	410.96	**100**	37.68	407.78	**4050**	1.0000	131
	Hexagonal	377.31	99.99	30.67	373.63	4680	1.1867	100
	Greedy	313.32	99.10	28.89	308.56	6340.62	0.7549	117
	Shape optimization	**285.32**	99.83	**20.26**	**282.07**	4313.67	**1.8111**	78
Map5	Square	320.77	**100**	35.72	317.87	3690	1.0000	102
	Hexagonal	314.73	99.99	30.91	312.00	3488	1.0795	84
	Greedy	261.6	98.48	32.02	256.24	7055.24	0.6727	104
	Shape optimization	**233.52**	99.85	**18.65**	**231.20**	**3114.83**	**1.9013**	65

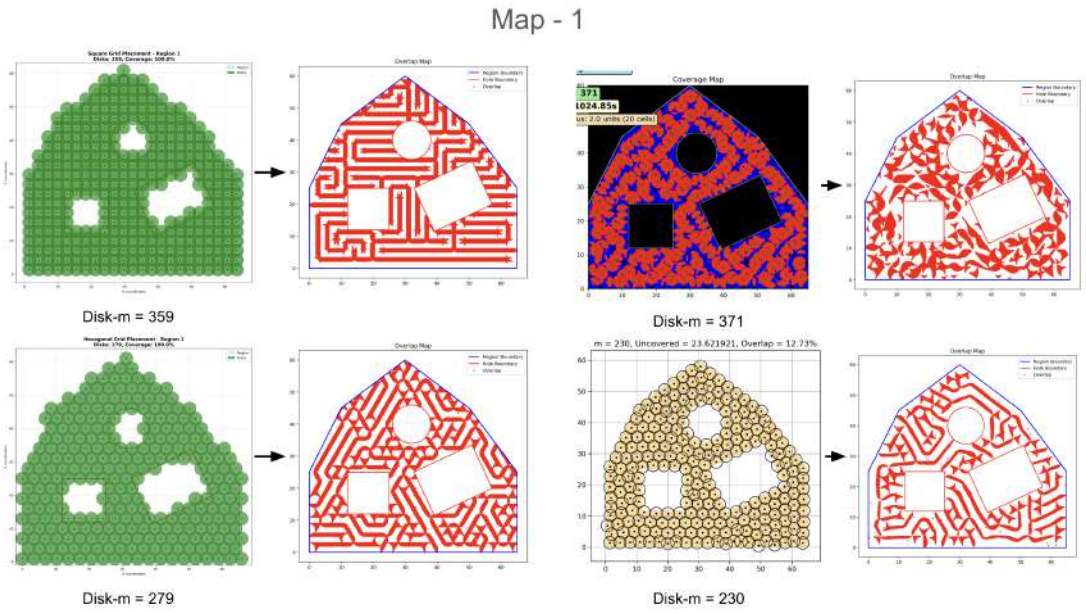


Figure 5.6: Comparison of coverage methods on Map 1

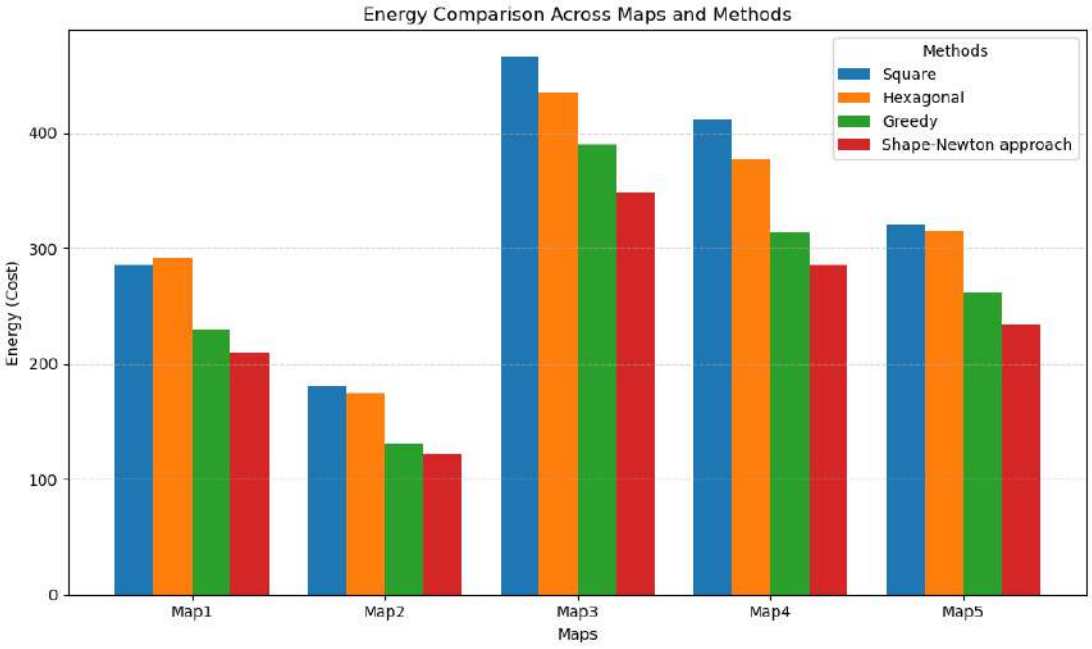


Figure 5.5: Energy Comparison Across Maps and Methods

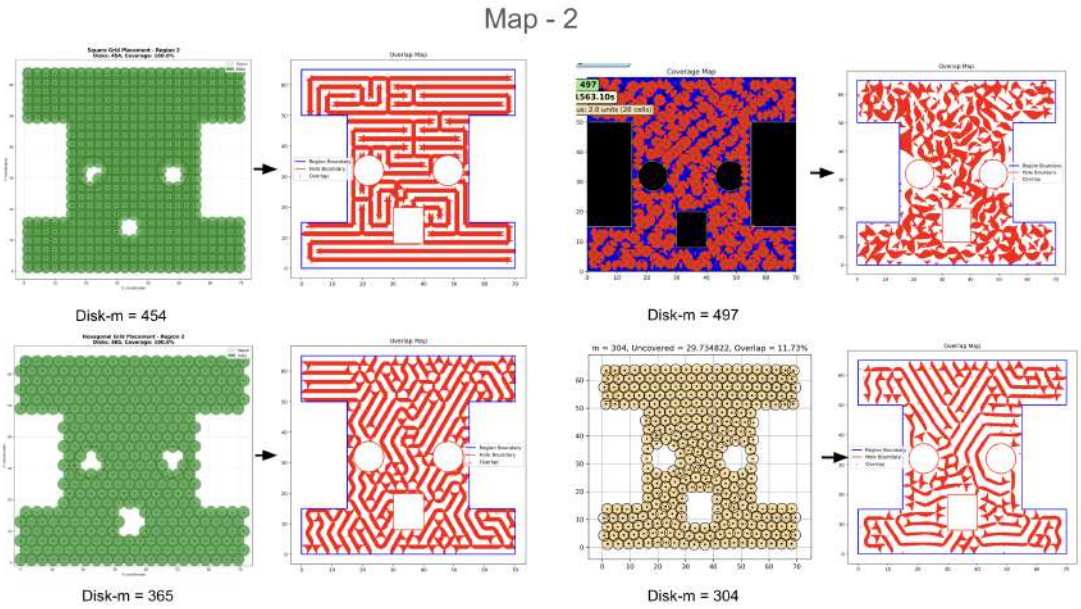


Figure 5.7: Comparison of coverage methods on Map 2

Table 5.2: Performance comparison across different methods and maps

Map	Method	Cost	Coverage (%)	Overlap (%)	Path Length	Turning (°)	Score	m
Map 6	Square	1150.84	100	37.82	1140.81	12780	1.0000	362
	Hexagonal	1085.99	99.99	31.60	1074.68	14400	1.1613	282
	Greedy	957.99	98.71	32.09	942.24	20974.3	0.6942	374
	Shape optimization	**871.61**	99.84	**20.24**	**863.18**	**11246.9**	**1.8760**	233
Map 7	Square	1438.74	100	37.06	1430.47	10530	1.0000	457
	Hexagonal	1419.90	99.99	31.86	1405.95	17760	1.0947	368
	Greedy	1274.04	98.81	33.15	1255.00	25642.9	0.6584	500
	Shape optimization	**1167.94**	99.86	**20.65**	**1157.56**	**13696.3**	**1.8824**	307
Map 8	Square	1359.48	100	37.09	1348.38	14130	1.0000	428
	Hexagonal	1323.79	99.99	31.71	1309.65	18000	1.1160	343
	Greedy	1191.08	98.68	34.29	1172.46	24983.9	0.6421	458
	Shape optimization	**1075.60**	99.87	**20.84**	**1066.00**	**12828.80**	**1.9015**	285

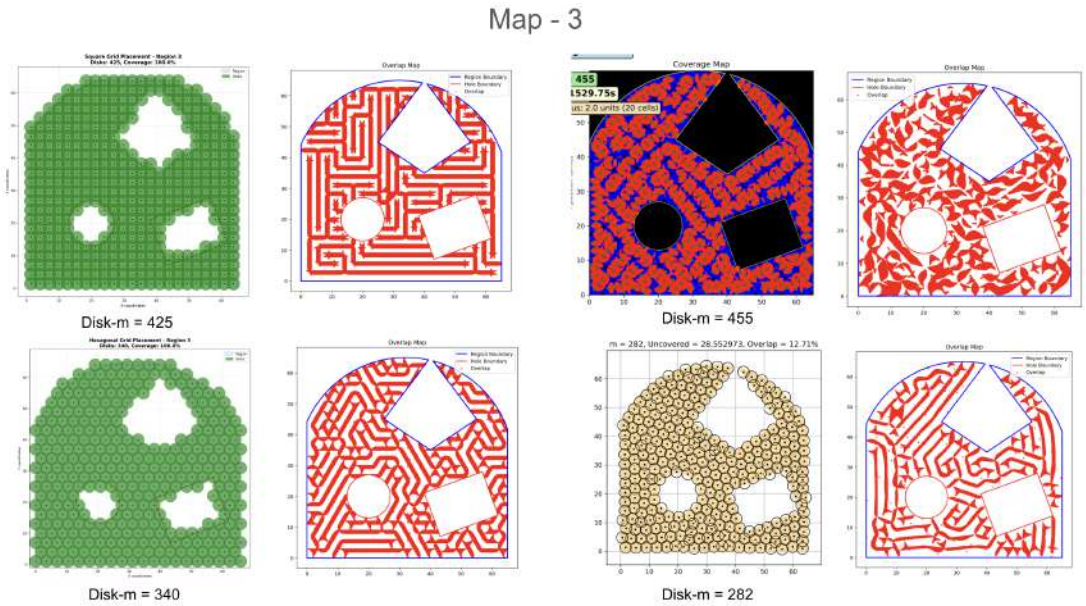


Figure 5.8: Comparison of coverage methods on Map 3

Discussion: A key observation from Table 5.2 is that all methods achieve near-complete coverage (approximately 99–100%), indicating that coverage completeness alone is not sufficient to distinguish performance. Instead, the primary differences arise in terms of efficiency metrics such as cost, overlap, path length, and turning effort.

Across all maps, the proposed Shape optimization method consistently achieves the **lowest cost**, demonstrating its effectiveness in reducing the overall energy required to cover the region. For instance, in Map 1, the cost is reduced from 1150.84 (square) and 1085.99 (hexagonal) to 871.61, corresponding to a reduction of approximately **24% and 19%**, respectively. Similar

trends are observed in Map 2 and Map 3, where the proposed method yields a cost reduction of approximately **18–20%** compared to grid-based approaches.

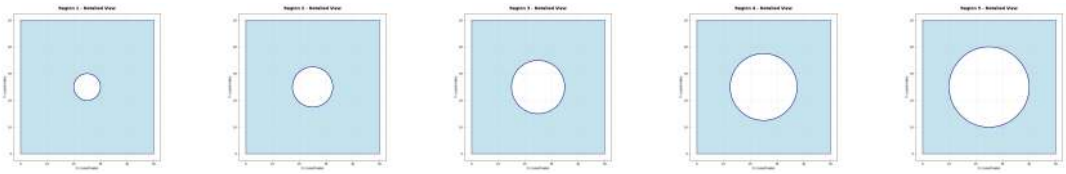
Another important factor is **overlap reduction**. Grid-based methods inherently introduce structured redundancy, leading to overlap values in the range of 31–38%. In contrast, the proposed method consistently maintains overlap around **20%**, resulting in an approximate **30–40% reduction in redundant coverage**. This behavior is clearly visible in Figures 5.6–5.8, where the coverage patterns generated by the proposed method appear more uniform and less cluttered.

Map 1 Analysis: In Map 1, the square grid deployment requires a significantly higher number of disks (m = 359) to achieve full coverage, resulting in redundant overlap and inefficient spatial utilization. The hexagonal deployment improves packing efficiency and reduces the number of disks (m = 279), but still suffers from structured placement that does not adapt well to irregular obstacle boundaries. In contrast, the proposed method achieves coverage with substantially fewer disks (m = 230), demonstrating a more efficient spatial distribution. The resulting coverage map shows reduced overlap and smoother traversal patterns, indicating improved energy efficiency.

Map 2 Analysis: For Map 2, which exhibits a more complex geometry with narrow passages and multiple obstacles, the limitations of discrete grid-based approaches become more pronounced. The square grid method requires m = 454 disks, while the hexagonal approach reduces this to m = 365. However, both methods exhibit excessive overlap and inefficient coverage in constrained regions. The proposed method significantly reduces the number of disks to m = 304, while maintaining complete coverage. The adaptive placement enables better handling of irregular boundaries and narrow regions, leading to reduced redundancy and improved coverage quality.

Map 3 Analysis: In Map 3, similar trends are observed. The square grid approach uses m = 425 disks, whereas the hexagonal method reduces this to m = 340. The proposed method further reduces the number of disks to m = 282, demonstrating consistent performance improvement across different environments. Additionally, the overlap maps show that the proposed approach produces more uniform coverage with fewer redundant regions, highlighting its ability to balance coverage completeness and efficiency.

The **path length** is also consistently lower for the proposed method across all maps, which directly contributes to reduced traversal time and energy consumption. For example, in Map



(a) (b) (c) (d) (e)

Figure 5.9: Variable obstacle size map.

2, the path length is reduced from 1430.47 (square) to 1157.56, highlighting the efficiency of optimized waypoint placement.

In terms of **turning cost**, the greedy method exhibits significantly higher values due to irregular and fragmented paths. Grid-based methods produce structured paths with discrete turning angles, whereas the proposed method generates smoother trajectories with continuous turning angles. This allows the system to better adapt to environmental geometry, reducing unnecessary directional changes while maintaining efficient coverage.

The qualitative results further support these observations. While square and hexagonal methods rely on rigid discretization, leading to inefficient handling of obstacles and irregular boundaries, the proposed method adapts dynamically to the environment. This results in fewer sensing units, reduced overlap, and smoother traversal paths.

5.7.2 Map with Variable Radius Circular Obstacle

To evaluate the sensitivity of energy performance with respect to obstacle size, we conduct experiments on a fixed map while systematically varying the obstacle radius. The environment boundary remains unchanged, see Fig. 5.9

Specifically, a circular obstacle is placed at the center of the map, and its radius is varied across five values: $r = 5$, $r = 7.5$, $r = 10$, $r = 12.5$, and $r = 15$.

All four deployment strategies—square grid, hexagonal grid, greedy, and the Shape optimization method were systematically evaluated across all experimental setups. This comprehensive analysis enables a detailed examination of the impact of increasing obstacle size on the key performance metric energy consumption.

As the obstacle radius increases, the available navigable space becomes constrained, leading to more restricted traversal paths. Under such conditions, grid-based approaches exhibit no-

table performance degradation due to their inherent rigidity and limited adaptability to irregular spatial configurations. In contrast, the proposed Shape optimization method demonstrates greater flexibility and robustness, effectively accommodating variations in obstacle geometry and maintaining superior performance across the evaluated metrics. These results are illustrated in Fig. 5.10.

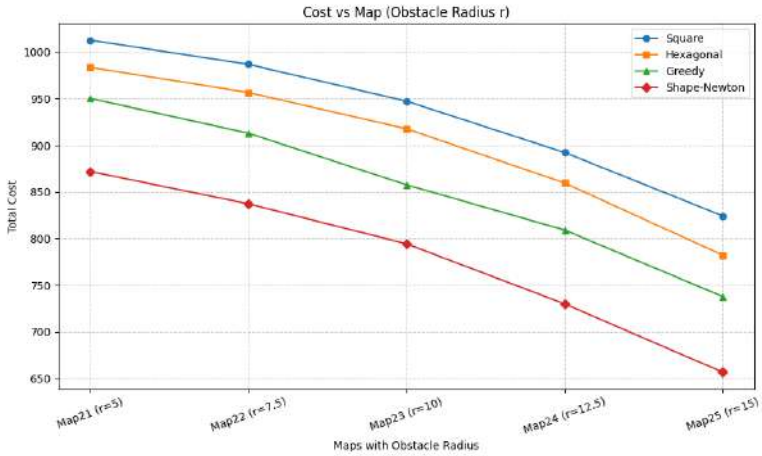


Figure 5.10: Energy (cost) vs. circular obstacle radius for different deployment strategies.

Chapter 6

Limitations

The ideal way to get optimal path to cover region should be done simultaneously perform decomposition and planning , but we have here decoupled the both problem and solve each problem individually. So far, this approach is limited in its application to static, fully observable environments as it presumes that the environment around us does not change. In practical applications, dynamic obstacles or other environmental variations necessitate additional online replanning and adaptive decision-making. Additionally, 100% coverage cannot be strictly ensured for every situation under the framework due to non-convexity of the underlying optimization problem. Concurrently, the computational complexity increases with the number of deployed disks, since both the continuous placement optimization and the path planning procedures scale with the overall problem size.

Chapter 7

Conclusion

We performed extensive experiments across multiple maps, observing that the proposed method consistently outperformed the classical deployment strategies, including Square, Hexagonal and greedy method. As the results indicate, Our method can provide similar or greater coverage and less redundant coverage due to a more uniform spatial distribution of the sensing disk. In this work, we proposed an end-to-end coverage path planning framework unifying continuous disk placement and energy-aware path optimisation. Given the Shape optimization deployment method determines a near optimal way-point distribution, resulting in a near optimal coverage.

Also, by utilizing an energy-aware LKH-D planner, the proposed approach produces a smoother and more efficient traversal path by handling both path length and turning cost. As a result, it achieves a significant reduction in overall energy consumption. On average, the proposed method reduces energy cost by approximately **9–10%** across all evaluated maps compared to the best baseline (greedy deployment). We experience that combining discrete-continuous optimised coverage generation with energy-aware path planning leads to efficiency improvement in both sensing and traversal. That makes a proposed framework particularly suitable for real-world robotics and UAV-based coverage applications. The implementation of the proposed framework is publicly available at: `https://github.com/Umesh-PersonalID/Adaptive_Placement`

Bibliography

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

[2] Sedat Dogru and Lino Marques. Eco-cpp: Energy constrained online coverage path planning. *Robotics and Autonomous Systems*, 157:104242, 2022.

[3] Andrew Kenneth Messing. Circle packing as a space decomposition method for robot path planning and coverage. *Master Thesis*, 2018.

[4] Nikunj Shah, Utsav Dey, and Kenji Nishimiya. End-to-end framework for robot lawnmower coverage path planning using cellular decomposition, 2025.

[5] Mark S Daskin. A maximum expected covering location model: formulation, properties and heuristic solution. *Transportation science*, 17(1):48–70, 1983.

[6] Zongyuan Shen, James P. Wilson, and Shalabh Gupta. C*: A coverage path planning algorithm for unknown environments using rapidly covering graphs. *IEEE Transactions on Robotics*, 42:1233–1253, 2026.

[7] Nguyen Duy Minh Phan, Yann Quinsat, Sylvain Lavernhe, and Claire Lartigue. Scanner path planning with the control of overlap for part inspection with an industrial robot. *The International Journal of Advanced Manufacturing Technology*, 98:629–643, 2018.

[8] Mohsen Ravanbakhsh, Yasin Abbasi-Yadkori, Maghsoud Abbaspour, and Hamid Sarbazi-Azad. A heuristic routing mechanism using a new addressing scheme, 2007.

[9] Jalil Modares, Farshad Ghanei, Nicholas Mastronarde, and Karthik Dantu. Ub-anc planner: Energy efficient coverage path planning with multiple drones. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6182–6189, 2017.

[10] Alexander V. Khorkov and Shamil I. Galiev. Optimization of a k-covering of a bounded set with circles of two given radii. *Open Computer Science*, 11(1):232–240, 2021.

[11] S. Poduri and G.S. Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 165–171 Vol.1, 2004.

[12] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

[13] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21:331 – 344, 2002.

[14] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331–344, 2002.

[15] David Ephraim Larkin. Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries. *arXiv preprint arXiv:1212.2476*, 2012.

[16] Megnath Ramesh, Frank Imeson, Baris Fidan, and Stephen L. Smith. Approximate environment decompositions for robot coverage planning using submodular set cover, 2024.

[17] Zhiyun Lin, Sijian Zhang, and Gangfeng Yan. An incremental deployment algorithm for wireless sensor networks using one or multiple autonomous agents. *Ad Hoc Networks*, 11(1):355–367, 2013.

[18] Ernesto G. Birgin, Antoine Laurain, Rafael Massambone, and Arthur G. Santana. A shape-newton approach to the problem of covering with identical balls. *SIAM Journal on Scientific Computing*, 44(2):A798–A824, 2022.

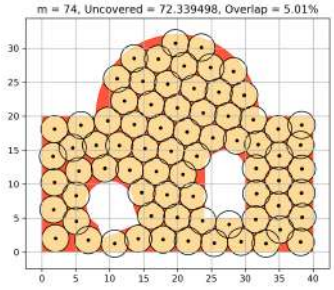
[19] Tauã M. Cabreira, Carmelo Di Franco, Paulo R. Ferreira, and Giorgio C. Buttazzo. Energy-aware spiral coverage path planning for uav photogrammetric applications. *IEEE Robotics and Automation Letters*, 3(4):3662–3668, 2018.

[20] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

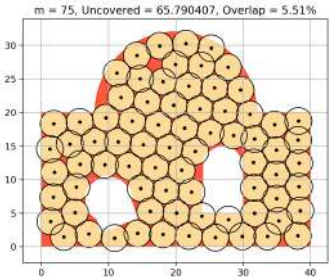
Appendix A

Appendix

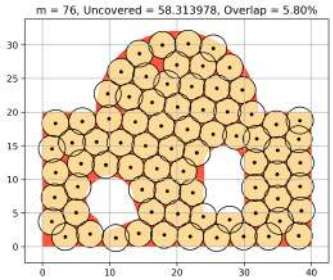
A.1 Experiments on Maps with Incremental Disk Placement Visualization



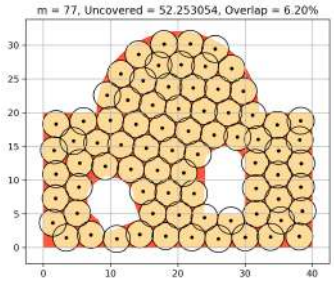
(a) Disk-m = 74



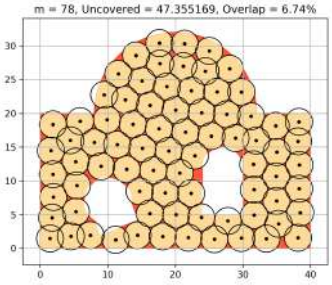
(b) Disk-m = 75



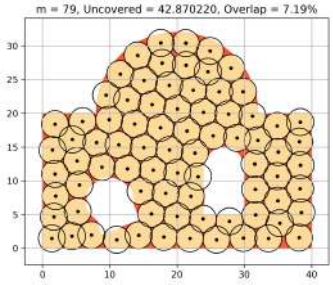
(c) Disk-m = 76



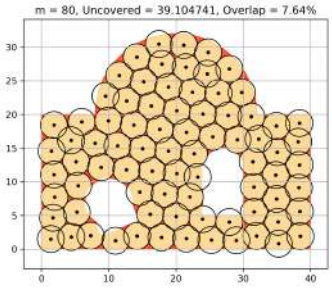
(a) Disk-m = 77



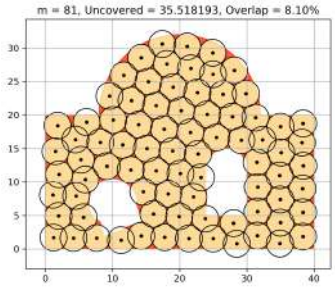
(b) Disk-m = 78



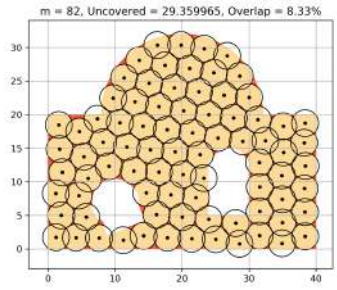
(c) Disk-m = 79



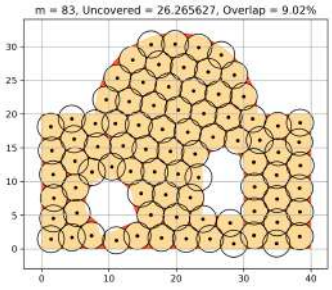
(a) Disk-m = 80



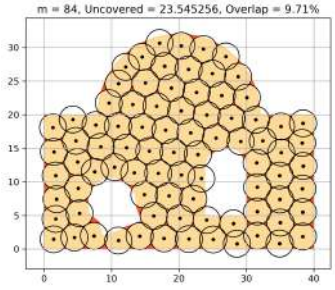
(b) Disk-m = 81



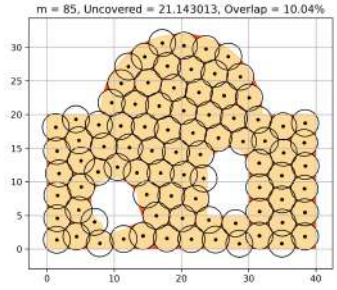
(c) Disk-m = 82



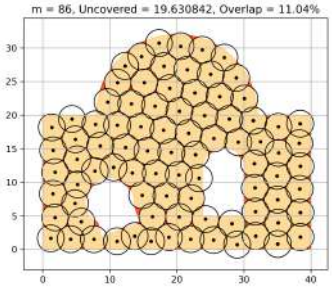
(a) Disk-m = 83



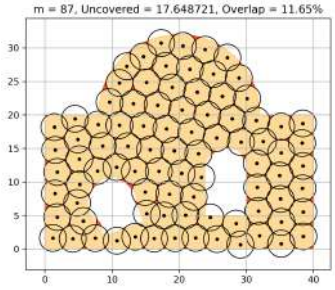
(b) Disk-m = 84



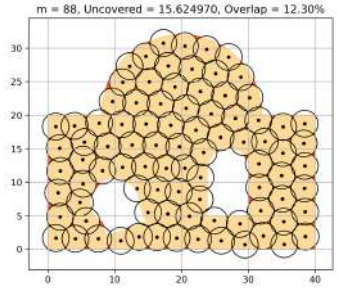
(c) Disk-m = 85



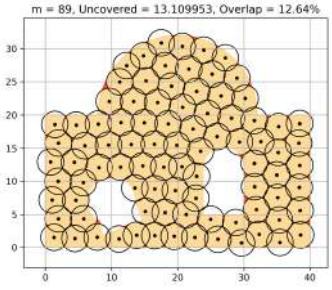
(a) Disk-m = 86



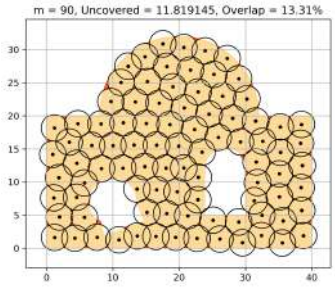
(b) Disk-m = 87



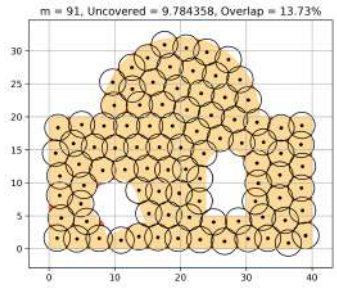
(c) Disk-m = 88



(a) Disk-m = 89



(b) Disk-m = 90



(c) Disk-m = 91

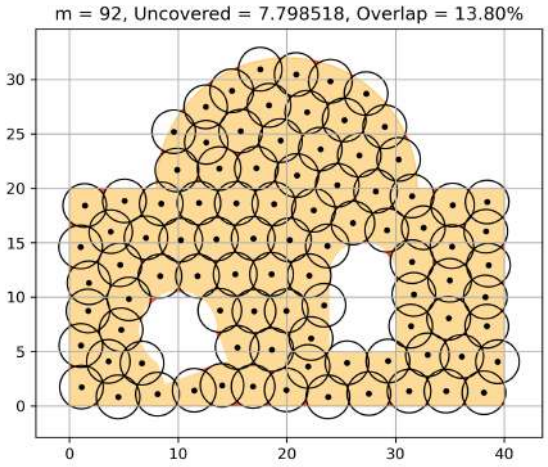


Figure A.7: Disk-m = 92