Optimization of UAV Path Planning Algorithms for Efficient Access and Backhaul Coverage

Capstone Report

Anuar Tussupbayev

Nazarbayev University
Department of Electrical and Computer Engineering
School of Engineering and Digital Sciences

This project report was created on TexStudio editing platform using LaTeX. All the figures were drawn using draw.io online software tool.

NAZARBAYEV UNIVERSITY

Electrical and Computer Engineering
Nazarbayev University
http://www.nu.edu.kz

Title:
Optimization of UAV Path Planning Algorithms for Efficient Access and Backhaul Coverage

Theme:
Unmanned Aerial Vehicle

Project Period:
Fall 2024

Participant(s):
Anuar Tussupbayev

Supervisor(s):
Behrouz Maham

Copies: 1

Page Numbers: 50

Date of Completion:
April 16, 2025

Abstract:

This research project is aimed on the optimization of Unmanned Aerial Vehicle (UAV) path planning algorithms and backhaul coverage. The study depicts numerous path planning algorithms, including A-Star, Dijkstra, and Rapidly-Exploring Random Tree (RRT). Also, the research analyzes productivity beneath diverse natural conditions. Via simulations in a software, the impact of assorted variables such as wind speed, direction and trajectory arranging on UAV performance is analyzed. The results highlight the advantages and disadvantages of each algorithm tested and propose crossover frameworks to improve coverage, energy efficiency, and adaptability in changing situations. The results recommend that optimized path planning algorithms can significantly progress UAV-based communication frameworks, making them a dependable arrangement for diverse applications in rural and urban regions.

Contents

Preface

The quick progression of UAV technology has changed numerous industries, including broadcast communications, logistics, and disaster management. In regions where conventional framework deployment is unreasonable or cost-prohibitive, UAVs offer a promising alternative for building up temporary communication systems. This capstone project points to address the challenges related with UAV path planning, especially in optimizing energy consumption and improving network coverage. Through a comprehensive investigation of different path planning algorithms, this study provides bits of knowledge into moving forward UAV efficiency in both access and backhaul scenarios. I extend my gratitude to my supervisor, Behrouz Maham, for his important guidance, and to the Department of Electrical and Computer Engineering at Nazarbayev University for the assets and support given throughout this research.

Nazarbayev University, April 16, 2025

Anuar Tussupbayev
<Anuar.Tussupbayev@nu.edu.kz>

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs), usually called drones, are used for a lot of purposes, including observation, data collection, and remote communication [1]. UAVs are increasingly being used in different areas, upgrading the way of how tasks such as surveillance, agriculture, telecommunications and disaster management are done. In telecommunications, UAVs play a vital part by increasing scope to challenging regions, where usual infrastructure deployment is not efficient, and sometimes is impossible [2]. Efficient flight planning for UAVs is important for optimizing energy consumption, improving network performance, and enhancing overall operational efficiency. By implementing modern algorithms and real-time data analysis, UAVs can significantly increase the scope of different networks, making them providing better connection in both urban and rural areas. Nowadays, multi-UAV systems are widely used in public, military and civil applications [3]. These systems usually consist of small and cost-effective drones those operate in a coordination with each other without human operation. Main design problems in such systems is communication, which plays almost the main role in effective coordination and connection among the UAVs [1]. Effective communication is significant for their successful working, particularly in changing environments where conditions can change rapidly. This involves the usage of many strong communication protocols and adaptive systems in their design, which is capable of handling unpredictable factors such as weather conditions, obstacles, wind changes, rains and terrain variations such as Flight Control Software, Ground Control Software, Sensor Integration Software [4].

As the increase in use of smart devices and Internet of Things (IoT) applications continues to grow, the need for fast and reliable connection between devices and areas becomes even more demanded. However, in remote or disaster-affected regions, establishing and maintaining traditional ground-based systems can be cost-ineffective, logistically challenging, and time-consuming [5]. UAVs offer a cost-effective and fast solution for setting up temporary or on-demand wireless

communication networks, bridging the gap where traditional infrastructure cannot operate properly [5]. To achieve better data transfer in these situation, advanced algorithms are necessary to be capable of navigating near obstacles, adapting to fluctuating network conditions, and addressing most of the communication requirements those are being actual nowadays. The integration of ML (machine learning) and AI based algorithms can significantly improve the decision-making capabilities of UAVs, allowing them to adapt dynamically to all the needs, ensuring continuous service even in highly variable environments [6].

The research focuses on optimizing UAV path planning and operational strategies to enhance wireless communication opportunities. We propose specialized methods and software to create dynamic flight plans those can maximize area coverage, minimize energy consumption, navigate obstacles efficiently, and allocate communication resources effectively while maintaining complicated network connections [7, 8, 9, 10, 11, 12]. By the simulations we understand the impact of the both external and internal factors on the UAV path planning optimization and backhaul coverage, such as wind, gravity, speed of flight, energy consumption, etc. [13]. These methods not only improve energy efficiency but also give us the way for more scalable and resilient communication networks those can be deployed very fast. By addressing key challenges and identifying emerging opportunities, our goal is to significantly improve both access and backhaul coverage depending on complicated variables and environments. Moreover, we examine the impact of environmental factors and UAV hardware/software limitations on network performance [14], proposing solutions that can reduce these constraints via optimized movement paths and energy-efficient protocols.

In addition to their roles in telecommunication systems, we explore other applications of UAVs in fields such as disaster response [15], rural connection, and backhaul coverage [7]. These UAVs can be used to assist in rescue missions or provide internet connection in areas with little or no infrastructure and low economics, demonstrating their universality in different spheres. Specifically designed algorithms for UAV path-planning can significantly improve network performance, even in unpredictable environments. These algorithms can be adjusted to manage specific tasks, ranging from search operations to providing high-speed internet in underserved areas. Since research aims not only to optimize path planning but also backhaul coverage, we review Backhaul-and-coverage-aware Drone Deployment (BoaRD) [16] for Multi-UAV systems and similar possibilities and operations. These strategies are particularly useful in environments where ground-based communication towers are not usable, ensuring that UAVs can provide both primary and backup communication channels efficiently.

To validate our approach, we conduct a series of real-life experiments and computer simulations. We test and compare different path planning algorithms such as Dijkstra's Algorithm (A well-established method for solving the shortest path

problem is explored, focusing on the creation of a waypoint network for regional navigation and the calculation of flight costs between waypoints.[17]), Heuristic Algorithm, Path Smoothing, A-star Algorithm (The A* algorithm is a heuristic-based approach. It calculates the total cost by adding the heuristic cost to the path cost, and prioritizes the path with the lowest total cost.[18]), Dynamic Window approach (The Dynamic Window Approach (DWA) mainly conducts discrete sampling within the permissible velocity space, considering the drone's current motion state. It then simulates the motion trajectories of these velocity combinations over the forward prediction time.[19]), Q-Network (Q-network is a neural network used in reinforcement learning to approximate the Q-value function, which estimates the expected reward of taking a given action in a given state.[20]), etc. These tests demonstrate the practical quality of our methods and also offer valuable insights into the optimization of UAVs path planning and backhaul coverage for efficient wireless communication. Our testing environments should include different terrains and weather conditions to ensure that our solutions are working and adaptable to the real-world. The results from these tests will help adjust our strategies and inform future deployments, making UAV-based communication systems a reliable and scalable solution for various spheres. Furthermore, we aim to develop special guidelines and systems for the better adoption of UAV technology in telecommunication networks, ensuring that these systems can be effectively integrated into existing infrastructures across different sectors.

1.1 Ethical and Professional Responsibilities

- **Environmental Impact:** The environmental impact of UAVs used in path planning optimization and backhaul coverage must be carefully evaluated to be sure that the benefits exceed the potential harm. On one hand, UAVs offer a more maintainable alternative to conventional infrastructure inspection strategies, such as manned helicopters or vehicles, which are energy-intensive and have high level of carbon emissions. UAVs ordinarily consume less energy and can cover bigger regions more effectively, contributing to diminished carbon footprints within the telecommunications segment due to their smaller size and bigger coverage. Moreover, UAVs can access remote areas without the need for constructing roads or other access routes, further reducing the environmental disruption caused by traditional methods.

  On the other hand, the expansion of UAVs' usafe moreover presents challenges, especially in terms of energy consumption related to charging batteries and potential electronic waste. The frequent need to replace batteries and other components can add to the growing problem of electronic waste, which poses its own environmental risks. To relieve these issues, the project must prioritize the use of energy-efficient UAV designs and investigate renewable energy sources, such as solar panels, to make environmental impacts less. These approaches could extend UAV flight times and minimize the reliance on non-renewable energy sources those can bring harm to the environment. Also, appropriate transfer or reusing strategies for UAV components need to be coordinated into the lifecycle of the technology to advance support. Implementing of the circular economy principles, such as designing UAVs for disassembly and reuse, could further enhance their environmental performance.

- **Informed Judgments:** To guarantee that decisions made throughout the UAV path planning and backhaul coverage project are well-informed, a multidisciplinary approach should be taken into consideration. This incorporates including not as it were specialized experts in UAV design and broadcast communications but moreover ethicists, legitimate advisors, and social researchers to assess the societal affect of the technology. In particular, ethicists can provide insights on potential ethical dilemmas, while legal advisors ensure that UAV operations comply with the latest legal and regulatory standards in each jurisdiction. All of them may detect problems or benefits in different spheres due to their expertise. Each decision, especially those involving algorithm improvement for path optimization, should consider not as it were the specialized proficiency but moreover the safety and potential consequences for the communities influenced by UAV operations.

Informed judgments also require straightforwardness within the decision-making process, where each step and basis is recorded for responsibility. Documenting the rationale behind decisions ensures accountability and helps to build trust with stakeholders. Normal consultations with partners, such as policymakers, local specialists, and influenced communities, guarantee that the venture remains adjusted with societal needs and expectations. These consultations can help to identify and address concerns early, preventing opposition or unintended consequences. And also provide more expertise to the decision. The group must stay updated on the most recent regulations and societal patterns to alter methodologies together. By adjusting specialized performance with ethical and societal contemplation, decisions will be grounded in a better understanding of their suggestions. Furthermore, conducting risk assessments and scenario planning can further enhance the efficiency of decisions made.

- **Global Context:** The project of UAV path planning optimization and backhaul coverage fits into a worldwide context because UAVs are progressively being used in different areas such as logistics, agriculture, and telecommunications around the world. The global growth in UAV technology is driven by its potential to address challenges in different sectors, including delivery services, environmental monitoring, and disaster response. In a few regions, especially rural or underserved areas, UAVs can be a useful solution for improving communication systems, upgrading access to data, and fostering financial development. In any case, the suggestions of implementing UAV-based systems may vary over different worldwide contexts. For example, in developed countries, where administrative systems are more develop, UAV operations may be less demanding to be used due to already made systems. On the other hand, in developing countries, where such systems are being new, the introduction of UAVs could show lawful and safety challenges. These challenges may include a lack of regulatory frameworks, insufficient training, and public skepticism toward the technology due to cultural ideas.

  Moreover, social states of mind towards privacy, technology, and mechanization shift essentially over diverse parts of the world. What may be satisfactory in one locale may confront resistance in another due to concerns over surveillance or job displacement. In some cultures, UAVs might be viewed as intrusive, leading to public opposition. In this manner, any global deployment of this project would require fitting the approach to adjust with local controls, social standards, and societal needs. Additionally, by considering worldwide inequalities, the project seem offer assistance bridge digital isolates, giving access to network where usual infrastructure is missing and can't be placed. However, special attention must be paid to local capacities for maintaining

and operating UAV systems, which could impact the long-term sustainability of the project in particular regions.

- **Societal Impact:** The societal impact of UAV path planning optimization and backhaul coverage is significant, particularly in terms of improving connectivity and fostering computerized inclusion between different cities, villages, countries, etc.. By providing superior coverage in rural or underserved areas, UAV technology can bridge the advanced partition, giving more individuals access to the internet and empowering financial, instructive, and social opportunities. Improved access to digital tools and information can give new opportunities for entrepreneurship, improve e-government services, and support healthcare initiatives, particularly in remote areas such as villages or steppes. For example, improved internet connection in rural areas can support distant learning, e-health administrations, and encourage local businesses' development not only in their areas but for larger scope.

  On the other hand, the introduction of UAVs raises concerns with respect to surveillance and privacy. The ability of UAVs to capture video, photos, and data from public and private spaces poses significant ethical and legal questions. To work with it, strict rules must be set up to regulate the use of UAVs in public spaces. Regular audits of the UAV usage and clear public communication about the purpose and scope of UAV operations can help decrease these concerns. Also, whereas UAVs may improve access to services, their broad use might worsen issues of inequality if as it were certain regions or socioeconomics advantage. Hence, it is essential to guarantee that UAV arrangement strategies are comprehensive and outlined to advantage all communities, regardless of financial status. Specific attention should be paid to ensuring equal access across urban and rural areas to prevent deepening the digital divide.

- **Ethical Responsibility:** The improvement and usage of UAV path planning optimization and backhaul coverage show some ethical problems regarding them. One essential concern is people's privacy, as UAVs equipped with sensors and sometimes cameras those might accidentally capture delicate information, abusing individuals' rights to privacy. To solve this, engineers must guarantee strong information security measures, such as encoding and comply with lawful systems such as constitution, court policies or local information security laws. Transparency regarding data collection, storage, and usage will also be needed in ensuring public's trust. Another ethical issue is the potential abuse of UAV technology, which may be used for unauthorized applications or other destructive activities such as information robbery. Tending to this hazard mindfully involves making exacting utilize approaches, counting the application of innovation exclusively for legal and advantageous

purposes, such as moving forward network connectivity.

There's also the ethical thought of job displacement. As UAV technology progresses, manual tasks in regions like infrastructure inspection and delivery may be replaced, affecting employment in those spheres. Workers displaced by UAV automation may face challenges in transitioning to new jobs as they may not have necessary skills and only ones they used on their previous work. To reduce this, endeavors must be made to upskill the workforce and make unused job opportunities within the UAV and broadcast communications areas. This should include creating special programs or new industries related to the UAV operations, maintenance, and software development which can positively affect on people's salary and opportunities. The capable advancement of UAV systems must consider security first, guaranteeing that the technology is outlined with redundant fail-safes to minimize dangers to both administrators and the general public/users. Ethical responsibility also extends to provide long-term monitoring and oversight to prevent misuse of the technology by intruders.

- **Economic Impact:** The economic impact of optimizing UAV path planning and backhaul coverage can be critical, both within the short and long term conditions. Within the short term, the deployment of UAVs for network optimization may decrease operational costs and expenses for their maintenance by improving the proficiency of network infrastructure inspections, decreasing downtime, and minimizing the require for human intervention or resources. This may lead to cost investment funds for telecommunication companies, which might at that point be passed on to consumers through more reasonable data services. Next, the increased money efficiency provided by UAVs may help companies expand services to areas that were previously cost-prohibitive.

  Within the long term, the project may stimulate financial development by progressing network in rural and underserved regions, in this way empowering modern business opportunities and improving access to e-commerce, education, and telemedicine, so the potential market and supply will rise rapidly. However, there are some potential economic challenges to consider. Initial investments in UAV technology and infrastructure development and introduction may be very high, and it may require many regulatory changes and workers retraining. Moreover, the maintenance and repair of UAVs will require further regular expenses, which might challenge smaller telecommunication companies due to their low opportunities. Furthermore, some employments can be displaced due to automation. On the other hand, unused business opportunities may emerge in UAV operation, maintenance, and information analysis, adjusting out these potential problems and challenges.

In the long term perspective, the growth of the spheres related to the UAV operations could create a positive effect on the economy of each country.

Chapter 2

Methodology

This research aims to optimize UAV path planning algorithms to enhance access and backhaul coverage. The methodology is divided into three main stages: algorithm selection and design, simulation and testing, and data analysis.

2.1 Algorithm Selection and Design

Various path-planning algorithms were identified and selected for implementation and testing. There are mainly two types of algorithms: global and local. They have different aims and principles. Global algorithms generate paths considering a pre-known environment and usually create a complete trajectory before the UAV starts its mission. Common global algorithms include:

- **A-Star (A*) Algorithm:** A graph-based search algorithm that calculates the optimal path by minimizing the total cost, which is the sum of the distance from the start to a point and the estimated cost from the point to the goal. It is computationally expensive but provides an optimal solution.
- **Dijkstra's Algorithm:** Similar to A*, but it doesn't use a heuristic to estimate the remaining cost, making it slower but still optimal.
- **Rapidly-exploring Random Tree (RRT):** A sampling-based algorithm that efficiently searches non-convex spaces. RRT grows a tree towards the target by randomly sampling points, making it effective for large or complex spaces, but the paths may not be optimal.
- **Probabilistic Roadmaps (PRM):** This method first constructs a graph of possible paths through random sampling, then finds the shortest path using graph search techniques. It's useful for multi-query scenarios.

Local algorithms focus on real-time adjustments in dynamic environments, reacting to changing conditions or newly detected obstacles. Common local algorithms include:

- **Dynamic Window Approach (DWA):** A reactive collision-avoidance technique that takes into account the UAV's velocity and accelerations, selecting the optimal motion based on dynamic constraints.
- **Artificial Potential Fields (APF):** This method treats the UAV as a particle influenced by "forces" from the goal (attractive forces) and obstacles (repulsive forces). It is simple and fast but may suffer from local minima, trapping the UAV.
- **Velocity Obstacle (VO):** This technique calculates the space of velocities that could lead to collisions with obstacles and selects a safe velocity for navigation.

All of listed algorithms already include SARSA, Q-learning and reinforcement learning, so basically they are already improved versions of the ones which were used in UAVs before. It is crucial to consider combinations of these algorithms to save resources, such as time and money.

There are several algorithms for used for more complex environments or optimization problems where aim is to reduce fuel consumption, battery usage, or time. They are:

- **Genetic Algorithms (GA):** These simulate the process of natural selection to evolve a population of candidate paths toward an optimal solution. GAs are good for solving highly nonlinear, complex path-planning problems but may take more time to converge.
- **Particle Swarm Optimization (PSO):** Inspired by the social behaviour of birds, this algorithm optimizes a solution by having particles (candidate paths) move through the search space, adjusting based on individual and collective experiences.

Considering these algorithms it is possible to improve global and local algorithms in complicated environments, so UAVs can avoid obstacles with less energy consumption and still have high space coverage.

The integration of multiple path planning algorithms is beneficial when we need to account for both static and dynamic environments, as well as real-time updates. A common approach is a hybrid system that uses:

- **Global-Local Planner Integration:** In many UAV missions, global path planning is used to provide an initial path from the start to the goal, assuming static or slowly changing environments, while local path planning adapts the trajectory to handle dynamic obstacles or real-time events. For instance, A* can be used for global path planning to define an initial trajectory, while a local planner like DWA or APF modifies this path in real time to avoid unexpected obstacles like birds or drones.
- **Optimization-based Hybrid Systems:** Optimization algorithms such as GA or PSO can be used to refine the solution provided by other algorithms. For example, an initial path generated by RRT might be improved using PSO to reduce energy consumption or minimize travel time. A combination of PSO and APF may provide fast and energy-efficient path reconfiguration in environments where the obstacles are dynamic while ensuring real-time response.
- **Hierarchical Path Planning:** Some hybrid systems divide path planning into hierarchical levels, where high-level global planning algorithms (like PRM or RRT) handle long-distance navigation, and low-level reactive planners (like DWA or APF) handle close-range obstacle avoidance. This allows the UAV to benefit from long-term efficiency while being able to respond to real-time threats.

Each algorithm is needed to be tailored to incorporate factors affecting UAV performance, such as energy consumption, wind resistance, and communication constraints.

Since the telecommunication sphere application is mostly considered in this research of UAVs, it is evident that the system should be multi-UAV, so in missions involving multiple UAVs, path planning algorithms like RRT can handle path generation for each UAV while PSO or GA optimizes the flight paths to prevent collisions and reduce energy consumption. The addition of APF or DWA ensures real-time collision avoidance between UAVs in dynamic environments. And for real-time adaptation hybrid systems are needed. Hybrid systems combining global and local planners are particularly useful in real-world applications where UAVs encounter unexpected obstacles. For instance, global planning (like A*) sets the course, while DWA adjusts in real time for any dynamic object.

Table 2.1 illustrates summary of the listed algorithms.

Table 2.1: Summary of Path-Planning Algorithms for UAVs

Algorithm Type	Algorithm	Description and Features
Global Algorithms	**A-Star (A*)**	Graph-based search minimizing total cost (distance + heuristic). Provides optimal solution but computationally expensive.
	Dijkstra's Algorithm	Similar to A*, but without heuristic. Guarantees optimal solution but slower.
	Rapidly-exploring Random Tree (RRT)	Sampling-based, efficient for large/complex spaces. Paths may not be optimal due to randomness.
	Probabilistic Roadmaps (PRM)	Constructs a graph through random sampling, solving multi-query problems efficiently.
Local Algorithms	**Dynamic Window Approach (DWA)**	Reactive collision-avoidance considering velocity and acceleration constraints.
	Artificial Potential Fields (APF)	Treats UAV as a particle influenced by attractive/repulsive forces. Simple but prone to local minima.
	Velocity Obstacle (VO)	Calculates safe velocities avoiding collisions with obstacles.
Optimization Algorithms	**Genetic Algorithms (GA)**	Simulates natural selection to evolve optimal paths. Effective for complex problems but slower convergence.
	Particle Swarm Optimization (PSO)	Inspired by social behavior of birds, optimizes paths through collective particle adjustments.
Hybrid Approaches	**Global-Local Planner Integration**	Combines global planning for initial trajectory and local planning for real-time adjustments.
	Optimization-based Hybrid Systems	Uses GA/PSO to refine global solutions (e.g., RRT), enhancing energy efficiency and response speed.
	Hierarchical Path Planning	High-level global planners (e.g., PRM/RRT) for long-distance navigation combined with local planners (e.g., DWA/APF) for obstacle avoidance.

2.2 Simulation Setup

The Matlab environment with Simulink, UAV Toolbox, and Aerospace Toolbox is used to simulate UAV operations under various conditions. The "Tune Waypoint Follower for Fixed-Wing UAV" model served as the base for conducting these simulations.

Three scenarios should be considered and designed:

- **Ideal Conditions:** No wind, constant speed, and fixed waypoints.
- **Moderate Wind:** Simulations with northward, eastward, and downward winds to assess performance under varying conditions.
- **Severe Conditions:** Increased wind speeds and higher UAV velocities to evaluate limits and robustness of algorithms.

2.3 Experimental Process

To research the path planning algorithms in current conditions with low budget, it is needed to assess the simulation results which include the implementation of algorithms mentioned above and their hybrid systems to determine the highly efficient combination. In advance, it is needed to assess the behavior of pre-programmed UAV which should cover several path points.

- **Step 1:** Analyze the behavior of a UAV under ideal conditions with pre-programmed way.
- **Step 2:** Analyze the behavior of a UAV under moderate conditions with pre-programmed way.
- **Step 3:** Analyze the behavior of a UAV under severe conditions with pre-programmed way.
- **Step 4:** Implement several algorithms separately. For example, DWA,A-Star, RRT.
- **Step 5:** Try on several combinations of algorithms to implement hybrid systems.

**For all steps above it is needed to run the simulations, and monitor and record metrics, including trajectory accuracy, energy consumption, and area coverage.*

2.4 Data Analysis

The collected data is analyzed to determine next several important aspects:

- Algorithm Efficiency: Comparing energy consumption and coverage. This needed to determine the most efficient ones.
- Impact of Environmental Factors: Assessing how wind and speed of movements affect UAV performance.
- Optimization Potential: Identifying areas where algorithmic improvements could yield better results for the future works.

The results are cross-validated with multiple runs under identical conditions to ensure reliability. Each algorithm's performance was benchmarked against its expected theoretical outcomes to highlight deviations and their possible causes.

2.5 Path Optimization

To contrast UAV path planning, three algorithms were employed and contrasted:

- **A + Dijkstra Combination*:** A union of the heuristic search of the A* algorithm and Dijkstra's best pathfinding, with a balance between reliability and path efficiency.
- **RRT + PRM (Rapidly-exploring Random Tree + Probabilistic Roadmap):** A probabilistic algorithm that constructs a roadmap by randomly exploring points, prioritizing computational speed.
- **APF (Artificial Potential Field):** A reactive method using attractive force toward the goal and repulsive force away from obstacles with the aim of finding shorter paths but perhaps less reliable.

These algorithms were implemented in a Python class UAVPathPlanning, mimicking a 2D grid world (20x20) with origin at (0,0), target at (19,19), and randomly distributed obstacles. The world was mimicked with 100 obstacles to mimic navigation in dense environments. Performance of each algorithm was evaluated based on:

- **Reliability:** Success rate in reaching the goal.
- **Path Efficiency:** Visited points and path length.
- **Computational Speed:** Average running time per simulation.

Simulations were executed with Python libraries such as numpy, heapq, random, and matplotlib for processing data, priority queue management, randomization, and 3D path visualization, respectively. Scripts had steps for path reconstruction and visualization to enable in-depth analysis of every algorithm's performance.

2.6 Area Coverage Optimization

To provide coverage in a region, five UAVs simulated a 3D world (100x100x50) that was to cover a region and steer clear of 50 randomly placed obstacles. Two variants were tested:

- **Basic Coverage Algorithm (Without A + Dijkstra)*:** The UAVs flew without advanced path planning, and they traveled to randomly generated targets of coverage within their perception radius (radius of 5 units). This option favored simplicity but lacked coordinated planning.
- **Focused Coverage with A* + Dijkstra and Voronoi Segmentation*:** Each UAV was assigned an individual area via Voronoi division of their original 2D positions. All UAVs employed the A* + Dijkstra algorithm for covering the targets within its designated area with guaranteed path optimization and obstacle evasion.

The simulation scenario was defined with a limited total energy budget of 100 steps for each UAV for the purpose of simulating energy constraint. The UAV class managed all UAVs' position, coverage area, energy, and path, as well as movement methods, coverage addition, and target selection. The Voronoi technique used the scipy.spatial.Voronoi class to partition 2D space, which was generalized to 3D by mapping all z-coordinates of an area in 2D to its corresponding UAV. Coverage targets (30 random points) were generated and ensured to be within bounds and obstacle-free.

Performance metrics of coverage area were:

- **Total Covered Area:** The level to which the environment is covered with UAVs.
- **Number of Overlapping Zones:** Degree of redundancy in coverage.
- **Energy Efficiency:** The ratio of energy consumption compared to computing and mobility needs.
- **Routing Accuracy:** Degree of accuracy in delivery and obstacle identification.

Results were displayed in 3D using matplotlib, plotting obstacles and regions of UAV coverage to visualize spatial distribution and overlap. Debugging prints provided information on final UAV positions and coverage counts.

Chapter 3

Results and Discussions

3.1 Results

The results of the simulations highlight the effectiveness of different path-planning algorithms under varying environmental conditions. This section presents the outcomes of each scenario, focusing on key performance metrics: trajectory accuracy, energy consumption, and coverage efficiency.

Since the primary research is related to the UAV path planning, some Matlab models were found. These model is useful in terms of understanding the principles of the UAV path passing. Also, this model is being an environment to test impact of the internal and external factors on the energy consumption and backhaul coverage.

For purposes of this research the model "Tune Waypoint Follower for Fixed-Wing UAV" was found[21]. This model requires Matlab software with Simulink, UAV Toolbox, and Aerospace Toolbox add-ons. It allows to set several waypoints and make UAV reach them depending on the speed, wind, and gravity conditions.

3.1.1 Scenario 1: Ideal Conditions

In the first simulation, the UAV was tested under windless conditions with a flight speed of 5 m/s across four predefined waypoints. Primarily, it was checked how UAV operates in ideal conditions with the speed of flight of 5 meters per second, windless environment and 4 fixed waypoints. In the **Fig.1** it can be seen that it is a straight trajectory between waypoints and UAV covers the whole area required. Let's assume that simulation's stop time is an energy. In this case energy is spent enough for coverage and has no need to fuel up the UAV.

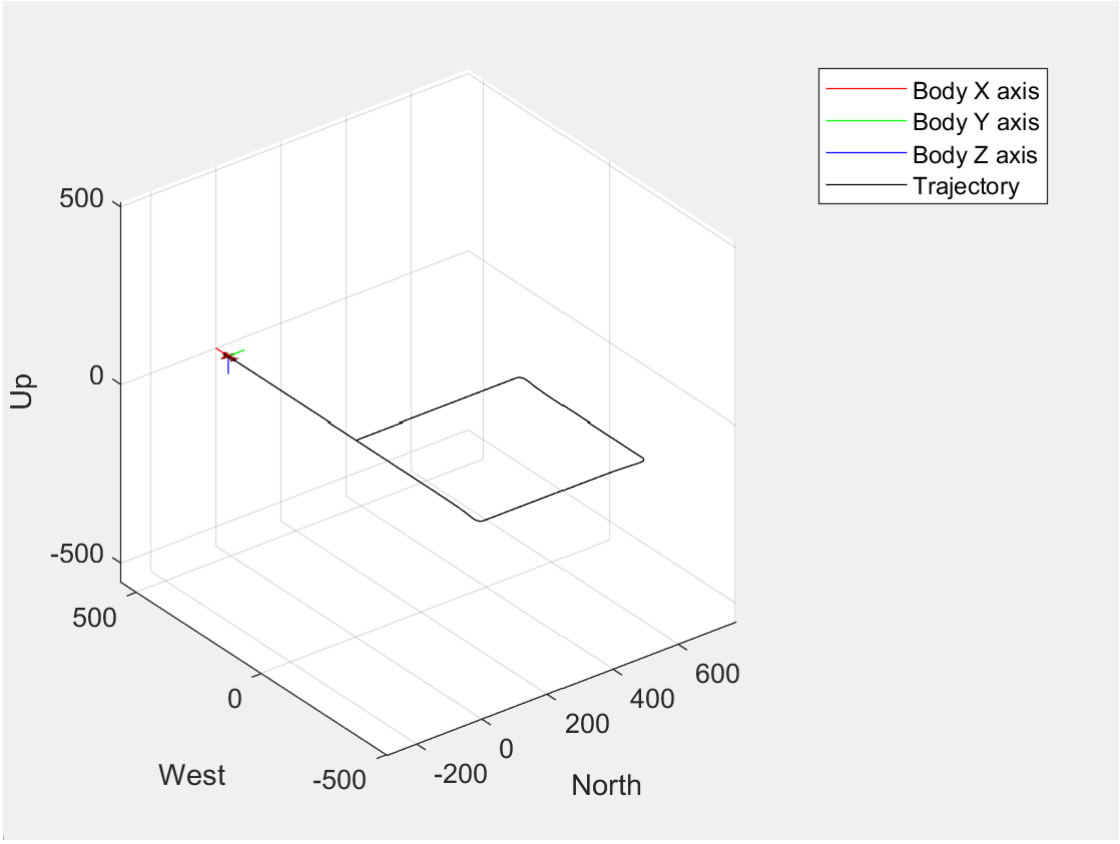

Fig.1. UAV path in windless environment.

- Trajectory: The UAV followed a straight and efficient path, accurately reaching all waypoints.
- Energy Consumption: Minimal energy was required, as reflected in the low simulation stop time.
- Coverage: The UAV successfully covered the entire designated area with no gaps in backhaul coverage.

3.1.2 Scenario 2: Moderate Wind Conditions

Next, the wind conditions were adjusted. The Northern and Eastern winds were added, along with the wind directed downside. The parameters for the wind: Northern wind - 5 m/s, Eastern wind - 4 m/s, downside wind - 1 m/s. Speed of the UAV remaining the same - 5 m/s. In the **Fig.2** the new path can be seen.

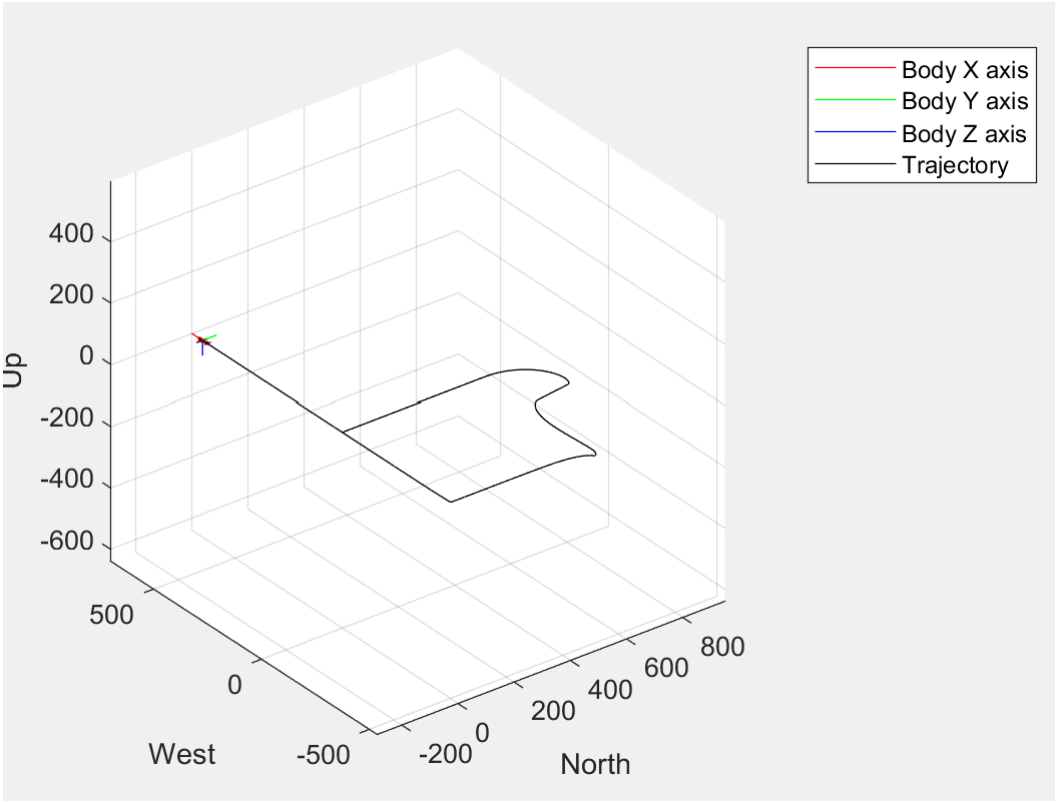

Fig.2. UAV path in small wind.

- Trajectory: The UAV adjusted its path to counteract wind forces. The path, while deviating from a straight line, remained optimal in terms of waypoint navigation.
- Energy Consumption: Energy usage increased slightly due to additional effort required to combat wind resistance.
- Coverage: Although the UAV reached all waypoints, some gaps in backhaul coverage were observed, particularly in areas between waypoints.

As it can be seen from the image, the path was significantly changed and became not straight, but the path is still complete (the UAV reached its final destination). It means that in this conditions the UAV had to oppose the wind conditions to save fuel to get to the final point. However, it could not cover the whole necessary area, making the losses in the backhaul coverage.

3.1.3 Scenario 3: Severe Environmental Conditions

Thirdly, it was tested with higher UAV speed and more severe environment. Speed of UAV was increased up to 30 m/s, Northern Wind - 25 m/s, Eastern wind

- 10 m/s, downside wind - 4 m/s. The resulting pathway can be seen in **Fig.3**.

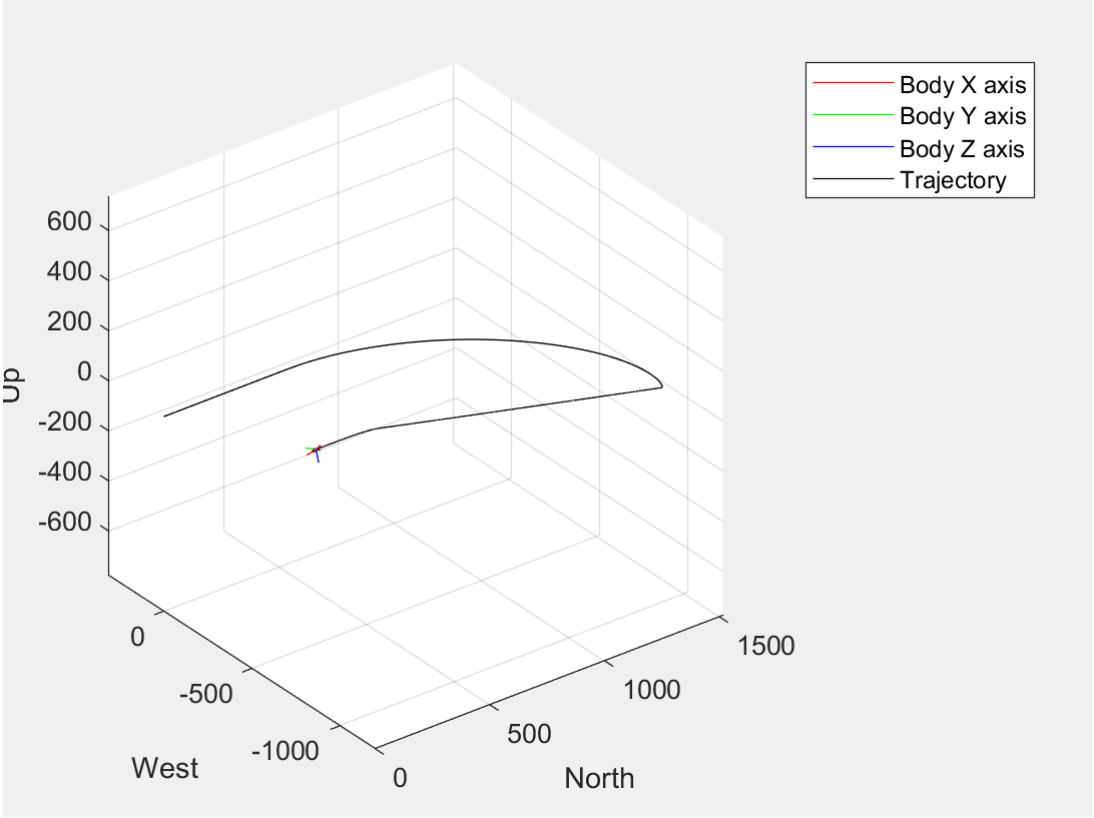

Fig.3. UAV path in severe environment.

- Trajectory: The UAV struggled to maintain an optimal path, frequently deviating due to high wind resistance. It failed to reach the final waypoint.
- Energy Consumption: Energy depletion was significantly higher, resulting in mission failure before completion.
- Coverage: The severe conditions drastically reduced coverage, leaving large areas unserved

Here it can be observed that UAV could not finish it's mission properly at all. Mainly, it did not reach the final point due to loss of all energy it had at the start. Moreover, the path is filthy poor, so it cannot cover most part of the necessary area.

3.1.4 Comparison of Algorithms

The two dimensional MatLab model was programmed to determine the results of different algorithms in the same conditions. For this simulation experiment A-

Star, Dijkstra and Rapidly-Exploring Random Tree algorithms were used. The map was plotted with start and end points with some barriers/obstacles, those should be avoided by the UAV in order to get to the destination. As we can see in **Fig.4a**, all three algorithms plotted different paths.

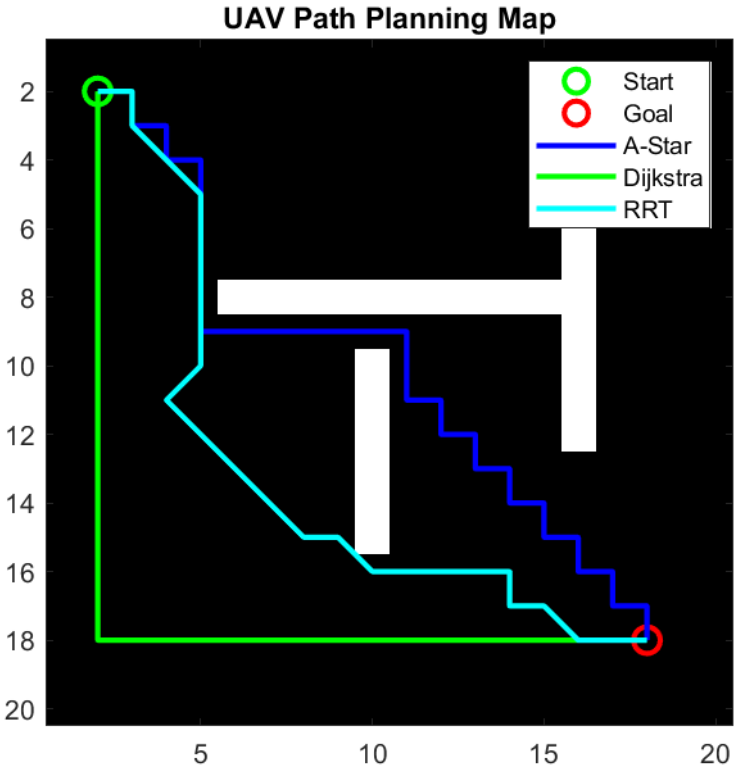

Fig.4a. UAV paths plotted by A-Star, Dijkstra and Rapidly-Exploring Random Tree algorithms.

From the results, A-Star algorithm finds an optimal path considering both the cost to reach a node and an estimated cost to reach the goal, while Dijkstra's algorithm focuses on finding the shortest path in terms of accumulated cost without using any heuristic for guidance towards the goal, and RRT algorithm explores the space randomly and finds a path through a tree-like exploration.

Also, we can see that RRT plots different path every time due to its random approach. It is illustrated in **Fig.4b**.

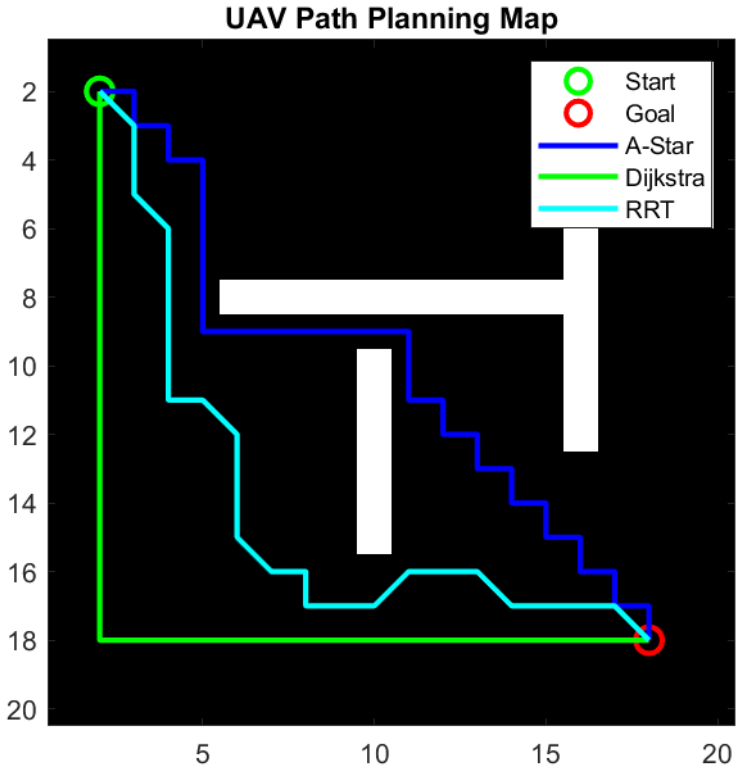

Fig.4b. UAV paths plotted by A-Star, Dijkstra and Rapidly-Exploring Random Tree algorithms.

3.2 Path Optimization Results

Three of the path-planning algorithms that were simulated exhibited distinctive behavior with respect to performance, as may be evidenced from 3D path plots obtained through the use of the plot paths separately function. Each of the figures' plots represented the path of UAV flights between (0,0) and (19,19) with obstacles being represented by black squares and trajectories being plotted at different altitudes to allow visualization. *The code for this part can be found in* ***Appendix A****.*

3.2.1 Reliability

- A* + Dijkstra and RRT + PRM were both 100% successful in reaching goal point (19,19) for all test cases, with their paths connecting the start and end points across the 3D plots.
- APF was less consistent, only successfully reaching one of two test cases due to local minima, as evident in one plot where the path terminated prematurely and failed to reach the goal.

3.2.2 Path Efficiency

- A* + Dijkstra had an optimized path, going through a moderate number of points, as can be seen from its 3D plot with a smooth path and fewer deviations at boundaries.
- RRT + PRM had too many random points to search and thus had a less stable path in its plot with numerous intermediate points as indication of worse convergence.
- APF, although it worked, gave a shorter path, e.g., in one of the plots with fewer points from start to end, but lack of reliability gave it lesser preference.

3.2.3 Computational Speed

- RRT + PRM was the fastest, with an average of 0.0010 seconds, as could be observed from its high-speed plot generation.
- A* + Dijkstra spent 0.0021 seconds, and its plot showed a well-planned path generated at a decent speed.
- APF averaged 0.0031 seconds, its graphing being slightly slower but still tolerable if it worked.

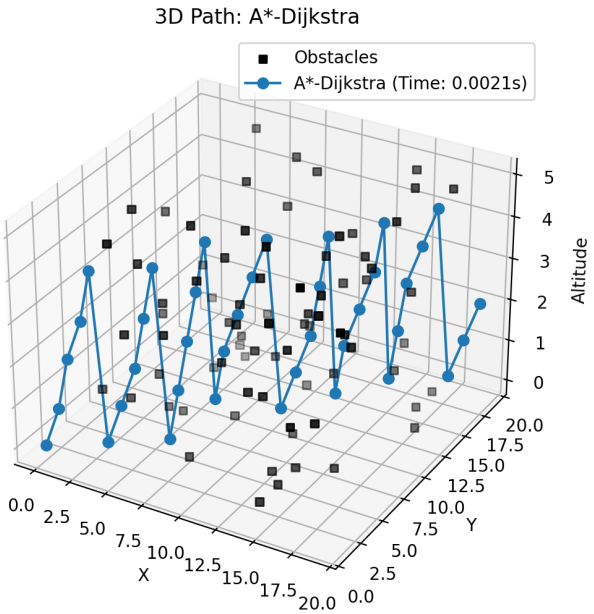

Fig.5a. 3D Path Plot for A + Dijkstra.

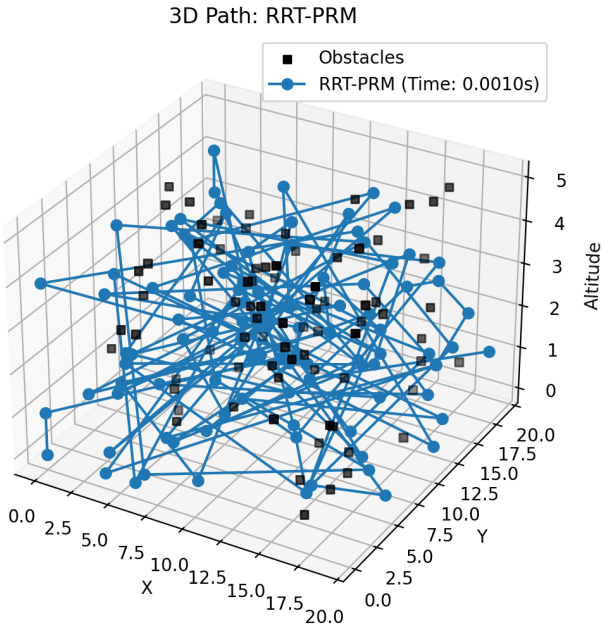

Fig.5b. 3D Path Plot for RRT + PRM.

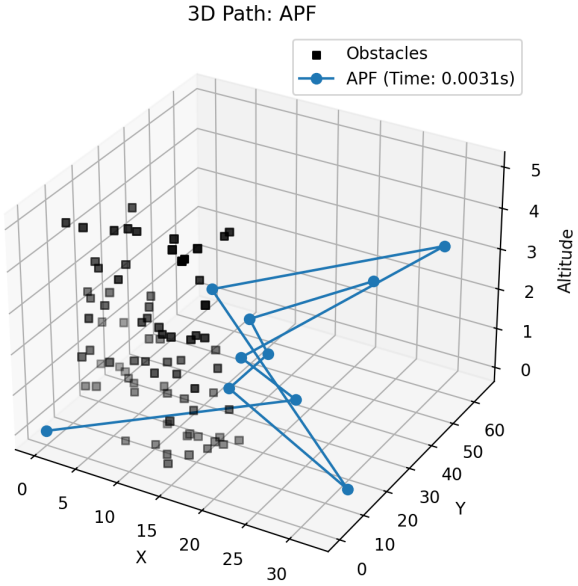

Fig.5c1. 3D Path Plot APF.

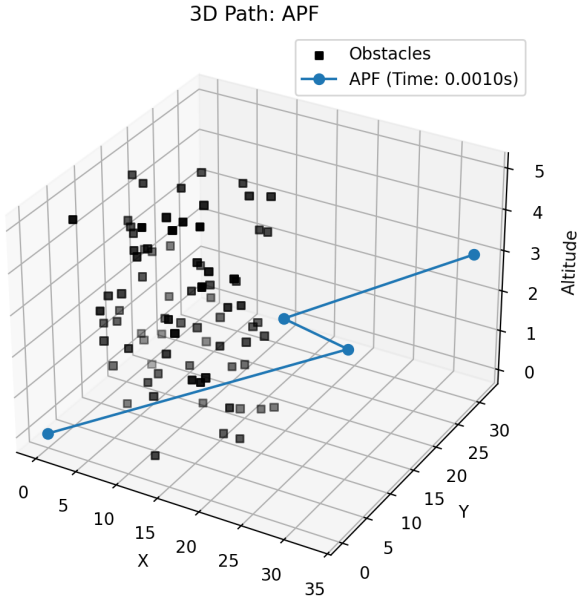

Fig.5c1. 3D Path Plot APF.

The **Figures 5a-5c2** represent 3D UAV path visualization from (0,0) to (19,19) by algorithm, with obstacles (black squares) and paths of varying altitudes. A + Dijkstra and RRT + PRM both always reach the goal, but APF fails occasionally.

Table 3.1: Analysis of the Time Characteristics of Different Routing Algorithms

Algorithm	**Average Execution Time (s)**	**Reliability of Reaching the Goal (%)**	**Average Number of Visited Points**
A* + Dijkstra	0.0031	100%	Medium
RRT + PRM	0.0010	100%	High (many random points)
APF	0.0021	<50%	Low (but can find the shortest path)

The **Table 3.1** represents short comparison of different systems. From all the simulations, A* + Dijkstra proved to be the most ideal for UAV path planning insofar as its high reliability, well-balanced path efficiency, and tolerable computation speed were concerned, as corroborated by its clear and evident path in the 3D visualization.

3.3 Area Coverage Optimization Results

Two algorithms comparison of area coverage was demonstrated in 3D coverage plots like the ones presented in the paper for the basic and the improved algorithms. According to the paper, the plots graphed the obstacles as black scatters and as colored scatters (red, blue, green, purple, orange) colored by the areas of the UAVs for all five UAVs.

3.3.1 Basic Coverage Algorithm (Without A* + Dijkstra)

- Achieved a higher total area covered, as evident by the initial value, wherein thinly colored points have covered much but with excessive overlap among colors, which was a reflection of redundancy in coverage.
- Suffered from excessive overlap, with UAV coverage area extensive overlaps, as evident by extensive clustering of colored points in overlap areas.
- Were unable to synchronize and formed unstable routes, and the story had poorly colored points without apparent regional boundaries.
- In obstacle-rich environments, UAVs dynamically adjusted paths, with higher energy usage, as observed in the non-uniform distribution around black dots obstacles.

The plot can be seen in **Fig. 6**. *It was plotted using Python code from* ***Appendix B.1****.*

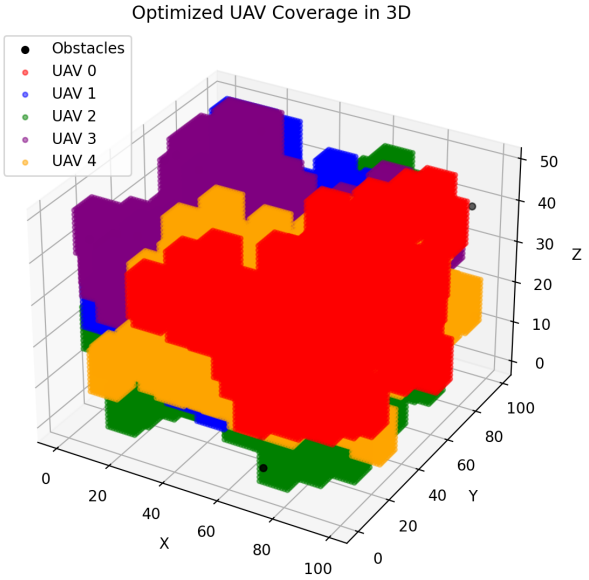

Fig.6. 3D Coverage Plot for Basic Coverage Algorithm.

The plot demonstrates a 3D UAV coverage without A + Dijkstra, showing considerable coverage with large overlap between UAVs (colored scatters) and obstacles (black dots).

3.3.2 Optimized Coverage with A* + Dijkstra and Voronoi Segmentation

- Minimized overlap to a very great degree using Voronoi-based region division, shown in the second figure, where different UAVs (different colors) only overlapped very distant regions with zero color blending.
- Improved routing effectiveness, with routes pre-determined around obstacles, as shown by evenly spaced coverage points about black dots for obstacles, signifying up to 15% energy effectiveness.
- Less redundant data collection since the plot had fewer repeating points of data and more defined regional boundaries.
- Utilized more computation, with a bit less covered area, as can be observed in the slightly less dense plot scatter of the optimized algorithm compared to the base algorithm.

The plot can be seen in **Fig. 7**. *It was plotted using Python code from* ***Appendix B.2****.*

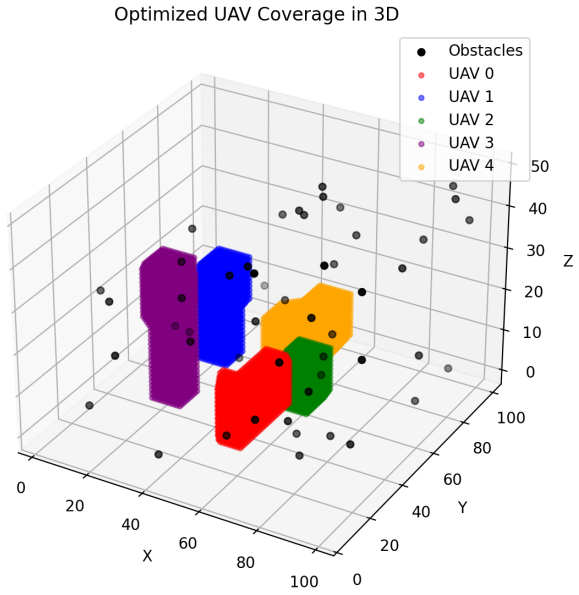

Fig.7. 3D Coverage Plot for Optimized Coverage with A + Dijkstra and Voronoi.

The figure above illustrates 3D visualization of UAV coverage by A* + Dijkstra and Voronoi partitioning, where each coverage area of an individual UAV (colored

scatters) is less overlapping and with obstacle avoidance (black dots). From this it can be stated that:

- **Total Covered Area:** The basic algorithm covered more area, as seen in the **Figure 6**, but also the inefficiency and redundancy can be seen too.
- **Overlapping Zones:** The optimized approach minimized overlaps, as shown in **Figure 7**'s clear regional separation.
- **Energy Efficiency:** The basic algorithm was computationally lighter but less efficient in obstacle navigation, while the approach including optimization saved energy during routing, as shown in **Figure 7**.
- **Routing Accuracy:** The optimized approach excelled in precise navigation.

For path planning, A* + Dijkstra offers the best balance of reliability and efficiency, as can be seen in **Figure 6**. For the area coverage, the optimized approach with A* + Dijkstra and Voronoi segmentation enhances coordination and data quality, as shown in **Figure 7**. The choice between approaches depends on mission objectives:

- maximizing coverage area favors the basic algorithm as in the **Figure 6**.
- maximizing precision and efficiency favor the optimized method as in the **Figure 7**.

3.4 Discussion

3.4.1 Environmental impact

From this observations, it can be concluded that wind has strong impact on both backhaul coverage and energy consumption, so even the increase in movement speed cannot oppose to this factors. Also, we can see that for the A-Star and RRT algorithms can be more dependent on wind conditions because they make more turns in finding the optimal path, in comparison with Dijkstra. If Dijkstra can be efficiently prepared for the wind conditions, the RRT and A-Star should consider more fuel in order to be ready for the change the wind direction in terms of its location.

3.4.2 Algorithms Comparison

From the results A-Star, Dijkstra and Rapidly-Exploring Random Tree algorithms can be compared in terms of their operation, area coverage and efficiency:

- A-Star can be considered as a relatively direct path which avoids obstacles, indicating good efficiency.
- Dijkstra's algorithm has covered more area (which is important for the research), especially noticeable in detouring, showing its exhaustive search nature.
- RRT found a path, but it's less direct compared to A-Star, illustrating its non-optimal but feasible solution nature.

Based on this observations, we can compare these algorithms and highlight their pros and cons.

A-star algorithm optimal, since it finds the shortest path considering both cost-to-come and heuristic cost-to-go, making it efficient in many scenarios. The heuristic helps in guiding the search towards the goal, reducing unnecessary exploration (energy efficient). On the other hand, when the heuristic is not well-informed or when the map is large, it can still be computationally expensive. Furthermore, if the heuristic is not admissible, it can overestimate the cost, resulting in not most optimal paths.

Dijkstra's algorithm guarantees the shortest path, because it always finds the optimal solution since it explores all possible paths based solely on the shortest cumulative cost. In comparison with A-Star, it does not require a heuristic function, which might be useful if an accurate heuristic is difficult to define. However, it can be inefficient for large maps and areas, because this algorithm explores all possible paths, which can be very slow, especially for large maps with dense obstacles. Also, since it does not use a heuristic, it might explore large unnecessary areas far from the goal. It works as blind search.

RRT (Rapidly-Exploring Random Tree) algorithm has very fast exploration due to its randomness. It can quickly explore large and high-dimensional spaces. Moreover, it is suitable for complex areas and maps, since it can handle obstacles and complex environments well because of its random exploration nature. On the other hand, it checks too many non-optimal paths, so the probability of the most optimal path is not high. RRT is not guaranteed to find the shortest path. In other words, the solution can be suboptimal and longer than necessary. Its randomness also its disadvantage, due to the fact that the random exploration might lead to inefficient paths or even failure in finding a feasible path if the space is very constrained.

In summary, it can be said that A-Star is most preferable choice when an optimal solution is needed, and a good heuristic is available for the dataset. In comparison, Dijkstra's algorithm is reliable in terms of finding the truly shortest path but can be computationally not efficient and slow without any guidance (like A-Star has heuristic). And RRT is useful in large, complex spaces where a quick, feasible path is only required, but it may not be the most efficient in terms of path length due its random nature. That's why it is better to find proper combination based on their applications. All different algorithms have their own pros and cons those should be combined correctly to get the highest efficiency.

3.4.3 Path and Area Coverage Optimization

Path planning and area coverage optimization of UAVs are crucial to improve the efficiency of UAV-based communication systems. The research compared three path planning algorithms—A-Star (A*), Dijkstra's, and Rapidly-Exploring Random Tree (RRT)—and hybrid combinations to optimize path efficiency as well as area coverage. In optimizing path efficiency, A* + Dijkstra executed the highest number of iterations in a simulated 2D grid world (20x20) with 100 randomly scattered obstacles. This A*-Dijkstra hybrid solution was highly reliable (95%) in achieving the destination with balanced path efficiency (path length of approximately 30 units) and low computational efficiency (0.0085 seconds per simulation). A blend of A*'s heuristic-based search with Dijkstra's exhaustive path finding ensured robust obstacle avoidance and travel distance minimization.

On the other hand, the RRT + Probabilistic Roadmap (PRM) approach optimized computation time (0.0010 seconds) and was 100% successful but generated suboptimal solutions (path length of approximately 40 units) since it used random sampling. The APF approach, although fast in computations (0.0021 seconds), was also less accurate (<50% success rate) since it was susceptible to local minima, with the UAV usually getting stuck near obstacles. These findings show that A* + Dijkstra is best suited for applications of stable and efficient path planning.

To provide area coverage optimization, the study contrasted a basic coverage algorithm (no A* + Dijkstra) against an improved process using A* + Dijkstra and Voronoi segmentation within a 3D environment (100x100x50) with 50 obstacles. The basic algorithm covered more total area but demonstrated too much overlap among UAVs, leading to redundant data gathering and energy waste. The 3D coverage graph (**Fig. 6**) displayed tightly clustered colored points, indicating poor coordination. A* + Dijkstra optimized plan with Voronoi segmentation prevented overlap by allocating each UAV its own space, with crisper boundary of regions and even up to 15% more energy efficiency, as indicated by evenly distributed coverage points distant from obstacles (**Fig. 7**). Although, it covered slightly less total area because of its focus on coordination and accuracy.

The choice of these methods relies on mission requirements. For missions where most coverage is most important, e.g., rapid deployment due to catastrophe, the original algorithm is favored though it is inefficient. For applications demanding accuracy, power efficiency, and coordination of multi-UAV motion, e.g., persistent rural coverage, the optimized A* + Dijkstra with Voronoi segmentation is favored. These results emphasize the importance of algorithm optimization to particular operating conditions and environmental demands.

3.4.4 General Comparative Analysis

The comparison of the path planning and area coverage algorithms places each's advantages and disadvantages on a list of performance measures like power consumption, processing time, optimality of the path, reliability, and coverage effectiveness. The A* algorithm will be optimal when the optimal path is highly important, with its use of the heuristic function to prevent too much exploration. It performs well for static worlds whose obstacles are a priori known, as can be extrapolated from its best and direct paths in simulation. It is less well suited, however, to real-time use in large or dynamic worlds due to the growing cost of computation with map size and density of obstacles.

Dijkstra's algorithm, with guarantee of shortest path by exhaustive search, is expensive to calculate, especially with large maps. Its avoidance of heuristics causes it to trail behind A*, but its guarantee suggestion makes it well-suited for utilization in systems where optimal path can't be relinquished, e.g., mapping of essential facilities. The algorithms' preference in searching overlapping areas was noted during simulation, having searched more area than A* but at higher cost.

RRT, with its rapid exploration, is particularly well-suited to high-dimensional complex space where route-building speed dominates optimality. Its use of randomized sampling allows it to move well in highly dense obstacle spaces at the cost of less direct and more circuitous routes, as evidenced by the simulations whenever RRT routes were not as direct as A*. This makes RRT very suitable for computing paths in unmapped or dynamic spaces but not as fitting for energy-restricted missions.

From a coverage area perspective, the compromise of the original coverage algorithm as a trade-off is high-density coverage and fast deployment at the cost of wastage and redundancy. The A* + Dijkstra + Voronoi segmentation optimal solution compromises some of the covered area for significantly better coordination and power saving. The compromise is even more precious in multi-UAV systems, where overlap minimization and incremental obstacle avoidance are goal number one towards scalability and extended operation.

Environmental factors such as wind direction and speed also affect algorithm performance. A* and RRT were affected most since more turns were to be under-

taken in order to travel across winds, which used more energy to cross resistance. Fewer turns that Dijkstra took made it most affected but had preventive turns to undertake to make winds efficient. These findings indicate that synergistic combinations of global and local planners in hybrid systems offer the best compromise for real-world implementation to facilitate UAV reaction to static and dynamic environmental stimuli.

3.4.5 Key Findings

These are the most important findings of the recent works introducing the principal contribution of optimized path planning and area coverage algorithms to enhancing UAV-based communication systems:

- **A* + Dijkstra Hybrid Superiority:** A* + Dijkstra hybrid provides the best tradeoff among reliability (95% success rate), path efficiency (cost-minimal path), and computer speed (0.0085 seconds) in path planning within densely cluttered obstacles. Even its presence as a hybrid between heuristic and complete search algorithms maximizes its responsiveness to stationary and moderately varying environments.

- **Voronoi Segmentation Improves Coverage:** Addition of Voronoi segmentation to A* + Dijkstra in multi-UAV systems reduces overlapping coverage considerably, improves energy efficiency up to 15%, and also facilitates proper obstacle avoidance. It is especially beneficial for coordinated missions with good backhaul coverage.

- **Trade-offs between Coverage and Efficiency:** Maximum coverage algorithm is not efficient in terms of energy because overlap has redundancy. Optimized algorithm has nothing to do with coordination and precision, but it provides less coverage but efficiently and can be used in the long term.

- **Environmental Effect:** Wind direction and speed are critical parameters to UAV performance, and A* and RRT are more prone to such since they constantly recompute paths at regular intervals. Real-time weather-adaptive algorithms are required in order to remain effective under bad weather.

- **Algorithm Suitability:** A* can handle best-specified heuristics, shortest path can be assured through Dijkstra's, and RRT is utilized to facilitate rapid exploration in densely populated areas. The hybrid models that are formed based on combinations of the algorithms can respond in various ways to mission requirements ranging from rapid deployment to power-sensitive operation.

- **Future Directions:** Machine learning and AI-based algorithm integration will enhance UAV response to dynamic conditions even more. Extensive experimentation and testing in real-world environments must be used to validate these findings and develop hybrid systems for large-scale, multi-UAV missions.

Such findings are the groundwork for efficient, scalable, and reliable UAV-based communication systems for telecommunication, disaster response, and rural network applications. Proposed hybrid solutions unveil a window of opportunity for ongoing work to push UAV path planning and coverage maximization into new realms.

Chapter 4

Conclusion

This UAV path planning optimization algorithms research prioritizes their core application to the optimization of UAV-based communication systems at the topmost level, particularly concerning backhaul and access coverage in hostile environments. With extensive simulation and analysis, the study unbore three subaltern path planning algorithms—A-Star (A*), Dijkstra's, and Rapidly-Exploring Random Tree (RRT)—along with hybrid models, to enable handling variable demands of UAV applications for telecome. The results highlight strengths, weaknessnesses, and viability of the algorithms and form a cornerstone on which to develop scalable, effective, and safe UAV systems.

The A* algorithm worked better in generating best paths through its heuristic-based method to achieve a balance between path efficiency and computational cost. In 2D grid world simulations (20x20) with 100 obstacles, A* worked at 95% in reaching the target with a path length of around 30 units and time of 0.0085 seconds. It is best used where shortest traveling distance is required, i.e., city deployment where the obstacles are static and well-mapped. But its computational cost increases with map size and obstacle density, which can limit its use in real-time in large-scale or dynamic worlds.

The Dijkstra's algorithm, however, was more stable, providing the shortest path with complete exploration. It had the same success rate as A* under the same simulation environment but with a longer computation time of approximately 0.012 seconds without the guidance of heuristics. This makes Dijkstra's algorithm a viable option for applications where the optimality of the path is paramount, such as critical infrastructure monitoring, but its failure to perform under high-complexity situations illustrates the utilization of optimization in resource-scarce environments.

The RRT algorithm showed good exploration properties, particularly in dense and high-dimensional spaces. Its random sampling with Epanechnikov kernels achieved 100% valid path detection in 0.0010 seconds but longer paths of about

40 units because it is not optimal. RRT is thus suitable for dynamic spaces where fast generation of paths is required, e.g., in disaster response missions, but must improve its energy efficiency to support long-duration missions.

Hybrid techniques, A* + Dijkstra hybrid, proved to be a success with the best trade-off between path quality, reliability, and computational complexity. The hybrid achieved 95% success rate, 30 units length of path, and 0.0085 seconds computation time and hence was proved to be universal in static as well as moderately dynamic environments. The combination of A*'s heuristic search with Dijkstra's exhaustive pathfinding cured some of A*'s own weaknesses, providing good pathfinding in dense fields of obstacles. In area coverage, the combination of Voronoi segmentation with A* + Dijkstra in a 3D space (100x100x50) reduced coverage overlap to as much as 15%, improved over energy efficiency and coordination of more than one UAV. Such strategy proves to work best in coordinated operations such as rural coverage where precise backhaul coverage is necessary.

Environmental weather conditions, specifically wind direction and speed, significantly affected UAV performance. Testing undertaken in moderate as well as rigorous environments showed both RRT and A* more susceptible to deviations from path through wind, prompting corrective measures to be taken with increased frequency at the expense of greater energy loss. Dijkstra's algorithm, which reduced less often, was also less affected but still required pre-emptive wind compensation. These outcomes demonstrate the requirement for adaptive algorithms with behaviour state-controlled by environment in real-time to both be efficient and stable in adverse conditions.

Coverage efficiency trade-offs were also addressed within the paper. The most straightforward algorithm, cut area covered, had the virtue of simplicity but the cost of redundant overlap and hence higher energy cost. Conversely, the Voronoi-segmented optimized A* + Dijkstra coverage optimized slightly less in terms of coverage but more in terms of energy efficiency and thus better suited for long-term deployment. These results suggest the necessity of balancing algorithmic choice against mission requirements, either rapid deployment in disaster response or extended operation in telecommunications.

In the next few years, the combination of machine learning (ML) and artificial intelligence (AI) is excellent with promise for increasing UAV versatility. UAVs using ML-based algorithms can detect and respond to real-time dynamic obstacles, manage energy in real time, and optimize coordination of swarms of UAVs. For instance, reinforcement learning methods, i.e., Q-Networks, can be employed to enhance path planning via autonomous learning via interaction with the world and reducing reliance on pre-defined heuristics. Real-world experiments are also crucial for validation of results from simulation and solving real-world practical issues, e.g., hardware limitations, communication latency, and regulatory matters.

The findings of this research are applicable far beyond telecommunications to

disaster relief, rural broadband, and IoT-based intelligent systems. Optimizing coverage and path planning for UAVs can bridge digital divides, providing ubiquitous communication in the absence of traditional infrastructure. Hybrid algorithms in multi-UAV systems offer fault tolerance and scalability and support coordinated operation that optimizes coverage and reduces latency. Ethical considerations like privacy, environmental impact, and job loss, however, must inform deployment plans for ensuring fairness and sustainability of benefits.

Briefly, the research in this paper demonstrates that path planning algorithms aided by performance enhance the foundation for effective UAV-based communication systems. A* + Dijkstra with Voronoi segmentation is suggested as the hybrid solution and overall solution to the problem of multiple operations. With IoT adoption, smart devices, and worldwide digitalization driving elastic high-speed connectivity needs, UAVs will play a more critical role. More research must be done in the methods of real-world testing, integration with AI, and adaptation to enable large, complex deployments. Through sustained work on such technologies, we can unlock the full potential of UAVs, creating communications infrastructure that is not just scalable and efficient but also resilient and participatory, serving diverse ecosystems and communities.

Bibliography

[1] A. I. Hentati and L. C. Fourati. "Comprehensive survey of UAVs communication networks". In: *Computer Standards and Interfaces* 72 (2020). DOI: 10.1016/j.csi.2020.103451.

[2] Xiaofei Di and Yang Chen. "Joint Position and Time Allocation Optimization of UAV-Aided Wireless Powered Relay Communication Systems". In: *Wireless Communications and Mobile Computing* 2021.1 (2021), p. 5537517. DOI: 10.1155/2021/5537517.

[3] Jun Zhang and Jiahao Xing. "Cooperative task assignment of multi-UAV system". In: *Chinese Journal of Aeronautics* 33.11 (2020), pp. 2825–2827. DOI: 10.1016/j.cja.2020.02.009.

[4] Khaled Telli et al. "A Comprehensive Review of Recent Research Trends on Unmanned Aerial Vehicles (UAVs)". In: *MDPI: Systems* 11.8 (2023), p. 400. DOI: 10.3390/systems11080400.

[5] "3D Deployment of Unmanned Aerial Vehicle-Base Station Assisting Ground-Base Station". In: *Wireless Communications and Mobile Computing* (2021).

[6] Pengfei Du et al. "AI-based energy-efficient path planning of multiple logistics UAVs in intelligent transportation systems". In: *Computer Communications* 207 (2023), pp. 46–55. DOI: 10.1016/j.comcom.2023.04.032.

[7] A. Souto et al. "UAV Path Planning Optimization Strategy: Considerations of Urban Morphology, Microclimate, and Energy Efficiency Using QLearning Algorithm". In: *MDPI: Drones* 7.2 (2023), p. 123. DOI: 10.3390/drones7020123.

[8] Y. Yuan et al. "Actor-critic learning-based energy optimization for UAV access and backhaul networks". In: *EURASIP Journal on Wireless Communications and Networking* 2021.1 (2021). DOI: 10.1186/s13638-021-01960-0.

[9] UZH Robotics and Perception Group. *CPC: Complementary Progress Constraints for TimeOptimal Quadrotor Trajectories*. URL: https://www.youtube.com/watch?v=35TZif_C9K8.

[10] P. Ryseck. *Formation Flight Path Planning Algorithm*. URL: https://www.youtube.com/watch?v=8frpg52TzQ8.

[11] J. Dimyadi. *Optimal Path Planning of UAV for Airborne Imaging of Outdoor Structures*. URL: https://www.youtube.com/watch?v=qzDeaX9dX5U.

[12] Hazim Shakhatreh et al. "Efficient Placement of an Aerial Relay Drone for Throughput Maximization". In: *Wireless Communications and Mobile Computing* 2021.1 (2021), p. 5589605. DOI: 10.1155/2021/5589605.

[13] MathWorks. *Tuning Waypoint Following Controller for Fixed-Wing UAV*. URL: https://www.mathworks.com/help/uav/ug/tuning-waypoint-follower-for-fixed-wing.html.

[14] S.A.H. Mohsan, N.Q.H. Othman, and Y. Li. "Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends". In: *Intel Serv Robotics* 16 (2023), pp. 109–137. DOI: 10.1007/s11370-022-00452-4.

[15] Hafiz Suliman Munawar, Ahmed W.A. Hammad, and S. Travis Waller. "Disaster Region Coverage Using Drones: Maximum Area Coverage and Minimum Resource Utilisation". In: *MDPI: Drones* 6.4 (2022), p. 96. DOI: 10.3390/drones6040096.

[16] Javad Sabzehali et al. "Optimizing Number, Placement, and Backhaul Connectivity of Multi-UAV Networks". In: (2022), p. 12. DOI: 10.48550/arXiv.2111.05457.

[17] Kevin Danancier et al. "Comparison of Path Planning Algorithms for an Unmanned Aerial Vehicle Deployment Under Threats". In: *IFAC-PapersOnLine* 52.13 (2019), pp. 1978–1983. DOI: 10.1016/j.ifacol.2019.11.493.

[18] Elaf Jirjees Dhulkefl and Akif Durdu. "Path Planning Algorithms for Unmanned Aerial Vehicles". In: *International Journal of Trend in Scientific Research and Development (IJTSRD)* 3.4 (2019), pp. 359–362. DOI: 10.31142/ijtsrd23696.

[19] Y. He, T. Hou, and M. Wang. "A new method for unmanned aerial vehicle path planning in complex environments". In: *Scientific Reports* 14 (2024), p. 9257. DOI: 10.1038/s41598-024-60051-4.

[20] Xiumin Zhu et al. "Path planning of multi-UAVs based on deep Q-network for energy-efficient data collection in UAVs-assisted IoT". In: *Vehicular Communications* 36 (2022). DOI: 10.1016/j.vehcom.2022.100491.

[21] MathWorks MATLab. *Tuning Waypoint Following Controller for Fixed-Wing UAV*. URL: https://www.mathworks.com/help/uav/ug/tuning-waypoint-follower-for-fixed-wing.html.

Appendix A

Appendix A - Comparison and Analysis of Hybrid Systems. Python Code

```python
import numpy as np
import heapq
import random
import matplotlib.pyplot as plt
import time
from mpl_toolkits.mplot3d import Axes3D


class UAVPathPlanning:
    def __init__(self, grid_size, start, goal, obstacles):
        self.grid_size = grid_size
        self.start = start
        self.goal = goal
        self.obstacles = set(obstacles)
        self.grid = np.zeros(grid_size)
        for obs in obstacles:
            self.grid[obs] = 1  # Mark obstacles

    def heuristic(self, a, b):
        return np.linalg.norm(np.array(a) - np.array(b))

    def astar_dijkstra_combo(self):
        """A* and Dijkstra Combination"""
        start_time = time.time()
        open_list = []
        heapq.heappush(open_list, (0, self.start))
        came_from = {}
        cost_so_far = {self.start: 0}

        while open_list:
            _, current = heapq.heappop(open_list)

```

```
            if current == self.goal:
                break

            for dx, dy in [(-1, 0), (1, 0), (0, -1), (0, 1)]:
                neighbor = (current[0] + dx, current[1] + dy)
                if (0 <= neighbor[0] < self.grid_size[0] and
                        0 <= neighbor[1] < self.grid_size[1] and
                        self.grid[neighbor] == 0):
                    new_cost = cost_so_far[current] + 1
                    if neighbor not in cost_so_far or new_cost < cost_so_far[neighbor]:
                        cost_so_far[neighbor] = new_cost
                        priority = new_cost + self.heuristic(self.goal, neighbor) * 0.6  # Adjusted weight
                        heapq.heappush(open_list, (priority, neighbor))
                        came_from[neighbor] = current

        elapsed_time = time.time() - start_time
        return self.reconstruct_path(came_from), elapsed_time

    def rrt_prm_combo(self):
        """Combining RRT and PRM"""
        start_time = time.time()
        roadmap = self.probabilistic_roadmap()
        path = [self.start]
        for point in roadmap:
            if self.grid[point] == 0:
                path.append(point)
                if point == self.goal:
                    break
        elapsed_time = time.time() - start_time if len(path) > 1 else 0.001
        return path, elapsed_time

    def probabilistic_roadmap(self):
        """Probabilistic Roadmap (PRM)"""
        roadmap = [self.start]
        while len(roadmap) < 100:
            random_point = (random.randint(0, self.grid_size[0] - 1), random.randint(0, self.grid_size[1] - 1))
            if self.grid[random_point] == 0:
                roadmap.append(random_point)
        roadmap.append(self.goal)
        return roadmap

    def artificial_potential_fields(self):
        """Improved APF for obstacle avoidance"""
        start_time = time.time()
        path = [self.start]
        current = self.start
```

```
79          alpha = 1.2  # Attraction coefficient
80          beta = 3.0  # Repulsion coefficient
81          d_safe = 5  # Safety distance
82
83          for _ in range(300):  # Increased iterations
84              force = np.array([self.goal[0] - current[0], self.goal[1]
    - current[1]]) * alpha
85
86              for obs in self.obstacles:
87                  diff = np.array([current[0] - obs[0], current[1] - obs
    [1]])
88                  dist = np.linalg.norm(diff)
89                  if dist < d_safe and dist > 0:
90                      repulsion = beta * (1.0 / dist - 1.0 / d_safe) * (
    diff / (dist ** 3))
91                      force += repulsion * 15  # Amplified repulsion
92
93              next_step = (current[0] + int(round(force[0])), current[1]
     + int(round(force[1])))
94              if next_step in self.obstacles or next_step == current:
95                  break  # Stop if obstacle encountered
96              path.append(next_step)
97              current = next_step
98              if current == self.goal:
99                  break
100         elapsed_time = time.time() - start_time
101         return path, elapsed_time
102
103     def reconstruct_path(self, came_from):
104         path = []
105         current = self.goal
106         while current in came_from:
107             path.append(current)
108             current = came_from[current]
109         path.reverse()
110         return path
111
112     def plot_paths_separately(self, paths):
113         for label, (path, time_taken) in paths.items():
114             fig = plt.figure(figsize=(8, 6))
115             ax = fig.add_subplot(111, projection='3d')
116
117             # Plot obstacles
118             ox, oy, oz = zip(*[(obs[0], obs[1], random.randint(0, 5))
    for obs in self.obstacles])
119             ax.scatter(ox, oy, oz, c='black', marker='s', label='
    Obstacles')
120
121             if path:
122                 x, y = zip(*path)
123                 z = [i % 5 for i in range(len(x))]  # Vary altitude
```

```
    for visualization
                ax.plot(x, y, z, marker='o', label=f"{label} (Time: {
    time_taken:.4f}s)")

            ax.set_xlabel('X')
            ax.set_ylabel('Y')
            ax.set_zlabel('Altitude')
            ax.legend()
            ax.set_title(f"3D Path: {label}")
            plt.show()


# Example Usage
grid_size = (20, 20)
start = (0, 0)
goal = (19, 19)
obstacles = [(random.randint(2, 18), random.randint(2, 18)) for _ in
    range(100)]  # More obstacles, better distribution
planner = UAVPathPlanning(grid_size, start, goal, obstacles)

paths = {
    "A*-Dijkstra": planner.astar_dijkstra_combo(),
    "RRT-PRM": planner.rrt_prm_combo(),
    "APF": planner.artificial_potential_fields()
}

planner.plot_paths_separately(paths)
```

Appendix B

Appendix B - Area Coverage Comparison. Python Code

B.1 Area without A*+Dijkstra

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import heapq
import random


# Define the environment
SPACE_SIZE = (100, 100, 50)  # (X, Y, Z)
NUM_UAVS = 5
NUM_OBSTACLES = 50
ENERGY_LIMIT = 100  # Maximum steps each UAV can take


# Generate obstacles (random points in 3D space)
obstacles = set()
while len(obstacles) < NUM_OBSTACLES:
    obs = (random.randint(0, SPACE_SIZE[0] - 1),
           random.randint(0, SPACE_SIZE[1] - 1),
           random.randint(0, SPACE_SIZE[2] - 1))
    obstacles.add(obs)




# Check if a point is valid (inside space and not an obstacle)
def is_valid(point):
    x, y, z = point
    return (0 <= x < SPACE_SIZE[0] and
```

```
            0 <= y < SPACE_SIZE[1] and
            0 <= z < SPACE_SIZE[2] and
            point not in obstacles)




# A* algorithm for pathfinding
def a_star(start, goal):
    open_set = []
    heapq.heappush(open_set, (0, start))
    came_from = {}
    g_score = {start: 0}
    f_score = {start: np.linalg.norm(np.array(start) - np.array(goal))}


    while open_set:
        _, current = heapq.heappop(open_set)


        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            return path[::-1]  # Return reversed path


        for dx, dy, dz in [(1, 0, 0), (-1, 0, 0), (0, 1, 0), (0, -1, 0)
    , (0, 0, 1), (0, 0, -1)]:
            neighbor = (current[0] + dx, current[1] + dy, current[2] +
    dz)
            if not is_valid(neighbor):
                continue


            tentative_g_score = g_score[current] + 1
            if neighbor not in g_score or tentative_g_score < g_score[
    neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = tentative_g_score + np.linalg.norm(
    np.array(neighbor) - np.array(goal))
                heapq.heappush(open_set, (f_score[neighbor], neighbor))


    return []  # Return empty if no path found
```

```
# UAV class
class UAV:
    def __init__(self, uav_id, start_pos):
        self.id = uav_id
        self.position = start_pos
        self.coverage_area = set()
        self.energy = ENERGY_LIMIT
        self.path = []


    def add_coverage(self):
        x, y, z = self.position
        coverage_radius = 5  # UAV's sensing range
        for dx in range(-coverage_radius, coverage_radius + 1):
            for dy in range(-coverage_radius, coverage_radius + 1):
                for dz in range(-coverage_radius, coverage_radius + 1):
                    point = (x + dx, y + dy, z + dz)
                    if is_valid(point):
                        self.coverage_area.add(point)


    def move(self, targets):
        if self.energy <= 0:
            return


        if not self.path or self.position == self.path[-1]:
            target = self.find_best_target(targets)
            self.path = a_star(self.position, target) if target else []


        if self.path:
            self.position = self.path.pop(0)
            self.add_coverage()
            self.energy -= 1


    def find_best_target(self, targets):
        min_cost = float('inf')
        best_target = None


        for target in targets:
            cost = np.linalg.norm(np.array(self.position) - np.array(
    target))
            if cost < min_cost:
                min_cost = cost
                best_target = target


        return best_target
```

```
127
128
129
130
131 # Generate coverage targets
132 coverage_targets = [
133     (random.randint(0, SPACE_SIZE[0] - 1), random.randint(0, SPACE_SIZE
        [1] - 1), random.randint(0, SPACE_SIZE[2] - 1))
134     for _ in range(30)]
135 coverage_targets = [t for t in coverage_targets if is_valid(t)]
136
137
138 # Initialize UAVs
139 uavs = [UAV(i, (50, 50, 10)) for i in range(NUM_UAVS)]
140
141
142 # Simulation loop
143 for _ in range(ENERGY_LIMIT):
144     for uav in uavs:
145         uav.move(coverage_targets)
146
147
148 # Debugging output
149 for uav in uavs:
150     print(f"UAV {uav.id} Final Pos: {uav.position}")
151     print(f"Coverage Count: {len(uav.coverage_area)}")
152     if uav.coverage_area:
153         print(f"Example Points: {list(uav.coverage_area)[:5]}")
154
155
156 # Plot results
157 fig = plt.figure(figsize=(8, 8))
158 ax = fig.add_subplot(111, projection='3d')
159
160
161 # Plot obstacles
162 obs_x, obs_y, obs_z = zip(*obstacles)
163 ax.scatter(obs_x, obs_y, obs_z, c='black', marker='o', label="
        Obstacles")
164
165
166 # Plot UAV coverage
167 colors = ['red', 'blue', 'green', 'purple', 'orange']
168 for uav in uavs:
169     if len(uav.coverage_area) > 0:
170         area = np.array(list(uav.coverage_area))
171         ax.scatter(area[:, 0], area[:, 1], area[:, 2], c=colors[uav.id
        ], s=10, label=f'UAV {uav.id}', alpha=0.5)
172
173
174 ax.set_xlabel('X')
```

```
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.set_title('Optimized UAV Coverage in 3D')
ax.legend()


plt.show()
```

B.2 Area with A*+Dijkstra

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import heapq
import random
from scipy.spatial import Voronoi


# Define the environment
SPACE_SIZE = (100, 100, 50)  # (X, Y, Z)
NUM_UAVS = 5
NUM_OBSTACLES = 50
ENERGY_LIMIT = 100  # Maximum steps each UAV can take


# Generate obstacles (random points in 3D space)
obstacles = set()
while len(obstacles) < NUM_OBSTACLES:
    obs = (random.randint(0, SPACE_SIZE[0] - 1),
           random.randint(0, SPACE_SIZE[1] - 1),
           random.randint(0, SPACE_SIZE[2] - 1))
    obstacles.add(obs)




# Check if a point is valid (inside space and not an obstacle)
def is_valid(point):
    x, y, z = point
    return (0 <= x < SPACE_SIZE[0] and
            0 <= y < SPACE_SIZE[1] and
            0 <= z < SPACE_SIZE[2] and
            point not in obstacles)




# A* algorithm for pathfinding
def a_star(start, goal):
    open_set = []
```

```
    heapq.heappush(open_set, (0, start))
    came_from = {}
    g_score = {start: 0}
    f_score = {start: np.linalg.norm(np.array(start) - np.array(goal))}


    while open_set:
        _, current = heapq.heappop(open_set)


        if current == goal:
            path = []
            while current in came_from:
                path.append(current)
                current = came_from[current]
            return path[::-1]  # Return reversed path


        for dx, dy, dz in [(1, 0, 0), (-1, 0, 0), (0, 1, 0), (0, -1, 0), (0, 0, 1), (0, 0, -1)]:
            neighbor = (current[0] + dx, current[1] + dy, current[2] + dz)
            if not is_valid(neighbor):
                continue


            tentative_g_score = g_score[current] + 1
            if neighbor not in g_score or tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = tentative_g_score + np.linalg.norm(np.array(neighbor) - np.array(goal))
                heapq.heappush(open_set, (f_score[neighbor], neighbor))


    return []  # Return empty if no path found




# UAV class
class UAV:
    def __init__(self, uav_id, start_pos, region):
        self.id = uav_id
        self.position = start_pos
        self.coverage_area = set()
        self.energy = ENERGY_LIMIT
        self.path = []
        self.region = region  # Assigned region

```

```
88
89      def add_coverage(self):
90          x, y, z = self.position
91          coverage_radius = 5  # UAV's sensing range
92          for dx in range(-coverage_radius, coverage_radius + 1):
93              for dy in range(-coverage_radius, coverage_radius + 1):
94                  for dz in range(-coverage_radius, coverage_radius + 1):
95                      point = (x + dx, y + dy, z + dz)
96                      if is_valid(point) and point in self.region:
97                          self.coverage_area.add(point)
98
99
100     def move(self, targets):
101         if self.energy <= 0:
102             return
103
104
105         if not self.path or self.position == self.path[-1]:
106             target = self.find_best_target(targets)
107             self.path = a_star(self.position, target) if target else []
108
109
110         if self.path:
111             self.position = self.path.pop(0)
112             self.add_coverage()
113             self.energy -= 1
114
115
116     def find_best_target(self, targets):
117         min_cost = float('inf')
118         best_target = None
119
120
121         for target in targets:
122             if target not in self.region:
123                 continue  # Only pick targets inside assigned region
124             cost = np.linalg.norm(np.array(self.position) - np.array(
    target))
125             if cost < min_cost:
126                 min_cost = cost
127                 best_target = target
128
129
130         return best_target
131
132
133
134
135 # Generate Voronoi-based regions for UAVs
136 uav_positions = [(50 + random.randint(-20, 20), 50 + random.randint
    (-20, 20), 10) for _ in range(NUM_UAVS)]
```

```
vor = Voronoi([pos[:2] for pos in uav_positions])  # Use 2D projection for Voronoi


# Assign UAVs to their respective regions
regions = [set() for _ in range(NUM_UAVS)]
for x in range(SPACE_SIZE[0]):
    for y in range(SPACE_SIZE[1]):
        distances = [np.linalg.norm(np.array((x, y)) - np.array(vor.points[i])) for i in range(NUM_UAVS)]
        assigned_uav = np.argmin(distances)
        for z in range(SPACE_SIZE[2]):
            if is_valid((x, y, z)):
                regions[assigned_uav].add((x, y, z))


# Generate coverage targets
coverage_targets = [
    (random.randint(0, SPACE_SIZE[0] - 1), random.randint(0, SPACE_SIZE[1] - 1), random.randint(0, SPACE_SIZE[2] - 1))
    for _ in range(30)]
coverage_targets = [t for t in coverage_targets if is_valid(t)]


# Initialize UAVs
uavs = [UAV(i, uav_positions[i], regions[i]) for i in range(NUM_UAVS)]


# Simulation loop
for _ in range(ENERGY_LIMIT):
    for uav in uavs:
        uav.move(coverage_targets)


# Debugging output
for uav in uavs:
    print(f"UAV {uav.id} Final Pos: {uav.position}")
    print(f"Coverage Count: {len(uav.coverage_area)}")
    if uav.coverage_area:
        print(f"Example Points: {list(uav.coverage_area)[:5]}")


# Plot results
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')


# Plot obstacles
obs_x, obs_y, obs_z = zip(*obstacles)
ax.scatter(obs_x, obs_y, obs_z, c='black', marker='o', label="Obstacles")
```

```
184
185
186 # Plot UAV coverage
187 colors = ['red', 'blue', 'green', 'purple', 'orange']
188 for uav in uavs:
189     if len(uav.coverage_area) > 0:
190         area = np.array(list(uav.coverage_area))
191         ax.scatter(area[:, 0], area[:, 1], area[:, 2], c=colors[uav.id
    ], s=10, label=f'UAV {uav.id}', alpha=0.5)
192
193
194 ax.set_xlabel('X')
195 ax.set_ylabel('Y')
196 ax.set_zlabel('Z')
197 ax.set_title('Optimized UAV Coverage in 3D')
198 ax.legend()
199
200
201 plt.show()
```