COVERAGE PATH PLANNING

SYED MAZHAR
u1988609

Supervisor: Tamara Petrovic
Co-supervisor: Felix

A Final Year Report submitted to School of Physical and Mathematical Sciences, Nanyang Technological University in partial fulfilment of the requirements for the Degree of

Intelligent Field Robotic Systems

June 2024

Abstract

Vivamus non vehicula tortor. Aenean tincidunt vel nisi id tincidunt. Aliquam eget sollicitudin ipsum. Curabitur quis malesuada magna. Integer nec condimentum sapien. Donec finibus rutrum tellus vel feugiat. Fusce auctor lectus id nunc dignissim, et ultricies eros volutpat. Pellentesque condimentum sed mi et placerat. Morbi semper, erat vitae sodales lobortis, orci enim viverra urna, et mollis mi dui ut ante. Vestibulum sodales lacinia scelerisque. Donec ut augue tellus. Etiam sit amet lacinia ipsum. Morbi commodo, sem sollicitudin dictum fringilla, libero quam varius erat, vel elementum felis velit ut elit. Fusce at sagittis eros. Phasellus tempor ullamcorper massa, consectetur scelerisque purus facilisis in. Vestibulum tempus eros vel sapien rutrum ullamcorper. Fusce at facilisis risus. Donec porttitor augue eu quam tempor, eget cursus risus commodo. Pellentesque sollicitudin, ipsum in tincidunt sagittis, nibh metus ultrices eros, egestas iaculis arcu justo non lectus. Praesent mollis orci ac sodales mollis. Sed suscipit sollicitudin ex id volutpat. Praesent laoreet molestie cursus. Nam dapibus dapibus bibendum. Donec maximus, velit non commodo porttitor, eros lacus ultrices ante, nec interdum purus mauris quis turpis. Nam dapibus quis lacus sed finibus. Aliquam venenatis, ex sit amet suscipit tempor, leo metus porta nulla, a egestas dui mi non risus. Duis vel libero ut mauris sodales cursus nec nec eros. Donec accumsan justo dolor. Quisque varius sem id dui dignissim, sed porta massa sodales. Nulla facilisi. Phasellus id metus ut enim eleifend volutpat. Fusce iaculis sodales tincidunt. Fusce quis convallis justo. Phasellus elementum, felis vel ultrices ultricies, ex nulla euismod justo, et.

Keywords: Blind source separation, Independent component analysis, Independent vector analysis, Sparse component analysis, Automatic speech recognition, Android application development.

Acknowledgement

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Table of Contents

Lists of Figures

List of Tables

1. Introduction

In recent years, the intersection of robotics and agriculture has garnered significant attention due to its potential to revolutionize farming practices and address various challenges in crop cultivation and management. One of the critical issues faced by farmers worldwide is the proliferation of weeds in agricultural fields, which not only compete with crops for essential resources but also pose significant threats to livestock health and food safety. In particular, weeds such as Rumex have been identified as major nuisances in grasslands, where their presence can contaminate forage intended for grazing animals, including cows.

The purity of grassland vegetation directly impacts the nutritional quality of forage consumed by livestock, thereby influencing the health and productivity of animals. The ingestion of contaminated forage, tainted with toxic or harmful weed species like Rumex, can lead to adverse health effects in cattle, affecting milk quality and quantity, as well as overall animal welfare. Therefore, the effective removal of weeds, particularly Rumex, from grasslands is imperative to ensure the integrity and safety of livestock feed, as well as the sustainability of agricultural operations.

In response to this pressing agricultural challenge posed by weed infestation in grasslands, this thesis endeavors to pioneer the development and implementation of an innovative coverage path planning (CPP) algorithm. Unlike conventional approaches, this algorithm prioritizes straight paths to optimize energy efficiency, thereby offering versatility across a spectrum of coverage path planning methodologies. While its primary application lies in weed removal within grasslands, its adaptability extends to diverse agricultural contexts, promising enhanced efficiency and sustainability in autonomous robotic operations. CPP plays a crucial role in guiding autonomous robotic systems to systematically traverse and cover an entire area of interest while minimizing overlap and maximizing efficiency. By leveraging advancements in robotics, artificial intelligence, and sensing technologies, CPP algorithms can enable autonomous agricultural robots to navigate complex terrain, identify weed-infested areas, and perform targeted weed removal tasks with precision and efficacy.

The primary objective of this research is to develop a coverage path planning (CPP) algorithm tailored for the intelligent and systematic removal of Rumex weeds from grasslands, thereby promoting the production of high-quality forage and safeguarding the health of grazing animals. This algorithm aims to optimize coverage efficiency, ensuring that all targeted weeds are effectively identified and removed by the robotic system. Through the integration of advanced robotics algorithms, the proposed approach seeks to enhance the efficacy of weed management practices while minimizing reliance on chemical herbicides and manual labor.

This thesis will explore various aspects of coverage path planning (CPP) in the context of agricultural robotics, including but not limited to:

- Review of Literature: A comprehensive overview of path planning techniques, including coverage path planning (CPP) and alternatives accommodating non-holonomic constraints. It explores classical and heuristic algorithms, their optimization strategies, and computational complexities, underscoring the importance of effective path planning in agricultural robotics. Through synthesizing diverse research streams, this review informs the development of an innovative CPP algorithm for weed removal.

- Methodology: We provide an in-depth exploration of the proposed CPP algorithm designed specifically for autonomous weed removal operations. We elucidate the underlying design principles governing the algorithm's development, emphasizing its adaptability to varying terrain and distinct dataset. Furthermore, we outline the computational framework, detailing the algorithm's implementation and optimization strategies to ensure efficient coverage and weed removal. Additionally, we discuss sensor integration strategies, elucidating how sensor data is utilized for perception and decision-making during weed removal tasks. Finally, we delve into the decision-making processes guiding the robotic system's actions, encompassing strategies for global and local path planning and obstacle avoidance to maximize efficiency and effectiveness in weed management operations.

- Experimental Setup: This section outlines the key considerations for field deployment and validation of the CPP algorithm in real-world grassland environments. It discusses the selection of robotic hardware, focusing on factors such as mobility, durability, and compatibility with the intended application. Additionally, it elaborates on the sensor configurations essential for environmental perception and obstacle detection during weed removal operations. The section also addresses field testing protocols, detailing the procedures for data collection, performance evaluation, and validation of the CPP algorithm's efficacy under actual operating conditions.

- Results and Analysis: This section presents a comprehensive analysis of experimental findings obtained through both simulation and real-world field trials, encompassing various parameters crucial for field robotics. It includes quantitative assessments of weed removal efficiency, coverage completeness, Energy usage, and to name a few. Furthermore, it compares the performance of the proposed CPP algorithm with other state-of-the-art approaches, considering factors such as path optimality, computational efficiency, and adaptability to different environments. Through a rigorous evaluation framework, this analysis provides valuable insights into the effectiveness and applicability of the CPP algorithm in practical agricultural settings.

- Discussion and Implications: This section engages in a thorough analysis of the research findings in the context of existing literature, elucidating the alignment and disparities between the proposed CPP algorithm and previous studies. It identifies the strengths and limitations of the algorithm, considering factors such as computational efficiency, scalability, and adaptability to diverse agricultural environments. Furthermore, it explores potential applications of the CPP algorithm beyond weed removal, highlighting its broader implications for sustainable agriculture and livestock management practices. By

synthesizing empirical evidence with theoretical insights, this discussion offers valuable perspectives on the future trajectory of autonomous robotics in agriculture.

By addressing these key components, this thesis aims to contribute to the advancement of autonomous weed management technologies in the agricultural sector, with a specific focus on improving the health and productivity of grassland ecosystems and ensuring the safety and quality of livestock feed. Through interdisciplinary collaboration and innovative engineering solutions, the integration of CPP algorithms into agricultural robotics holds promise for mitigating weed-related challenges and fostering sustainable farming practices for the benefit of farmers, consumers, and the environment alike.

By meticulously addressing these critical components, this thesis endeavors to significantly propel the evolution of autonomous weed management technologies within the agricultural domain. With a targeted emphasis on enhancing the health and productivity of grassland ecosystems, as well as safeguarding the safety and quality of livestock feed, the research aims to effect tangible improvements across multiple facets of agricultural sustainability.

Through a concerted effort to foster interdisciplinary collaboration and devise innovative engineering solutions, the integration of Coverage Path Planning (CPP) algorithms into agricultural robotics emerges as a beacon of hope for mitigating the myriad challenges posed by weed infestation. This holistic approach not only promises to alleviate immediate concerns for farmers but also extends its benefits to end consumers and the environment at large.

By empowering farmers with efficient and effective weed management tools, the research seeks to enhance agricultural productivity while reducing reliance on conventional, often environmentally detrimental, weed control methods. Moreover, by promoting sustainable farming practices, the integration of CPP algorithms holds the potential to foster long-term environmental stewardship and contribute to the preservation of natural ecosystems.

Ultimately, the overarching goal of this thesis is to usher in a new era of agricultural innovation—one characterized by the harmonious coexistence of technological advancement and environmental conservation. By harnessing the power of robotics and leveraging the principles of sustainable agriculture, the research endeavors to create a brighter future for farmers, consumers, and the planet alike.

need to include the algorithm of the paper as did in the first review paper

2. Literature Review

Path planning algorithms play a crucial role in the field of robotics and autonomous navigation, enabling vehicles to navigate efficiently through complex environments while adhering to various constraints. Over the years, researchers have developed a myriad of algorithms to address different aspects of path planning, ranging from basic techniques to more advanced methodologies.

2..1 Dubin's Path

In the realm of geometric analysis and constrained path planning, the pioneering work of L.E. Dubins stands as a cornerstone, providing profound insights into the properties and characteristics of paths subject to curvature constraints. Dubins' seminal exploration, outlined in his paper titled "On Curves of Minimal Length with a Constraint on Average Curvature," lays a robust foundation for understanding the fundamental principles governing constrained path planning, which has profound implications for various fields, including robotics, autonomous navigation, and geometric optimization.

At the heart of Dubins' research lies the fundamental concept of R-geodesics, which represent paths of minimal length under specified curvature constraints. This notion encapsulates the geometric essence of constrained paths, defining them as combinations of straight lines and circular arcs with a minimum radius of curvature, denoted as R. Dubins' theorem regarding the structure of R-geodesics in two dimensions provides a clear geometric understanding, asserting that such paths consist of no more than three segments, each comprising either a straight line or an arc of a circle with radius R. This theorem not only delineates the structure of minimal paths but also imposes precise constraints on their composition, revealing the inherent elegance and simplicity of paths subject to curvature constraints.

Furthermore, Dubins' exploration extends beyond characterization to the rigorous proof of the existence of R-geodesics. By leveraging mathematical tools such as Ascoli's theorem and drawing upon concepts from E. Schmidt's proof of A. Schur's Lemma, Dubins establishes the existence of paths of minimal length under average curvature constraints. This proof not only confirms the theoretical existence of such paths but also sheds light on their analytical and geometric properties, offering deeper insights into the nature of constrained curves.

Dubins' work represents a significant milestone in the study of geometric analysis and constrained path planning, providing not only a solution to a specific geometric problem but also a methodological framework applicable to a broader class of problems in path planning and optimization. The concept of R-geodesics, along with the analytical techniques employed by

Dubins, has inspired further research and development in the field, serving as the cornerstone for the design and implementation of algorithms for autonomous navigation systems.

In summary, Dubins' pioneering research has not only advanced our understanding of constrained path planning but has also catalyzed the development of algorithms and methodologies for efficient and optimal navigation in complex environments. His work continues to influence and inspire contemporary research in geometric analysis and autonomous systems, underscoring its enduring significance in the field of robotics and beyond. The profound insights gleaned from Dubins' contributions have paved the way for the development of innovative solutions to challenging path planning problems, shaping the landscape of autonomous systems and robotics research.

2..2 Travelling Salesman Problem (TSP) with Neighborhoods

The Traveling Salesman Problem (TSP) stands as a classic conundrum in the realm of optimization, challenging researchers to find the most efficient route for a salesman to visit a set of locations and return to the starting point while minimizing the total distance traveled. This problem, renowned for its computational complexity and practical applications in logistics and route planning, has spurred numerous investigations into variants that reflect real-world scenarios more accurately.

One such variant, explored in the second paper titled "Approximation Algorithms for TSP with Neighborhoods in the Plane," introduces the concept of TSP with neighborhoods (TSPN). In this formulation, destinations are not singular points but rather areas or neighborhoods, complicating the problem by requiring the salesman to visit each neighborhood at least once without specifying exact points for each visit.

To address the challenges posed by TSPN, the paper introduces innovative approximation algorithms tailored to different types of neighborhoods, such as line segments or complex shapes described as "fat" regions. A notable contribution is the development of a constant factor approximation algorithm for neighborhoods represented as line segments, signifying a significant advancement in the field of geometric optimization.

Central to the paper's methodology is the m-guillotine method, a novel approach that recursively subdivides the plane to maintain a near-optimal solution. Theoretical insights, including key theorems, underpin the effectiveness of the approximation algorithms, providing a solid foundation for their practical application in logistics, robotics, and geographic information systems.

Moreover, the research underscores the importance of geometric optimization in solving real-world problems, emphasizing the potential for these algorithms to be adapted and extended to other planning and routing problems with geographical constraints. By bridging the gap

between theoretical computer science and practical applications, the paper sets a precedent for future research, encouraging further exploration into more efficient algorithms, different neighborhood shapes, and the integration of dynamic elements into the TSPN framework.

This investigation into TSP with neighborhoods not only expands the scope of traditional TSP solutions but also offers valuable insights into optimizing spatial planning tasks in diverse fields. The methodologies and algorithms developed in this research pave the way for more efficient route planning and logistical operations, driving advancements in both theoretical and applied aspects of optimization and geometric analysis.

2..3 On the point-to-point and traveling sales- person problems for Dubins' vehicle.

In the pursuit of efficient and realistic navigation solutions for autonomous vehicles and robotics, the challenges associated with path planning for Dubins' vehicles, constrained by their minimum turning radius, have garnered significant attention. The paper titled "On the point-to-point and traveling salesperson problems for Dubins' vehicle" offers a comprehensive analysis of these challenges and presents innovative methodologies to address them.

At the core of the research are two fundamental problems: the point-to-point shortest path problem (PTP) and the traveling salesperson problem (TSP) tailored for Dubins' vehicles. These problems are crucial for developing algorithms capable of computing optimal or near-optimal paths for vehicles subject to nonholonomic constraints.

The study introduces a novel approach to solving the point-to-point problem for Dubins' vehicles, leveraging the vehicle's constrained dynamics to compute the shortest possible path between two points. By providing a detailed analysis of mathematical models and computational algorithms, the authors emphasize the importance of considering nonholonomic constraints in path planning for autonomous vehicle navigation.

Expanding the complexity of navigation problems, the paper addresses the traveling salesperson problem (TSP) for Dubins' vehicles, where the vehicle must visit a set of locations in the shortest route without revisiting any. Recognizing the computational difficulty of finding exact solutions for larger sets of points, the authors explore heuristic and approximation algorithms to approximate solutions efficiently.

The methodology proposed in the paper is grounded in rigorous mathematical formulations and computational geometry, offering a blend of theoretical and practical solutions to the challenges of path planning for Dubins' vehicles. By addressing both the PTP and TSP, the

authors provide a holistic view of navigation problems, showcasing the interplay between optimal path planning and the inherent constraints of vehicle motion.

Overall, this paper significantly contributes to the literature on autonomous vehicles and robotics, particularly in the context of path planning for nonholonomically constrained vehicles. The exploration of point-to-point and traveling salesperson problems for Dubins' vehicles not only elucidates inherent challenges but also provides practical solutions applicable in real-world scenarios. The rigorous approach and innovative solutions presented lay a solid foundation for future research, promising the development of more efficient and realistic navigation systems for autonomous vehicles.

2..4 Algorithms for the Traveling Salesman Problem with Neighborhoods Involving a Dubins Vehicle

The paper under review delves into the intricacies of the Dubins Traveling Salesman Problem (DTSP), a variant of the classic Traveling Salesman Problem (TSP) that incorporates curvature constraints into the salesman's path between waypoints. This problem poses significant computational challenges due to its NP-hard nature, necessitating the development of approximation algorithms for practical solutions.

The authors begin by highlighting the limitations of existing algorithms for the DTSP, particularly those derived from the Euclidean TSP (ETSP) framework. Through rigorous analysis, they demonstrate that these methods fail to provide satisfactory solutions when waypoints are densely distributed relative to the minimum turning radius of the salesman's path. This critical insight underscores the need for DTSP-specific algorithms tailored to address the curvature constraint effectively.

To overcome these limitations, the paper introduces two novel heuristics: the nearest neighbor heuristic and an algorithm based on heading discretization. The nearest neighbor heuristic constructs a solution iteratively by selecting the closest not-yet-visited waypoint according to the Dubins metric, which accounts for the curvature constraint. This approach provides a complete DTSP solution, including waypoint ordering and the required heading at each waypoint.

Building upon insights from curvature-constrained shortest path problems, the second heuristic discretizes potential headings at each waypoint and formulates a generalized asymmetric traveling salesman problem (ATSP). This ATSP is then reduced to a standard ATSP, solvable using existing algorithms. Additionally, the authors propose an algorithm that discretizes potential headings and constructs a graph representing Dubins distances between

waypoints. By solving the resulting ATSP, this algorithm approximates a tour through all waypoints, directly addressing the curvature constraint.

In conclusion, the paper contributes significantly to understanding and solving the DTSP by proving its NP-hardness, illustrating the limitations of existing approximation methods, and introducing innovative heuristics that better account for curvature constraints. These advancements not only provide more effective solutions to the DTSP but also pave the way for future research in developing polynomial-time approximation algorithms free from the highlighted limitations.

2..5 Optimizing Autonomous Vehicle Paths with the Dubins Traveling Salesman Problem

In the realm of autonomous vehicle navigation, particularly for Unmanned Aerial Vehicles (UAVs), the task of efficiently covering areas with intersecting regions of interest poses a significant challenge. This problem, often encountered in surveillance, reconnaissance, and search-and-rescue operations, involves finding the shortest route for a vehicle to visit multiple target areas while adhering to its physical constraints, such as minimum turning radius and flight restrictions. Mathematically, this challenge can be framed as the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN), an optimization problem that seeks to minimize the total distance traveled while ensuring that each neighborhood is visited at least once.

The Intersecting Regions Algorithm (IRA), as discussed in the literature, presents a novel approach to address the complexities of the DTSPN. This algorithm leverages the concept of sampling within intersecting regions to construct feasible tours for autonomous vehicles. Unlike traditional methods that overlook the overlap among regions, IRA explicitly considers this overlap, resulting in more efficient path planning and optimized route selection.

One crucial aspect of IRA is its theoretical underpinning, which provides insights into the efficiency and completeness of the generated paths. Theorem 3.4 establishes that the path length produced by IRA will not exceed that of previous methods, highlighting its effectiveness in minimizing travel distances. Additionally, Corollary 3.5 ensures the resolution completeness of IRA, guaranteeing adequate coverage of all designated regions, even as the sample size increases. These theoretical assurances bolster confidence in IRA's reliability and suitability for real-world applications.

Moreover, IRA addresses the computational complexity associated with path planning by employing an algorithmic structure that scales polynomially with the problem size. This scalability is essential for practical applications where computational resources are limited, ensuring that IRA remains viable even for large-scale scenarios with numerous regions of interest.

Monte Carlo simulations further validate IRA's practical value, demonstrating significant performance improvements in scenarios with high degrees of region overlap.

In summary, the Intersecting Regions Algorithm (IRA) represents a significant advancement in solving the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN) for autonomous vehicles, particularly UAVs navigating through environments with overlapping regions of interest. By efficiently sampling within these intersections and optimizing the resultant path, IRA ensures comprehensive coverage while maintaining optimal or near-optimal path lengths. Its proven efficiency, backed by both theoretical analysis and empirical validation, positions IRA as a promising solution for enhancing the operational capabilities of UAVs in complex surveillance and reconnaissance missions.

To further substantiate the significance of IRA and its implications for autonomous vehicle navigation, additional sources from the literature can be explored. These may include research papers, conference proceedings, or technical reports that discuss related topics such as path planning algorithms, optimization techniques, and applications of UAVs in various domains. By integrating insights from multiple sources, a comprehensive understanding of IRA and its contributions to autonomous vehicle navigation can be attained, enriching the discussion and analysis in the literature review section.

2..6 Optimizing Dubins Paths with Convex Optimization Techniques

The optimization of paths for Dubins vehicles, constrained by their minimum turning radius, presents a complex problem that has garnered significant attention in the field of robotics and autonomous navigation. In a recent research paper, authors delve deep into this intricate challenge, aiming to find the shortest path for Dubins vehicles while adhering to their unique constraints.

To tackle this problem, the authors adopt a systematic approach, focusing on polygonal paths with specified waypoints. A crucial aspect of their methodology involves spacing these waypoints at a distance that facilitates the construction of Dubins paths while avoiding overly complex path components. By initially setting the minimum distance between waypoints at 8.6 units, they streamline the path construction process and enable the application of convex optimization techniques to identify the shortest possible path.

However, the optimization problem posed by Dubins paths is not without its complexities. Standard optimization methods face challenges due to the nature of the function being minimized, which involves inverse trigonometric functions. Recognizing this, the authors turn to the ellipsoid method for solving the convex optimization problem. Through detailed analysis, they demonstrate that, under specific constraints and in the absence of sharp turns, the optimal path

can be approximated within a desired accuracy in a time complexity that aligns with theoretical expectations for the ellipsoid method in high-dimensional spaces.

An intriguing extension of their work addresses scenarios with fixed initial and final path directions, introducing additional complexities due to the potential need for circular arcs greater than π at the path's endpoints. The authors adeptly navigate this challenge by considering multiple path types with fixed orientations, ensuring robust analysis and applicability even under constrained conditions. This adaptability enhances the practical utility of their findings, offering solutions that can accommodate various initial and final path configurations.

Moreover, the potential applications of the authors' techniques extend beyond Dubins path optimization. By reducing the problem to a series of convex optimization tasks, the paper sets the stage for more efficient path planning in environments with polygonal obstacles. Once the sequence of contact points and the path's interaction with obstacles are determined, this approach promises significant advancements in robotic navigation, autonomous vehicle routing, and other domains where obstacle avoidance is crucial.

However, the paper also acknowledges certain limitations and suggests avenues for further research. Exploring the convexity properties of paths involving more complex maneuvers, such as consecutive circular arcs (CCC-paths), could lead to a more generalized and flexible framework for path planning. Additionally, alternative strategies for identifying the shortest path warrant exploration, hinting at the potential for deeper insights into the underlying structure of the problem.

In summary, the research paper presents significant advancements in the optimization of Dubins paths, offering novel insights and methodologies that could greatly benefit the fields of robotics and autonomous navigation. By addressing both theoretical challenges and practical considerations, the paper lays the groundwork for future explorations into more efficient and adaptable path planning strategies.

2..7 Efficient Path Planning for Autonomous Vehicles in Dynamic Environments

Navigating narrow and complex environments poses significant challenges for autonomous vehicles, particularly car-like robots operating in constrained spaces. In a recent paper, researchers propose a sophisticated path planning method tailored to address these challenges, emphasizing the generation of paths with continuous curvature to ensure smooth operation.

The authors introduce a two-phase planning approach that combines global and local strategies to efficiently navigate through narrow areas. In the first phase, they utilize the RTR (Rotate-Translate-Rotate) planner, an adaptation of the Rapidly Exploring Random Tree (RRT) method. This planner focuses on generating paths comprised of straight movements

and in-place turning, simplifying complex maneuvering into manageable actions suited for navigating constrained spaces.

The second phase involves refining the global path using a local planning procedure called TTS. This planner approximates the initial path by a sequence of paths that adhere to the vehicle's curvature constraints, ensuring smooth and feasible trajectories. The TTS planner can generate paths with continuous curvature turns (CC-turns) and straight segments, providing a flexible solution adaptable to various environmental constraints.

Extensive simulation studies compare the proposed RTR+TTS planning algorithm with other methods based on metrics such as success rate, computation time, and path quality. Results demonstrate that the proposed method achieves high success rates in finding feasible paths while generating paths that are comfortable and "natural" for passengers. In scenarios like parallel parking and navigating narrow corridors, the RTR+TTS method outperforms other approaches in terms of path simplicity and adherence to curvature constraints.

The paper concludes by highlighting the effectiveness and potential of the planning method for autonomous vehicle navigation in challenging environments. The authors propose further research to explore the algorithm's application in dynamic and unknown spaces, as well as its integration with online map-building algorithms. Ultimately, the goal is to extend the capabilities of autonomous vehicles to operate safely and efficiently in a wider range of scenarios, enhancing their practicality for everyday use.

2..8 Optimizing Dubins Traveling Salesman Problem with Neighborhoods: A Novel Algorithmic Approach

The Dubins Traveling Salesman Problem with Neighborhoods (DTSPN) poses a unique challenge in autonomous vehicle navigation, particularly for unmanned aerial vehicles (UAVs) constrained by minimum turning radius requirements. This paper addresses the DTSPN by introducing a specific formulation known as the Dubins Touring Regions Problem (DTRP), aimed at finding an optimal sequence of configurations for entering and exiting regions while minimizing the total path length.

Unlike the traditional Traveling Salesman Problem (TSP), where the goal is to find the shortest route visiting a set of points, the DTSPN involves regions or neighborhoods, requiring the vehicle to enter each region at least once. This problem is crucial in applications such as autonomous drone surveillance, delivery systems, and robotic exploration, where vehicles must efficiently navigate through specified areas while considering nonholonomic constraints.

The proposed algorithm for solving the DTRP leverages local iterative optimization techniques, taking into account the unique dynamics of Dubins vehicles. It begins with an initial sequence of region visits derived from solving an Euclidean TSP (ETSP), which serves

as a proxy for the region centers. The algorithm then iteratively refines the entry and exit configurations to minimize the total tour length.

A key aspect of the solution method is its decoupled approach, optimizing the heading and position of each entry point independently. This simplification, facilitated by mathematical techniques that project the problem into a more manageable form, allows for efficient local optimization. The algorithm iterates until no further improvements can be made or a termination criterion is met, incorporating strategies to escape local minima by adjusting vehicle headings and repositioning at region boundaries.

Empirical validation of the algorithm demonstrates its effectiveness and efficiency across various scenarios, including different region shapes and configurations, particularly in dense environments where regions are close together. Comparative analysis against existing evolutionary algorithms showcases the proposed method's ability to produce high-quality solutions with significantly reduced computational time, making it suitable for real-time applications on modest hardware, such as onboard computers in UAVs.

In conclusion, the paper significantly advances the applicability of Dubins vehicle path planning in practical scenarios by introducing the DTRP formulation and proposing an efficient local iterative optimization algorithm to solve it. This approach holds promise for enhancing autonomous vehicle routing and navigation in complex environments, with potential real-world implementation in various autonomous systems applications.

2..9 Optimal Solution for Generalized Dubins Interval Problem and Its Application to Dubins Traveling Salesperson Problem with Neighborhoods

The Generalized Dubins Interval Problem (GDIP) extends the traditional Dubins path problem by incorporating constraints on the initial and final orientations within specified intervals. This paper introduces and evaluates an optimal solution for the GDIP and explores its application to the Dubins Traveling Salesperson Problem with Neighborhoods (DTRP). The GDIP seeks to find an optimal path that minimizes the distance between two points while adhering to turning radius constraints and constraints on starting and ending orientation intervals.

The authors propose a novel algorithm to efficiently solve the GDIP, demonstrating its utility in solving the DTRP by providing a tight lower bound for the solution cost. This lower bound facilitates informed sampling strategies and enables accurate assessment of solution quality.

In evaluating the GDIP solution, the paper assesses its computational requirements and compares them with traditional Dubins maneuvers. The findings indicate that while the GDIP

solution computes optimal solutions faster, it incurs a higher computational cost due to the increased complexity of considering orientation intervals and additional maneuver types.

Furthermore, the performance of the algorithm in solving the DTRP is evaluated by analyzing its convergence and efficiency as the number of target regions increases. Results show that the solution cost closely approximates the lower bound with increasing resolution, and the computational time scales approximately linearly with the problem size, demonstrating the algorithm's effectiveness and efficiency in solving complex DTRP instances.

The paper concludes that the GDIP significantly enhances the solution of the DTRP by providing a tight lower bound for the solution cost, enabling effective informed sampling and accurate estimation of the solution's optimality gap. The experimental results support the hypothesis that the proposed solution can achieve high-quality solutions for problems with up to 100 target regions with an optimality gap of around 1 percent within reasonable computational times.

In summary, this research extends the Dubins path problem to include orientation interval constraints, offering an optimal solution for the GDIP and applying it to improve the efficiency and accuracy of solving the DTRP. The proposed algorithm demonstrates computational efficiency, scalability, and practical means to evaluate solution quality, making it a valuable tool for applications requiring efficient and reliable path planning in complex environments.

2..10 Reeds-Shepp Paths: Enhancing Navigation Flexibility for Autonomous Vehicles

Reeds-Shepp paths, introduced by J.A. Reeds and L.A. Shepp in their seminal 1990 paper, offer a versatile solution to the optimal path planning problem for vehicles capable of moving both forwards and backwards. Unlike Dubins paths, which only consider forward movement, Reeds-Shepp paths provide increased flexibility and maneuverability in navigating tight or complex environments, making them well-suited for a range of applications, including robotics and autonomous vehicle navigation.

Dubins paths, optimized for scenarios where vehicles can only move forward and are constrained by the minimum turning radius, offer simplicity in path computation but may lack maneuverability in obstacle-dense environments. In contrast, Reeds-Shepp paths introduce reverse movement capabilities, effectively doubling the set of possible maneuvers. This additional capability allows for shorter or more feasible paths in scenarios where Dubins paths may fail to provide a viable solution or result in longer paths.

In agricultural robotics, where precise navigation is crucial for tasks such as weed removal, Reeds-Shepp paths offer significant benefits. Agricultural fields often pose challenges such as tight spaces between crop rows, irregular boundaries, and obstacles like rocks or uneven terrain. The flexibility provided by Reeds-Shepp paths enables agricultural robots to navigate more

efficiently in these environments, reducing time and energy consumption. Moreover, the ability to move in reverse allows for more precise positioning during weed removal, enhancing task effectiveness and minimizing crop damage.

Reeds and Shepp not only introduced Reeds-Shepp paths but also conducted a comprehensive analysis of their mathematical properties, including minimum-length paths under both forward and backward movement constraints. Their work laid the groundwork for subsequent research in optimal control and path planning, emphasizing the importance of considering a vehicle's full range of capabilities when designing navigation algorithms.

In summary, while Dubins paths offer simplicity for forward movement constraints, Reeds-Shepp paths extend this capability by incorporating reverse movements, providing a more flexible and efficient solution for navigating complex environments. For agricultural ground robots engaged in weed removal tasks, the use of Reeds-Shepp paths can significantly enhance navigation efficiency and task performance, making them a preferred choice for such applications. The concepts and mathematical foundations established in the Reeds-Shepp paper continue to influence the development of path planning algorithms and autonomous systems design across various domains.

later Obstacle literature review:

2..11 Obstacle Avoidance in Static Environments with Non-Holonomic Constrained Robots: Intro

Obstacle avoidance is a critical aspect of path planning in robotics, particularly in scenarios where the environment contains static obstacles. The primary objective of obstacle avoidance algorithms is to enable robots to navigate safely through cluttered environments while avoiding collisions with stationary objects. This task becomes especially challenging when considering non-holonomic constraints, which restrict the motion of the robot, often resulting in complex and constrained maneuvering capabilities.

In environments with known static obstacles, the focus shifts towards static obstacle avoidance approaches, where the complete information about the map and obstacle locations is available at the beginning of the planning process. These approaches typically leverage the concept of configuration space (C-space), which represents all feasible configurations that the robot can attain without encountering obstacles. By mapping the robot's motion in the real world to its corresponding configurations in the C-space, path planning algorithms can efficiently search for collision-free paths.

One of the primary challenges in static obstacle avoidance is the computational complexity involved in searching through the configuration space to find feasible paths. As the dimensionality of the C-space increases with the number of degrees of freedom of the robot and the complexity of the environment, the search space grows exponentially. This exponential growth presents a significant computational burden, making it challenging to find optimal paths, especially in high-dimensional spaces.

Non-holonomic constraints further exacerbate the complexity of the path planning process. Non-holonomic constraints arise when the robot's motion is restricted, preventing it from executing certain maneuvers or moving in certain directions. For instance, vehicles with differential steering, such as cars or robots with caster wheels, exhibit non-holonomic constraints as they cannot move sideways or rotate freely like a holonomic robot.

Incorporating non-holonomic constraints into path planning algorithms requires careful consideration to ensure that the generated paths adhere to the robot's kinematic limitations while still achieving the desired navigation objectives. This often involves constraining the search space in the configuration space to only consider feasible motions and incorporating motion primitives tailored to the robot's kinematics. Motion primitives are pre-defined elementary motion segments that capture the feasible motions of the robot within its kinematic constraints.

Despite the challenges posed by static obstacles and non-holonomic constraints, advancements in path planning algorithms have led to the development of efficient and

reliable approaches for obstacle avoidance in static environments. Techniques such as C-space representation, heuristic search algorithms, and motion primitives have been instrumental in enabling robots to navigate safely and effectively through cluttered environments while adhering to their non-holonomic constraints.

Overall, obstacle avoidance in static environments with non-holonomic constrained robots represents a challenging yet crucial problem in robotics. By addressing these challenges through innovative algorithmic techniques and leveraging advancements in computational resources, researchers continue to make strides towards developing robust and efficient path planning solutions for a wide range of real-world applications.

2..12 Random Walk Algorithms for Coverage Path Planning in Robotics

Random walk (RW) algorithms represent a stochastic approach to coverage path planning (CPP), drawing inspiration from the random movements observed in natural phenomena, such as animal foraging behavior. In environmental exploration and coverage tasks, RW algorithms have gained attention for their simplicity and adaptability. These algorithms typically involve the robot making random movements within the environment to scan and explore uncharted areas.

One variant of the RW method involves a fixed linear approach, where the robot randomly turns at angles and moves in straight lines until it encounters a wall or obstacle boundary. This approach, although straightforward, may lead to inefficient coverage, as the robot's movement is not optimized for thorough exploration. However, it has been used in cleaning systems where the objective is to cover as much area as possible within a confined space.

On the other hand, the variable step method in RW algorithms offers more flexibility and adaptability, particularly in collaborative mobile robot swarm systems. In this method, the robot computes a set of RW directions based on the probability distribution of step lengths, allowing for more diverse and exploratory movements. Variants of the variable step method include Brownian motion (BM) and Lévy flight (LF). BM involves the robot moving in step lengths with a given distribution and randomly turning, while LF entails the robot traveling distances based on Lévy's probability distribution.

Researchers have explored the application of RW algorithms in various contexts, including swarm robotics and multi-robot systems. For example, Martinez et al. proposed a swarm robot system using BM-based RW to enhance area coverage. In this approach, each robot behaves like a particle controlled by signals in the environment, contributing to collective exploration and coverage. Additionally, pheromone-based communication and LF search strategies have been employed to improve efficiency in unknown environments.

One of the main advantages of RW algorithms is their simplicity and minimal sensor requirements. Unlike some other CPP algorithms, RW algorithms do not rely heavily on

localization sensors, making them easy to deploy and implement. However, their effectiveness is limited in larger environments with obstacles, as the random movements may lead to inefficient coverage and the potential for revisiting the same areas multiple times.

In summary, RW algorithms offer a simple yet effective approach to coverage path planning, particularly in small and relatively obstacle-free environments. While they may lack the sophistication of some heuristic-based algorithms, RW algorithms remain a valuable tool in robotics for tasks requiring exploration and coverage in constrained spaces.

2..13 Chaotic Coverage Path Planning Algorithms in Robotics

Chaotic Coverage Path Planner (CPP) algorithms offer a deterministic approach to coverage path planning by leveraging chaotic systems to generate trajectories for robotic exploration and surveillance tasks. Unlike stochastic methods like random walk, chaotic CPP ensures high coverage efficiency across the entire workspace by pre-determining the robot's trajectory. One well-known chaotic system used in CPP is Arnold's dynamical system, initially introduced by Sekiguchi and Nakamura. By combining chaotic dynamic variables with the kinematic equations of the mobile robot, controllers are designed to produce chaotic motion, facilitating efficient coverage without the need for obstacle avoidance along boundaries.

In addition to Arnold's dynamical system, other chaotic systems such as the Lorenz dynamical system and Chua circuit have been employed in CPP to achieve high coverage rates. For instance, in a 3D non-linear chaotic system, the Lorenz system has been utilized to speed up workspace coverage using hyperchaotic techniques with non-linear open-loop controllers. Similarly, Chua patterns have been integrated into mobile robots to enhance coverage performance. Various chaotic attractors, including those derived from the Chua circuit and Lorenz system, have been proposed for generating coverage trajectories, contributing to improved exploration efficiency.

Moreover, discrete-time dynamical systems like the standard (Taylor-Chirikov) and logistic map have been employed in CPP to generate coverage trajectories. These maps, serving as models for 2D and 1D iterated maps, respectively, have been utilized to design random bit generators for trajectory generation. Angular transformations and inverse pheromone methods have also been explored to improve coverage uniformity and reduce memory requirements in chaotic CPP algorithms.

Compared to random walk methods, chaotic CPP offers continuous motion, enabling robots to search and locate targets more effectively with a more uniform coverage density. The continuous motion characteristic of chaotic CPP algorithms facilitates faster scanning in unknown environments, making them particularly suitable for exploration and surveillance missions. However, it's important to note that the unpredictable trajectories generated by chaotic

CPP algorithms are highly dependent on the kinematic motion of the robot and may require further study, especially concerning coverage time and trajectory predictability.

Overall, chaotic CPP algorithms represent a promising approach to coverage path planning, offering unique advantages in terms of efficiency and exploration effectiveness. However, further research is needed to fully understand the implications of their unpredictable trajectories and their performance in various robotic applications.

2..14 Spanning Tree Coverage Algorithms in coverage path planning

Spanning Tree Coverage (STC) algorithms represent a methodical approach to coverage path planning by subdividing the workspace into a sequence of disjoint cells and constructing spanning trees within these cells to facilitate optimal pathfinding. One of the key distinctions of STC algorithms is their ability to handle obstacles within the workspace, allowing robots to navigate around or through them to achieve comprehensive coverage.

Initially, the workspace is divided into cells either through cell decomposition-based methods or grid-based approaches. Subsequently, a spanning tree is constructed within the corresponding cells, with each cell being further split into sub-cells, typically matching the size of the robot. This enables the robot to systematically cover each unoccupied sub-cell by traversing the spanning tree using algorithms like depth-first search. However, challenges arise when obstacles within mega-cells occupy sub-cells, hindering complete coverage.

To address this limitation, researchers have proposed various extensions and modifications to STC algorithms. For instance, the full-STC algorithm allows robots to cover free sub-cells to maximize area coverage. Additionally, online strategies have been developed to enhance coverage efficiency in multi-robot systems, though initial robot positions and backtracking issues may pose challenges.

Efforts have also been made to optimize cell assignment and task distribution in STC algorithms. Algorithms based on auction and bidding processes have been proposed for multi-robot CPP, while pseudo-STC methods and wall following algorithms enable robots to navigate around obstacles within mega-cells. Furthermore, improvements in path planning have focused on minimizing backtracking and increasing coverage rates, particularly in scenarios where mega-cells are partially occupied by obstacles.

Recent advancements in STC algorithms have aimed to address energy efficiency concerns and fault tolerance in real-world scenarios. Hybrid CPP approaches combining frontier-based exploration with STC algorithms have been proposed to reduce energy usage, while decentralized strategies aim to distribute coverage tasks evenly among robots and ensure task completion even in the event of robot failure. However, challenges such as unbalanced workload distribution and

fault tolerance remain significant areas of research in the development of STC algorithms for coverage path planning.

Overall, STC algorithms offer a systematic and efficient approach to coverage path planning, particularly in environments with static obstacles. Continued research and innovation in this field are essential to overcome existing challenges and further improve the effectiveness and robustness of STC-based coverage path planning algorithms in real-world applications.

2..15 Dynamic Programming in Coverage Path Planning

Dynamic Programming (DP) presents a powerful methodology for optimizing complex problems by breaking them down into simpler sub-problems and then efficiently combining the solutions. In the context of Coverage Path Planning (CPP), DP is employed to tackle the challenge of finding the most efficient coverage path while considering factors such as distance, obstacles, and turns.

The key advantage of DP lies in its ability to handle overlapping sub-problems and exploit optimal substructure, making it well-suited for optimizing the sequence of coverage sub-spaces in CPP. By recursively solving these sub-problems and combining their solutions, DP algorithms can efficiently generate globally optimal coverage paths.

One application of DP in CPP involves optimizing the sequence of segments and connections to minimize the path length and the number of turns. This approach, demonstrated in various studies, enables the construction of shorter and more efficient coverage paths. Additionally, DP frameworks have been developed to address specific challenges such as coverage overlaps within a given area of interest, with strategies like the bottom-up approach to save memory space and accelerate computation.

However, scalability can be a concern when applying DP to large-scale CPP instances, as the sheer complexity of the problem can lead to suboptimal solutions. To mitigate this, researchers have explored hybrid approaches, combining DP with techniques like nearest neighbor algorithms or genetic algorithms (GA) to optimize tours more effectively. By leveraging the strengths of both DP and heuristic methods, these hybrid approaches can produce high-quality solutions for CPP with many regions.

Furthermore, advancements in DP-based CPP algorithms have focused on enhancing adaptability and energy efficiency, particularly in dynamic environments. By integrating DP into online CPP algorithms, robots can dynamically adjust their coverage paths based on real-time information, improving overall performance while conserving energy.

Despite its effectiveness, DP-based CPP algorithms may face challenges in adapting to highly dynamic environments where obstacles or terrain conditions change frequently. Nonetheless,

ongoing research aims to address these challenges and further refine DP-based approaches for CPP, ensuring their applicability and efficiency in various real-world scenarios.

2..16 Artificial Potential Field Algorithms for Obstacle Avoidance in Coverage Path Planning

The artificial potential field (APF) algorithm stands out as a widely used method for obstacle detection and navigation toward a goal position in various robotic applications, including Coverage Path Planning (CPP). This approach employs a virtual repulsive force generated around obstacles and an attractive force toward the goal, effectively guiding the robot while maintaining a safe distance from obstacles.

In CPP, the APF algorithm plays a crucial role in path planning by ensuring that the robot navigates efficiently through the environment while avoiding collisions with obstacles. Studies such as the one conducted by Sutantyo et al. have demonstrated the effectiveness of combining the APF algorithm with other techniques, such as the LF algorithm, to explore unknown environments and enhance dispersion among multiple robots.

However, one limitation of the APF algorithm is its susceptibility to local optima, where the robot may become trapped in certain regions of the environment. To address this challenge, researchers have proposed various strategies to optimize robot trajectories and escape local minima. For instance, Wei et al. introduced an inspection strategy that combines the APF algorithm with particle swarm optimization (PSO), allowing the robot to dynamically adjust its speed and position to avoid local optima.

Furthermore, advancements in APF-based CPP algorithms have focused on improving path planning efficiency and scalability. Wang et al. introduced a potential field approach based on information gain and path cost, enabling the robot to find optimized trajectories and prevent entrapment in local minima. Similarly, authors like Jiang and Deng have modified the repulsive potential function of the APF algorithm to enhance obstacle avoidance during inspection missions, particularly in narrow spaces.

Despite these advancements, challenges remain, particularly in scenarios involving multiple robots simultaneously navigating toward the same goal. Collision avoidance between robots becomes crucial in such situations to prevent interference and ensure safe and efficient navigation. Addressing this challenge requires further research into developing collision avoidance strategies that can be seamlessly integrated with the APF algorithm, enabling coordinated and collision-free movement among multiple robots in complex environments.

In summary, while the APF algorithm offers an effective approach for obstacle avoidance and navigation in CPP, ongoing research efforts aim to enhance its robustness, scalability, and adaptability to address challenges such as local optima and collision avoidance in multi-robot

scenarios. These advancements hold the potential to significantly improve the performance and applicability of APF-based CPP algorithms in real-world robotic applications.

2..17 Sampling-Based Planning Algorithms for Coverage Path Planning

Sampling-based planning algorithms have emerged as powerful tools for solving complex planning problems in a variety of robotic applications, including Coverage Path Planning (CPP). These algorithms, which utilize random sampling methods, offer heuristic and optimal solutions to navigate through challenging environments effectively.

One of the recent advancements in sampling-based planning is the integration of probability sampling techniques. Probability sampling-based planning (SBP) algorithms leverage probabilistic approaches to address planning problems more efficiently. These algorithms involve mapping the environment from configuration space using a node sampling strategy, which entails randomly generating a set of nodes in the search environment. This approach allows for a more comprehensive exploration of the environment, particularly in scenarios where sensor-based inspection, such as visual-based inspection, is essential for coverage.

Two notable sampling-based planners commonly used in CPP are the Probabilistic Roadmap (PRM) and the Rapidly Exploring Random Tree (RRT). The PRM algorithm constructs a roadmap of the environment by sampling configurations and connecting them with collision-free paths. This roadmap serves as a global map of the environment, enabling efficient path planning between start and goal configurations. On the other hand, the RRT algorithm rapidly explores the configuration space by iteratively sampling random configurations and expanding the tree toward unexplored regions. RRTs are particularly effective in dynamic environments where real-time planning is required.

The probabilistic completeness of SBP algorithms ensures that they can effectively handle complex planning scenarios and provide solutions that are both heuristic and optimal. By leveraging randomness in the sampling process, these algorithms can navigate through uncertain and cluttered environments, making them suitable for various robotic applications, including CPP.

Overall, sampling-based planning algorithms, especially those incorporating probability sampling techniques like PRM and RRT, offer versatile and efficient solutions for CPP by effectively exploring and navigating complex environments while ensuring probabilistic completeness and optimality in path planning.

2..18 Probabilistic Roadmap Planning for Coverage Path Planning

The Probabilistic Roadmap (PRM) planner is a widely used approach for path planning and query in robotics, particularly in scenarios where the configuration space is complex and obstacle-ridden. The PRM algorithm consists of two main phases: planning and query.

In the planning phase, the PRM algorithm randomly generates a specified number of nodes within the robot's configuration space. These nodes represent potential configurations or states of the robot. The algorithm then connects pairs of nodes in the configuration space using straight lines while ensuring that these lines do not intersect with any obstacles in the environment. This process results in the creation of a roadmap, which is essentially a graph representing feasible paths through the configuration space.

During the query phase, the PRM algorithm is tasked with planning a path between the initial and goal configurations provided by the user or the system. This involves utilizing the roadmap generated in the planning phase to efficiently search for a collision-free path from the initial configuration to the goal configuration. The path planning process often involves employing search algorithms like the A* algorithm to find the optimal path through the roadmap.

Researchers and practitioners have applied PRM in various robotics applications, ranging from search and rescue operations to industrial automation. For example, in a search and rescue scenario following an earthquake, PRM can help autonomous robots navigate through cluttered and hazardous environments to locate and assist survivors. In industrial settings, PRM has been used to optimize paths for robotic arms, ensuring collision-free movement and efficient task completion.

One notable application of PRM is its combination with the A* algorithm to generate collision-free and optimal paths for industrial robots. By leveraging PRM for path generation and A* for path optimization, researchers have been able to reduce cycle times and improve the efficiency of robotic operations in manufacturing environments.

Despite its effectiveness, PRM does have some limitations. One challenge is that the random placement of nodes during the planning phase may limit the coverage area of the robot, particularly near boundaries and obstacles. Additionally, when a collision occurs with an obstacle, PRM removes the corresponding nodes and edges associated with that collision, which can lead to discontinuities in the roadmap. Furthermore, PRM may incur high computational complexity and time overhead, especially when dealing with a large number of nodes in densely populated environments, despite its probabilistic completeness.

Overall, while PRM offers a powerful solution for path planning in robotics, its effectiveness depends on careful parameter tuning and consideration of its limitations in specific application contexts.

2..19 Rapidly Exploring Random Tree (RRT) Algorithm for Coverage Path Planning

The Rapidly Exploring Random Tree (RRT) algorithm is a powerful planning technique used to efficiently explore and search in high-dimensional configuration spaces, making it particularly well-suited for robotics applications where the environment is complex and dynamic. Unlike traditional methods, RRT employs an incremental approach to construct a tree structure that represents feasible paths through the configuration space.

The RRT algorithm is designed to handle kinodynamic planning, allowing robots to navigate through environments with dynamic obstacles and constraints effectively. One of its key advantages is its ability to rapidly search for feasible paths without requiring a precomputed roadmap during the learning phase, making it faster than some other planning techniques like Probabilistic Roadmap (PRM) for single-query problems.

Researchers have extensively studied and improved the RRT algorithm to enhance its performance and versatility in various robotics applications. For instance, the bidirectional RRT approach accelerates the exploration process by growing trees from both the initial and goal configurations simultaneously, enabling the algorithm to find shorter paths by connecting the trees. Although the paths generated by RRT may not always be optimal, modified variants like RRT* have been developed to provide asymptotically optimal solutions, improving path quality and efficiency.

Applications of RRT extend beyond path planning to cover diverse tasks such as coverage sampling and multi-goal planning. By combining RRT with other algorithms like Genetic Algorithms (GA), researchers have developed hybrid approaches that address complex planning problems more effectively. For instance, the multi-directional fixed nodes RRT* algorithm optimizes trajectory planning for multiple points of interest (POIs) by exploring the neighborhood and using GA to find the shortest path to visit a sequence of POIs.

Furthermore, recent advancements in RRT-based algorithms, such as the Random Kinodynamic Inspection Tree (RKIT), integrate kinodynamic planning with coverage path planning to address challenges in 3D environments. RKIT efficiently identifies feasible coverage plans by generating intermediate points and employing a steering function to navigate through obstacles while considering differential constraints.

Despite its strengths, challenges remain in applying RRT to narrow passage scenarios where robots must navigate through cluttered and confined spaces. Future research efforts could focus on adapting RRT variants to optimize area coverage in challenging environments, thereby advancing the capabilities of robotics systems in real-world applications.

2..20 Next-Best-View (NBV) Planning for Coverage Path Planning

The sampling-based view planning approach is a sophisticated solution for addressing optimization problems in robotics, particularly in tasks requiring both view planning and motion planning. This approach combines sensor-based planning with algorithms that determine the optimal viewpoints for sensing and exploration tasks. View planning plays a crucial role in various applications, such as modeling and exploration, where sensors guide the robot's vision system to achieve specific objectives, such as coverage or target observation.

One common objective in view planning is to identify the minimal set of viewpoints necessary to cover the entire target structure or region. This problem is akin to the Set Cover Problem (SCP) or the Traveling Salesman Problem (TSP), where the goal is to minimize the number of viewpoints or optimize the sequence of viewpoints to maximize coverage efficiency. Researchers have developed variant planning algorithms, including greedy strategies, optimal strategies, and decomposition planners, to solve these coverage planning problems effectively.

Many studies have focused on addressing online Coverage Path Planning (CPP) problems by utilizing approaches like the Next-Best-View (NBV) planner. The NBV planner selects suitable viewpoints based on the current robot location and sensor information, enabling efficient reconstruction of target structures or regions. Structural Inspection Planner (SIP) algorithms and probabilistic analysis techniques have been employed to optimize the tour of viewing poses, ensuring comprehensive coverage while minimizing energy consumption.

Incorporating sequential viewpoint planning is another critical aspect of view planning, involving modeling information gain in 3D environments using techniques like voxel or surface mesh representations. Learning-based NBV algorithms estimate optimal viewpoints by evaluating a set of voxels and iteratively planning the next scan. Additionally, methods like Structure from Motion (SfM) are used to reconstruct target regions and generate high-quality 3D data for accurate coverage planning.

To improve coverage completeness and efficiency, researchers have explored various exploration strategies, including multi-layer CPP techniques and combinations of local and global exploration algorithms. These approaches leverage sampling-based algorithms like Rapidly Exploring Random Trees (RRT) or RRT* to navigate through unknown environments and optimize coverage paths while avoiding obstacles.

Despite significant progress, challenges remain in achieving a balance between model quality and computation time, particularly in large-scale environments. Future research directions may involve refining sampling algorithms, enhancing geometric accuracy, and developing real-time implementations to address the complexities of robotics applications effectively.

2..21 Greedy Search Algorithms in Coverage Path Planning

The greedy search algorithm is a heuristic method commonly used in optimization problems, where decisions are made at each step based solely on the local optimal choice, without considering the global implications of those choices. One well-known example of a greedy algorithm is Dijkstra's algorithm, often used in pathfinding and navigation tasks. While the greedy algorithm is simple to implement and fast, it does not guarantee finding the globally optimal solution due to its short-term decision-making process.

Graph search algorithms, such as the A* algorithm, D* algorithm, and Theta* algorithm, are commonly employed in robotics to plan and optimize coverage paths. These algorithms utilize various strategies, including boustrophedon motion or spiral patterns, to navigate through environments efficiently. In coverage path planning, the robot may encounter obstacles or blind spots, requiring re-planning of the path to ensure complete coverage of the region of interest. The search algorithms play a crucial role in identifying the shortest path between nodes in a graph, facilitating the robot's movement from one position to another while maximizing coverage efficiency.

When the robot encounters obstacles or falls into a blind spot, the search algorithm dynamically re-plans the path to guide the robot to the next optimal position, ensuring continuous coverage of the region of interest. However, path searching in large grid maps poses computational challenges due to the high computation cost associated with exploring a vast search space.

Despite these challenges, advancements in search algorithms have significantly improved search efficiency and coverage path planning in robotics. Future research may focus on developing more efficient search algorithms capable of handling large-scale environments and reducing computation costs to further enhance coverage path planning in robotics applications.

2..22 Depth-First Search (DFS) and Breadth-First Search (BFS) in Coverage Path Planning

Depth-first search (DFS) and breadth-first search (BFS) are fundamental graph traversal algorithms used to explore nodes in a graph data structure. While both algorithms have their advantages and disadvantages, they are commonly employed in coverage path planning (CPP) to optimize path sequences.

DFS is a recursive algorithm that explores as far as possible along each branch before backtracking. It is well-suited for scenarios with finite depth spaces and can optimize path sequences by minimizing overlap and the number of turns. However, DFS may not be suitable

for infinite depth spaces, and it does not guarantee finding an optimal solution, such as the shortest coverage path.

In contrast, BFS systematically explores all neighbor nodes at the present depth before moving on to nodes at the next depth level. While BFS guarantees finding the shortest path in terms of number of steps, it can consume large memory space due to the high branching factor in the search space.

Researchers have applied both DFS and BFS techniques in various CPP scenarios to optimize coverage paths. For example, Kabir et al. utilized DFS to create cleaning trajectories, while Barrientos et al. suggested a waveform planner based on BFS to generate coverage paths with minimal turns. Wang et al. employed BFS to reduce the uncovered area in CPP, although this approach may lead to uncovered edges in certain scenarios.

Additionally, DFS and BFS have been combined with other techniques to address specific challenges in CPP. For instance, knowledge reasoning combined with BFS can help robots avoid dynamic obstacles in uncertain environments, reducing repetition rates and computation time. Miao et al. proposed a distribution technique using sub-map decomposition and BFS methods to decompose unknown maps into sub-areas and distribute robots efficiently for coverage.

Overall, while DFS and BFS algorithms can effectively optimize coverage paths in scenarios with small graphs, their applicability may vary depending on the specific characteristics of the environment and the requirements of the CPP task at hand.

2..23 Dijkstra's Algorithm in Coverage Path Planning

Dijkstra's algorithm is a versatile graph search technique used to find the shortest path from a single source node to all other nodes in a graph with non-negative edge costs. It operates by iteratively selecting the node with the lowest total cost from the source node and updating the costs of its neighboring nodes accordingly.

In the context of coverage path planning (CPP), Dijkstra's algorithm has been applied in various scenarios to optimize path sequences and minimize traversal costs. Almadhoun et al. demonstrated an efficient path coverage approach by employing Dijkstra's algorithm to explore and visit all nodes with minimum cost in an indoor environment. This application ensures thorough coverage while minimizing resource consumption.

Yehoshua et al. introduced a spiral spanning tree coverage (STC) approach for optimizing coverage paths, complemented by Dijkstra's algorithm to find the minimum weighted path.

By leveraging Dijkstra's algorithm, they achieve efficient path planning, enhancing the overall coverage percentage.

Additionally, Cheng et al. utilized Dijkstra's algorithm to calculate the shortest paths between subgraphs within stripe layers, facilitating fast path searching and reducing the total action cost to maximize area coverage. This strategy aims to minimize revisited nodes, optimizing the coverage process.

Furthermore, Rosa et al. demonstrated task planning for multi-robot systems using Dijkstra's algorithm within a honeybee (hexagonal) structure. This application showcases the algorithm's adaptability to various planning tasks within complex environments.

While Dijkstra's algorithm is effective for finding paths with minimum cost, it may not always produce optimal results in terms of travel distance. Zhang et al. addressed this limitation by considering additional factors such as turning times and angles in the cost function, enhancing the algorithm's performance.

Overall, Dijkstra's algorithm plays a crucial role in optimizing coverage paths in CPP by efficiently finding paths with minimal cost, although it may require further refinement to address specific optimization objectives such as minimizing travel distance.

2..24 A* Algorithm in Coverage Path Planning

The A* algorithm is a widely used graph search algorithm that efficiently determines the shortest path from a starting node to a goal node in a graph, taking into account both the actual cost of reaching a node from the start node and the estimated cost of reaching the goal node from the current node. This heuristic approach allows A* to prioritize nodes that are likely to lead to the optimal solution, making it particularly effective in pathfinding tasks where minimizing cost is crucial.

In the context of coverage path planning (CPP), the A* algorithm has been utilized to optimize path sequences and reduce processing time while ensuring comprehensive coverage. Viet et al. implemented CPP using the A* algorithm in conjunction with a backtracking approach to achieve optimal coverage. Despite the potential for large memory requirements to store backtracking points, this method allows for efficient path planning and coverage optimization.

Cai et al. described the application of the A* algorithm to search for the shortest path from escaping a dead node to an uncovered area, addressing challenges associated with obstacle avoidance. However, traditional A* algorithms may encounter difficulties covering cells around

obstacles, particularly when moving diagonally, and may lead to high rates of cell revisitation and overlapping without fully covering adjacent cells.

To mitigate these challenges, Le et al. proposed a modified version of the A* algorithm tailored specifically for CPP. This modified algorithm incorporates boundary waypoints and obstacle waypoints to reduce revisitation ratios and increase coverage ratios compared to traditional A* implementations. By optimizing the path planning process, this approach enhances coverage efficiency and reduces unnecessary revisits, resulting in more effective coverage paths.

Overall, the A* algorithm offers significant advantages over traditional search algorithms like DFS and BFS, particularly when the location of the target is known. Its ability to balance efficient pathfinding with comprehensive coverage makes it a valuable tool in CPP applications, especially when combined with specialized modifications to address specific challenges encountered in complex environments.

2..25 D* Algorithm in Coverage Path Planning

The D* algorithm, a variant of the optimal A* algorithm, has emerged as a powerful tool for pathfinding in dynamic environments. Unlike traditional algorithms that require complete replanning from scratch when encountering obstacles, D* is capable of dynamically replanning paths by efficiently applying cost path optimization solutions as the robot navigates through changing environments.

Researchers such as Dakulovic et al. have demonstrated the effectiveness of the D* algorithm in reducing node revisitation and minimizing overlapping paths during the path re-planning process. By computing cost values strategically, D* algorithm implementations can adapt to dynamic scenarios, ensuring efficient and obstacle-free navigation.

Incorporating advancements such as active SLAM (Simultaneous Localization and Mapping), Maurovic et al. have extended the capabilities of the D* algorithm by introducing modifications with negative edge weights. This enhancement enables D* to explore dynamic environments more effectively, facilitating robust navigation even in complex and evolving surroundings.

The D* Lite algorithm, a refined version of D*, further improves path re-planning efficiency by leveraging information from previous searches. Luo et al. utilized D* Lite as a global path planner, complementing it with techniques like ant colony optimization (ACO) to address tasks such as the Traveling Salesman Problem (TSP). This integration enhances overall navigation performance, minimizing distances traveled along planned trajectories in exploration missions.

Moreover, recent advancements like the AD* algorithm offer optimal pathfinding through online re-planning for dynamic obstacle avoidance. AD* builds upon the strengths of D*

Lite, providing even more robust navigation solutions for environments with rapidly changing conditions.

In summary, the choice between algorithms like A* and D* Lite depends on specific task requirements. While A* remains a versatile option for pathfinding, D* Lite excels in dynamic environments where efficient replanning is essential. These advancements in pathfinding algorithms have significant implications for robotics applications, enabling agile and adaptive navigation strategies in real-world scenarios.

2..26 Theta* Algorithm in Coverage Path Planning

The Theta* algorithm presents a significant advancement over discrete search methods like A* and D* by enabling pathfinding in continuous space. Unlike its predecessors, Theta* considers any angle pathfinding, allowing for more flexible and efficient navigation in grid maps. This flexibility is particularly useful in scenarios where precise path planning is essential.

Choi et al. demonstrated the practical application of Theta* in online Coverage Path Planning (CPP) for cleaning robots, leveraging boustrophedon motion to optimize local backtracking paths. By determining backtracking points based on pass knowledge, the algorithm streamlines navigation, improving coverage time in unknown environments. However, Theta* may not always yield globally optimal solutions regarding path length.

In three-dimensional spaces, the Lazy Theta* algorithm proves more adept at pathfinding on cubic grids due to the increased number of neighbors per node compared to two-dimensional grids. Faria et al. implemented frontier cell exploration with Lazy Theta* in a 3D Octomap framework, effectively navigating and avoiding obstacles. Additionally, efforts to enhance Lazy Theta* efficiency include reducing the number of generated neighbors to lower computation costs and incorporating flyby sampling techniques for smoother path generation and coverage without overlap.

Despite its advantages, Lazy Theta* does not guarantee optimal path lengths, highlighting a potential area for further improvement. Nevertheless, these advancements in continuous space pathfinding algorithms have significant implications for robotics applications, offering more robust and adaptable navigation solutions in complex and dynamic environments.

2..27 Evolutionary Algorithms

In the realm of path planning for robotics, evolutionary algorithms (EAs) and human-inspired approaches have garnered considerable attention for their ability to find optimal or near-optimal solutions to complex optimization problems. One prominent example is Genetic Algorithms (GA), a metaheuristic inspired by natural genetic evolution, which has been extensively employed in solving various path planning problems. GA operates by iteratively evolving a population

of potential solutions through mechanisms like crossover and mutation, eventually converging towards a solution that meets predefined criteria.

While GA offers the advantage of global search capability, it often suffers from poor stability and high computation time, particularly in scenarios with large search space complexity. To address these limitations, researchers have proposed enhancements such as multi-objective GA and hybrid approaches combining GA with other techniques like Dynamic Programming (DP) or simulated annealing. These adaptations aim to improve convergence speed and solution quality while mitigating the computational burden.

Another noteworthy EA is Differential Evolution (DE), which offers advantages such as quick convergence and robustness. DE operates by iteratively generating trial vectors through mutation, recombination, and selection processes, making it particularly suitable for optimization problems with complex search spaces. Researchers have explored various modifications to DE, such as combining it with roulette and multi-neighborhood operations, to enhance its performance in path planning tasks.

In addition to EAs, swarm intelligence algorithms have gained prominence for their ability to emulate collective behavior observed in natural systems. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Bee Colony Optimization (BCO) are notable examples of swarm intelligence algorithms applied to path planning. These algorithms leverage the collective intelligence of swarm agents to efficiently explore and optimize paths in complex environments. However, they may face challenges such as local optima trapping and slow convergence rates, prompting researchers to propose enhancements like distributed algorithms and improved pheromone updating rules.

On the other hand, human-inspired algorithms, such as neural networks and reinforcement learning (RL), draw inspiration from the workings of the human brain to optimize decision-making processes in path planning. Neural networks, including feedforward and convolutional architectures, have been utilized to learn complex mappings between sensory inputs and actions, enabling robots to navigate and plan paths in dynamic environments. Reinforcement learning, a subset of machine learning, allows agents to learn optimal behaviors through trial-and-error interactions with the environment. RL algorithms like Q-learning and Deep Q-Networks (DQN) have shown promise in optimizing path planning tasks, albeit with challenges related to convergence speed and scalability.

Despite their potential, evolutionary and human-inspired approaches may not be suitable for scenarios requiring real-time path planning or where computational efficiency is paramount. These methods typically involve iterative optimization processes that may incur significant computation time, making them less practical for time-sensitive applications. Thus, while these

approaches offer valuable insights into path planning optimization, their adoption may depend on the specific requirements and constraints of the robotics task at hand.

Genetic Evolution:

Genetic Algorithms (GAs) have been extensively explored in the field of path planning for robotics, drawing inspiration from natural genetic evolution. By mimicking biological processes such as crossover and mutation, GAs iteratively evolve a population of potential solutions towards optimal or near-optimal paths. Researchers have applied GAs to various path planning problems, including the Travelling Salesman Problem (TSP) and coverage path planning (CPP). However, despite their global search capability, GAs often suffer from poor stability and high computational complexity, particularly in scenarios with large search spaces. While efforts have been made to enhance GA performance through multi-objective optimization and hybrid approaches, these methods still require significant computational resources and may not be suitable for real-time path planning applications.

Swarm Intelligence:

Swarm intelligence algorithms, such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Bee Colony Optimization (BCO), have gained popularity for their ability to emulate collective behaviors observed in natural systems. These algorithms leverage the collective intelligence of swarm agents to efficiently explore and optimize paths in complex environments. However, they may face challenges such as local optima trapping and slow convergence rates, limiting their applicability in time-sensitive path planning tasks. While enhancements like distributed algorithms and improved pheromone updating rules have been proposed to address these challenges, swarm intelligence algorithms still require significant computational resources and may not meet the real-time constraints of robotics applications.

Ecology:

Ecological algorithms, such as Invasive Weed Optimization (IWO), offer an alternative approach to path planning inspired by the distribution and colonization behaviors of natural systems. These algorithms aim to efficiently explore and optimize paths by mimicking ecological processes. While IWO has shown promise in optimizing path planning tasks, it may still suffer from high computational complexity and limited scalability, particularly in large-scale environments. Furthermore, the application of ecological algorithms in robotics may require extensive parameter tuning and optimization, making them less suitable for real-time path planning applications.

Human-Inspired Approaches:

Human-inspired approaches, including neural networks and reinforcement learning (RL), leverage insights from human cognition to optimize decision-making processes in path planning. Neural networks, such as feedforward and convolutional architectures, have been applied to learn complex mappings between sensory inputs and actions, enabling robots to navigate and plan paths in dynamic environments. Similarly, RL algorithms like Q-learning and Deep Q-

Networks (DQN) allow agents to learn optimal behaviors through trial-and-error interactions with the environment. However, these approaches may suffer from slow convergence speeds and scalability issues, particularly in large-scale and dynamic environments. Additionally, the training and optimization of neural networks and RL algorithms often require extensive computational resources and may not meet the real-time constraints of robotics applications.

Reasons for Not Using:

While evolutionary, swarm intelligence, ecological, and human-inspired approaches offer valuable insights into path planning optimization, they may not be suitable for real-time applications requiring computational efficiency and responsiveness. These methods typically involve iterative optimization processes that may incur significant computation time, making them less practical for time-sensitive robotics tasks. Additionally, the complexity and scalability of these algorithms may pose challenges in real-world deployment, particularly in dynamic and unpredictable environments. Thus, while these approaches contribute to the body of knowledge in path planning research, their adoption in practical robotics applications may depend on specific task requirements and constraints.

2..28 Combinatorial Planning Techniques

Combinatorial planning techniques emerged as a response to the need for efficient and systematic methods to navigate robots through complex environments cluttered with obstacles. As robotics applications expanded into domains such as manufacturing, logistics, and exploration, the demand for reliable path planning algorithms became increasingly pronounced. Traditional approaches often struggled to cope with the intricacies of real-world environments, leading to the development of combinatorial planning techniques.

These techniques were invented to provide a structured framework for path planning by discretizing the continuous configuration space into a graph-based representation. By breaking down the problem into manageable components, combinatorial planning methods aimed to overcome the challenges posed by obstacles and non-trivial workspace geometries. They offered a systematic way to explore the configuration space, enabling robots to navigate from an initial pose to a desired goal while avoiding collisions.

Several successful algorithms have emerged within the realm of combinatorial planning, each tailored to address specific challenges and requirements. Visibility Graphs, for instance, construct a graph connecting the initial and goal configurations through vertices of obstacles, allowing for the derivation of optimal paths. Voronoi diagrams leverage geometric principles to define regions of influence around obstacles, facilitating path planning by identifying areas of potential clearance. Exact and approximate cell decomposition methods partition the workspace

into cells, simplifying the path planning process by focusing on individual regions rather than the entire space.

These algorithms have found applications in various domains, including robotics, computer-aided design, and video game development. Their success lies in their ability to provide deterministic and robust solutions to path planning problems, even in highly complex and dynamic environments. By offering a systematic approach to configuration space exploration, combinatorial planning techniques have become indispensable tools in the arsenal of roboticists and engineers striving to enable autonomous navigation in challenging environments.

2..29 Visibility Graphs

Visibility Graphs represent a fundamental combinatorial planning technique used to navigate robots through environments cluttered with obstacles. This method constructs a graph connecting the initial and goal configurations through vertices representing the obstacles. By leveraging the concept of visibility between vertices, Visibility Graphs provide a systematic way to derive collision-free paths while optimizing for efficiency and optimality.

The concept behind Visibility Graphs is intuitive yet powerful: if two configurations (vertices) are mutually visible, meaning there is a straight-line path between them that does not intersect any obstacles, they are connected in the graph. This process effectively abstracts the environment into a graph-based representation, enabling the derivation of collision-free paths using graph search algorithms.

One of the key advantages of Visibility Graphs is their ability to produce optimal paths. By connecting configurations through direct lines of sight, these paths inherently minimize distance and traversal time, leading to efficient navigation. Additionally, Visibility Graphs exhibit deterministic behavior, ensuring consistent results across different environments and scenarios.

Another strength of Visibility Graphs lies in their robustness to complex environments. Whether the workspace contains obstacles of irregular shapes or varying densities, this method can adapt by constructing a graph that captures the visibility relationships between configurations. This robustness extends to the avoidance of local minima, as Visibility Graphs provide a global view of the configuration space, enabling the identification of paths that circumvent potential deadlocks.

However, like any planning technique, Visibility Graphs have limitations. One drawback is the computational overhead associated with constructing the graph, especially in environments with a large number of obstacles or intricate geometries. Additionally, Visibility Graphs are

limited to polygonal obstacles, restricting their applicability in environments with non-polygonal or complex-shaped obstacles.

Despite these limitations, Visibility Graphs remain a powerful and widely used tool in combinatorial planning. Their ability to generate optimal, deterministic, and robust paths makes them invaluable for applications ranging from robotic navigation in warehouses to autonomous vehicle route planning in urban environments. As technology advances and computational resources become more accessible, Visibility Graphs continue to play a vital role in enabling efficient and reliable path planning in complex environments.

2..30 Voronoi Diagrams

Voronoi diagrams represent another powerful approach in combinatorial planning, offering a distinct method for generating collision-free paths in complex environments. These diagrams are derived from a set of points scattered across the free space, where each point defines a region encompassing all locations that are closer to it than to any other point in the set. This partitioning of space results in a tessellation of regions, known as Voronoi cells or polygons, as illustrated in Fig-03a.

The process of constructing Voronoi diagrams begins by identifying the midpoints between every pair of points in the set. These midpoints serve as the boundaries between adjacent Voronoi cells, delineating the regions of influence for each point. By extending these boundaries to the obstacles in the environment, Voronoi diagrams effectively define regions with varying degrees of clearance, providing valuable insight into potential paths for navigation.

One of the key advantages of Voronoi diagrams lies in their robustness to complex environments. By partitioning space based on proximity to points, these diagrams can adapt to irregular obstacle geometries and varying obstacle densities, ensuring comprehensive coverage of the configuration space. Additionally, Voronoi diagrams offer scalability, as they can accommodate larger environments without significantly increasing computational complexity.

Another benefit of Voronoi diagrams is their ability to provide increased clearance from obstacles compared to other planning techniques. By defining regions based on distance to points, these diagrams inherently prioritize paths that maintain a safe distance from obstacles, reducing the risk of collisions and enhancing overall safety.

However, Voronoi diagrams also have limitations that must be considered. One drawback is their tendency to produce sub-optimal paths, especially in environments with complex obstacle configurations. Additionally, the computational cost of constructing Voronoi diagrams can be significant, particularly in environments with a large number of points or obstacles. Moreover,

integrating Voronoi-based planning into existing approaches may require additional heuristics to address specific constraints or objectives, adding complexity to the planning process.

Furthermore, like Visibility Graphs, Voronoi diagrams may not adhere to non-holonomic constraints when generating paths, potentially limiting their suitability for certain types of robotic platforms or vehicles.

Despite these limitations, Voronoi diagrams remain a valuable tool in combinatorial planning, offering a scalable and robust approach to path generation in complex environments. By leveraging the principles of proximity-based partitioning, Voronoi diagrams provide a versatile framework for navigation that can be tailored to various applications and scenarios, contributing to the advancement of autonomous systems and robotics.

2..31 Exact Cell Decomposition

Exact cell decomposition offers another approach to path planning by decomposing the free configuration space into trapezoids with vertical side segments, as depicted in Fig-04. This method involves shooting rays upward and downward from each polygon vertex to delineate the boundaries of the trapezoids. Subsequently, a vertex is placed in the interior of every trapezoid, typically chosen as the centroid, along with additional vertices in each vertical segment. Finally, these vertices are connected to form a graph, facilitating pathfinding through graph search algorithms.

One of the primary advantages of exact cell decomposition is its effectiveness in area coverage tasks. By decomposing the configuration space into trapezoidal cells, this approach ensures comprehensive exploration of the environment, making it well-suited for applications such as surveillance, mapping, and search-and-rescue missions.

Additionally, exact cell decomposition offers increased clearance from obstacles, similar to Voronoi diagrams. By placing vertices within the interior of trapezoids, this method inherently prioritizes paths that maintain a safe distance from obstacles, enhancing overall safety and robustness.

Another notable advantage of exact cell decomposition is its scalability. This approach can accommodate environments of varying sizes and complexities, making it suitable for a wide range of robotic applications, from small-scale indoor navigation to large outdoor environments.

Furthermore, exact cell decomposition excels in handling complex environments characterized by irregular obstacle geometries and varying obstacle densities. By partitioning the configuration space into geometrically simple trapezoidal cells, this method can effectively

navigate through cluttered and intricate environments, providing reliable path planning capabilities.

However, similar to other combinatorial planning techniques, exact cell decomposition has certain drawbacks that must be considered. One such limitation is its tendency to produce sub-optimal paths, particularly in environments with complex obstacle configurations. Additionally, exact cell decomposition is only applicable for polygonal obstacles, restricting its utility in environments with non-polygonal or irregularly shaped obstacles.

Moreover, like Voronoi diagrams, exact cell decomposition does not explicitly consider non-holonomic constraints when generating paths, which may limit its suitability for certain robotic platforms or vehicles with kinematic constraints.

To effectively integrate exact cell decomposition into practical robotic systems, additional heuristics and algorithms may be required to address specific constraints or objectives, adding complexity to the planning process and potentially increasing computational overhead.

Despite these limitations, exact cell decomposition remains a valuable technique in combinatorial planning, offering a scalable, robust, and versatile approach to path planning in diverse environments. By leveraging the principles of geometric decomposition, this method contributes to the advancement of autonomous systems and robotics, enabling efficient and reliable navigation in real-world scenarios.

2..32 Approximate Cell Decomposition

Approximate cell decomposition offers a grid-based approach to path planning, dividing the configuration space into either fixed or variable-sized grids, as illustrated in Fig-05. Each grid cell is labeled as either free or occupied to represent the corresponding regions of free space and obstacles.

One of the primary advantages of approximate cell decomposition is its simplicity of implementation. By representing the environment as a grid of cells, this approach provides a straightforward framework for path planning that is easy to understand and implement, making it accessible to developers and researchers alike.

Another advantage of approximate cell decomposition is its flexibility in handling variable obstacle shapes. Unlike methods that rely on geometric decomposition or graph-based representations, approximate cell decomposition can accommodate obstacles of varying shapes and sizes, allowing for greater versatility in navigating complex environments.

Additionally, approximate cell decomposition offers ease of integration into existing code bases. By leveraging grid-based representations, this approach can be seamlessly incorporated

into robotic systems and simulation environments, facilitating rapid prototyping and development of path planning algorithms.

Furthermore, approximate cell decomposition benefits from pre-generated grids, which help reduce computational time during path planning. By generating the grid beforehand, computational overhead is minimized, resulting in faster planning times compared to approaches that require dynamic graph generation and search.

Moreover, in many cases, grid-based approaches such as approximate cell decomposition are faster than other techniques, particularly those that involve graph generation and search. The inherent simplicity and efficiency of grid-based representations make them well-suited for real-time applications and resource-constrained robotic systems.

However, like other path planning methods, approximate cell decomposition has certain limitations that must be considered. One such limitation is the need for heuristics to guide the path planning process. While grid-based approaches provide a systematic framework for navigation, heuristic strategies are often required to optimize path selection and avoid obstacles effectively.

Additionally, approximate cell decomposition may encounter challenges when dealing with dynamic obstacles, as frequent updates to grid cells are necessary to accommodate changes in the environment. This computational overhead can lead to increased complexity and decreased efficiency, particularly in scenarios with rapidly changing obstacles or environments.

Furthermore, approximate cell decomposition is constrained by fixed resolution, which may limit its ability to represent fine-grained details in the environment. This limitation can impact the accuracy and precision of path planning, particularly in environments with intricate geometries or narrow passages.

In summary, approximate cell decomposition offers a simple, flexible, and efficient approach to path planning, particularly suitable for applications where real-time performance and ease of implementation are paramount. Despite its limitations, this method remains a valuable tool in the toolkit of roboticists and researchers, contributing to the advancement of autonomous navigation and robotic systems in diverse environments.

2..33 State Lattice Planning

State lattice planning involves a systematic five-step process to generate feasible paths for a robot navigating in a continuous environment with non-holonomic constraints. The process begins with discretization, where the continuous space is divided into a finite number of cells or

nodes to create a discrete representation of the environment. Each node represents a possible robot pose, accounting for the robot's position and orientation.

Following discretization, the next step is node expansion, where each node is expanded to include states that adhere to the robot's non-holonomic constraints. This expansion process ensures that the generated states are physically feasible for the robot to traverse.

Connectivity comes next, where all the expanded states are connected to form a graph representing the possible paths through the environment. This graph serves as the basis for path generation, where a graph search algorithm is employed to find the optimal path from the start to the goal state.

Finally, trajectory planning involves generating a smooth trajectory along the selected path, often accomplished through techniques such as path smoothing or trajectory optimization.

State lattice planning offers several advantages for path planning in robotic navigation. Firstly, it ensures completeness, meaning that if a feasible path exists, the algorithm will eventually find it. Moreover, it provides an optimal solution by identifying the shortest path from the start to the goal state, minimizing travel distance and time.

Another advantage is the ability to dynamically replan paths in real-time, depending on changes in the environment or robot constraints. This flexibility allows for adaptive navigation in dynamic scenarios, enhancing the robot's performance and safety.

Furthermore, state lattice planning can incorporate kinematic constraints, such as maximum velocity or acceleration limits, ensuring that the generated paths are physically feasible for the robot to execute.

However, there are certain drawbacks associated with state lattice planning that may limit its suitability for certain applications. For example, the algorithm may not cover intermediate points or obstacles that lie between discrete nodes, potentially leading to suboptimal paths or collisions.

Additionally, generating an approximate straight path may require a dense tree of nodes, leading to increased computational time, especially when the path encounters frequent obstacles.

Moreover, while the algorithm aims to find the optimal path, there is no guarantee of optimality, particularly in complex environments with dynamic obstacles or changing constraints.

In our approach, which relies on trajectory generation using Dubin's paths, there are specific considerations to address. For instance, the algorithm only requires points as input, generating

trajectories autonomously. Therefore, intermediate points must be determined post-trajectory generation to ensure smooth navigation.

Observations regarding the state lattice planning process include the generation of a dense tree to connect all grid points, the selection of optimal orientations for endpoints, and the importance of balancing distance and time constraints in path planning.

Overall, state lattice planning offers a systematic approach to path planning, providing completeness, optimality, and dynamic replanning capabilities. However, its effectiveness may vary depending on the specific requirements and constraints of the robotic application.

2..34 OTHER CLASSICAL AND HEURISTIC ALGORITHMS

The exploration and coverage problem (CPP) is fundamental in robotics, especially in scenarios where robots need to navigate and cover large or complex areas efficiently. Combinatorial planning techniques offer a diverse set of algorithms designed to address this challenge by planning paths while avoiding obstacles. These techniques have been developed to handle various types of environments, obstacles, and constraints, making them invaluable in the field of robotics.

Combinatorial planning encompasses several major techniques, each with its own approach to planning paths and avoiding obstacles. Visibility graphs, Voronoi diagrams, exact cell decomposition, and approximate cell decomposition are among the prominent methods employed in combinatorial planning.

Visibility graphs construct a path by connecting the start and goal configurations through vertices of obstacles, resulting in a graph representing feasible paths. This method ensures optimality, deterministic behavior, robustness in complex environments, and avoidance of local minima. However, generating the graph and performing graph search to find optimal paths can be computationally expensive, especially as the complexity increases with the number of edges. Additionally, visibility graphs are limited to working with polygonal obstacles and do not account for non-holonomic constraints during path generation.

Voronoi diagrams scatter points in free space and define boundaries between points, creating regions around each point. Paths are planned through these points, offering robustness in complex environments and increased clearance from obstacles. However, Voronoi diagrams may result in sub-optimal paths and require additional heuristics to integrate into existing approaches. Additionally, they do not adhere to non-holonomic constraints during path generation.

Exact cell decomposition divides the free space into trapezoids with vertical side segments, connecting vertices to generate a graph for path planning. This method is suitable for area coverage, provides increased clearance from obstacles, and is scalable for complex environments.

However, it may produce sub-optimal paths and is applicable only to polygons, neglecting non-holonomic constraints.

Approximate cell decomposition adopts a grid-based approach, dividing space into fixed or variable-sized grids labeled as free or occupied. This method is simple to implement, allows integration of variable obstacle shapes, and is suitable for integrating into existing code bases. However, it requires the use of heuristics, is computationally expensive for dynamic obstacles, and operates at fixed resolutions.

State lattice algorithms discretize continuous space into nodes representing robot poses with non-holonomic constraints, forming a graph for path planning. These algorithms offer completeness, optimal solutions, and dynamic replanning capabilities but may overlook intermediate points and encounter computational challenges in dynamic environments.

In addition to these classical techniques, several heuristic algorithms address exploration and coverage problems. These include boustrophedon motion, internal spiral algorithms, Voronoi partition approaches, and Brick and Mortar algorithms. Each algorithm offers unique strategies and approaches to address specific challenges in exploration and coverage, contributing to the diverse landscape of combinatorial planning in robotics.

To summarize the evolution of the algorthims:

At the foundational level, algorithms such as the Dubins path and the Reeds-Shepp path have laid the groundwork for understanding optimal path planning for vehicles with specific motion constraints. The Dubins path, introduced by L.E. Dubins, focuses on finding the shortest path between two points for vehicles constrained by minimum turning radius. This algorithm, which considers only forward movement, provides a simple yet effective solution for navigating environments with curvature constraints. On the other hand, the Reeds-Shepp path, pioneered by J.A. Reeds and L.A. Shepp, extends the capabilities of path planning by incorporating both forward and backward movements. This increased flexibility allows vehicles to navigate more efficiently in tight or obstacle-dense environments, making it particularly useful for applications such as agricultural robotics.

Building upon these foundational algorithms, researchers have developed more advanced techniques to address specific challenges in path planning. One such challenge is the Traveling Salesman Problem (TSP), where a vehicle or agent must visit a set of locations while minimizing the total distance traveled. The classical TSP has been extensively studied, and numerous approximation algorithms have been proposed to find near-optimal solutions efficiently. However, when the TSP involves neighborhoods or regions instead of precise points, as in the TSP with Neighborhoods (TSPN), the problem becomes significantly more complex. Algorithms like the Intersecting Regions Algorithm (IRA) have been introduced to tackle this challenge by optimizing paths through overlapping regions efficiently.

Moreover, variants of the TSP tailored for specific vehicles or constraints have garnered attention in recent years. For instance, the Dubins Traveling Salesman Problem (DTSP) addresses path planning for vehicles constrained by minimum turning radius, such as UAVs. Algorithms like the Nearest Neighbor Heuristic and the heading discretization approach have been proposed to find approximate solutions to the DTSP efficiently. Additionally, the Generalized Dubins Interval Problem (GDIP) extends the DTSP by considering orientation intervals, introducing new challenges in path planning. Advanced optimization techniques, such as local iterative optimization algorithms, have been developed to solve the GDIP effectively and efficiently.

Furthermore, path planning algorithms for autonomous vehicles operating in narrow or constrained environments have received significant attention. Techniques like the RTR+TTS planning algorithm combine global and local planning strategies to navigate through tight spaces efficiently while maintaining continuous curvature paths. These algorithms leverage the vehicle's dynamics and environmental constraints to generate feasible and comfortable trajectories, enhancing the practicality of autonomous navigation systems.

In conclusion, path planning algorithms continue to evolve, addressing a wide range of challenges in robotics and autonomous navigation. From foundational algorithms like the Dubins and Reeds-Shepp paths to advanced optimization techniques for complex environments,

researchers are continuously pushing the boundaries of path planning to enable safe and efficient navigation in real-world scenarios.

3. Problem Statement

We consider a robotic scenario wherein a four-wheel robot equipped with a mechanical CNC-inspired system for extraction operates within a defined area. The robot, characterized as non-holonomic, integrates the mechanical system beneath it, enabling movement along the x and y directions by approximately 60 cm, and vertically until ground contact. Due to its non-holonomic nature, the robot imposes kinematic constraints on turning, enforcing a minimum turning radius of 2 meters.

The operational environment comprises grass fields assumed to be uniform without any slopes (z=0), covering a real area of (120, 90) square meters. Weed distribution within this area is heterogeneous, with approximately 60 percent of points clustered following a Gaussian distribution with varying variances. Weed positions are obtained via drone-based data collection, utilizing a deep learning model to identify weed locations. This dataset serves as the basis for complete coverage path planning.

Given the robot's mechanical implementation width of 60 cm, the operational region is discretized into points with each point representing an area of 30 cm, facilitating path planning optimization. The robot's velocity is constrained to 0.8 m/s on straight paths and 0.4 m/s on curved paths.

The primary objective of this research is to develop a path planning algorithm capable of covering all weed points within the designated area while adhering to the robot's non-holonomic constraints. Additionally, the algorithm aims to generate paths that approximate straight lines where feasible, ensuring comprehensive coverage of all points.

The objectives of the proposed algorithm include:

- **Realistic Path Generation:** Develop paths that mimic natural movement patterns, enhancing operational realism.
- **Computational Efficiency:** Minimize processing time and resources required for path planning.
- **Energy Conservation:** Optimize energy consumption during path execution, enhancing overall operational efficiency.
- **Field Operation Efficiency:** Facilitate efficient field operations, reducing time spent on weed detection and removal processes.

The algorithm should prioritize finding the shortest path distance to cover all weed points effectively while meeting the aforementioned objectives. The algorithm's performance will be evaluated based on the time taken to generate paths, the distance covered, and the energy consumed during path execution.

4. Experimental Setup

4..1 Environment

The environment for our coverage motion planning algorithm is situated in a grass field characterized by small grass uniformly distributed across the area. The grass field is representative of typical agricultural settings, where weed management is essential for maintaining crop quality and productivity. As the grass matures, it is common for various unwanted plants, particularly weeds, to grow sporadically throughout the field. Our primary focus in this study is on the removal of Rumex (commonly known as dock weeds), which pose a significant challenge to farmers. However, the algorithm can be adapted to target other types of weeds based on specific requirements, the only that change would be the detection system to identify the target weed.

4..2 Importance of Weed Removal

Importance of Weed Removal We are concentrating on Rumex plants due to their detrimental impact on agricultural productivity and livestock health. Although Rumex plants are not inherently toxic, their presence in cattle feed can adversely affect the quality of milk production. Ensuring the purity of grass fed to cattle is crucial for producing high-quality, bio milk, which is not only more beneficial for human consumption but also promotes better health and well-being of the cattle. Pure grass feed leads to higher nutritional value in milk, contributing to improved dairy products. Therefore, effective weed management, particularly the removal of Rumex plants, is essential for maintaining the quality and productivity of grass fields.

Traditionally, farmers have resorted to manually removing these weeds, a labor-intensive and time-consuming process. Manual removal becomes particularly arduous in large fields, imposing significant physical strain on farmers and limiting the efficiency of weed management. Automating this process with robotic systems offers a promising solution to enhance agricultural practices and improve farmers' quality of life.

4..3 The Robot

4..3.1 Overview

To efficiently remove weeds from the grass field, we employ a robust, four-wheeled robot specifically designed for this task. This section provides an in-depth look at the robot's external and internal features, highlighting the components that enable it to navigate the field and execute precise weed extraction.

4..3.2 External Features

The robot features a four-wheeled design, ensuring stability and effective maneuverability across uneven terrain. The choice of a four-wheeled configuration is crucial as it supports the internal

weed extraction system, which occupies most of the internal space. The wheels are equipped with treads suitable for grass fields, providing the necessary traction and mobility.

Localization and Orientation: For accurate positioning and orientation in the open field, the robot is equipped with two Real-Time Kinematic (RTK) GPS systems. RTK GPS technology is ideal for this environment, offering centimeter-level accuracy in localization, which is essential for precise navigation and weed targeting.

Vision System: The robot incorporates two strategically placed cameras. The front-facing camera is oriented towards the ground, scanning for weeds ahead of the robot. This early detection allows the local planner to adjust the robot's path accordingly, ensuring efficient navigation and weed targeting. The second camera is mounted at the bottom center of the robot, directly above the extraction mechanism. This camera provides an accurate view of the weed's position beneath the robot, facilitating precise extraction.

4..3.3 Internal Features

The internal mechanism of the robot is crucial for the precise and efficient removal of weeds. It comprises several key components: a processing unit, a battery system, and the main weed extraction system.

Processing Unit: The processing unit serves as the brain of the robot, orchestrating the various functions and ensuring smooth operation. It processes data from the cameras and GPS systems, making real-time decisions to control the navigation and extraction processes. The unit is equipped with advanced algorithms for path planning, weed detection, and tool control, enabling the robot to perform its tasks autonomously and efficiently.

Battery System: The robot is powered by a robust battery system designed to provide sufficient energy for extended field operations. The battery pack is engineered for easy replacement, ensuring minimal downtime and continuous operation. This power system supports all onboard electronics, including the processing unit, cameras, GPS, and the weed extraction mechanism.

Weed Extraction System: The heart of the weed removal process is the weed extraction system, inspired by CNC machine technology. This system features two moving rails that allow the internal extraction mechanism to move in both the x and y directions. Controlled by the processing unit, these rails provide precise positioning capabilities. The extraction mechanism can move approximately 60 cm in both the x and y directions, covering a significant area beneath

the robot. Once a weed is detected, the system moves the tool directly above the weed. The mechanical tool is then lowered in the negative z direction to engage the weed.

The extraction tool is designed with sharp implements and rotates at high speed to destroy the weed effectively. This rotation ensures that the weed is thoroughly eradicated and cannot regrow. After destroying the weed, the remnants are left on the field, eliminating the need to carry them, which would otherwise burden the robot. The robot's internal system, with its 60 cm movement range in both directions, allows for efficient navigation. Each detected weed point can be associated with a circular region within which the robot can maneuver to remove the weed. This approach not only aids in accurate weed extraction but also facilitates efficient path planning. By considering these regions, the planning algorithm can optimize the robot's path to cover all weed points effectively.

Adaptive Regions and Uncertainty Management: The regions around each weed point also serve to manage uncertainty in weed detection. If the position of a weed is detected with less accuracy, the associated region can be adjusted accordingly. A larger uncertainty reduces the size of the region to ensure the robot remains closer to the center, enhancing the likelihood of successful extraction. This adaptive approach allows the robot to handle variations in detection accuracy, maintaining high efficiency and precision in weed removal. The robot's internal features are meticulously designed to support the complex task of weed removal. The combination of precise movement capabilities, robust processing power, and adaptive region management ensures that the robot can effectively and efficiently eradicate weeds from the grass field.

4..4 Constraints

With a comprehensive understanding of the robot's capabilities and features, it is essential to delve into the constraints that arise due to its mechanical system and the nature of the field in which it operates. These constraints significantly influence the development of an effective motion planning algorithm.

Non-Holonomic Nature: The robot is equipped with four regular wheels, limiting its movement capabilities. Unlike holonomic robots, which can move in any direction, our robot cannot move sideways. It is constrained to forward and backward movements and must make turns to change its direction. This non-holonomic nature adds a layer of complexity to the motion planning algorithm, as it must account for the robot's inability to change its heading instantaneously.

Turning Radius: The robot has a minimum turning radius of 2 meters, meaning it cannot make sharp turns. This constraint necessitates careful planning to ensure the robot can navigate

around obstacles and reach all designated weed points without making turns that exceed its turning capabilities.

Speed Variations: The robot's speed must be carefully regulated to prevent damage to the grass and ensure precise weed removal. When moving in a straight path, the robot operates at a velocity of 0.8 m/s. However, when making a turn, the velocity is reduced to 0.4 m/s to maintain the 2-meter turning radius. This adjustment helps in maintaining stability and accuracy during turns, which is critical for avoiding collateral damage to the grass and ensuring efficient weed extraction.

Kinematic Constraints: Given its non-holonomic nature and the minimum turning radius, the robot cannot change its heading angle at a specific position. Instead, it requires a certain amount of space to turn and align itself in the desired direction. These kinematic constraints must be factored into the motion planning algorithm to ensure smooth and feasible navigation across the field.

Field Constraints: The field is covered with grass and scattered weeds, particularly the rumex plants. The random distribution of weeds means the robot must navigate the entire field efficiently to locate and remove all weeds. The algorithm must account for this scattered distribution and plan paths that ensure comprehensive coverage without unnecessary overlaps.

With a clear understanding of the robot's constraints, we can now proceed to develop a motion planning algorithm that leverages these constraints to ensure comprehensive and efficient weed removal. By integrating considerations for non-holonomic movement, adaptive speed control, optimized coverage, obstacle avoidance, and environmental adaptability, the algorithm will facilitate the robot's autonomous operation, making the weed removal process more efficient and reducing the burden on farmers. This strategic approach not only enhances the robot's performance but also contributes to maintaining the health of the grass and improving the overall quality of the field.

4..5 Remnants

Benefits of Robotic Weed Removal Deploying robots for weed removal in grass fields presents several advantages:

Increased Efficiency: Robots can operate continuously and systematically cover large areas, ensuring thorough removal of unwanted plants without fatigue. Labor Savings: Automating weed removal reduces the need for manual labor, freeing farmers to focus on other essential tasks and reducing physical strain. Precision: Advanced sensors and algorithms enable robots to identify and target specific weeds, minimizing damage to the surrounding grass and ensuring effective weed management. Health Benefits: By eliminating the need for manual weed removal, farmers can avoid the physical exertion and potential health risks associated with prolonged exposure to outdoor conditions and repetitive movements. Enhanced Livestock Health: Ensuring

the purity of grass feed leads to healthier cattle, which in turn produce higher-quality milk, enhancing overall dairy production.

Experimental Objective The primary objective of our experimental setup is to develop and validate a coverage motion planning algorithm that enables robots to efficiently navigate and manage weed removal in grass fields. By leveraging advanced robotic technology, we aim to automate the process of identifying and removing Rumex plants, ultimately improving agricultural productivity and promoting sustainable farming practices.

Methodology The experimental setup involves the following key steps:

Field Analysis: Detailed mapping of the grass field to identify the distribution of grass and Rumex plants. Algorithm Development: Creating a coverage motion planning algorithm tailored to navigate the field and target Rumex plants for removal. Robotic Implementation: Equipping robots with necessary sensors and tools to detect and remove weeds, followed by field testing to ensure operational efficiency. Performance Evaluation: Assessing the effectiveness of the robotic system in terms of coverage, weed removal accuracy, and operational efficiency. By addressing the challenges of weed management through robotic automation, we aim to revolutionize agricultural practices, making farming more sustainable, efficient, and farmer-friendly. This innovative approach not only enhances the productivity of grass fields but also contributes to the overall well-being of livestock and farmers alike.

Integrating Constraints into Motion Planning Understanding the mechanical and field constraints is crucial for developing an effective coverage motion planning algorithm. These constraints guide the design of the algorithm to ensure it is practical and feasible under real-world conditions.

Path Planning with Non-Holonomic Constraints: The algorithm must plan paths that respect the robot's non-holonomic nature. This involves generating smooth, continuous paths that the robot can follow without needing to make sharp turns. Techniques such as Dubins curves, which are designed for non-holonomic vehicles, can be employed to create feasible paths that the robot can navigate.

Adaptive Speed Control: Incorporating variable speed control into the algorithm allows for safe and efficient navigation. The robot should move faster in straight paths to cover more ground quickly and slow down during turns to maintain stability and precision. This adaptive speed control also helps in preventing damage to the grass, ensuring that the robot's operations are minimally invasive.

Coverage Optimization: The algorithm should optimize the coverage pattern to ensure that all weeds are detected and removed efficiently. By considering the robot's 60 cm movement range in both directions, the algorithm can plan paths that cover larger areas without unnecessary backtracking. This optimization reduces the total time and energy required for weed removal.

Obstacle Avoidance and Navigation: The algorithm must include robust obstacle avoidance capabilities to navigate around natural terrain features and any unforeseen obstacles. This ensures the robot can continue its operation smoothly without interruptions.

Handling Environmental Variability: To ensure reliable performance under varying environmental conditions, the algorithm can incorporate sensor feedback to adjust the robot's path and speed in real-time. This adaptability enhances the robot's resilience and operational efficiency.

5. Methodology

5..1 Brief Description

Start with the algorithm, then you can modify it later.

5..2 Data Preprocessing

Data preprocessing is a pivotal step in the research methodology, especially in the context of coverage path planning for weed removal in agricultural fields. This section delineates the comprehensive preprocessing procedures employed to ensure the precision and efficacy of the data used in the subsequent analysis.

Data Acquisition: The initial phase of preprocessing involves the acquisition of data pertaining to the weed positions within the field. This data collection is facilitated through the use of a drone equipped with a deep learning model capable of identifying and pinpointing the locations of weeds. The drone performs a systematic survey of the field, capturing high-resolution images and employing the deep learning model offline to detect weed positions. The resultant data is then utilized as input for the coverage path planning module.

Due to the developmental status of the deep learning model, the current implementation involves manual data acquisition using a real-time kinematic GPS (RTK) system. The RTK system offers centimeter-level accuracy in pinpointing weed locations, which, for the purpose of this research, is considered sufficiently precise.

Handling Data Uncertainty: In a practical scenario, the deep learning model's output would include positional data of weeds along with an associated uncertainty measure, reflecting the inherent imprecision of the system. However, given the reliance on RTK data for this research phase, we assume perfect positional accuracy. Future implementations incorporating the deep learning model will necessitate the adjustment of weed regions based on the uncertainty associated with each data point. This uncertainty measure will be used to define the regions of influence for each weed, ensuring that the coverage path planning algorithm accounts for the imprecision in the data.

Region Definition and Overlap Management: Given the robot's extraction width of 60 cm, the field is divided into fixed regions. When data points from the drone indicate weed positions, these points often come with overlapping regions due to the growth patterns of weeds like Rumex, which tend to cluster.

Overlap Reduction: To address overlaps, the preprocessing algorithm calculates the centroids of overlapping regions. Points within overlapping regions are consolidated to avoid duplication,

ensuring that each weed cluster is represented by a single centroid. This consolidation reduces redundancy and enhances the efficiency of the coverage path planning.

Non-Overlapping Weeds: For weeds whose regions do not overlap with others, the center of each weed is directly considered as a centroid. This step ensures that isolated weeds are accurately accounted for in the data set.

Data Integration and Optimization: By processing the data to identify centroids and manage overlaps, we achieve a significant reduction in the total number of points. Preliminary results indicate a reduction of at least 40% in the number of data points, optimizing the dataset for coverage path planning. This reduction not only minimizes the total traversal length required for weed removal but also conserves energy and reduces redundant revisits.

The processed data is then fed into the coverage path planning module, which utilizes the optimized set of points to devise an efficient path for weed removal, ensuring comprehensive coverage with minimal resource expenditure. The following figure illustrates the data preprocessing workflow, highlighting the steps taken to transform raw positional data into an optimized dataset ready for coverage path planning.

This detailed preprocessing approach ensures that the data fed into the coverage path planning algorithm is both accurate and efficient, laying a robust foundation for effective weed removal operations in agricultural fields.

I can also discuss two clustering techniques, first one implemented for amny clusters and the second one is the above one. Also write in detail about the procedure followed for the above mentioned steps.

5..3 Algorithm Description

Behavioral Approach for Coverage Path Planning in Agricultural Fields

Following the preprocessing phase aimed at resolving the regions associated with designated points, the subsequent imperative lies in formulating an algorithm capable of comprehensively covering all identified points. With the completion of preprocessing, the focus narrows down to ensuring the precise coverage of all points to effectively identify and extract weed infestations. Although the agricultural robot in question operates under non-holonomic constraints, the overarching objective transcends mere efficiency and the identification of the shortest path. Instead, the primary emphasis lies in achieving exhaustive point coverage while strategically favoring approximately linear trajectories.

The rationale behind prioritizing linear trajectories over strictly adhering to non-holonomic paths is multifaceted. Firstly, the adoption of more curved paths significantly heightens the risk of grass damage, thereby undermining the fundamental objective of preserving grass quality. Given the paramount importance of maintaining optimal grass conditions within agricultural fields, any approach that compromises this aspect inherently fails to align with the core objectives. Secondly, the energy consumption associated with traversing curved paths is substantially higher compared to linear trajectories. For the specific agricultural robot under consideration, empirical estimates suggest that traversing an equivalent path length via curved trajectories incurs an energy expenditure four times greater than that of linear paths. Consequently, the central objective of the algorithm resides in identifying the shortest path capable of encompassing all designated points while mitigating curvature and prioritizing linear trajectories.

The development of this behavioral approach necessitates a nuanced understanding of agricultural terrain dynamics, robot kinematics, and energy efficiency considerations. By integrating these facets into the algorithmic design process, the resultant solution seeks to strike a delicate balance between point coverage efficacy, grass preservation, and energy optimization.

Vision Cone Strategy:

At this stage, having acquired comprehensive global information about the points in the field, the next step involves prioritizing straight paths while minimizing computational complexity and time. An innovative approach is employed wherein the robot is equipped with a vision cone mechanism. This vision cone is defined by two lines extending from the robot at a fixed angle and distance. The angle of these lines is determined based on the robot's minimum turning radius. For instance, for a robot with a minimum turning radius of 2 meters, the angle of the

cone on either side is set at 11 degrees. The distance to the end of the cone depends on the operational area of the robot but is set to a fixed distance of 100 meters for this scenario.

The vision cone allows the robot to consider only those points within this cone from its current position as potential next travel points. This selective consideration significantly reduces the computational effort required to determine the path, as it disregards points outside the cone. By narrowing the focus to relevant points within the vision cone, computational efficiency is enhanced, thus making the vision cone an intelligent and effective strategy.

Algorithmic Framework:

The algorithmic framework is designed to facilitate comprehensive point coverage while minimizing curvature and prioritizing linear trajectories. The behavioral approach adopted for the algorithm is hierarchical, comprising three distinct behaviors that are sequentially activated. The transition from one behavior to the next is contingent upon the degree of point coverage achieved.

1. **Initial Behavior:** The algorithm commences with the first behavior, designed to initiate coverage from the starting point. This stage focuses on covering a substantial portion of the field, leveraging the vision cone to select the next travel points and maintaining the priority on straight paths.
2. **Intermediate Behavior:** Upon achieving a certain threshold of point coverage, the algorithm transitions to the second behavior. This intermediate stage aims to further optimize coverage by adjusting the strategy based on the points that remain. The robot continues to utilize the vision cone but adopt a slightly more flexible criteria for point selection to ensure efficient coverage progression.
3. **Final Behavior:** Once the intermediate behavior reaches its saturation point—where additional coverage gains diminish—the algorithm shifts to the final behavior. This stage is designed to ensure complete, 100% coverage of all remaining points. The final behavior will incorporate more refined strategies to target any residual areas, ensuring no point is left uncovered.

This hierarchical behavioral algorithm ensures a methodical and efficient approach to coverage path planning. By starting with broad coverage strategies and progressively refining the approach, the algorithm effectively balances the need for comprehensive point coverage with the constraints of the robot's kinematic capabilities and the operational goal of preserving grass

quality. The vision cone mechanism plays a pivotal role in this process, enhancing computational efficiency and enabling intelligent path selection.

Rationale for Hierarchical Approach:

The hierarchical approach is adopted to improve convergence rate and coverage efficiency. If a single algorithm or behavior were followed throughout, the robot would cover many points initially but gradually cover fewer points over time, reducing the algorithm's accuracy and increasing the overall path length and operational time. By transitioning between different behaviors, the algorithm can adapt to the changing density and distribution of points, maintaining high efficiency throughout the coverage process.

This hierarchical strategy ensures that the algorithm remains effective even as the number of uncovered points decreases. By tailoring the approach to the specific conditions encountered at each stage, the robot can optimize its path, reduce unnecessary movements, and maintain high precision in point coverage. This not only conserves energy but also preserves the quality of the grass by minimizing excessive traversal. The adaptive nature of the hierarchical approach thus represents a robust and efficient solution for coverage path planning in agricultural fields.

5..3.1 First Behavior: Initial Coverage (change its name)

The algorithm begins by receiving the centroids of the overlaps between the regions of the raw points as input. It takes the initial position and orientation of the robot, where the position remains fixed while the orientation is treated as a temporary orientation. From this temporary orientation, the algorithm considers an angular range of twenty-four degrees on either side. Within this range, it samples six orientations on each side, resulting in a total of thirteen orientations, including the initial orientation.

The algorithm designates the current orientation as the leftmost extreme orientation. Using this position and orientation, it then starts selecting the next point to navigate to. At this stage, the algorithm employs the vision cone mechanism, characterized by an 11-degree angle on either side and extending 100 meters forward. The primary objective at this juncture is to select the next most suitable point within the vision cone.

To determine the next best suitable point, the algorithm extracts the five closest points from the current position and orientation within the vision cone. These points are regarded as intermediate potential points. For each of these intermediate points, the algorithm computes the distribution of points on either side of the current orientation across the entire vision cone. A combined score is then calculated for each intermediate point based on the distance from the robot and the distribution of points in the long run. Both the distance and the distribution are normalized before calculating the combined score. The point with the highest combined

score is deemed the best potential candidate and is selected as the next point to navigate to. This scoring method is intelligent as it considers not only the distance but also the future distribution of points, thereby facilitating more efficient coverage as the robot progresses.

Once the best potential point is selected, it becomes the next point for navigation. The robot's orientation at this new point is updated based on the direction vector from the robot to the selected point. The robot then moves to this potential point with the updated orientation. This process is repeated, with the robot continually navigating to the next potential point until no points are visible within the vision cone. This pattern encourages the robot to move in an approximately straight line whenever possible. When the robot can no longer find any points within the vision cone, it indicates that the robot has reached the extreme end of the points. At this point, the number of points covered in this orientation is recorded.

Returning to Initial Position and Orientation

After completing the initial coverage path, the robot will return to its starting position and orientation. The robot then considers the next orientation in the set of thirteen sampled orientations and proceeds to complete an approximately straight path in that direction, recording the number of points covered along each path. This process is repeated for all thirteen orientations. Upon completing paths for all orientations, the orientation that results in the maximum number of points covered is selected as the optimal orientation for further navigation.

Incorporating Non-Holonomic Constraints for Turns

When making a turn, the robot must adhere to its non-holonomic constraints. To address this, the algorithm considers a circular region around the robot with defined minimum and maximum radii. The minimum radius is set to one and a half times the robot's minimum turning radius, ensuring feasible turns. The initial maximum radius is set to three times the minimum turning radius. If no points are found within this initial circular region, the outer radius is incrementally increased by two units until the maximum radius limit is reached.

Within this circular region, the algorithm identifies all potential points and filters them further. The selection process involves evaluating each point based on a combined score, considering both the distance from the robot and the angle differnce between the robot's orientation and the vector from the robot to the potential point. both the entities are normalized before comparison. A lower distance and a smaller orientation deviation result in a higher score for the point. The point with the highest score is selected as the next potential point for the robot to navigate to when making a turn.

Orientation Towards Centroid

At the beginning of the algorithm, the centroid of the entire set of points is computed. When selecting a point for the turn, the orientation is determined by the vector from the selected point

towards this centroid. This selected point becomes the next navigation target, and the orientation towards the centroid becomes the new temporary orientation for the robot.

Continuing the Path and Making Turns

Upon reaching the end of a complete path, the robot needs to execute a turn. This turn is represented by a single completed path, after which the robot continues the same pattern of operation. From the point where the turn is made, the robot selects thirteen orientations and navigates along each one, completing a path and recording the number of points covered. The orientation that results in the maximum number of points covered is then selected as the optimal orientation for continued navigation. This process is iterative and continues throughout the robot's operation.

Focus on Points Near the Centroid

This first behavior of the robot emphasizes covering more points near the centroid of the data and not covering points on the outer sides. Initially, the robot covers a significant number of points with each path. However, as the number of turns increases, the coverage rate decreases. This reduction in efficiency occurs because the robot focuses on points near the centroid as after each turn it faces toward the centroid, resulting in majority of the points covered are in the central region. Consequently, while this behavior ensures substantial coverage, it becomes less efficient over time as fewer new points are covered with each turn.

Coverage Efficiency and Convergence Rate

Although this behavior can cover most or all points in the data, the coverage rate diminishes, and convergence takes longer. This limitation highlights the need for a hierarchical approach with multiple behaviors. By transitioning between behaviors, the algorithm can maintain high efficiency and improve the convergence rate, ensuring comprehensive and timely coverage of all points.

Second behavior: Intermediate behavior:

Boundary-Focused Coverage Strategy

The second behavior of the algorithm is designed to enhance the coverage rate and improve the convergence efficiency. Upon transitioning from the first to the second behavior, the algorithm takes all remaining points as input, considering the current position and orientation of the robot. At this stage, the density of uncovered points is lower near the centroid and higher towards the boundaries of the operational area. Therefore, the second algorithm focuses on covering points located along the periphery rather than near the centroid.

Orientation Sampling and Path Selection

The algorithm begins similarly by selecting thirteen orientations from the robot's current position and orientation. The robot navigates along each orientation, completing a path and recording the number of points covered. The orientation resulting in the highest number of

points covered is selected as the optimal direction for continued navigation. This ensures that the robot always moves in the direction that maximizes point coverage.

Refined Criteria for Selecting Turning Points

A critical difference in this behavior lies in the method for selecting the next best point to make a turn. The algorithm considers a circular region around the robot with predefined minimum and maximum radii similarly as of first behavior. Within this region, the algorithm evaluates all potential points, applying refined selection criteria. The criteria prioritize points with the smallest absolute orientation difference from the robot's current orientation and the shortest distance from the robot's current position. By doing so, the algorithm ensures that the robot makes smooth, efficient turns that align with its non-holonomic constraints.

The point with the least orientation difference and the shortest distance is chosen as the next point for making a turn. Unlike the first behavior, where the orientation towards the centroid was prioritized, the orientation of this potential point is directed towards the next point along the same moving orientation. For example, if the potential point lies clockwise to the robot's current orientation, the robot's new orientation will aim towards the next nearest point with the smallest clockwise orientation difference, and the same applies for counterclockwise points. This new orientation becomes the temporary orientation, and the robot continues the process of orientation sampling and selection with the goal of maximizing point coverage.

Enhanced Coverage and Convergence

This behavior is particularly effective in covering points along the boundaries and in regions with higher point density, thereby increasing the overall coverage rate and convergence efficiency of the algorithm. While the first behavior predominantly focuses on points near the centroid, the second behavior shifts attention to the boundary points, ensuring a more balanced and comprehensive coverage pattern.

By following a systematic circular pattern, the robot can cover most of the points in the data set. This transition from centroid-focused coverage to boundary-focused coverage allows the algorithm to address the limitations of the first behavior, which tends to become less efficient as more points near the centroid are covered. The hierarchical combination of these behaviors ensures that the robot effectively covers the entire field, optimizing both the coverage rate and the overall efficiency of the operation.

Enhancing Coverage with Intermediate Points

To further improve the coverage rate and convergence of the algorithm, an additional mechanism is incorporated into both the first and the second behaviors. This enhancement involves covering intermediate points along each path between two selected points. The algorithm checks for points that lie close to the current path within a certain threshold distance.

Regardless of the distance between the two primary points on the path, these intermediate points are included in the coverage plan.

The orientation for these intermediate points is determined based on their position. If multiple points exist close to the path, the orientation is directed towards the next intermediate point in sequence. For the last intermediate point, the orientation is towards the final end point.

Another criterion for selecting intermediate points depends on the distance between the two primary points of the path. If this distance is substantial, points that are slightly farther but within the threshold are also considered. In such cases, the algorithm allows for a minor compromise on the straightness of the path to cover more points, thereby further enhancing the coverage rate and convergence efficiency.

Efficiency of Combined Behaviors

Both the first and second behaviors, along with the enhancement for intermediate points, ensure complete coverage of the points. These behaviors have been experimentally validated to provide good coverage rates and convergence for the algorithm. However, the efficiency in terms of coverage rate, convergence speed, and path length diminishes after approximately eighty-five to ninety percent of the points are covered. At this stage, the algorithm tends to cover fewer points per turn and makes multiple turns to cover just two or three points.

This inefficiency results in longer paths and higher energy consumption, which contradicts the goal of minimizing path length and conserving energy. This challenge highlights the necessity for a third behavior. The third behavior aims to optimize the final stages of the coverage process by allowing slight deviations from straight paths to cover more points efficiently, thus reducing the overall path length and energy consumption.

Final Behavior: Completing Coverage:

Final Behavior: Completing Coverage The third and final behavior of the algorithm addresses the coverage of the few remaining points, which are sparsely distributed across the area. At this stage, the concept of the vision cone is removed, as maintaining strict straightness is no longer prioritized. Although straight paths are typically more energy-efficient, the robot would consume more energy traveling long straight paths while covering fewer points. Instead, small circular turns with a radius of 3 meters are preferred, allowing the robot to avoid long straight

paths of up to 100 meters. This approach optimizes energy usage by prioritizing the coverage of remaining points over path straightness.

To determine the optimal path for these remaining points, the algorithm employs the Dubins open traveling salesman problem (TSP), leveraging the concepts described in a seminal paper by X.

Overview of the Traveling Salesman Problem (TSP) The traveling salesman problem is a classic optimization challenge in computer science and operations research. It seeks to find the shortest route that visits a set of cities and returns to the original city, with the primary challenge being to determine the optimal sequence of cities to minimize total travel distance. This problem is NP-hard, meaning there is no known polynomial-time algorithm that can solve it exactly for large instances.

Open Traveling Salesman Problem (Open TSP) The open TSP is a variant of the classic TSP where the salesman does not need to return to the starting city after visiting all other cities. This relaxation allows for more flexibility in route planning and can lead to different optimal solutions. The open TSP is particularly useful in scenarios where a return trip is unnecessary or infeasible, such as delivery routes or exploration missions. In our case, the robot does not need to return to the starting point, making the open TSP more suitable for our problem. Nevertheless, the non-holonomic constraints of the robot must be considered.

Dubins Open Traveling Salesman Problem (Dubins Open TSP) The Dubins open TSP extends the open TSP by accounting for the non-holonomic constraints of vehicles, such as turning radius and orientation. It seeks to find the shortest path that visits a set of points while respecting the vehicle's kinematic constraints. The Dubins path, named after mathematician L. F. Dubins, provides the shortest path for vehicles with a fixed turning radius. By applying Dubins paths to the open TSP, the algorithm can generate efficient and feasible routes for vehicles with non-holonomic constraints.

Application to the Final Coverage The final behavior leverages the Dubins open TSP to efficiently cover the remaining points. Initially, the shortest path between the points is computed using the open TSP. Subsequently, Dubins constraints are applied to ensure the path is feasible for the robot. This method not only ensures complete coverage of the remaining points but also reduces the overall path length and energy consumption. By adopting this approach, the algorithm significantly improves the coverage rate and convergence speed, achieving comprehensive coverage in an efficient manner.

This integrated strategy, comprising the first, second, and final behaviors, enables the robot to cover all points in the agricultural field effectively. Each behavior is tailored to optimize different stages of the coverage process, from focusing on dense central areas to sparsely populated boundaries and finally to the remaining isolated points. This hierarchical and adaptive

approach ensures that the robot operates efficiently, minimizing both path length and energy consumption throughout the coverage process.

Hierarchical Behavioral Algorithm for Coverage Path Planning These three behaviors are meticulously designed to optimize the coverage of points in an area while minimizing the path length and energy consumption of the robot. The algorithm demonstrates a sophisticated approach to coverage path planning, particularly suited to agricultural fields, but adaptable to other domains as well.

Automatic Shift Between Behaviors: A critical aspect of this behavioral algorithm is the decision-making process for shifting from one behavior to the next. Different data distributions require varying coverage percentages to optimize this transition. Determining the optimal shift point is challenging and necessitates extensive experimentation and testing to achieve the best coverage rate and convergence.

The performance of the algorithm can significantly vary depending on when the shift between behaviors occurs. For the same dataset, different coverage percentages for behavior shifts can greatly influence the coverage rate and convergence efficiency. Therefore, it is essential to tailor the shift points for each specific dataset rather than relying on a fixed percentage.

Computing the Optimal Coverage Percentage One of the standout features of this behavioral algorithm is its ability to determine the near-optimal coverage percentage for shifting behaviors. Once the dataset is pre-processed, the algorithm computes the best percentage coverage for behavior shifts as follows:

- **Centroid and Concentric Circles:** The algorithm first computes the centroid of the dataset. Imaginary concentric circles are then centered at the centroid, with varying radii extending outward until all points are encompassed.
- **Point Distribution Analysis:** The percentage of points within each concentric region is calculated. The behavioral shift percentage is determined based on a threshold percentage found through experimentation and testing, which is 50 percent. The points are counted in each region, starting from the innermost circle, and the percentages are accumulated until the threshold is reached.
- **Threshold Determination:** The region where the threshold percentage is exceeded dictates the coverage percentage for the behavior shift. If the threshold is reached in the innermost region, indicating a dense point distribution near the centroid, the first behavior will be more effective, and the shift percentage will be automatically set to a higher percent (e.g., 60-70%). Conversely, if the threshold is reached in the outer regions, suggesting denser distribution near the boundaries, the second behavior becomes more pertinent, and the shift percentage is automatically set to a lower percent (e.g., 30-40%) by the algorithm.

This intellectually behavior shifting enables the algorithm to determine the optimal coverage percentage, thereby enhancing the overall coverage rate and convergence efficiency. By integrating these calculated shift points, the hierarchical behavioral algorithm adapts dynamically to different datasets, ensuring that the robot operates efficiently across various scenarios. This adaptive approach significantly improves the algorithm's performance, as demonstrated in the subsequent results.

5..4 Obstacle Avoidance

Obstacle Avoidance in Coverage Path Planning Obstacle avoidance is a crucial component of coverage path planning (CPP), applicable in various contexts such as agricultural fields, warehouses, and more. In agricultural fields, static obstacles include trees, rocks, houses, and other structures, collectively referred to as keep-out zones. This discussion focuses on the avoidance of static obstacles within the scope of coverage path planning.

The CPP algorithm considers several inputs and constraints. The robot operates with non-holonomic constraints, meaning it cannot move directly sideways and must follow a path that considers its motion limitations. The total area to be covered is defined, ensuring the robot stays within the designated field. Obstacles are represented as polygons of various shapes and sizes, which the robot must avoid. Additionally, the vision cone defines the area that the robot can see and use for straight-path approximation, aiding in real-time obstacle detection and avoidance.

Prior to executing the algorithm, comprehensive data about the field is collected using a drone. The drone flies over the field, capturing images to generate a detailed map. This map is instrumental in defining the grid for path planning and identifying and mapping out obstacles accurately. By having all the necessary data beforehand, the CPP algorithm can be initialized with a complete overview of the field, including obstacles.

One of the critical aspects of the CPP algorithm is its efficiency in obstacle avoidance. Given that the robot will encounter obstacles multiple times during its operation, it is essential that the algorithm minimizes the computation time required for each avoidance maneuver. The algorithm should be optimized to quickly navigate around obstacles. Path adjustments must be computed swiftly to ensure minimal disruption to the overall coverage path. By focusing on these efficiency improvements.

Obstacle Avoidance in Coverage Path Planning The obstacle avoidance algorithm in coverage path planning (CPP) consists of two main components:

Initial setup once all the data is received. Real-time operation where the robot follows the path generated by the algorithm.

Initial Setup The initial setup begins once the algorithm receives the total area, obstacles represented by their vertices, and the vision cone. The first crucial step is to account for the robot's physical dimensions by creating a configuration space. This ensures that the algorithm considers the robot's actual operating space, not just a point in the field.

CPP operates in continuous space for path planning, but representing obstacles solely in continuous space is computationally prohibitive, especially in obstacle-rich environments. On

the other hand, completely using discrete space is impractical for large fields, such as agricultural areas, because methods like occupancy grids would be too expensive to generate and manage over extensive areas.

To address this, a hybrid approach is adopted. Continuous space is used for general path planning, while obstacles are represented in both continuous and discrete spaces. Discrete space involves creating separate occupancy grids for each obstacle. A critical challenge here is determining the size of the grid cells: they must be small enough to accurately represent the obstacle yet large enough to avoid excessive computational load. The algorithm addresses this by generating dynamic grids based on the size of the obstacle and the robot's non-holonomic constraints.

Dynamic Grid Generation The initial objective is to dynamically generate an occupancy grid for each obstacle. The algorithm begins by extending the vertices of each polygonal obstacle to create a safe zone around it. This extended obstacle is then approximated as a rectangle using the minimum and maximum coordinates of the extended polygon.

To generate the grid for this approximated obstacle, the algorithm considers the robot's non-holonomic constraints. It calculates a curve using the two extreme angles of the vision cone and a step length, ensuring the robot can navigate this curve with the minimum turning radius until a 90-degree turn is achieved. At this point, the changes in the x and y coordinates (dx and dy) are recorded. These values are used to determine the grid's dimensions, with an allowance of 1.5 times the curve distance along the robot's heading and half the obstacle's width. This setup ensures that the robot can navigate around the obstacle even with its non-holonomic constraints.

Once the grid size is determined, the grid cells are generated. The cell size is chosen based on the robot's dimensions: larger robots do not require fine grids as they cannot move cell by cell. Therefore, an appropriate grid cell size is selected to suit the robot's size.

The setup includes two main components:

Occupancy Grid: Represents the grid cells. Grid Centers: Points within the grid cells used to obtain salient points for obstacle avoidance in the world space.

For each obstacle, point data is generated over the occupancy grid. The algorithm extracts points around the obstacle's boundary, known as buffer points. These buffer points help in navigating around the obstacles efficiently.

Dynamic Obstacle Avoidance in Coverage Path Planning This section outlines the second phase of the obstacle avoidance algorithm, focusing on the real-time aspects of navigation once

the initial setup is complete. After setting up the obstacles and generating the necessary grids, the algorithm proceeds to follow a regular behavioral approach for path planning.

The path generation process begins as described previously. However, this time, the algorithm includes a mechanism to detect collisions with obstacles. If no obstacles are detected along the planned path, the behavioral algorithm operates normally. When an obstacle is detected, the algorithm initiates a sequence of steps to navigate around it.

Collision Detection and Salient Point Identification Upon detecting an obstacle in the path, the algorithm first identifies a line along the robot's heading and checks for intersection points with the obstacle. At this stage, buffer points—previously defined around the obstacle—become critical. The algorithm identifies the furthest buffer point along the robot's heading, which marks the end of the line. It then calculates the perpendicular distances from this line to all buffer points, selecting two points on either side of the line at the maximum distance. These points are designated as salient points, serving as key navigational targets to avoid the obstacle.

Once the salient points are determined, they become the goal points for the robot to bypass the obstacle. The robot's current position is the starting point for this avoidance maneuver. The algorithm then checks whether the robot is inside the grid. If the robot is outside the grid, the salient points are directly used as goal points, and the robot navigates towards them following its non-holonomic constraints. The grid is designed such that if the robot is at the edge, it can avoid the obstacle by reaching these extreme points directly.

Graph-Based Path Finding If the robot is inside the grid, a more complex process ensures that the robot can reach the salient points while adhering to its motion constraints. A graph-based approach is utilized to find the shortest and feasible path from the robot's current position to the salient points using the grid cells.

Efficient and rapid graph generation is crucial, as this process must occur each time an obstacle is encountered. The algorithm employs two extreme angles of the vision cone and a step length to create the graph. Each iteration produces a new generation of grid cells. For instance, if the robot occupies one cell, the next generation, based on the two extreme angles and step length, will occupy two cells. Subsequent generations expand similarly, covering more cells until the goal point is included in the graph.

One challenge in this graph-based approach is balancing the step length. If the step length is too short, the graph requires many generations to reach the goal point, increasing computational time. Conversely, if the step length is too long, the graph may become sparse, risking the possibility of not adequately covering the goal point and potentially passing through it.

To ensure an efficient and adaptive approach to obstacle avoidance, the algorithm dynamically determines the step length for graph generation. By allowing the algorithm to decide the

step length, it can find a near-optimal length that generates a sparse graph initially, gradually becoming denser as it approaches the goal point. This adaptive strategy optimizes computational efficiency while ensuring thorough coverage.

Automating the selection of step length involved an experimental analysis to understand its dependence on various parameters. Notably, the distance to the salient point emerged as a significant factor. By conducting experiments with different salient point positions and step lengths, a linear relationship between the distance and step length was identified. Consequently, the algorithm fits a line to this data at the outset, enabling it to automatically determine the dynamic step length based on the distance to the salient point. This near-optimal step length encourages the algorithm to generate a sparse graph initially, gradually densifying it as it approaches the goal point. This approach significantly enhances computational efficiency.

Once the graph is generated, multiple paths from the robot's position to the salient point are available. The algorithm employs an A* search over the graph to find the shortest path. The intermediate points along this path serve as the route to avoid the obstacle and reach the goal point efficiently and swiftly. Subsequently, the regular behavior resumes for further coverage of the designated points.

If the generated graph does not include the goal point, indicating that the goal point is unreachable within the constraints, the algorithm retraces one step back in the path. It then repeats the process from the point where the obstacle was detected, ensuring that the robot can circumvent the obstacle and complete its coverage path planning seamlessly.

This iterative process repeats each time an obstacle is encountered, enabling the robot to efficiently navigate around obstacles and reach its designated goal points. This comprehensive approach ensures robust obstacle avoidance within the coverage path planning algorithm, facilitating efficient and timely completion of tasks.

The obstacle avoidance approach in coverage path planning addresses several critical challenges with innovative solutions. These include dynamic grid selection to balance computational complexity and accuracy, employing a hybrid space approach to represent obstacles, and optimizing computational time through dynamic step length determination. The algorithm dynamically adjusts step length based on the distance to the salient point, ensuring near-optimal path planning efficiency. Additionally, utilizing both continuous and discrete spaces allows for efficient representation of obstacles. Moreover, employing an A* search algorithm facilitates the determination of the optimal shortest path, enabling swift and effective obstacle avoidance. These integrated strategies ensure robust obstacle avoidance within the coverage path planning algorithm, enhancing overall efficiency and effectiveness.

6. Simulation Test and Analysis

This section presents a thorough exploration of the proposed complete coverage path planning algorithm tailored for agricultural field applications. It unfolds into two distinct sections:

Simulation Setup:
This section is dedicated to providing a detailed exposition of the simulation setup. Meticulous attention is directed towards delineating the experimental framework within which the simulation tests are conducted.

Simulation Results and Analysis:
Comprehensive empirical findings obtained from the meticulously designed simulation experiments within the aforementioned framework are detailed in this section. Key performance metrics are analyzed to provide a holistic assessment of the algorithm's performance under diverse scenarios. we conduct a thorough analysis of the empirical results, meticulously scrutinizing each detail. Through this examination, we aim to discern the operational dynamics of the algorithm, identify its strengths, and pinpoint areas for potential refinement.

6..1 Simulation Setup

The simulation setup is meticulously designed to emulate the real-world operational environment of an agricultural field, ensuring that the algorithm's performance can be rigorously tested under realistic conditions. The global coverage path planning is initially performed using Matplotlib, followed by testing in a simulation environment leveraging the Robot Operating System (ROS) and the Gazebo simulator.

In the context of coverage path planning for agricultural fields, it is crucial to account for varying plant distributions. Therefore, different datasets are used to represent diverse scenarios, with the aim of assessing the algorithm's robustness under varying initial conditions and plant distributions. Specifically, the simulation setup incorporates different datasets with the same initial position and orientation of the robot to demonstrate the algorithm's adaptability.

Dataset Description:
To comprehensively evaluate the algorithm, six distinct datasets are employed within the flat field of 120m x 120m area, with each dataset containing different numbers of points: 250, 500, 2000, 4000, 6000, and 10,000. Each primary dataset is further subdivided into four sub-datasets, characterized by different spatial distributions:

- Random Distribution: Points are distributed randomly throughout the field.
- Clustered Distribution (4 Clusters): 20% of the points are distributed randomly, while 80% are distributed in four clusters with distinct Gaussian distribution variances.

- Clustered Distribution (6 Clusters): 20% of the points are distributed randomly, with the remaining 80% distributed in six clusters, each with distinct means and variances.
- Clustered Distribution (10 Clusters): Similar to the previous sub-datasets, 20% of the points are randomly distributed, and 80% are in ten clusters, each characterized by unique Gaussian distribution parameters.

This results in a total of 24 different datasets, meticulously crafted to test the algorithm's robustness across various scenarios, including variations in the number of points, their distribution patterns, the number of clusters, and the statistical properties of these clusters. Such comprehensive testing ensures that the algorithm is robust and adaptable to real-world agricultural environments, where plant distributions can significantly vary.

Performance Metrics The performance of the algorithm is evaluated using several critical metrics across all datasets:

- Computational Time: The time required for the algorithm to compute the coverage path.
- Field Operation Time: The actual time taken for the robot to complete the coverage task.
- Total Path Length: The total distance traveled by the robot.
- Number of Turns: The total number of turns the robot makes during its operation.
- Energy Consumption: The energy used by the robot to complete the coverage task.
- Coverage Efficiency: The percentage of points covered by the robot within the operational time.

These metrics provide a comprehensive assessment of the algorithm's efficiency, effectiveness, and overall performance under different test scenarios.

Processor Specifications:
Processor: AMD® Ryzen 5 Pro 5675U with Radeon Graphics x 12.
Processing Speed: 2.3 GHz, Number of Cores: 6, and RAM: 16 GB.

Visual Representation of the Dataset
The distribution of points across the different datasets can be visualized in (Figure 1), illustrating the varying scenarios used to test the algorithm's robustness.

6..2 Simulation Results and Analysis

This section presents the detailed results and analysis of the simulation experiments conducted to evaluate the robustness of the proposed complete coverage path planning behavioral algorithm. The empirical findings are meticulously analyzed to provide insights into the algorithm's

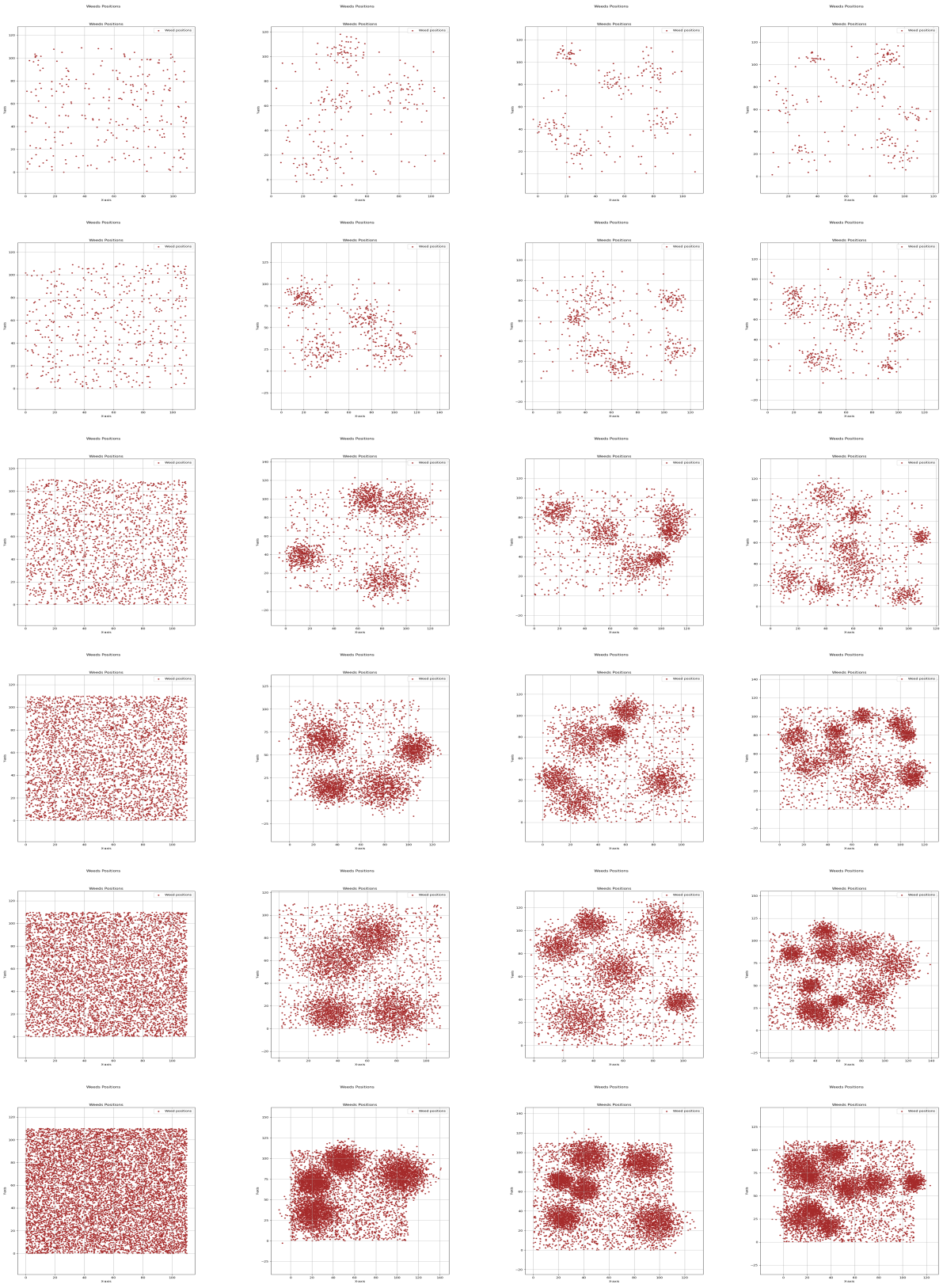

Figure 1: Dataset.

performance under diverse scenarios, enabling a comprehensive assessment of its efficiency and effectiveness.

Experimental Consistency To ensure the validity of comparisons and isolate the effects of different datasets on the algorithm's performance, the initial position and orientation of the robot were kept constant throughout all experiments. Additionally, the parameters for both the robot and the algorithm were consistently maintained across all trials.

Robot Constraints and Algorithm Parameters:
The robot's constraints and operational parameters are set to realistic values to ensure the validity of the simulation results. The parameters' data can be visualized in (Table 1), offering a comprehensive overview of the robot's physical and operational characteristics.

Table 1: Robot and algorithm Constraints and Parameters

Constraints and Parameters	***Values***
Robot initial position	[60.0, 1.0]
Robot initial orientation	120
Search angle of the vision cone	±11
Minimum search radius of the vision cone.	1.25
Maximum search radius of the vision cone.	100
Minimum turning radius allowed for the robot.	2.0
robot velocity on curved paths.	0.4
robot velocity on straight paths.	0.8
Distance used to compute field time taken on curved paths.	7.85
Number of points to consider in the vision cone.	5
Minimum distance between intermediate points, and start and end point.	2.7
Minimum distance between intermediate points.	2
Maximum distance of the point form the path.	0.6

The algorithm parameters are also meticulously defined to ensure consistent performance across all experiments. These parameters are set based on the robot's physical constraints and operational requirements, ensuring that the algorithm operates within realistic boundaries.

The algorithm was designed to initially compute the global coverage path using straight lines. Subsequently, Dubins paths are generated for each line segment to ensure the shortest feasible path between each pair of points, optimizing the overall path length. The (Figure 2) illustrate the straight paths generated by the algorithm for different datasets.

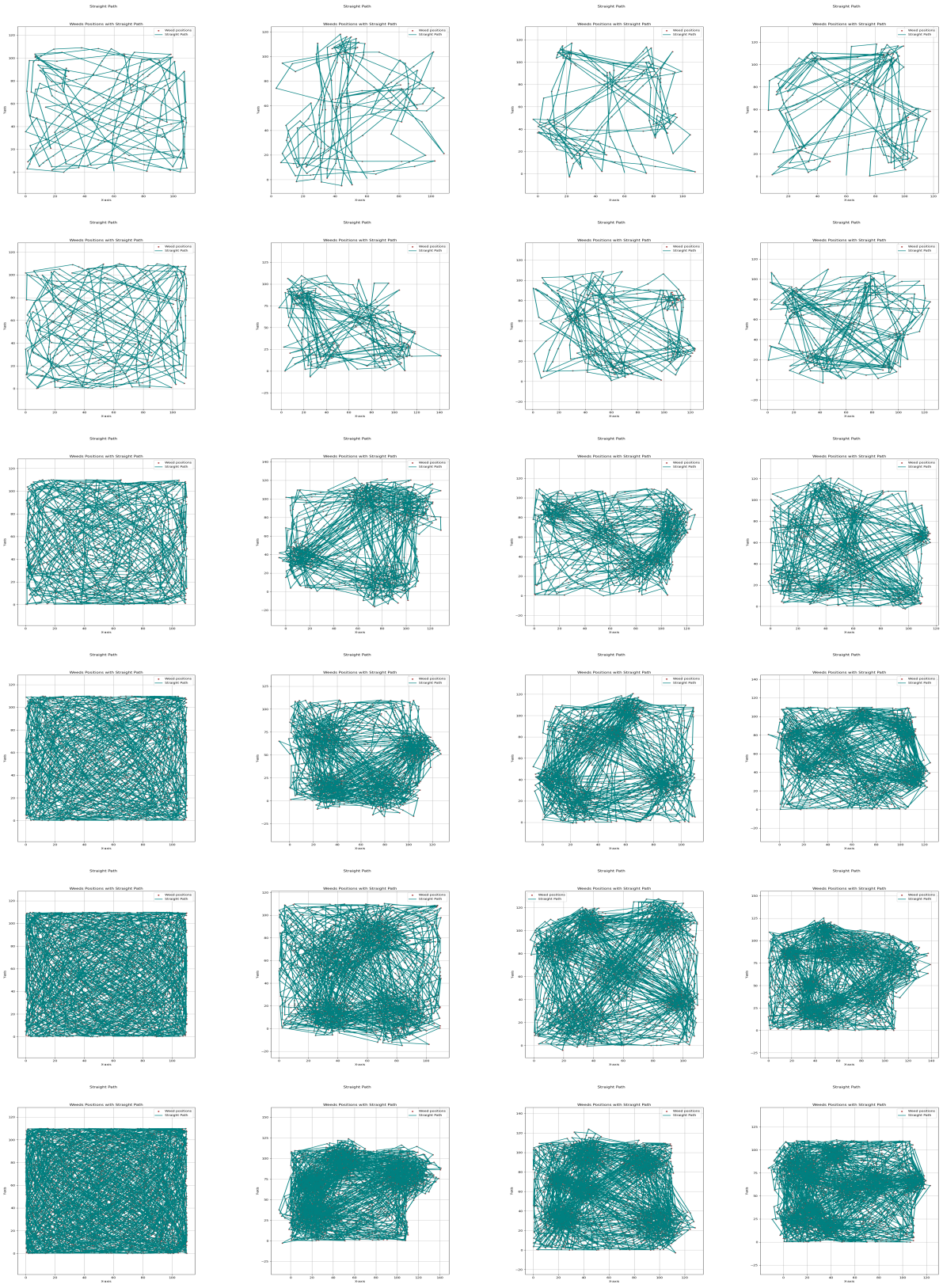

Figure 2: Straight Path.

Table 2: Results for the performance metrics.

No. of points	*Cases*	*Computation Time in secs.*	*Field Operation Time in sec.*	*Route length in meters*	*No. of Turns.*	*Energy Consumption in KWh.*
250	case 1	0.858553886	6282.400938	4652.945	48	5783.34459
	case 2	0.775789261	5125.168209	3769.005	41	4734.555207
	case 3	0.788151503	5413.822753	3985.978	44	5022.177564
	case 4	0.825942278	6216.01818	4620.695	44	5656.894864
500	case 1	1.842700005	9520.140257	7027.363	75	8793.612844
	case 2	1.696401358	9151.211911	6788.619	67	8366.468569
	case 3	1.940592766	8589.717644	6336.544	67	7914.394434
	case 4	1.609200478	9761.624722	7250.2	70	8898.700417
2000	case 1	9.04812026	21751.64778	16189.858	154	19816.55759
	case 2	8.591621161	22226.37676	16650.701	144	20041.90077
	case 3	7.65038085	18446.73883	13728.721	131	16813.77138
	case 4	8.155776024	20839.67217	15547.469	139	18820.91894
4000	case 1	24.18394971	32301.69933	24191.259	210	29136.75914
	case 2	20.85941482	29871.08874	22356.011	198	27018.91147
	case 3	18.23963118	27659.4681	20620.054	192	25141.65448
	case 4	19.22349644	29701.58872	22222.31	200	26932.31049
6000	case 1	45.23673987	39851.80224	29777.382	262	35947.48179
	case 2	33.23154497	34821.16749	26039.854	236	31597.65399
	case 3	30.80121374	35035.83592	26223.169	230	31639.66906
	case 4	32.34737062	35496.36509	26570.832	226	31893.13207
10000	case 1	122.6808474	50681.89334	38128.675	308	45382.07468
	case 2	74.37674475	45367.87779	34118.252	277	40641.60223
	case 3	65.54510021	42291.24514	31650.696	278	38197.59611
	case 4	62.76576424	42801.16425	32154.031	274	38606.73138

Once the Dubins paths are generated from the straight paths, the resultant paths can be visualized in the (Figure 3). These figures showcase the robot's traversal across different datasets, illustrating how the algorithm adapts to various point distributions.

The performance metrics for all datasets are summarized in the (Table 2). These metrics provide insights into the algorithm's computational efficiency, operational effectiveness, and overall robustness.

The coverage rate plot over the field time offers a visual representation of the algorithm's performance across the whole datasets. This plot demonstrates how effectively the algorithm achieves coverage over time, allowing for a comparative analysis of its efficiency under varying point distributions and densities. This plot can be visualized in (Figure 4).

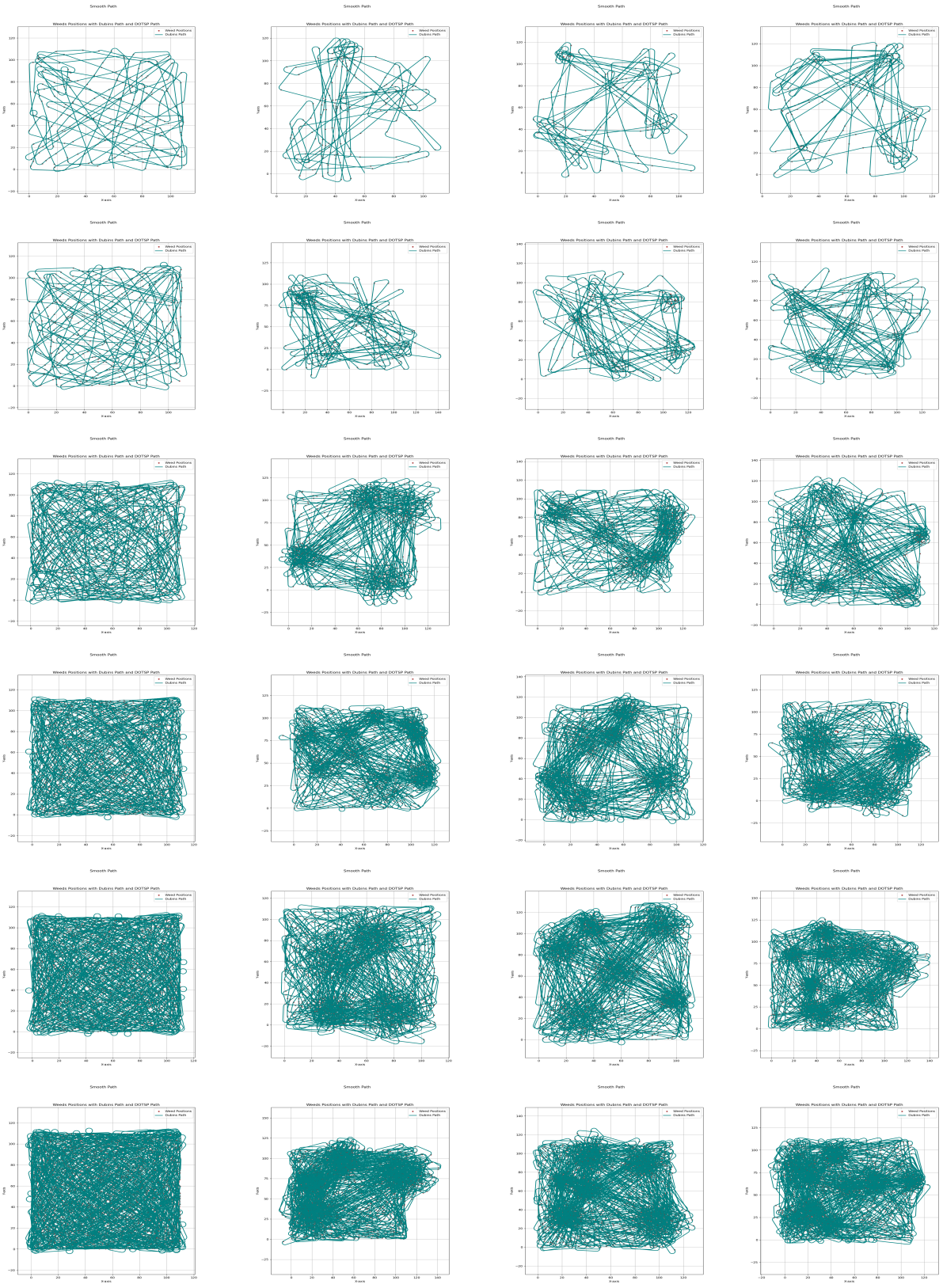

Figure 3: Dubin's Path.

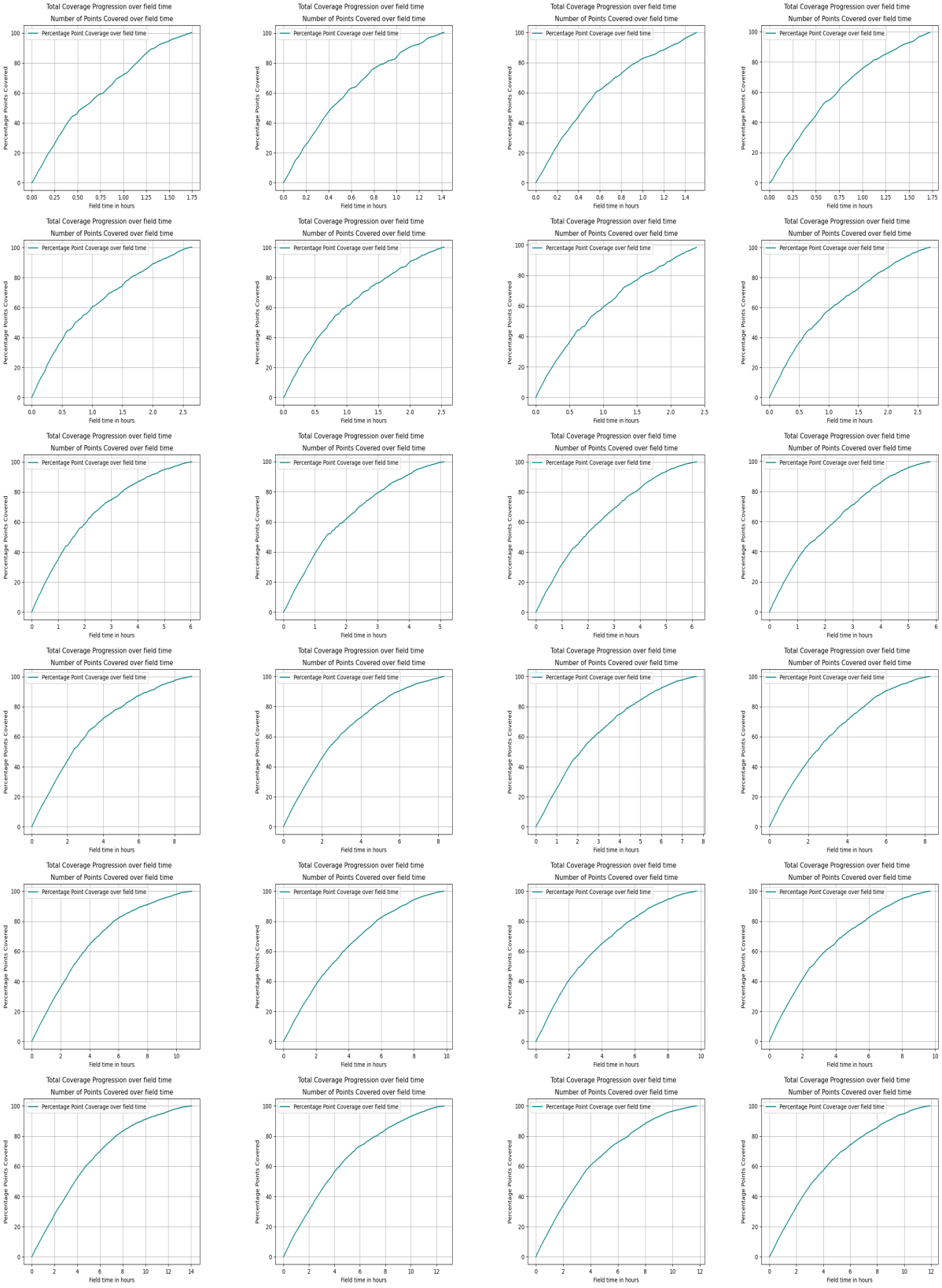

Figure 4: Coverage plots.

Performance Metrics plots.

Analysis:

Need to write it properly after putting all the results.

The analysis of the results reveals several key insights into the algorithm's performance. Coverage efficiency remains consistently high across all datasets, demonstrating the algorithm's robustness in various scenarios. Computational time and field operation time scale predictably with the number of points, highlighting the algorithm's scalability. The number of turns and energy consumption metrics indicate that while Dubins path smoothing enhances path traversal efficiency, it also demands more computational resources.

The coverage rate plot further corroborates the algorithm's effectiveness, showing high coverage efficiency across varying point distributions and densities. These findings suggest that the algorithm is well-suited for real-world agricultural applications, where plant distributions can vary significantly.

By analyzing the coverage rate, we can gain deeper insights into the algorithm's ability to maintain high coverage efficiency consistently. The results from this plot are crucial in validating the robustness and adaptability of the proposed path planning algorithm.

7. Simulation Test and Analysis with obstacles

This section presents a comprehensive examination of the proposed complete coverage path planning algorithm, tailored to handle static polygonal obstacles in agricultural field applications. The section is divided into two key parts: Simulation Setup and Results & Analysis.

7..1 Simulation Setup

This subsection provides a detailed description of the simulation setup, focusing on the experimental framework within which the tests are conducted.

The simulation environment is meticulously crafted to replicate real-world agricultural fields, incorporating static polygonal obstacles to simulate common structures such as trees, rocks, or buildings that serve as keep-out zones. This realistic setup ensures that the algorithm's performance is rigorously evaluated under practical conditions.

This simulation environment is setup utilizing a realistic field data obtained from the field manually and inclusion of the obstacles within the field. While the point distribution remains unchanged, the obstacles' shapes, sizes, and numbers vary across different scenarios. Strategic placement of obstacles tests the algorithm's adaptability and robustness. Five distinct datasets are used to assess the algorithm's performance under varying obstacle configurations, increasing in complexity with each set.

Dataset Description
To thoroughly evaluate the algorithm, five distinct datasets are employed within a flat 120m x 120m field, each containing different numbers and configurations of obstacles:

- Set1: Contains one rectangular obstacle of size 10m x 10m.
- Set2: Contains two rectangular obstacles of sizes 10m x 10m and 4m x 4m.
- Set3: Contains three convex polygonal obstacles of distinct sizes and shapes.
- Set4: Contains six convex polygonal obstacles of different sizes and shapes.
- Set5: Contains six polygonal obstacles of varying sizes and shapes, including concave and extremely long obstacles.

This hierarchical arrangement of datasets, with increasing complexity, ensures comprehensive testing of the algorithm's robustness and adaptability to real-world agricultural environments, where obstacles can vary significantly.

Performance Metrics The algorithm's performance is evaluated using several critical metrics across all datasets:

- Computational Time: The time required for the algorithm to compute the coverage path.
- Field Operation Time: The actual time taken for the robot to complete the coverage task.
- Total Path Length: The total distance traveled by the robot.
- Number of Turns: The total number of turns the robot makes during its operation.
- Energy Consumption: The energy used by the robot to complete the coverage task.
- Coverage Efficiency: The percentage of points covered by the robot within the operational time.

These metrics provide a comprehensive assessment of the algorithm's efficiency, effectiveness, and overall performance under different test scenarios.

Visual Representation of the Dataset

The (Figure 5) illustrates the five different datasets used in the simulation experiments, showcasing the shapes, sizes, numbers, and locations of obstacles within the field. This visual representation aids in understanding the varying complexities introduced by the obstacles.

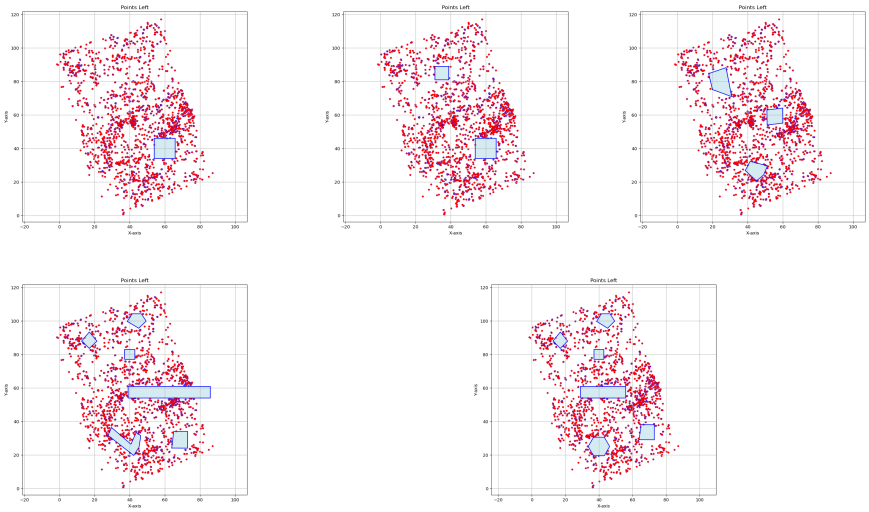

Figure 5: Obstacles Dataset.

7..2 Simulation Results and Analysis

This section details the results and analysis of simulation experiments designed to evaluate the robustness of the proposed complete coverage path planning algorithm in the presence of static polygonal obstacles. The empirical findings are meticulously analyzed to provide insights into

Table 3: Constraints and Parameters in presence of obstacles.

Constraints and Parameters	***Values***
Robot initial position	[60.0, 1.0]
Robot initial orientation	45
Search angle of the vision cone	±11
Minimum search radius of the vision cone.	1.25
Maximum search radius of the vision cone.	100
Minimum turning radius allowed for the robot.	2.0
robot velocity on curved paths.	0.4
robot velocity on straight paths.	0.8
Distance used to compute field time taken on curved paths.	7.85
Number of points to consider in the vision cone.	5
Minimum distance between intermediate points, and start and end point.	2.7
Minimum distance between intermediate points.	2
Maximum distance of the intermediate point form the path.	0.6
Graph search extreme angles	±11
buffer distance around obstacles.	1
Extend polygon distance for safe region.	1.4
Graph generation step reduction iteration.	3
Grid diameter.	0.4
Max generation for graph generation.	18
Minimum step length for graph generation.	1.2
Total operational area.	[[-2,0], [90,120]]
Distance between waypoints.	10

the algorithm's performance across diverse scenarios, offering a comprehensive assessment of its efficiency and effectiveness.

Experimental Consistency To ensure the validity of comparisons and isolate the effects of different datasets on the algorithm's performance, the initial position and orientation of the robot were kept constant throughout all experiments. Additionally, the parameters for both the robot and the algorithm were consistently maintained across all trials.

Robot Constraints and Algorithm Parameters:

The robot's constraints and operational parameters are set to realistic values to ensure the validity of the simulation results. These data can be visualized in (Table 3).

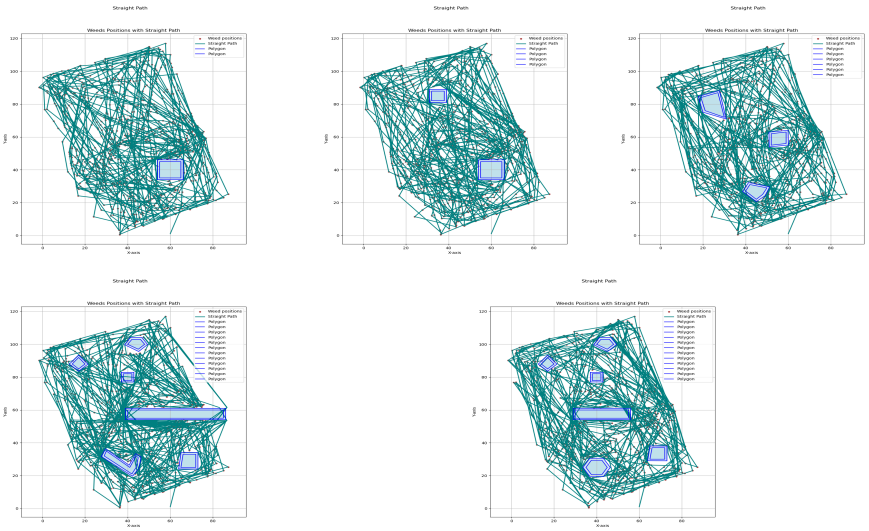

Figure 6: Straight Path.

The algorithm parameters are also meticulously defined to ensure consistent performance across all experiments. These parameters are set based on the robot's physical constraints and operational requirements, ensuring that the algorithm operates within realistic boundaries.

The algorithm was designed to initially compute the global coverage path using straight lines. Subsequently, Dubins paths are generated for each line segment to ensure the shortest feasible path between each pair of points, optimizing the overall path length. The (Figure 6) illustrate the straight obstacle free paths generated by the algorithm for all the datasets.

The resulting path after applying dubins constraints are depicted in the (Figure 7). These visual representations portray the robot's movement across diverse datasets, providing insight into the algorithm's adaptability amidst varying obstacles.

The (Table 4) summarizes the performance metrics for all datasets, offering valuable insights into the algorithm's computational efficiency, operational effectiveness, and overall robustness.

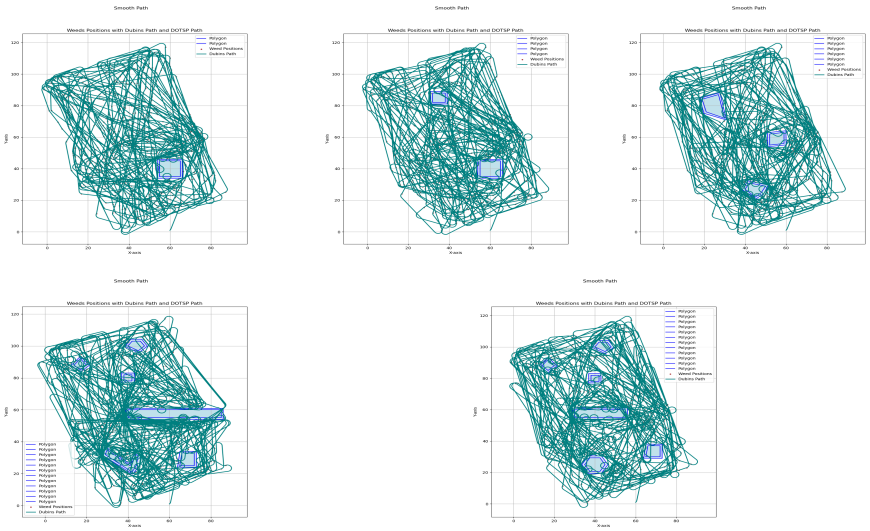

Figure 7: Dubins' Path.

Table 4: Results for the performance metrics.

Cases	*Computation Time in secs.*	*Field Operation Time in sec.*	*Route length in meters*	*No. of Turns.*	*Energy Consumption in KWh.*
case 1	13.32	14612.4	10637.003	134	13792.7026
case 2	20.762	15170.4	11028.086	141	14348.6364
case 3	35.71	15094.8	11079.932	127	14070.7815
case 4	93.918	18694.8	13738.529	155	17388.7787
case 5	58.307	16160.4	12010.546	117	14765.8961

Additionally, the coverage rate plot over the field time provides a visual depiction of the algorithm's performance across all datasets. This plot illustrates how efficiently the algorithm achieves coverage over time, facilitating a comparative analysis of its efficiency under varying scenarios. Please refer to (Figure 8) for visualization of the coverage plots.

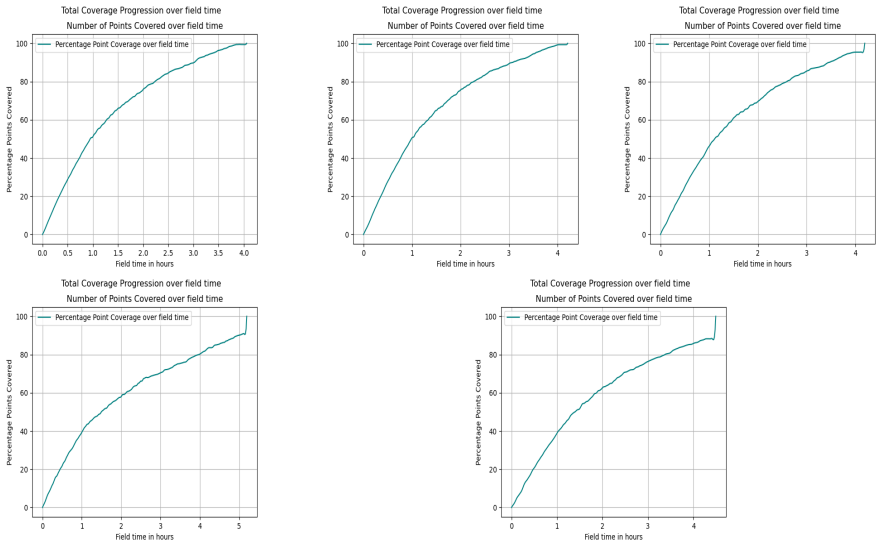

Figure 8: Coverage plots.

The remaining points after the completion of the second behavior in presence of obstacles are depicted in the (Figure 9).

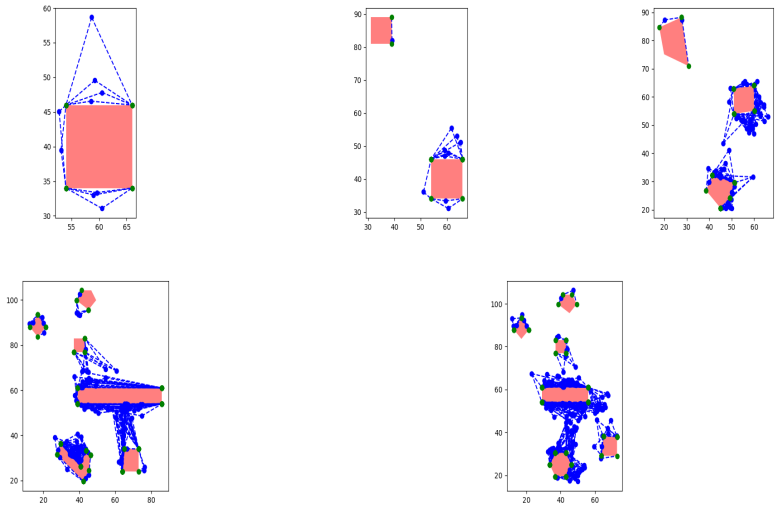

Figure 9: Remaining points after second behavior.

The path obtained for the remaining points utiizing the third behavior can be visualized in the (Figure 10).

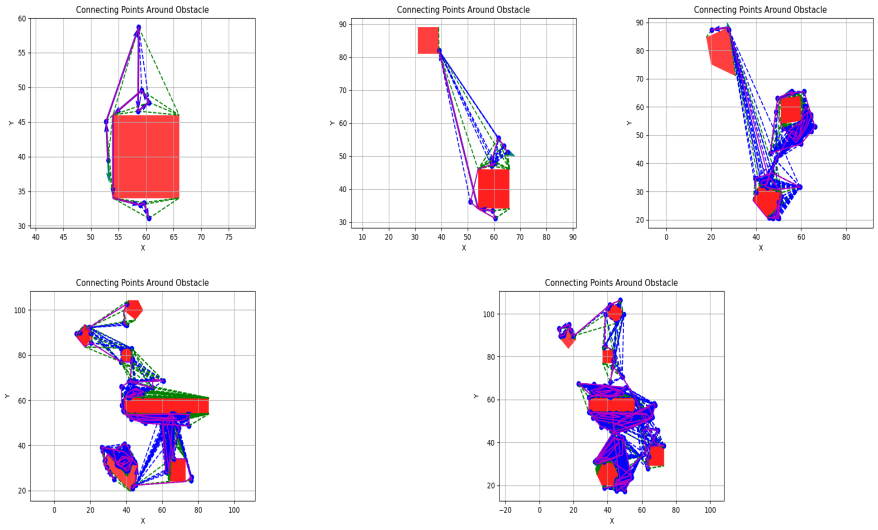

Figure 10: Path for remaining points.

Analysis:
Need to write it properly after putting all the results.

7..3 Real robot testing and analysis

This section details the implementation of the proposed Complete Coverage Path Planning (CCP) algorithm on a real robot platform, addressing both scenarios with and without obstacles. The subsequent analysis of the experimental results is provided. The section is organized into two main parts: Experimental Setup and Results & Analysis.

7..4 Experimental Setup

This subsection provides a comprehensive description of the real robot setup, emphasizing the experimental framework within which the tests are conducted.

The real-world experiments were conducted using a specially designed robot platform equipped with a camera for weed detection and extraction, and a GPS module for localization. This robot is specifically engineered to operate in grass fields. The robot can be visualized in (Figure 11).

Figure 11: Real robot.

To perform the tests on the real robot platform, a small region of the complete field was selected, and only the weed positions within this area were considered. Testing on the entire field was impractical due to its extensive size and the significant time required for complete coverage. Therefore, a smaller region was chosen for practical testing purposes and is demonstrated in the (Figure 12). Additionally, complete coverage was not pursued due to time constraints. Instead, the robot was programmed to cover 42% of the points, with the expectation that the robot would behave similarly even with full coverage. This assumption is based on the robot's ability to mimic its behavior in the simulation accurately. The selected region for experimentation and the specific points utilized for the tests are demonstrated in the (Figure 12).

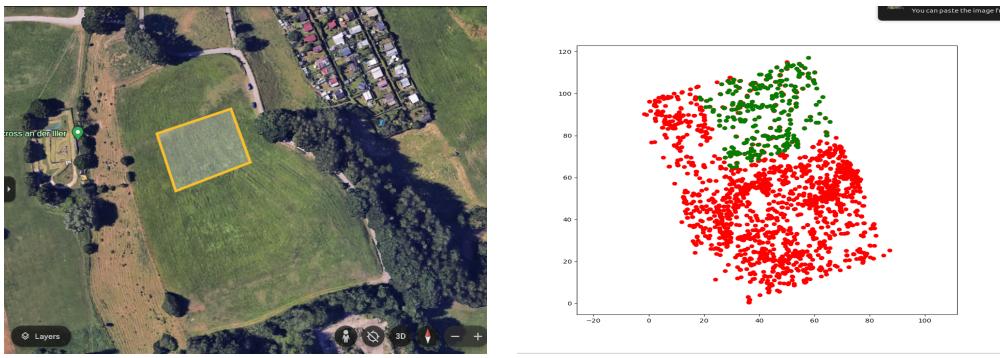

Figure 12: Selected field region.

Apart from these adjustments, all the robot constraints and parameters were kept consistent with those discussed in the simulation setup. Two experimental cases were considered: one without obstacles and the other with obstacles. For the scenario involving obstacles, two polygonal obstacles (one concave and one convex) were introduced. The figures below depict both experimental scenarios.

The objective of demonstrating the CCP algorithm on a real robot platform is twofold: to showcase the alignment between the simulated global trajectory and the actual trajectory followed by the robot, and to evaluate the algorithm's effectiveness in weed removal. Specifically, the analysis focuses on how many weeds the robot successfully removes compared to the number planned by the global planner. Since the robot relies on its local planner, any deviation from the global planner's trajectory could result in missed weeds or the coverage of additional, unintended weed points.

This approach aims to highlight the practical applicability and robustness of the CCP algorithm in real-world agricultural settings, emphasizing its ability to adapt to various environmental conditions and obstacles while maintaining efficient operational performance.

Real Robot Results and Analysis:

Initially, the real-world experiment was conducted on the first case without obstacles. Given that only 42% coverage was used to demonstrate the results on the real robot, the global trajectory obtained from the simulation is depicted below. After executing the test on the real robot in the field, the actual path followed by the robot using the local planner is provided in the subsequent figure.

Global trajectory

Local trajectory

The number of weeds planned by the local planner to be extracted was X, and the number of weeds actually removed by the robot was Y. The alignment between the global and local

trajectories can be visualized in the figure below. To quantify this alignment, the absolute trajectory error (ATE) and relative pose error (RPE) were calculated, metrics commonly used in visual odometry literature. The ATE was x, and the RPE was y. Additional results, if available, can be visualized in the figure below.

Both trajectories on same plot

Similarly, the same procedure was performed for the second case with obstacles. The global trajectory from the simulation and the local trajectory from the real robot are shown below. In this scenario, the number of weeds planned for extraction by the local planner was X, and the number of weeds actually removed by the robot was Y. The alignment between the two trajectories is illustrated in the figure below. The absolute trajectory error was x, and the relative pose error was y. Any additional results obtained are also visualized in the figure below.

Global trajectory

Local trajectory

Both trajectories on same plot

Analysis:

From the results, it can be summarized that the local planner (real robot) closely follows the global planner, demonstrating the accuracy of the global planner in modeling the robot's constraints in a manner that aligns well with the local planner. A key focus is the comparison between the number of weeds planned for extraction and the number of weeds actually removed. Ideally, these numbers should match as closely as possible, depending on the alignment between the local and global trajectories.

If the robot accurately follows the global trajectory, it will cover and extract all the planned weeds. Conversely, if there is a deviation, the robot may fail to extract some of the planned weeds and instead remove others that were not targeted. These additional weeds would have been covered in subsequent passes, and the missed weeds would remain uncovered, resulting in incomplete coverage.

Thus, the alignment between the global and local trajectories is crucial and can be assessed using the absolute trajectory error and relative pose error. These trajectory error metrics indirectly indicate the efficiency of the algorithm and the alignment between the global and local planners. Therefore, complete coverage is more likely if the trajectory deviation is minimal, whereas greater deviation compromises coverage completeness, leaving some weed points untreated.

By ensuring minimal deviation, the proposed CCP algorithm can achieve efficient and effective coverage, thus validating its applicability in real-world agricultural scenarios. The detailed analysis underscores the importance of trajectory alignment in achieving the desired operational outcomes.

7..5 Comparison with other approaches.

In this section, we will conduct a comparative analysis of the proposed complete coverage path planning (CCP) algorithm against two alternative approaches: the lawnmower approach, which serves as the baseline for this thesis, and the graph search approach. The comparison will be conducted under the same experimental conditions described in the simulation section, utilizing identical datasets and performance metrics to ensure consistency and fairness in evaluation.

Simulation Setup As with the initial simulations, the setup is designed to emulate the operational environment of an agricultural field. The same datasets used in the evaluation of the proposed CCP algorithm will be employed here to facilitate a direct comparison. These datasets are characterized by different point distributions and densities, ensuring a comprehensive assessment of each approach's performance.

The key performance metrics for comparison include:
Computational Time Field Operation Time Total Path Length Number of Turns Energy Consumption Coverage Efficiency
These metrics will provide a detailed view of each algorithm's efficiency, effectiveness, and overall robustness.

Lawnmower Approach The lawnmower approach, often referred to as the boustrophedon path, is a widely used baseline method in coverage path planning. It involves the robot traversing the field in parallel lines, akin to a lawnmower, ensuring that the entire area is covered systematically. This method is straightforward but may not be optimal in terms of path length and energy consumption, particularly in fields with irregular plant distributions or obstacles.

Graph Search Approach The graph search approach utilizes a graph-based representation of the field, where points of interest (weeds and cluster of weeds, in this context) are treated as nodes, and paths between them are edges. Algorithms such as A* or Dijkstra's can be employed to find an optimal path that covers all nodes following the non-holonomic constraints. However, this apprach does not guarantee prioritization of approximate straight paths but guarantee respect non-holonomic constraints.

Performance Metrics The performance metrics for all datasets and approaches will be summarized in tables and visualized in figures below. These metrics will provide a comprehensive view of each algorithm's performance under different scenarios, enabling a detailed comparison of their efficiency and effectiveness. The results will be analyzed to identify the strengths and weaknesses of each approach, facilitating an informed decision on the most suitable method for complete coverage path planning in agricultural fields.

No. of points	Cases	Route Length (meters)			Computation Time (seconds)			Field Time (seconds)			Energy Expenditure (units)		
		Behavioral Approach	Graph search Approach	Baseline (Lawnmover)	Behavioral Approach	Graph search Approach	Baseline (Lawnmover)	Behavioral Approach	Graph search Approach	Baseline (Lawnmover)	Behavioral Approach	Graph search Approach	Baseline (Lawnmover)
250	(case 1)	4652.945	2288.31	9896.291776	0.858553886	4.435761562	0.601133347	6282.400938	1741.022767	13910.92722	5783.34459	3632.750509	13593.64178
	(case 2)	3769.005	2083.73	8253.784905	0.775789261	4.341929366	0.587094307	5125.168209	1576.077269	11632.10613	4734.555207	3447.362011	11409.4849
	(case 3)	3985.978	2092.1	7564.831982	0.788151503	4.599930199	0.590665579	5413.822753	1572.229368	10819.97748	5022.177564	3613.97386	10838.28198
	(case 4)	4620.695	2061.77	8564.113438	0.825942278	4.318975639	0.613069534	6216.01818	1546.992294	12186.8293	5656.894864	3598.218862	12120.16344
500	(case 1)	7027.363	3710.8	17071.30153	1.842700005	8.956043648	0.758333683	9520.140257	2806.494905	23468.43942	8793.612844	6143.002153	22181.65153
	(case 2)	6788.619	3248.08	11616.47502	1.696401358	8.61580853	0.695278645	9151.211911	2442.859621	16679.34377	8366.468569	5582.217349	16797.47502
	(case 3)	6336.544	3304.11	12034.99812	1.940592766	8.740781215	0.669975042	8589.717644	2469.994472	16947.37264	7914.394434	5903.641572	16603.69812
	(case 4)	7250.2	3215.45	11852.70452	1.609200478	9.350487668	0.683098078	9761.624722	2396.719655	16945.19315	8898.700417	5850.162864	16963.05452
2000	(case 1)	16189.858	655.67	28265.5625	9.04812026	40.44517501	1.046939611	21751.64778	6456.38521	37873.39062	19816.55759	15678.01424	34365.0125
	(case 2)	16650.701	7850.07	25402.09418	8.591621161	36.87743123	1.091753483	22226.37676	5819.37001	34519.74273	20041.90077	14760.4663	32043.19418
	(case 3)	13728.721	7257.3	21144.39368	7.65038085	34.6019571	0.995180845	18446.73883	5373.163304	29256.4921	16813.77138	13747.4777	27926.79368
	(case 4)	15547.469	7828.97	23561.53673	8.155776024	36.02791135	1.016644001	20839.67217	5806.381665	32091.48341	18820.91894	14680.92948	29896.48673
4000	(case 1)	24191.259	13033.92	31436.1282	24.18394971	80.41146149	1.38740325	32301.69933	9654.118538	41846.41025	29136.75914	24629.20129	37559.1282
	(case 2)	22356.011	11583.92	28932.5249	20.85941482	71.84240132	1.366778374	29871.08874	8577.056512	39050.53112	27018.91147	21935.23424	35856.2249
	(case 3)	20620.054	10643.04	26886.435	18.23963118	66.78159156	1.275558949	27659.4681	7849.986297	36257.41874	25141.65448	20609.80285	33244.935
	(case 4)	22222.31	11076.75	28340.7056	19.22349644	68.28330584	1.293629885	29701.58872	8162.427239	38310.757	26932.31049	21561.45279	35264.4056
6000	(case 1)	29777.382	16816.83	33040.58685	45.23673987	286.3729786	1.66720438	39851.80224	12440.13841	43851.98356	35947.48179	32016.74242	39163.58685
	(case 2)	26039.854	14100.83	30366.86311	33.23154497	246.6400248	1.570202589	34821.16749	10412.05816	40500.01639	31597.65399	27130.02134	36466.31311
	(case 3)	26223.169	13599	31659.26981	30.80121374	237.7119713	1.675028086	35035.83592	9984.599781	42135.14976	31639.66906	27018.06546	37805.81981
	(case 4)	26570.832	14609.54	32148.11261	32.34737062	244.7778758	1.63052392	35496.36509	10738.4984	43227.01576	31893.13207	28846.61331	39448.61261
10000	(case 1)	38128.675	21623.33	35598.33764	122.6808474	490.2239986	2.21295166	50681.89334	15947.4934	47049.17205	45382.07468	41890.90133	41721.33764
	(case 2)	34118.252	18820.21	36001.3554	74.37674475	411.4919939	2.168357611	45367.87779	13852.97698	48288.88175	40641.60223	36868.19144	43890.6054
	(case 3)	31650.696	17269.61	33119.52466	65.54510021	376.7545004	2.029067278	42291.24514	12686.21149	44245.03083	38197.59611	34211.77633	39949.02466
	(case 4)	32154.031	17527.4	34293.53829	62.76576424	375.6292722	2.238646507	42801.16425	12876.94731	45859.73536	38606.73138	34702.42134	41476.28829

Figure 13: This is an example image.

The lawnmover approach and behavioral approach was performed on the same device. However, the graph search approach was performed on a different device with the following specifications: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz

8 Cores

32GB RAM

To visualize and compare properly the following figures will be presented:

Put all the figures and then explain those figures in the text.

Analysis of the comparison results