



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Alan Kunz Cechinel

UAV Architecture for Persistent Environmental Monitoring:
Wind-Informed Sweep Path Planning and Autonomous Landing on Tree Branches

Florianópolis
2026

Alan Kunz Cechinel

UAV Architecture for Persistent Environmental Monitoring:
Wind-Informed Sweep Path Planning and Autonomous Landing on Tree Branches

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Edson Roberto De Pieri, Dr.
Coorientadora: Prof. Patricia Della Múa Plentz, Dr.

Florianópolis
2026

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Kunz Cechinel, Alan
UAV Architecture for Persistent Environmental
Monitoring : Wind-Informed Sweep Path Planning and
Autonomous Landing on Tree Branches / Alan Kunz Cechinel ;
orientador, Edson Roberto de Pieri, coorientadora,
Patricia Della M  a Plentz, 2026.
157 p.

Tese (doutorado) - Universidade Federal de Santa
Catarina, Centro Tecnol  gico, Programa de P  s-Gradua  o em
Engenharia de Automa  o e Sistemas, Florian  polis, 2026.

Inclui refer  ncias.

1. Engenharia de Automa  o e Sistemas. 2. Coverage Path
Planning. 3. Unmanned Aerial Vehicle. 4. Autonomous
Landing. I. de Pieri, Edson Roberto. II. Della M  a Plentz,
Patricia. III. Universidade Federal de Santa Catarina.
Programa de P  s-Gradua  o em Engenharia de Automa  o e
Sistemas. IV. T  tulo.

Alan Kunz Cechinel

UAV Architecture for Persistent Environmental Monitoring:
Wind-Informed Sweep Path Planning and Autonomous Landing on Tree Branches

O presente trabalho em nível de doutorado foi avaliado e aprovado, em 10 de Dezembro de 2025, pela banca examinadora composta pelos seguintes membros:

Prof. Kalinka Castelo Branco, Dr.
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

Prof. André Bittencourt Leal, Dr.
Departamento de Engenharia Elétrica
Universidade do Estado de Santa Catarina

Prof. Ubirajara Franco Moreno, Dr.
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de doutor em Engenharia de Automação e Sistemas.



Documento assinado digitalmente
Julio Elias Normey Rico
Data: 23/01/2026 09:31:36-0300
CPF: ***.840.859-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Julio Elias Normey Rico, Dr.
Coordenador do Programa de Pós-Graduação



Documento assinado digitalmente
Edson Roberto de Pieri
Data: 22/01/2026 19:03:34-0300
CPF: ***.042.498-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Edson Roberto De Pieri, Dr.

Orientador



Documento assinado digitalmente
PATRICIA DELLA MEA PLENTZ
Data: 22/01/2026 19:50:33-0300
CPF: ***.243.700-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Patricia Della Méa Plentz, Dr.
Coorientadora

Universidade Federal de Santa Catarina – UFSC
Florianópolis, 2026.

This work is dedicated to my parents, Silvio and Salete,
for their unwavering support, and to my wife, Valéria, for
her love and companionship throughout this journey.

ACKNOWLEDGEMENTS

First of all, I thank God for blessing me with a wonderful family, especially my parents, Silvio Cechinel and Maria Salete Kunz Cechinel, whose love, support, and perseverance have been my lifelong inspiration and the driving force behind every achievement. I also extend this acknowledgment to my wife, Valéria Barcelos Soares, who came into my life at the beginning of my doctoral journey and has been patient, loving, and supportive through every challenging moment.

I am also grateful to all the professors who have, in some way, contributed to my academic and professional development. In particular, I thank Professor Edson Roberto De Pieri for his guidance throughout this work and for his advice that extended beyond academic matters, and Professor Patrícia Della Méa Plentz for her valuable contributions to scientific papers, her support in obtaining the grant for the sandwich doctorate, and our shared efforts in entrepreneurship. I also express my gratitude to Professor Juha Rönning, who served as my advisor during the sandwich doctorate period, for his insightful guidance and support.

Furthermore, I would like to acknowledge my colleagues from the Department of Automation and Systems for their intellectual support and the many enjoyable moments we shared, which made this journey lighter and more collaborative.

Finally, I would like to acknowledge CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) for their financial support, both for my Ph.D. scholarship and for the sandwich doctorate internship.

*Every researcher walks through the maze of the unknown,
guided only by a faint light that grows brighter with each discovery.
Through resilience and curiosity, we carve our own path to the maze's end.*

RESUMO

Veículos Aéreos Não Tripulados (VANTs) têm se mostrado uma alternativa valiosa ao monitoramento ambiental realizado por humanos, devido à sua flexibilidade e à capacidade de cobrir áreas amplas, remotas ou perigosas. No entanto, a implantação eficaz desses veículos permanece um desafio complexo, exigindo estratégias coordenadas para garantir cobertura eficiente da área, otimização energética e operação autônoma de longo prazo. O monitoramento de áreas requer uma sequência de pontos de passagem (waypoints) que o VANT deve visitar para cobrir completamente a Área de Interesse (AI). Encontrar este caminho é conhecido como o problema de Planejamento de Caminho de Cobertura (CPP), classificado como NP-Difícil em termos de complexidade computacional. Um único VANT geralmente não consegue cobrir grandes áreas sem recarga ou substituição da bateria. Nesses casos, múltiplos VANTs podem ser utilizados para dividir a tarefa. No entanto, isso introduz complexidade adicional, pois é necessário particionar a AI de maneira equilibrada entre os veículos, ao mesmo tempo que se otimizam os trajetos individuais. Além disso, mesmo com vários VANTs, o monitoramento de longo prazo pode exigir que drones permaneçam estacionários em determinados momentos, funcionando temporariamente como sensores estáticos. Uma revisão sistemática da literatura revelou que nenhum dos estudos analisados considerou os efeitos do vento no planejamento, apesar de seu impacto conhecido sobre a estabilidade de voo, o consumo de energia e a confiabilidade da missão. Essa omissão representa uma lacuna significativa e uma oportunidade para aprimorar o realismo e a robustez nas estratégias de planejamento de trajetórias. Esta tese propõe uma arquitetura na qual uma frota de VANTs de longo alcance, com alta capacidade de carga, é utilizada para realizar missões de vigilância em larga escala sobre regiões de interesse como florestas, pomares e plantações comerciais. Esses VANTs de longo alcance são capazes de lançar drones menores para o monitoramento sob a copa das árvores. Esta arquitetura foi parcialmente alcançada nesta dissertação. Primeiro, foi desenvolvido um método leve para CPP com múltiplos VANTs, considerando explicitamente os consumos energéticos adicionais causados pelo vento no voo. O método, denominado Planejamento de Varredura Considerando o Vento (WISP), utiliza um variante de K-Means com restrições para garantir uma divisão justa da AI e aplica uma Árvore Geradora Mínima (MST) para encontrar os caminhos de cobertura em cada sub-região. Em um segundo estudo, foi desenvolvido um sistema de pouso autônomo em galhos de árvores. A abordagem emprega inteligência artificial para detectar galhos e aplica princípios de visão estéreo, utilizando apenas uma câmera monocular, permitindo que o VANT pouse e atue como sensor estático, reduzindo o consumo de energia. Os resultados demonstraram que o WISP superou os métodos K-Nearest Neighbors (KNN), Simulated Annealing (SA) e abordagens gananciosas (Greedy) em critérios como tempo de computação, uso de memória e consumo de energia no caminho. Já o sistema de pouso autônomo obteve sucesso em cenários controlados. No entanto, como o WISP foi avaliado apenas em simulação, estudos futuros devem validar sua eficácia em condições reais.

Palavras-chave: Planejamento de Caminho de Cobertura, Múltiplos Veículos Aéreos Não Tripulados, Pouso Autônomo, Robôs Móveis.

RESUMO EXPANDIDO

INTRODUÇÃO

Nos últimos anos, os veículos aéreos não tripulados (VANTs) têm se tornado ferramentas estratégicas em aplicações que exigem sensoriamento remoto, impulsionados pela miniaturização de componentes e pelo avanço de sistemas embarcados. No entanto, ainda enfrentam limitações significativas quanto à autonomia de voo, especialmente em missões prolongadas ou realizadas em ambientes adversos. Além disso, fatores ambientais, como o vento, são frequentemente negligenciados no planejamento de trajetórias, o que impacta negativamente o consumo energético e a confiabilidade das missões.

OBJETIVOS

O objetivo geral desta tese é projetar, implementar e validar uma arquitetura hierárquica baseada em VANTs multirrotores voltada ao monitoramento ambiental escalável, energeticamente eficiente e persistente, por meio de planejamento de trajetória que considere os efeitos do vento no consumo energético e o pouso autônomo em estruturas naturais.

Esta tese concentra-se em dois componentes principais da arquitetura proposta, levando às seguintes hipóteses:

- *A incorporação da direção, magnitude e variação do vento, junto com a variação de altitude do VANT, no planejamento de trajetórias de VANTs reduz o consumo energético estimado em comparação com métodos tradicionais que desconsideram os efeitos do vento.*
- *Um sistema baseado em visão monocular, utilizando visão estéreo por movimento e um mecanismo de gancho, permite pousos autônomos confiáveis e repetíveis de VANTs em estruturas naturais.*

Com base no objetivo geral e nessas duas hipóteses, foram definidos os seguintes objetivos específicos:

1. Projetar uma arquitetura hierárquica de sistema multi-VANT para monitoramento coordenado de longo alcance e sob a copa das árvores.
2. Estender modelos existentes de consumo energético de VANTs para considerar forças de arrasto verticais e horizontais.
3. Desenvolver um plugin para o simulador Gazebo utilizando o modelo estendido para estimar o consumo energético instantâneo.
4. Implementar um método de alocação de áreas entre os VANTs visando o balanceamento da carga de trabalho.
5. Desenvolver um método de planejamento de trajetória de cobertura baseado em Árvores Geradoras Mínimas (MST), com o objetivo de minimizar o consumo energético considerando os efeitos do vento.

6. Integrar o sistema ao ecossistema PX4+ROS+Gazebo para simulações reprodutíveis e controle de missão em tempo real.
7. Desenvolver um sistema de pouso autônomo para VANTs em estruturas naturais, utilizando visão monocular e um mecanismo de gancho.

METODOLOGIA

Este trabalho teve início com um levantamento do estado da arte em Planejamento de Caminho de Cobertura (CPP) para sistemas multirrobôs, com foco especial na análise de como fatores ambientais, como o vento, têm sido considerados. Esse levantamento foi realizado por meio de uma Revisão Sistemática da Literatura (RSL), a qual confirmou que os efeitos do vento são subestimados nas estratégias existentes de CPP para Veículos Aéreos Não Tripulados (VANTs), revelando uma lacuna clara na pesquisa.

Paralelamente, pesquisas anteriores conduzidas pelo *Biomimetics and Intelligent Systems Group* (BISG), da Universidade de Oulu, identificaram a necessidade de um protótipo de prova de conceito de um sistema de pouso autônomo capaz de operar sobre estruturas naturais. Um sistema com essas características seria particularmente útil em aplicações florestais, permitindo, por exemplo, que bombeiros implantem sensores temporários sob as copas das árvores para monitorar o avanço de incêndios florestais.

Com base na lacuna identificada pela RSL e nas exigências práticas levantadas pelo grupo BISG, foi proposta uma Arquitetura Hierárquica Multi-VANT para Monitoramento Ambiental Contínuo e de Longa Duração. Essa arquitetura descreve como VANTs de grande porte, com longo alcance e alta capacidade de carga, podem lançar VANTs menores para executar tarefas de monitoramento localizadas e prolongadas em ambientes complexos.

Os VANTs de maior porte empregam um planejador de cobertura denominado Planejamento de Varredura Considerando o Vento (WISP), que otimiza as rotas de voo considerando os efeitos do vento sobre o consumo energético. Enquanto isso, os VANTs menores, que operam sob as copas das árvores, são equipados com um sistema de pouso autônomo baseado em visão computacional, o qual permite pousar em estruturas naturais (como galhos de árvores), viabilizando um monitoramento persistente e energeticamente eficiente.

Essas soluções foram implementadas e validadas por meio de uma combinação de experimentos em simulação e testes em ambiente real controlado. O WISP foi avaliado por meio de simulações que analisaram seu desempenho sob diferentes condições de vento e em diversos cenários de áreas de interesse. Simultaneamente, o sistema de pouso autônomo foi integrado e testado em um VANT real, a fim de demonstrar sua viabilidade para pousos sobre estruturas naturais.

RESULTADOS E DISCUSSÕES

Os resultados demonstraram que o planejador WISP supera abordagens tradicionais tanto em consumo energético quanto em tempo de solução, especialmente em cenários com influência do vento. O uso do algoritmo K-Means com restrições proporcionou uma divisão mais equilibrada das áreas de cobertura entre os VANTs. No sistema de pouso, os testes em ambiente controlado indicaram uma taxa de sucesso de 95%, comprovando a viabilidade do pouso autônomo com mecanismo de gancho em galhos horizontais. Embora promissor, o sistema ainda demanda aprimoramentos para operação em ambientes reais, como florestas, exigindo maior capacidade de localização e resistência a condições variáveis.

CONSIDERAÇÕES FINAIS

Este trabalho contribui com soluções práticas e inovadoras para o monitoramento ambiental de longa duração por VANTs. O planejador WISP e o sistema de pouso representam avanços importantes rumo à autonomia energética e operacional dessas plataformas. No entanto, desafios permanecem, como a necessidade de um sistema de coordenação online e robusto entre os VANTs e a validação em campo das soluções propostas. Trabalhos futuros devem concentrar-se na integração completa da arquitetura, na otimização dos algoritmos para hardware embarcado e em testes em ambientes naturais reais.

Palavras-chave: VANTs. Planejamento de cobertura. Pouso autônomo. Monitoramento ambiental. Consumo energético.

ABSTRACT

Unmanned Aerial Vehicles (UAVs) have emerged as a valuable alternative to human-based environmental monitoring due to their flexibility and ability to cover large, remote, or hazardous areas. However, the effective deployment of UAVs remains a complex challenge, requiring coordinated strategies to ensure efficient area coverage, energy optimization, and long-term autonomous operation. Area monitoring requires a sequence of waypoints that the UAV must visit to fully cover the Area of Interest (AOI). Computing such a path is known as the Coverage Path Planning (CPP) problem, which is NP-Hard in terms of computational complexity. A single UAV is often unable to cover large areas without recharging or replacing its battery. In such cases, multiple UAVs can be deployed to share the task. However, this introduces additional complexity, as it becomes necessary to partition the AOI among the UAVs in a balanced manner while also optimizing the individual coverage paths. Moreover, even with multiple UAVs, long-term monitoring may still be required, where drones might remain stationary at times and temporarily function as static sensors. A systematic review of the literature revealed that wind effects on CPP have been neglected in the reviewed studies, despite their well-documented impact on flight stability, energy consumption, and overall mission reliability. This omission highlights a significant gap and presents an opportunity to enhance the realism and robustness of UAV path planning strategies. This thesis proposes an architecture in which a fleet of long-range UAVs, equipped with large payload capacities, is deployed to perform wide-area surveillance missions over regions of interest such as forests, orchard fields, and commercial timber plantations. These high-endurance UAVs are able to deploy smaller UAVs for long-term monitoring tasks beneath the canopy layer. Together, these components support efficient and persistent monitoring in challenging environments. This objective is partially attained in this thesis. First, a lightweight method for multi-UAV CPP was developed, explicitly considering the additional energy consumption caused by wind. The method, named Wind-Informed Sweep Planner (WISP), builds on K-Means Constrained clustering to ensure a fair division of the AOI, and uses a Minimum Spanning Tree (MST) to compute the coverage path within each subdivided region. In a second study, a system for autonomous landing on tree branches was developed. This approach uses Artificial Intelligence (AI) to detect branches and applies stereo vision principles, using only a monocular camera, to enable landings. When landed, the UAV can operate as a static sensor while significantly reducing power consumption. The results showed that WISP outperformed K-Nearest Neighbors (KNN), Simulated Annealing (SA), and the Greedy approaches across all evaluated criteria: computation time, memory usage, and energy consumption during path execution. Additionally, the autonomous landing system successfully performed landings in controlled scenarios. These findings indicate that the proposed approach makes a solid contribution to multi-UAV monitoring. However, since WISP was only evaluated in simulation, future work should focus on validating whether the generated routes reduce energy consumption in real-world conditions.

Keywords: Coverage Path Planning, Multiple Unmanned Aerial Vehicles, Autonomous Landing, Mobile Robots.

LIST OF FIGURES

Figure 1 – Scheme for autonomous Mobile Robot (MR)s navigation.	25
Figure 2 – Examples of land-based autonomous MRs for agricultural applications.	29
Figure 3 – Examples of aerial-based autonomous MRs for agricultural applications.	30
Figure 4 – NAURU 500C ISR from XMobots	31
Figure 5 – Landmark-based Simultaneous Localization and Mapping (SLAM).	36
Figure 6 – Grid map.	36
Figure 7 – Path Planning Categories.	40
Figure 8 – Classical navigation pipeline.	42
Figure 9 – Example of planned and executed paths.	42
Figure 10 – Trapezoidal cellular decomposition in coverage path planning.	44
Figure 11 – Diferent AOI types in polygonal space.	46
Figure 12 – Simple coverage paterns.	46
Figure 13 – Paterns for search and rescue coverage.	47
Figure 14 – Typical approach for CPP.	48
Figure 15 – Decomposition methods.	48
Figure 16 – Typical approach for CPP using grid decomposition.	50
Figure 17 – CPP methods for grid.	50
Figure 18 – CPP sub-problems.	52
Figure 19 – Hierarchical UAV-based environmental monitoring system overview.	72
Figure 20 – WISP’s steps overview.	75
Figure 21 – Compass rose showing wind directions and an example of relative wind angle θ_R	76
Figure 22 – Heuristic forces and angles demonstration.	77
Figure 23 – Walls created to compute the coverage path.	82
Figure 24 – Diagram of interconnections among systems.	84
Figure 25 – UAV’s views.	86
Figure 26 – System’s architecture. Green arrows represent Visual Inertial Odometry data transference. Orange arrows represent image data transference. The blue arrow represents command sending.	88
Figure 27 – Example of training data with class annotations. 0 represents a branch, and 1 represents an aluminum frame.	90
Figure 28 – Graphs illustrating box loss, classification loss, detection fidelity loss, precision, recall, and Mean Average Precision (mAP) across training epochs for both the training and validation sets.	91
Figure 29 – A simplified example of the geometry behind the LSES.	93
Figure 30 – Possible starting points.	94

Figure 31 – Frontal view of the Holybro X500 UAV.	98
Figure 32 – Top view of the Holybro X500 UAV.	98
Figure 33 – Theoretical results from the conceptual validation, illustrating the average power consumption of the UAV during level flight.	99
Figure 34 – Experimental results from the conceptual validation, illustrating the average power consumption of the UAV during level flight.	100
Figure 35 – Theoretical results from the conceptual validation, illustrating the average power consumption of the UAV during descending flight.	100
Figure 36 – Experimental results from the conceptual validation, illustrating the average power consumption of the UAV during descending flight.	100
Figure 37 – Theoretical results from the conceptual validation, illustrating the average power consumption of the UAV during climbing flight.	101
Figure 38 – Experimental results from the conceptual validation, illustrating the average power consumption of the UAV during climbing flight.	101
Figure 39 – Mean computation time measured across different grid sizes.	103
Figure 40 – Mean memory consumption measured across different grid sizes.	103
Figure 41 – Example of the WISP method across all environments and wind directions.	105
Figure 42 – Example of the KNN-Energy method across all environments and wind directions.	106
Figure 43 – Example of the KNN-Distance method across all environments and wind directions.	107
Figure 44 – Example of the Greedy method across all environments and wind directions.	108
Figure 45 – Example of the Greedy+SA method across all environments and wind directions.	109
Figure 46 – Example of the SA method across all environments and wind directions.	110
Figure 47 – Mean total energy cost per environment and method (kJ). Each cell reports the mean total cost (kJ) $\pm$ Standard Deviation (STD).	111
Figure 48 – Mean number of turns per environment and method.	112
Figure 49 – Mean energy saved (%) per method relative to the worst-performing method.	112
Figure 50 – Mean energy saved (%) per method excluding SA and KNN-distance.	113
Figure 51 – Mean number of battery swaps per environment and method.	114
Figure 52 – Energy Variation Coefficient for each environment by number of UAVs and method. Where: Km - K-Means, KMC - K-Means Constrained, and MBKM - Mini Batch K-Means	115

Figure 53 – Mean Time to find a solution for each environment by number of UAVs and method. Where: Km - K-Means, KMC - K-Means Constrained, and MBKM - Mini Batch K-Means	115
Figure 54 – Environment used for trajectory validation. Darker blue indicates a height of 280 m, while lighter blue corresponds to 30 m.	116
Figure 55 – Planned vs Actual energy by method.	117
Figure 56 – Planned vs Actual time by method.	118
Figure 57 – Planned vs Actual distance by method.	118
Figure 58 – Planned and executed trajectories for each method in simulation. . .	119
Figure 59 – Current consumption by scenario. (Kunz Cechinel. <i>et al.</i> , 2023) . . .	120
Figure 60 – Examples of roll and yaw variations within the experiment. (Kunz Cechinel. <i>et al.</i> , 2023)	121
Figure 61 – Experimental environment. The takeoff region is marked with a blue square. (Kunz Cechinel. <i>et al.</i> , 2023)	124
Figure 62 – Route in X and Y axes for the best result. (Kunz Cechinel. <i>et al.</i> , 2023)	127
Figure 63 – Route in Y and Z axes for the best result. (Kunz Cechinel. <i>et al.</i> , 2023)	127
Figure 64 – Route in X and Y axes for the worse result. (Kunz Cechinel. <i>et al.</i> , 2023)	128
Figure 65 – Route in Y and Z axes for the worse result. (Kunz Cechinel. <i>et al.</i> , 2023)	128
Figure 66 – Experimental environment for YOLO experiments.	129
Figure 67 – Route in X and Y axes for the best result.	130
Figure 68 – Route in Y and Z axes for the best result.	130
Figure 69 – Route in X and Y axes for the worse result.	131
Figure 70 – Route in Y and Z axes for the worse result.	131

LIST OF TABLES

Table 2 – Evaluation of some UAV platforms.	30
Table 3 – Global and local path planning.	41
Table 4 – General information summary of the works found.	53
Table 5 – Details summary of the works found.	54
Table 6 – Real-world landing site center location relative to the UAV’s camera. (Kunz Cechinel. <i>et al.</i> , 2023)	122
Table 7 – Estimated landing site center location relative to the UAV’s camera in the second image. (Kunz Cechinel. <i>et al.</i> , 2023)	122
Table 8 – Position accuracy and angle error to the landing site center location relative to the UAV’s camera. (Kunz Cechinel. <i>et al.</i> , 2023)	123
Table 9 – Error between actual and estimated landing points. Values in red in- dicate that the UAV landed farther from the expected position. (Kunz Cechinel. <i>et al.</i> , 2023)	125
Table 10 – Error between actual and estimated landing points.	132

LIST OF ABBREVIATIONS AND ACRONYMS

AC	Ant Colony
AI	Artificial Intelligence
ALS	Autonomous Landing System
AMSL	Above Mean Sea Level
ANN	Artificial Neural Network
ANT	Artificial Neural Tissue
AOI	Area of Interest
ASL	Above Sea Level
BiCC	Bipartite Cooperative Coevolution
BISG	Biomimetics and Intelligent Systems Group
CPP	Coverage Path Planning
CV	Coefficient of Variation
DDQN	Deep Double Q-learning Networks
Dec-PPCPP	Decentralized Predator-Prey-based Approach to Adaptive Coverage Path Planning
DGPS	Differential GPS
DPAR	Dynamic Predator Avoidance Reward
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
DWA	Dynamic Window Approach
EKF	Extended Kalman Filter
GA	Genetic Algorithm
GPS	Global Positioning System
ILP	Integer Linear Programming
IMU	Inertial Measurement Unit
IoU	Intersection over Union
IR	Infrared
KF	Kalman Filter
KNN	K-Nearest Neighbors
LKH	Lin-Kernighan Heuristic
LSES	Landing Site Estimation System
mAP	Mean Average Precision
MCL	Monte Carlo Localization
MCP	Multiple Coverage Path Planning
MHT	Multiple Hypothesis Tracking
MILP	Mixed Integer Linear Programming
MOVNS	Multiobjective Variable Neighborhood Search

MR	Mobile Robot
MST	Minimum Spanning Tree
MTSP	Multiple Traveling Salesman Problem
NFZ	No-Flight Zone
PF	Particle Filter
PI	Proportional-Integral
RA-MCPP	Reliability-Aware Multi-Agent Coverage Path Planning
RFID	Radio Frequency Identification
RL	Reinforcement Learning
RMTT	RoboMaster TT
ROS	Robot Operating System
RSH	Randomized Search Heuristic
RTK	Real-Time Kinematic
RW	Random Walk
RWD	Relative Wind Direction
SA	Simulated Annealing
SCoPP	Scalable Coverage Path Planning
SF	Shadow Following
SITL	Software In The Loop
SLAM	Simultaneous Localization and Mapping
SPAR	Stationary Predator Avoidance Reward
SR	Systematic Review
ST	Spanning Tree
STD	Standard Deviation
SV	Stereo Vision
SVM	Support Vector Machine
ToF	Time-of-Flight
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VIO	Visual Inertial Odometry
VRP	Vehicle Routing Problem
VTOL	Vertical Take-Off and Landing
WISP	Wind-Informed Sweep Planner

CONTENTS

1	INTRODUCTION	20
1.1	OBJECTIVES	22
1.2	METHODOLOGY OVERVIEW	23
1.3	CONTRIBUTIONS	23
1.4	THESIS STRUCTURE	24
2	AUTONOMOUS MOBILE ROBOTICS	25
2.1	INTRODUCTION	25
2.2	LOCOMOTION	26
2.3	PERCEPTION AND ENVIRONMENTAL REPRESENTATION	31
2.4	LOCALIZATION	37
2.5	PATH PLANNING	39
2.6	CONCLUSION	44
3	COVERAGE PATH PLANNING	45
3.1	INTRODUCTION	45
3.2	SURVEY ON MULTI-UAV COVERAGE PATH PLANNING	51
3.3	CONCLUSION	67
4	PROPOSED APPROACH	71
4.1	AREA MONITORING APPROACH OVERVIEW	71
4.2	WIND-INFORMED SWEEP PLANNER	74
4.2.1	Energy Heuristic	75
4.2.2	Discretization and Area Allocation	79
4.2.3	Coverage Path Planning	80
4.2.4	Integration with ROS and PX4 Environment	82
4.3	AUTONOMOUS LANDING SYSTEM	85
4.3.1	UAV preparation	86
4.3.2	Architecture	87
4.3.3	Controllers	88
4.3.4	YOLOv8 Module	89
4.3.5	Vision Module	91
4.3.6	Landing Site Estimation System	92
4.3.7	Landing Pipeline	94
4.4	CONCLUSION	95
5	RESULTS AND DISCUSSION	97
5.1	EXPERIMENTS WITH THE WIND-INFORMED SWEEP PLANNER	97
5.1.1	Unmanned Aerial Vehicle Model Used in the Experiments	97
5.1.2	Heuristic Conceptual Validation	98
5.1.3	Coverage Path Planning Algorithm Comparison	101

5.1.4	Comparative Analysis of K-Means Variants for Area Allocation .	114
5.1.5	Simulation-Based Validation of Theoretical Trajectories	116
5.2	EXPERIMENTS FOR THE AUTONOMOUS LANDING SYSTEM . . .	120
5.2.1	Battery Consumption	120
5.2.2	Experiment in Controlled Scenario	120
5.2.3	Experiment in Flight – Landing Site Selection by User	123
5.2.4	Experiment in Flight – Landing Site Selection by YOLOv8	129
5.3	CONCLUSION	132
6	CONCLUSION	134
6.1	LIMITATIONS AND FUTURE WORK	135
6.2	LITERATURE PUBLICATIONS	137
	REFERENCES	138
	APPENDIX A – SYSTEMATIC REVIEW PROTOCOL	156

1 INTRODUCTION

Recent advances in the miniaturization of onboard computers, sensors, actuators, and communication modules have significantly improved the robustness of Unmanned Aerial Vehicles (UAVs), enabling their integration into a wide range of applications. Laghari *et al.* (2023) highlight several potential uses, including photography, shipping and delivery, natural scenery mapping, disaster management, agricultural monitoring, wildlife monitoring, military operations, among others.

According to Mohsan *et al.* (2023), UAVs are useful in various military, industrial, and commercial applications, particularly in monitoring-related activities. They help reduce the labor and time required for ground-based surveying, inspection, and sampling, while also enabling collision avoidance and rapid convergence to targets. This allows personnel to dedicate more time to planning, analysis, and decision-making.

Despite these advantages, Bueren *et al.* (2015), Jeongeun Kim *et al.* (2019) and Mohsan *et al.* (2023) emphasize several technical constraints, including limited battery efficiency, short flight durations, restricted communication range, and reduced payload capacity. These constraints directly influence the selection and performance of onboard sensors. As observed by Tsouros, Bibi, and Sarigiannidis (2019), the sensors used in UAVs must be compact, lightweight, and energy-efficient. Furthermore, Radoglou-Grammatikis *et al.* (2020a) notes that such limitations affect the scalability of applications. For example, in the case of large agricultural fields, a single UAV may be insufficient to cover the entire area due to limited energy availability.

Typically, UAVs performing monitoring tasks must cover an Area of Interest (AOI), which requires solving the Coverage Path Planning (CPP) problem. According to Choset (2001), the CPP is a subtopic of motion planning in robotics, that involves constructing a path for a robot to traverse the entire environment. This path involves a representation of the environment, which might be a graph, cells in a grid, or a simple polygon. However, as mentioned earlier, more than a single robot might be needed to complete a coverage task. In such cases, multiple robots might solve the problem. However, instead of defining a single path, in the Multiple Coverage Path Planning (MCP) problem, it is necessary to divide the AOI between the robots and define their paths inside each AOI. Although simple to define, Rekleitis *et al.* (2008) demonstrated that this problem is at least NP-HARD.

The Systematic Review (SR) presented in Chapter 3 investigates the MCP problem. The reviewed approaches include complex Mixed Integer Linear Programming (MILP) models solved using mathematical optimization tools, techniques based on Artificial Intelligence (AI), clustering strategies, and custom algorithms specifically developed by the respective authors. In algorithm-based approaches, a common strategy involves dividing the area among the UAVs using clustering, optimization, or greedy

methods, followed by computing the coverage path within each assigned region. In contrast, AI-driven approaches typically involve training agents using a reward function designed to maximize coverage. Each agent independently defines the area it will cover and plans its path while sharing information with others. A common drawback of AI-based approaches is the repeated coverage of the same areas, while for MILP models, the primary limitation is the computational time required to find a solution, even for relatively small problem instances. As a result, alternative approaches tend to be preferred due to their greater scalability and efficiency.

The systematic review also investigated whether wind effects were considered in path planning. Surprisingly, none of the reviewed works addressed this factor, despite its well-known influence on flight stability, energy consumption, and overall mission reliability. This reveals a relevant gap in the current literature and suggests an opportunity for future research to enhance the realism and robustness of UAV operations.

Besides monitoring, autonomous landing is a critical capability with a wide range of applications, including package delivery (Feng, K. *et al.*, 2020), automated battery charging (Junaid *et al.*, 2017), and cooperative tasks in search and rescue missions (Persson; Wahlberg, 2019). Research in this area is typically categorized into two main domains: landing on static platforms (Xuan-Mung *et al.*, 2020; Junaid *et al.*, 2017; Gier-nacki *et al.*, 2017), and landing on moving platforms (Gautam *et al.*, 2022; Rodriguez-Ramos *et al.*, 2019; Feng, Y. *et al.*, 2018; Borowczyk *et al.*, 2017).

In both static and dynamic landing scenarios, UAVs typically complete their descent by making contact with a platform using their landing gear. Alternatively, recent research has investigated nature-inspired perching strategies, enabling UAVs to emulate birds by grasping onto tree branches (Kirchgeorg; Mintchev, 2022), power lines (Kitchen *et al.*, 2020), or architectural features such as edges and corners of buildings (Hang *et al.*, 2019). As far as the authors are aware, this work introduces a novel landing strategy involving a hook-based mechanism that allows a UAV to suspend itself from a tree branch. This introduces a new subproblem within the domain of autonomous landing: the autonomous hooking maneuver. The significance of this problem lies in its potential to enable long-duration environmental monitoring while drastically reducing battery consumption. A UAV suspended from a branch, power line, or pole at a suitable height can collect environmental data passively, avoiding the high energy demands associated with sustained flight.

Despite significant advancements in UAV technology, they still face limitations in environmental monitoring especially regarding endurance. Current CPP approaches often overlook the wind effects on energy consumption, leading to suboptimal energy efficiency and reduced mission reliability. Likewise, while several autonomous landing techniques have been explored, few address the challenges of energy-constrained, long-term monitoring in natural environments. These gaps highlight the need for in-

egrated solutions that couple wind-aware path planning with landing strategies for long-term monitoring. Motivated by these challenges, this work proposes a hierarchical multi-UAV architecture that integrates wind-informed coverage planning with a novel hook-based autonomous landing system.

1.1 OBJECTIVES

The general objective of this thesis is to design, implement, and validate a hierarchical multicopter-based UAV architecture that enables scalable, energy-efficient, and persistent environmental monitoring through wind-aware path planning and autonomous landing on natural structures.

This thesis focuses on two key components of the proposed architecture, leading to the following hypotheses:

- *Incorporating wind direction, magnitude, and altitude variation into UAV CPP reduces estimated energy consumption compared to traditional wind-agnostic planning methods.*
- *A monocular vision-based system using stereo-from-motion and a hook-based mechanism enables reliable and repeatable autonomous landing of UAVs on natural structures.*

Based on the general objective and these two hypotheses, the following specific objectives were defined:

1. Design a hierarchical multi-UAV system architecture for coordinated long-range and under-canopy monitoring.
2. Extend existing UAV energy consumption models to incorporate both vertical and horizontal drag forces.
3. Develop a Gazebo simulator plugin using the extended model to estimate instantaneous energy consumption.
4. Implement an area allocation strategy among UAVs aimed at achieving workload balance.
5. Develop a CPP method based on the Minimum Spanning Tree (MST) to minimize energy consumption considering wind.
6. Integrate the system with the PX4+ROS+Gazebo ecosystem for reproducible simulations and real-time mission control.
7. Develop an autonomous landing system that allows UAVs to land on natural structures using monocular vision and a hook-based mechanism.

1.2 METHODOLOGY OVERVIEW

This work began by surveying the state of the art in CPP for multi-robot systems, with a particular focus on whether environmental factors such as wind were being addressed. This was achieved through a SR, which confirmed that wind effects are underestimated in existing CPP strategies for UAVs, revealing a clear research gap.

In parallel, previous research conducted by the Biomimetics and Intelligent Systems Group (BISG) at the University of Oulu identified the need for a proof of concept for an autonomous landing system capable of operating on natural structures. Such a system would be particularly useful in forestry applications. For example, enabling firefighters to deploy temporary sensors beneath tree canopies to monitor fire spread during wildfires.

Based on the gap identified in the SR and the application-driven requirements from the BISG group, a hierarchical multi-robot architecture was proposed for continuous, long-term environmental monitoring. This architecture describes how large UAVs, equipped with extended range and high payload capacity, can deploy smaller UAVs to perform localized, long-duration monitoring tasks in complex environments.

The larger UAVs benefit from a wind-aware coverage path planner, referred to in this work as Wind-Informed Sweep Planner (WISP), which optimizes flight paths by accounting for both energy consumption and environmental conditions. The proposed method discretizes the AOI using multiple grid rotations and applies a clustering strategy to partition each discretization into a set of grid cells assigned to each UAV. For each configuration, a coverage path is computed within each cluster, and the solution with the lowest estimated energy consumption is selected. Meanwhile, the smaller under-canopy UAVs are equipped with a vision-based autonomous landing system that enables them to land on natural structures (e.g., tree branches) for energy-efficient, persistent monitoring.

These solutions were implemented and validated through a combination of simulation-based and real-world experiments. The WISP was evaluated in simulation to assess its performance under varying wind conditions and AOI scenarios. Meanwhile, the vision-based autonomous landing system was integrated and tested on a real UAV platform to demonstrate its feasibility for landing on natural structures.

1.3 CONTRIBUTIONS

The main contributions achieved while pursuing the specific objectives of this research are summarized as follows:

- Expanded the UAV energy model proposed by Sorbelli *et al.* (2021) to account for aerodynamic drag not only in horizontal motion but also in vertical maneuvers, which was not previously treated. This enhancement enables

more accurate energy consumption estimates by considering the influence of wind during climbing and descending flights.

- Implemented a plugin for the Gazebo simulator to model and simulate UAV energy consumption. Once released, this tool is expected to foster further research on energy-aware simulations and support the evaluation of energy-efficient flight strategies within Gazebo-based environments.
- Designed and implemented the *Wind-Informed Sweep Planner*, which uses the extended energy model to build a graph representation of the environment and computes a MST to generate paths that minimize energy consumption. The planner is computationally lightweight, addresses a gap identified in the literature, and has been integrated with the PX4 autopilot, supporting potential deployment on real UAV platforms.
- Developed a novel method for autonomous landing on tree branches using a simple hook-based mechanical system. A YOLO-based detector automatically identifies suitable branches, while a stereo-from-motion technique estimates their depth and orientation. The system was integrated and validated on a real UAV, enabling vision-guided landings for energy-efficient and persistent aerial operations.

1.4 THESIS STRUCTURE

The remainder of this thesis is organized into five chapters, each addressing a key aspect of the proposed approach, from foundations to experimental validation.

Chapter 2 introduces the core components of Mobile Robot (MR) navigation, including locomotion, perception, localization, and path planning, with an emphasis on both aerial and ground platforms.

Chapter 3 presents the Coverage Path Planning (CPP) problem and its extension to the Multi-Unmanned Aerial Vehicle (UAV) setting. A Systematic Review (SR) of relevant literature highlights the main approaches, challenges, and research gaps in Multiple Coverage Path Planning (MCP).

Chapter 4 describes the proposed UAV-based monitoring framework, including a hierarchical system architecture, the Wind-Informed Sweep Planner (WISP) planner, and an autonomous landing system based on monocular vision and a hook mechanism.

Chapter 5 outlines the experimental validation of the proposed methods, including simulation and field tests. These experiments evaluate energy efficiency, workload balance, and autonomous landing performance under various environmental conditions.

Finally, Chapter 6 presents the conclusions, discusses limitations found, and proposes directions for future research.

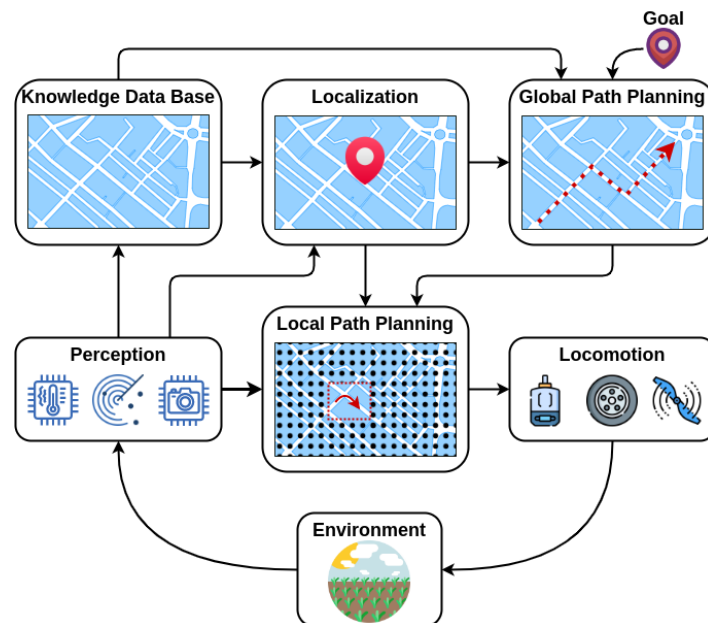
2 AUTONOMOUS MOBILE ROBOTICS

This chapter presents the core components of Mobile Robot (MR) navigation: locomotion, perception, localization, and path planning. It introduces key concepts, classical techniques, common applications, and the advantages and limitations of each topic. Both aerial and ground MRs are explored throughout the discussion.

2.1 INTRODUCTION

Walking through a room is a trivial task for humans, involving involuntary actions such as localization, obstacle avoidance, path selection, and speed regulation. Replicating these abilities is a significant challenge for MRs, requiring multiple mechanisms to navigate safely and effectively in the environment. Figure 1 illustrates a general architecture for autonomous MR navigation, highlighting its core components.

Figure 1 – Scheme for autonomous MRs navigation.



The robot may have prior knowledge of the environment before starting navigation. This information could have been collected earlier, though it may be incomplete. In such cases, the robot can enhance its understanding using perception during navigation. Typically, it is assumed that a map is already available. Navigation begins when the robot receives a goal or target position. Based on its knowledge of the environment (e.g., a map) and its current localization, the global path planner computes an optimal route to the destination, usually defined as a sequence of waypoints.

The local path planner then uses real-time perception and localization data to pursue these waypoints. Unlike the global planner, it has a limited view of the environment, constrained by the robot's sensor range and field of view. Using this local information,

the planner generates control commands for the locomotion system, enabling the robot to move safely and effectively.

2.2 LOCOMOTION

Mobile robots are characterized by their ability to navigate through the environments in which they operate. To achieve this, a MR uses its locomotion actuators to apply forces to the environment, which, according to Newton's third law, results in movement. Designing these mechanisms to suit the robot's operating medium (e.g., space, air, ground, or water) is essential but not sufficient for successful task execution. As noted by Rubio, Valero, and Llopis-Albert (2019), a MR's locomotion system must also meet several technical criteria, including maneuverability, controllability, terrain adaptability, efficiency, and stability.

Depending on its locomotion capabilities, a MR may walk, skate, slide, crawl, surf, dive, or fly. Based on these strategies, mobile robots are typically classified as land-based, aerial, or aquatic. This section focuses on land-based and aerial locomotion systems.

According to Siegwart, Nourbakhsh, and Scaramuzza (2011), land-based MRs primarily rely on locomotion mechanisms such as legs, wheels, tracks, or hybrid combinations of these. In the case of air-based MRs, Alvarenga *et al.* (2015) state that they typically adopt either fixed-wing or rotary-wing configurations. The following discussion presents these locomotion systems, addressing their design challenges, advantages and disadvantages, and their capabilities with respect to terrain traversability.

Alatise and Hancke (2020) point out that wheels have been the most widely used locomotion mechanism in MRs and vehicles. This choice offers a combination of simplicity and efficiency, particularly for flat, rigid, and non-rugged terrains. However, wheels are less suitable for navigating stony, uneven, or soft surfaces compared to other locomotion systems. Despite this limitation, wheels are generally cheaper and easier to design, build, and program than tracks or legs, which involve a larger number of mechanical components and result in more complex kinematic models. In addition, MRs with three or more wheels maintain continuous contact with the ground, improving balance and stability. As noted by Rubio, Valero, and Llopis-Albert (2019), wheeled robots tend to cause minimal wear on the terrain and are less complex to control.

Scientists and engineers have drawn inspiration from mammals, reptiles, and insects to design legged MRs. This type of locomotion relies on multiple contact points between the robot and the ground to enable movement. As Siegwart, Nourbakhsh, and Scaramuzza (2011) state, this configuration provides greater adaptability and maneuverability in rough terrain. By strategically coordinating these contact points, a robot can cross gaps and uneven surfaces, as long as it maintains ground clearance without slipping.

According to Rubio, Valero, and Llopis-Albert (2019), the main advantage of legged MRs over wheeled systems lies in their traversability and efficiency. They are particularly well-suited for soft, irregular, or unstable terrain, offering superior mobility, stability, and reduced impact on the ground surface. However, these benefits come at the cost of higher power consumption and greater mechanical complexity. Siegwart, Nourbakhsh, and Scaramuzza (2011) further explain that such systems require legs with multiple degrees of freedom, capable of supporting the robot's weight, and often its payload, while walking. The associated kinematic models are complex, and maintaining dynamic balance during locomotion demands a far more sophisticated control system than that required for wheels or tracks.

Bulldozers and war tanks are among the most familiar examples of tracked vehicles. Using the same principle, tracked MRs are well-suited for navigating uneven, rugged, stony, and muddy terrains. However, this type of locomotion also presents several drawbacks. According to Bruzzone and Quaglia (2012), tracked robots are subject to vibrations caused by the lateral track profile and are rarely equipped with effective damping systems. This limitation reduces mechanical efficiency and constrains the maximum operational speed, making tracked robots slower and more energy-intensive than their wheeled counterparts.

In wheeled MRs, it is common to assume that wheels do not skid against the ground surface (Siegwart; Nourbakhsh; Scaramuzza, 2011; Rubio; Valero; Llopis-Albert, 2019). This assumption enables direction changes through skid steering, which applies differential speeds to the wheels on each side of the robot. Tracked robots employ a similar approach. However, as Rubio, Valero, and Llopis-Albert (2019) observe, the larger contact surface of tracks improves maneuverability on loose terrain compared to wheels. At the same time, the increased skidding during turns makes it difficult to predict the robot's center of rotation. Combined with variations in ground friction, this reduces the accuracy of position and orientation estimates. As a result, Siegwart, Nourbakhsh, and Scaramuzza (2011) report that dead reckoning localization becomes highly unreliable when used with tracked locomotion.

Oliveira, Moreira, and Silva (2021) examined advances in agriculture robotics. Regarding locomotion systems, the authors found that 73% of agricultural robotics applications use wheeled locomotion, 8% aerial, 3% tracked, 2% legged, and the remaining distributed among boats, fixed arms, and others. Pathan *et al.* (2020) presented twelve land-based autonomous mobile robots for precision agriculture, where eleven of them use wheeled locomotion. Although the advantages of tracked and legged mobile robots, the simplicity of developing the locomotion and control systems in wheeled MRs plays a major role when choosing the mechanism system for robots in an agricultural environment. Besides that, farmers are already familiar with wheeled machinery. Robots with the same locomotion system benefit from this familiarity, increasing the likelihood

of integrating these new technologies in agricultural scenarios.

A similar trend is observed in other domains. Halder and Afsari (2023) conducted a systematic review of 269 papers on robotic inspection and monitoring in the built environment. Their results show that Unmanned Aerial Vehicles (UAVs) dominate the field, appearing in 161 studies. UAVs are favored for their ability to quickly access difficult and elevated locations. Unmanned Ground Vehicles (UGVs) follow distantly, with 32 mentions, valued for their ability to carry heavier payloads for extended inspections. In contrast, more specialized or mechanically complex systems, such as legged (2 papers), snake-like (3), wall-climbing (11), or hybrid robots (7), appear far less frequently. These findings reinforce the observation that simplicity, practicality, and operator familiarity strongly influence the selection of robotic platforms across domains, not only in agriculture but also in inspection and monitoring applications.

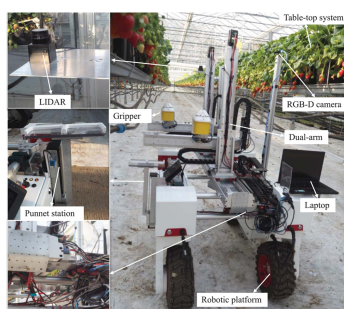
Figure 2 presents examples of wheeled, tracked, and legged autonomous mobile robots designed for agricultural applications. Xiong *et al.* (2020) proposed a wheeled autonomous MR (Figure 2 (a)) capable of continuously harvesting strawberries in poly-tunnels. The system achieved a success rate ranging from 50% to 97.1% when picking strawberries that were partially exposed or isolated. However, in more realistic scenarios, the system requires further improvements to handle targets that are heavily surrounded by obstacles.

Srinivasan *et al.* (2016) designed a tracked robot (Figure 2 (b)) capable of traversing undulating agricultural terrain and autonomously sowing seeds on prepared land. The system employed an ultrasonic sensor to detect the end of the field and used magnetometer data for navigation. Directional accuracy was enhanced with the implementation of a simple Proportional-Integral (PI) controller. Experimental results showed that the robot was able to sow seeds in evenly spaced rows with consistent uniformity. Additionally, the system demonstrated efficient power consumption, making it a viable solution for real-world agricultural applications.

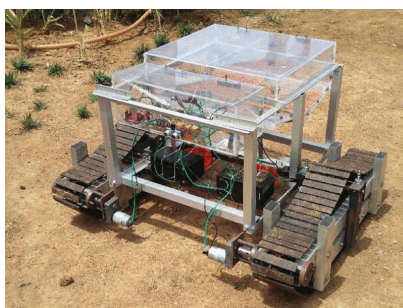
Iida *et al.* (2008) developed a legged MR (Figure 2 (c)) designed to explore agricultural fields and collect environmental data such as soil nutrients and plant growth. The hexapod robot, equipped with six legs, uses a gas sensor to detect ambient CO_2 levels and an anemoscope to determine the wind direction. By combining these two data sources, the robot is able to navigate toward the CO_2 source. However, the navigation speed is relatively slow; experimental results showed that the robot walked at a rate of only 2 m/min.

Figure 2 – Examples of land-based autonomous MRs for agricultural applications.

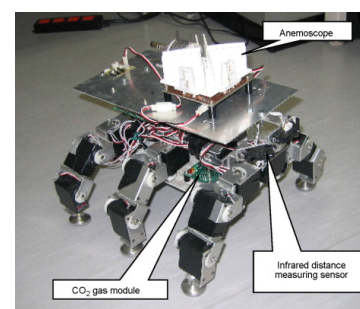
(a) Wheeled MR.

Source: Xiong *et al.* (2020)

(b) Tracked MR.

Source: Srinivasan *et al.* (2016)

(c) Legged MR.

Source: Iida *et al.* (2008)

Aerial-based MRs, also referred to as UAVs, may adopt fixed-wing configurations that follow the same locomotion principles as conventional airplanes. In this setup, a propeller attached to a motor generates forward airspeed. Combined with the aerodynamic properties of the wings, this forward motion produces lift, allowing the UAV to fly. Directional control is achieved through movable surfaces such as ailerons on the wings, and often elevators and a rudder on a tail assembly.

According to Garg (2021), fixed-wing UAVs feature a relatively simple design and offer higher flight speeds and greater payload capacity compared to rotary-wing platforms. However, they rely on continuous forward motion to maintain lift and typically require a runway for takeoff and landing. These limitations make them unsuitable for tasks that require hovering, unlike rotary-wing UAVs. Nevertheless, as noted by Nex and Remondino (2014), fixed-wing UAVs are capable of covering large areas in a short period of time. Budiharto *et al.* (2019) report that this capability is one of the main reasons why fixed-wing UAVs are favored by farmers.

Rotary-wing UAVs use locomotion principles similar to those of helicopters. Their propulsion mechanism comprises two or more propellers attached to an engine. The propellers create vertical airflow resulting in lift, which enables vertical take-off and landing, eliminating the necessity of a runway. According to Garg (2021), although Rotary-wing UAVs allow omnidirectional flight, this locomotion mechanism provides lower speed, smaller payload capacity, and a shorter range. Kanellakis and Nikolakopoulos (2017) highlight another advantage of rotary-wings UAVs: they can be operated both indoor and outdoor, and they are fast and agile, making it possible to follow a target in close quarters. Besides that, although the electrical and mechanical simple design, the control system handle the complexity of maintaining stability and speed.

Nex and Remondino (2014) evaluated UAV types through Table 2. The authors presented the pros and cons of different UAV typologies according to the literature review and their own experience.

Figure 3 shows examples of fixed-wing and rotary-wing autonomous UAV for agri-

Table 2 – Evaluation of some UAV platforms. The evaluation is from 1 (poor results or capacities) to 5 (good results or capacities).

	Fixed-wing		Rotary-wing	
	Electric	Combustion	Electric	Combustion
Payload	3	4	2	4
Wind resistance	2	3	2	4
Minimum speed	2	2	4	4
Flying autonomy	3	5	2	4
Portability	2	2	3	3
Landing distance	3	2	4	4

Source: Adapted from Nex and Remondino (2014).

cultural applications. Ruangwiset (2014) developed a prototype of an autonomous fixed-wing UAV, Figure 3 (a), for precision agriculture applications. The author conducted flight tests to analyze the effect of payload weight and altitude on power consumption. In experiment, without crosswinds, the prototype consumed approximately the same amount of propulsion current regardless of payload weights or altitude. However, in crosswind scenarios, the electrical current increased because of the additional throttle necessary to maintain the programmed flight path. Therefore, the author concluded that paths in agriculture missions must avoid crosswind to reduce power consumption and improve the range and endurance.

Alsalam *et al.* (2017), Figure 3 (b), proposed a modular and generic system to control a rotary-wing UAV autonomously using computer vision. Their system allows the UAV to adjust its planned path when necessary to approach a target for a close inspection, herbicide application, or to collect higher resolution agricultural images. Experiments showed that the system dynamically adapts its current goal and executes appropriated actions after detecting a target. The authors concluded that the system has strong potential to be adopted in precision agriculture applications.

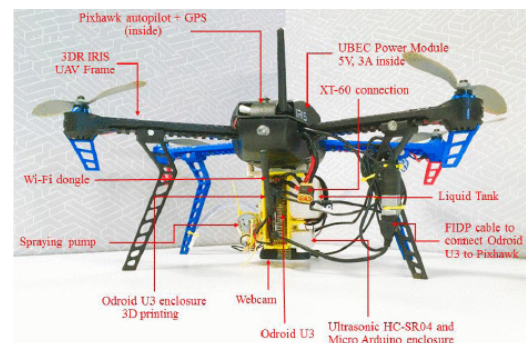
Figure 3 – Examples of aerial-based autonomous MRs for agricultural applications.

(a) Fixed-wing UAV.



Source: Ruangwiset (2014)

(b) Rotary-wing UAV.



Source: Alsalam *et al.* (2017)

Combining the best characteristics of both aerial robots types, Vertical Take-Off and Landing (VTOL) UAVs can take off and land vertically like a rotary-wing aircraft, while transitioning to horizontal flight to behave like a fixed-wing UAV, thereby combining the agility of quadcopters with the efficiency and range of airplanes.

An example of such a platform is the NAURU 500C ISR (Figure 4), developed by the Brazilian company XMobots. Designed specifically for intelligence, surveillance, and reconnaissance missions, this UAV is intended for security operations and is capable of flights at altitudes above 400 ft. With an endurance of up to 4 hours, the NAURU 500C ISR supports fully integrated ISR systems, making it a valuable asset for both public and private security applications.

Figure 4 – NAURU 500C ISR from XMobots



Source: xmboots (2025)

2.3 PERCEPTION AND ENVIRONMENTAL REPRESENTATION

Although MRs are known by their navigation abilities using locomotion systems, without sensors to perceive the environment, they become unable to accomplish even simple tasks. Therefore, as Siegwart, Nourbakhsh, and Scaramuzza (2011, p. 101) explain, robots acquire knowledge about themselves and their surroundings by taking measurements with various sensors and then extracting meaningful information from those measurements. With this knowledge, robots can determine their internal state, estimate their location in the environment, plan the paths to new destinations, avoid obstacles, control their locomotion actuators, and so on.

Siegwart, Nourbakhsh, and Scaramuzza (2011, p. 101) classified sensors into proprioceptive/exteroceptive and passive/active categories. This classification is based on the origin of data measured and on the working principle. Proprioceptive sensors collect information from the internal system of robots, such as motor speed, odometry, acceleration, battery voltage, internal temperature, and joint angles. Exteroceptive sensors get information from the environment, such as light, sound, distance, thermal and visual images, or scents.

An important application of proprioceptive sensors is to determine the current internal state of the robot. For example, in a wheeled robot, encoders and Hall-effect

sensors can verify whether a wheel is rotating at the demanded speed. In UAVs, a Hall sensor can monitor the current drawn by a motor to check if it is still functioning. Based on such readings, fail-safe behaviors can be triggered when malfunctions are detected.

Khalastchi and Kalech (2019) describe the use of Artificial Intelligence (AI) techniques to enhance fault detection and diagnosis. These methods include model-based reasoning, which identifies inconsistencies between expected and observed behavior. Distributed multi-agent reasoning allows robots to coordinate and detect disagreements in system behavior. Machine learning techniques, which learn normal patterns from sensor data and flag deviations as potential faults. Bio-inspired approaches may compare local behaviors across robots or apply artificial immune system principles to detect abnormal operation. These AI-driven strategies allow fault detection even in dynamic and uncertain environments where sensor reliability alone is not sufficient.

Sensors can also be classified based on their interaction with the environment. Passive sensors measure energy originating from the environment that comes into the sensor. Therefore, a passive sensor does not disturb the environment to make measurements. This type of sensor includes microphones, cameras, voltmeters, gyroscopes, and Inertial Measurement Units (IMUs). By contrast, active sensors emit energy into the environment and then measure its response. Examples of active sensors include ultrasonic and laser rangefinders, wheel quadrature encoders, and Infrareds (IRs) proximity sensors. According to Rubio, Valero, and Llopis-Albert (2019), active sensors provide more controlled interactions with the work environment, leading to improved performance. However, a disadvantage is that they are more susceptible to interference between their signals and external sources. Siegwart, Nourbakhsh, and Scaramuzza (2011, p. 104) presented a list of sensors relevant to mobile robot navigation, including those mentioned above. This list classifies each sensor by general category, typical use, sensor type, whether it is proprioceptive or exteroceptive, and whether it is passive or active.

Uncertainty and limitations are inherent to sensing and arise from several factors. Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005, p. 3) mention limitations related to physical properties, noise, and faults. For instance, cameras are constrained by their spatial resolution, and laser rangefinders may not function properly in environments with reflective surfaces such as glass. Noise can unpredictably distort sensor readings, reducing their accuracy and reliability. Moreover, sensors can fail without warning, and identifying faulty sensors is often extremely difficult. As a result, selecting appropriate sensors for a robot according to its application requirements becomes a challenging task. Everett (1995, p. 3) proposed guidelines for choosing sensors capable of acquiring high-resolution environmental data in a timely and practical manner, while accounting for limitations in onboard energy and computational capacity. These guidelines consider aspects such as field of view, range capability, accuracy, and resolution.

1. Field of view: sufficient width and depth for the application;
2. Range capability: effective minimum and maximum range suitable for the intended use of the sensor;
3. Accuracy and resolution: appropriate for the task requirements;
4. Detected objects: account for energy absorption, reflective properties and tolerance to ambient conditions and noise;
5. Real-time operation: data must be available at a rates suitable for platform speed and nearby vehicles;
6. Concise, interpretable data: the output should meet processing requirements, as excessive data can be as problematic as too little;
7. Redundancy: ensure graceful degradation; Multimodal sensing increases detection and confidence;
8. Simplicity: keep the systems low-cost, modular, and easy to maintain or upgrade;
9. Power: physical dimensions and weight must be practical for the robot.

These criteria offer a general foundation for selecting suitable sensors in mobile robotics. However, specific application domains may impose additional constraints. One of the most important applications of smart farming is crop monitoring. This task typically uses small UAVs with limited payload and power capacity. As Tsouros, Bibi, and Sarigiannidis (2019) point out, the main criteria for selecting sensors in this context are low weight, low energy consumption, compact size, and the ability to capture high-resolution data. The authors also highlight common onboard sensors that meet with these constraints: visible light sensors, multispectral sensors, hyperspectral sensors, and thermal sensors. Visible light sensors, or simply cameras, are used to capture high-resolution images. Multispectral and hyperspectral sensors collect data on vegetation's spectral absorption and reflection across multiple bands. Thermal infrared sensors collect temperature information from objects and generate images that display shapes and temperature. Furthermore, the authors highlight that it is common to integrate data from visible sensors with data from other types of sensors to overcome individual limitations. This enables the analysis of both visible and non-visible characteristics of crops, providing more comprehensive information to support improved agricultural management.

Robotic systems typically rely on multiple sensors to perform their tasks. In many cases, more than one sensor is used to measure the same environmental property. This redundancy is beneficial because it allows sensors to compensate for each other's limitations. However, combining data from different sources into a unified and consistent

representation is a complex task, addressed through a process known as multi-sensor fusion, also referred to as data fusion.

Mitchell (2007, p. 3) defines multi-sensor fusion as the theory, techniques, and tools which are used for combining sensor data, or data derived from sensory data, into a common representational format. Alatisse and Hancke (2020) discussed several other definitions and concluded that regardless of the specific wording, the fundamental goal of sensor fusion is to improve the accuracy and quality of the data, while reducing cost when feasible. These improvements are expected to enable results that are, in some way, superior to those achievable through the use of individual data sources.

Although sensor fusion brings improvements to the perception process, several problems must be addressed to achieve its full potential. Fung, Michael Z. Q. Chen, and Yong Hua Chen (2017) identified various issues related to data fusion, including registration errors during the integration process, uncertainty in sensor data, incomplete or inconsistent information, spurious data, challenges in correspondence and data association, differences in data granularity, and mismatched time scales. Additional advantages, disadvantages, and challenges associated with data fusion are discussed in R.C. Luo and Kay (1990), Nagla, Uddin, and Singh (2014) and Fung, Michael Z. Q. Chen, and Yong Hua Chen (2017).

Multi-sensor fusion is a multidisciplinary area that draws from several fields, making it difficult to establish a single classification for data fusion techniques. Castanedo (2013) found in the literature that the employed methods and techniques can be grouped according to multiple criteria. These include the relationships between the input data sources, the types and nature of input and output data, the abstraction level of the data used, the data fusion levels defined by Joint Directors of Laboratories (1991), and the architecture type. Based on these classifications, the authors concluded that sensor fusion techniques can be categorized into three nonexclusive groups: data association, state estimation, and decision fusion.

Data association aims to determine the set of measurements related to each target in the environment. State estimation focuses on defining the state of targets in motion, typically their position, using observations or sensor measurements. Decision fusion seeks to generate high-level inferences about events and activities based on the detected targets. Castanedo (2013) presented a list of widely used techniques to address the problems associated with each of these categories. According to Fung, Michael Z. Q. Chen, and Yong Hua Chen (2017), sensor fusion algorithms often rely on probability theory, classification methods, and artificial intelligence techniques. These include the Kalman Filter (KF) (and extended version), Support Vector Machine (SVM), Bayesian Inference Technique, Sequential Monte Carlo methods (particle filter), Dempster-Shafer Theory of Evidence, Artificial Neural Network (ANN), and Fuzzy Logic.

A common example of multi-sensor fusion in land and aerial mobile robots is the

use of the Extended Kalman Filter (EKF) to estimate localization. Titterton and Weston (2004, p. 402) explain this type of approach in detail, using measurements from IMU and Global Positioning System (GPS) sensors for a missile navigation system. The IMU provides real-time data but accumulates integrative errors over time in the localization estimate. In contrast, the GPS sensor offers high precision but with a slower update rate. By using the EKF, it is possible to reduce the error in the IMU measurements by incorporating data from the GPS. Each time a GPS update is received, the EKF estimates the error in the IMU data. This estimated error is then subtracted from the subsequent IMU measurements until the next GPS update. As a result, the localization accuracy based on IMU data is significantly improved.

Another important area of perception research is environment modeling. For activities such as localization and planning, the robot is expected to have an environment map, which was got before start working. This map can be constructed manually. However, it would be necessary to measure precisely the position of landmarks, walls, windows, and doors, which would be laborious and susceptible to errors. Simultaneous Localization and Mapping (SLAM) techniques allow a robot to autonomously build a map while simultaneously determining its own location within it. This is a challenging problem, often compared to the chicken and egg situation. The robot needs the map for localization, but mapping depends on localization to extract environmental features and place them correctly. EKF and Particle Filter (PF) based approaches are widely used to address the SLAM problem (Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter, 2005; Aulinas *et al.*, 2008; Gupta; Conrad, 2019).

J.J. Leonard and Durrant-Whyte (1991) proposed the first solution to SLAM using the EKF, which became known as EKF-SLAM. Years later, Montemerlo *et al.* (2002) introduced FastSLAM, a hybrid approach combining EKF and PF, which became one of the most widely adopted SLAM techniques. Both methods produce landmark-based maps, representing environments by estimating the position of fixed features, such as wall corners, windows, and doors. Figure 5 illustrates an environment alongside the map generated using the approach presented by Montemerlo *et al.* (2002).

Grisetti, Stachniss, and Wolfram Burgard (2007) proposed a technique based on the PF known as GMapping. This approach uses odometry and laser range sensors to build a grid map, where the environment is represented similarly to a blueprint. The map is composed of a grid of cells that can indicate free, occupied, or unknown space. Reducing the cell size improves the map's resolution and accuracy, but it also increases memory usage and computational cost. Therefore, the cell size must be chosen carefully according to the application's requirements. Figure 6 shows an example of a grid map generated using the GMapping approach.

One of the main challenges of SLAM is its reliance on the assumption that the environment is static. In low-dynamic settings, the techniques mentioned earlier can

Figure 5 – Landmark-based SLAM.

(a) Environment.



(b) Map.

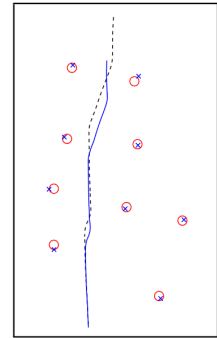
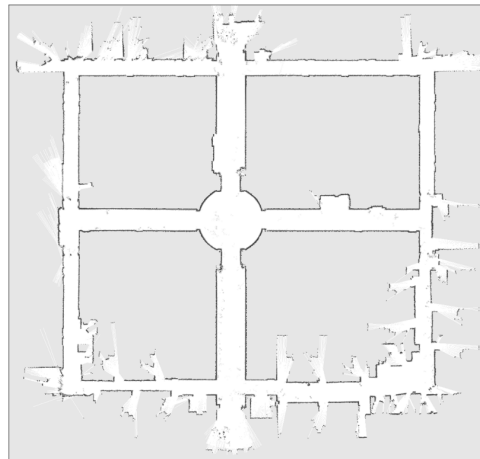
Source: Montemerlo *et al.* (2002)

Figure 6 – Grid map. White, free space. Grey, unknown environment. Black, obstacle.



Source: Grisetti, Stachniss, and Wolfram Burgard (2007)

generally handle small amounts of movement and typically ignore transient objects such as sporadically moving people. However, in highly dynamic environments, these traditional methods often fail by incorrectly mapping moving elements. In recent years, the growing accessibility of artificial intelligence techniques and improvements in computing power have enabled the integration of deep learning into SLAM to mitigate the problems caused by dynamic elements. A prominent example is the combination of traditional methods like ORB-SLAM (Mur-Artal; Montiel; Tardós, 2015) with deep learning object detection algorithms such as YOLO (Redmon *et al.*, 2016). Wenxin Wu *et al.* (2022) proposed YOLO-SLAM, a dynamic-environment-robust visual SLAM system that integrates a lightweight object detection network with a geometric constraint method. The system filters out dynamic features by analyzing depth differences and applying Random Sample Consensus (RANSAC), allowing it to exclude moving objects from the mapping process. Evaluations demonstrated that YOLO-SLAM significantly outperforms traditional SLAM techniques. It achieves lower localization error and offers

improved stability and accuracy in highly dynamic environments.

For other SLAM techniques and types of maps for indoor and outdoor environments, see Wolfram Burgard, Hebert, and Bennewitz (2016) and Stachniss, John J. Leonard, and Sebastian Thrun (2016). A comparative overview of pros and cons of SLAM techniques can be found in Aulinas *et al.* (2008).

2.4 LOCALIZATION

The execution of tasks by MRs involves perception, planning, and motion control, all of which require accurate knowledge of the robot's location to achieve its objectives. Localization refers to the process of estimating the robot's position and orientation within a given environment. For land-based robots, the position is typically represented by (x, y) coordinates and the orientation by an angle θ , which corresponds to the yaw relative to a reference frame. In the case of aerial-based systems, localization also includes the z coordinate, along with roll and pitch angles, to fully describe their three-dimensional pose.

Before presenting the classical localization approach, it is necessary to understand the types of localization problems, which can be categorized considering the position knowledge available initially and during operation. There are three categories of localization with an increasing degree of difficulty: position tracking, global localization, and kidnapped robot problem (Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter, 2005; Siegwart; Nourbakhsh; Scaramuzza, 2011; Panigrahi; Bisoy, 2021).

Position tracking, also known as local localization, is the simplest case of localization. It estimates the current location based on the previous position and sensor measurements, such as odometry or IMU. Therefore, the robot's initial location must be known. According to Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005) and Siegwart, Nourbakhsh, and Scaramuzza (2011), due to noise and uncertainty in such systems, the position is usually represented by a unimodal distribution, such as a Gaussian. Additionally, the noise must remain small. Otherwise, as position tracking relies on integration over time, the accumulated error can grow significantly, making it impractical to maintain an accurate estimate of the robot's location.

In global localization, the robot is placed somewhere in the environment without knowing its initial position. It must then begin navigating and determine its location within the environment. Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005) reported that in this type of problem, the pose error cannot be assumed to be bounded, which makes unimodal distributions generally unsuitable for representing the location. Additionally, Siegwart, Nourbakhsh, and Scaramuzza (2011) explain that the initial position is typically represented by a uniform distribution. As the robot navigates, the localization process gradually converges to several unimodal distributions, among which the one with the highest level of certainty is taken to represent the current location.

The kidnapped robot problem extends the global localization scenario. In this case, the robot must not only localize itself but also detect when it has been displaced to an unknown location and recover from this situation. Although such a scenario may seem unrealistic, Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005) note that testing for kidnapping events is a useful method for evaluating the robustness and failure recovery capability of global localization systems. Siegwart, Nourbakhsh, and Scaramuzza (2011) emphasize that since localization algorithms are not fail-proof, the ability to recover from localization failures is essential for truly autonomous navigation.

GPS is perhaps the most well-known localization method. It is widely used in military applications, everyday life, robotics, and more. In the context of autonomous navigation, GPS-based localization is applicable in outdoor environments for systems such as autonomous agricultural vehicles (Li, Ming; Wakabayashi; Yokoyama, 2009), UAVs (Radoglou-Grammatikis *et al.*, 2020b), and autonomous MRs (Ravankar *et al.*, 2021). The main advantage of the GPS system is that it provides absolute positioning. As a result, it is particularly effective for handling the kidnapped robot problem.

According to Bechar and Vigneault (2016), GPS systems have evolved, and techniques such as Differential GPS (DGPS) and Real-Time Kinematic (RTK) have enabled accurate real-time positioning. However, as reported by Ravankar *et al.* (2021), the cost of these sensors generally varies based on factors such as precision, accuracy, and robustness, making high-precision GPS systems (with centimeter-level accuracy) expensive. On the other hand, sensors with lower accuracy (ranging from tens to hundreds of centimeters) may be impractical for MRs operating in precision agriculture. Therefore, the need for high-accuracy localization significantly influences the overall cost of these robotic platforms.

Even high-cost GPS systems are not fail-proof. They may suffer from signal reception issues and have limitations in position update rates. Therefore, as mentioned by Bechar and Vigneault (2016), GPS is often integrated with dead-reckoning sensors, such as IMUs and odometry, to improve pose estimation accuracy. According to R. Khot *et al.* (2006), this integration is beneficial because errors in GPS (used for global localization) and dead-reckoning sensors (used for local or relative localization) are complementary by nature. This complementarity allows sensor fusion techniques to combine their measurements and reduce overall localization error.

Although GPS provides absolute localization, it is not suitable for indoor environments such as buildings, greenhouses, or even fields covered by canopies. Inside buildings, the GPS signal can be blocked and become unavailable for several seconds, which makes navigation impractical. This problem also occurs in forested environments. According to Roßmann, Krahwinkler, and Bücken (2009), signal absorption by the forest canopy and intermittent reception can lead to localization errors of up to 50 m. In such scenarios, better results are obtained with probabilistic map-based techniques,

including KF and its variants, Multiple Hypothesis Tracking (MHT), and Monte Carlo Localization (MCL) (Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter, 2005). Other approaches based on Radio Frequency Identification (RFID), Wi-Fi, and computer vision are presented by Yilmaz and Temeltas (2019) and Panigrahi and Bisoy (2021).

The KF theory requires the system to be linear with Gaussian noise. However, robotic systems usually exhibit nonlinear motion models. In such cases, the EKF is used for localization. This extension linearizes the system and then applies the KF. The robot's position is represented by a Gaussian distribution. According to Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005, p. 274), the EKF is more efficient in terms of time and memory usage than MCL methods and is typically applied with landmark-based maps. However, as noted by Panigrahi and Bisoy (2021), the EKF requires the robot's initial position to be known with reasonable accuracy and is suitable only for position tracking localization. Therefore, if the robot becomes lost or experiences a failure, it will not be able to recover its position using this method.

MCL methods use particles to represent hypotheses about the robot's position. These particles are initially distributed throughout the environment, allowing MCL to solve the global localization problem. After the algorithm converges, adding random particles to the map enables recovery from kidnaps or localization failures. According to Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005, p. 274), MCL can perform localization using raw sensor data combined with grid maps. They also state that the number of particles influences both accuracy and computational cost.

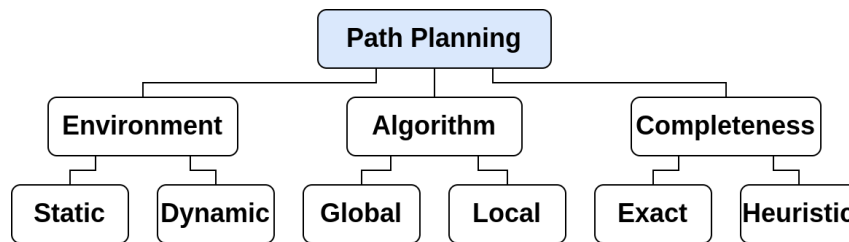
Dieter Fox (2003) proposed KLD-Sampling, a method that improves MCL performance by adapting the number of particles according to the required accuracy. This approach is used in the navigation stack of the Robot Operating System (ROS). Sebastian Thrun, Wolfram Burgard, and Dieter Fox (2005, p. 274) attribute the popularity of MCL-based methods to their ease of implementation and robustness, particularly their ability to represent different types of noise.

2.5 PATH PLANNING

A robot path planner defines a path from a starting position to a target position. Sánchez-Ibáñez, Pérez-del-Pulgar, and García-Cerezo (2021) states that the robot's movement capabilities and the characteristics of the terrain limit its motion. As a result, these constraints reduce the number of viable paths that the planner can generate. In addition, a robot's workspace typically contains both obstacles and free space. Therefore, the path planner is expected to provide the best feasible and obstacle-free route. According to Patle *et al.* (2019), safe path planning without collisions is the most critical function of any navigation technique. Thus, selecting an appropriate path planning method is a fundamental step in designing robots that operate in both simple and complex environments.

There are several challenges associated with mobile robot path planning. For instance, the environment may contain unexpected obstacles, including moving ones. As a result, the robot faces uncertainties regarding the map of its surroundings, requiring it to recalculate the path as new information becomes available. Additionally, the system may face computational constraints, making a good route more practical than the optimal one. Koubaa *et al.* (2018) present a categorization of these challenges, shown in Figure 7, based on the nature of the environment, the algorithm's focus, and the completeness of the solution.

Figure 7 – Path Planning Categories.



Source: Adapted from Koubaa *et al.* (2018)

Path planning in static environments is relatively straightforward. Since a static environment does not change over time, its obstacles and target position remain fixed during navigation. Although assuming a static environment simplifies the planning process, it does not reflect real-world conditions. In practical workspaces, MRs must deal with various challenges caused by unexpected static and dynamic objects such as doors, people, and other robots. Path planning in dynamic environments addresses these changes in the workspace. This capability strongly depends on the perception phase, which provides local information about the robot's surroundings within the effective range of its sensors. Therefore, while a global route is sufficient in static environments, dynamic workspaces require local adaptations to the path due to the uncertainty of the environment.

Dividing path planning algorithms into global and local categories is the most common approach to classifying them (Koubaa *et al.*, 2018; Gao *et al.*, 2018; Martins *et al.*, 2020; Sánchez-Ibáñez; Pérez-del-Pulgar; García-Cerezo, 2021). Table 3 presents the main characteristics of global and local path planning methods.

In global path planning, the robot has complete knowledge of the environment through a map constructed prior to navigation. Both the initial and target locations are defined on this map, and the path planner's role is to compute a route between them using the available free space. This type of planning is performed offline, meaning the path is calculated before the robot begins moving. The computational effort required depends on the size of the environment, which can lead to slow response times in large areas.

In contrast, local path planning relies on partial knowledge of the environment. Typically, the robot uses onboard sensors to construct a partial map in real time, enabling it to move toward the target while avoiding obstacles. These techniques must provide fast response times and are executed online. As a result, the path is adapted dynamically according to obstacles that appear along the way.

Table 3 – Global and local path planning.

Local path planning	Global path planning
Sensor-based	Map-based
Reactive navigation	Deliberative navigation
Fast response	Relatively slower response
Suppose that the workspace area is incomplete or partially incomplete	The workspace area is known
Generate the path and moving toward target while avoiding obstacles or objects	Generate a feasible path before moving toward the goal position
Done online	Done offline

Source: Koubaa *et al.* (2018)

Path optimization can be relaxed due to real-time requirements, the size of the environment, or hardware limitations. Heuristic algorithms offer good, though not optimal, results while requiring less computational processing. When using these approaches, the paths generated may vary even when the same algorithm is applied to identical start and target positions. In contrast, exact path planning algorithms aim to find the best feasible route. These algorithms consistently produce the same result for a given scenario, making it possible to formally prove that the computed path is optimal or that no feasible path exists.

The classical approach to mobile robot navigation combines global and local path planners. Xiao *et al.* (2022) present the classical navigation pipeline, illustrated in Figure 8. The authors explain that, given a target, perception data is processed to build a global representation of the environment, which may be based on recent sensor inputs or a pre-existing map. The global goal is then passed to the global path planner, which generates a global path, typically represented as a sequence of local goals, also known as waypoints. This step is analogous to a coarse plan made by a human. Taking the nearest local goal, the robot uses current perception data to construct a local representation of the surroundings. Based on this representation, the local path planner produces commands for the motor controllers, which generate raw motor outputs that are both safe and directed toward the goal. Figure 9 shows an example of waypoints and the resulting path executed by the local path planner.

Sánchez-Ibáñez, Pérez-del-Pulgar, and García-Cerezo (2021) analyzed several

Figure 8 – Classical navigation pipeline.

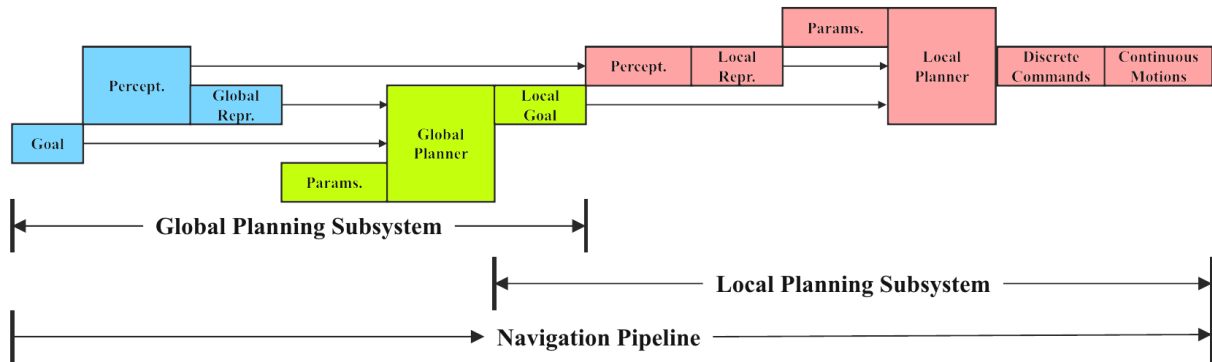
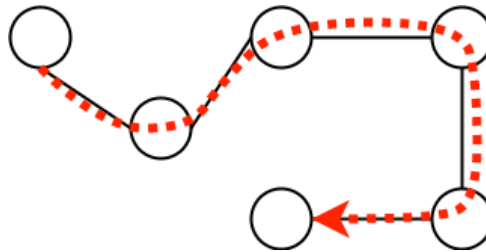
Source: Adapted from Xiao *et al.* (2022)

Figure 9 – Example of planned and executed paths. White circles are waypoints. The dotted red line is the performed path.



surveys and reviews regarding path planning from 2012 to 2021. They concluded the classifications proposed in those works do not represent the full range of path planning algorithms. To improve this scenario, the authors proposed four main categories: Reactive Computing, Soft Computing, Optimal Control, and C-Space search. The first two compose techniques are frequently used in local path planning. The two later, typically for global path planning. The paper presented several approaches for path planning. From them, the combination of A* and Dynamic Window Approach (DWA) was selected to be explained. These algorithms are the standard approach in the ROS navigation.

A* is a heuristic-based graph search algorithm proposed by Hart, Nilsson, and Raphael (1968). It combines exact and estimated information to guide the search process. The exact component represents the cost from the starting point to the current vertex. This part of the heuristic favors vertices closer to the start, similar to the Dijkstra algorithm proposed by Dijkstra (1959). The estimated component incorporates heuristic information such as Manhattan, diagonal, or Euclidean distance to prioritize vertices closer to the goal, as in Greedy Best-First Search. At each iteration, the algorithm selects the vertex that advances furthest from the start while remaining nearest to the goal. A* is also more computationally efficient than Dijkstra in many scenarios because it focuses on finding the shortest path from a single source to a single destination. In addition, A* supports searches in weighted graphs, which are not suited to Greedy Best-First Search. Because of these characteristics, A* is commonly used as a global

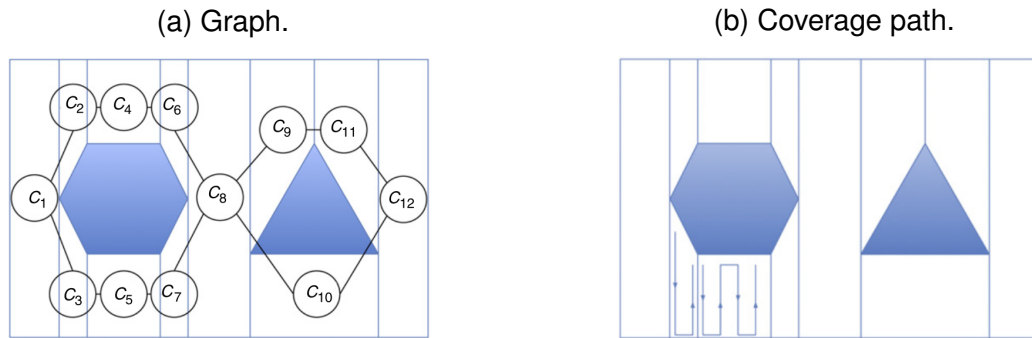
path planner, producing a sequence of waypoints to be followed by the DWA during local path planning.

The DWA was proposed by D. Fox, W. Burgard, and S. Thrun (1997). The authors explained that, in this technique, the search for commands that control the robot is performed directly in the velocity space. The method incorporates the robot's dynamics by limiting the search space to velocities that are reachable within the robot's dynamic constraints. Additionally, the approach considers only safe velocities with respect to nearby obstacles, ensuring that the robot can come to a complete stop if necessary. The local path is computed using a local map of the environment. In grid maps, this local representation is typically formed by a sliding window generated from exteroceptive sensors such as ultrasonic or laser range finders. As a result, the performance and accuracy of the approach depend on the update rate of this temporary model. After computing the robot's velocity, this data is sent to the motor controllers, which generate electrical signals to drive the motors. Therefore, the algorithm cannot avoid obstacles moving faster than the speeds predicted within its planning horizon.

In agricultural, surveillance and monitoring scenarios, robots usually operate under energy constraints and must cover large areas to complete their tasks. According to Kazim *et al.* (2021), path planning is essential for UAVs because it is directly related to their autonomy, onboard components, guidance, endurance, and overall functionality. UAVs face critical energy limitations. As discussed in the Locomotion section, adding extra batteries or power banks is not feasible due to payload restrictions. As a result, path planning becomes one of the most important factors in reducing energy consumption and enabling UAVs to fulfill their essential duties. Land-based MRs can also benefit from optimized path planning to cover larger areas while performing tasks such as seeding, weeding, and harvesting. The problem of determining a path that ensures complete coverage of all points in an Area of Interest (AOI) is known as Coverage Path Planning (CPP). Tan, Mohd-Mokhtar, and Arshad (2021) explain that CPP algorithms aim to compute an optimal, collision-free trajectory that allows the robot to fully cover an AOI within a specific time frame.

According to Tan, Mohd-Mokhtar, and Arshad (2021), when the environment is known in advance, cellular decomposition is one of the simplest methods for dividing an area into smaller sub-areas for CPP. Galceran and Carreras (2013) and Bormann *et al.* (2018) explain that this approach partitions the free space into non-overlapping cells, ensuring that their union covers the entire navigable area. After the decomposition, a graph is generated to connect the cells, and this graph is used to determine the most efficient sequence for exploring the sub-areas. Each cell is then covered using a lawn mower pattern. This process, based on trapezoidal cellular decomposition, is illustrated in Figure 10.

Figure 10 – Trapezoidal cellular decomposition in coverage path planning.



Source: Galceran and Carreras (2013)

2.6 CONCLUSION

This chapter explored the mechanisms that compose MRs navigation, preparing the reader to understand how navigation functions in robotics. It presented classical approaches and widely adopted techniques related to each topic. Review and survey papers were referenced for readers interested in further exploration. Since path planning is the central topic of this work, it is addressed in a dedicated chapter, which provides more detailed analysis through a systematic review.

3 COVERAGE PATH PLANNING

This chapter introduces the Coverage Path Planning (CPP) problem by examining its key concepts, requirements, challenges, environment representations, and existing approaches to route planning. The discussion is then extended to the Multi-UAV CPP problem. A Systematic Review (SR) of the most cited works outlines the main characteristics, methods, and sub-problems in Multi-CPP, as well as current research gaps. The protocol adopted to conduct the SR is detailed in Appendix A.

3.1 INTRODUCTION

Traditional path planning techniques aim to define the shortest path between a robot's starting position and a goal position. However, such approaches are not suitable for tasks like cleaning, painting, seeding, harvesting, or inspection, which require a CPP algorithm. Given an Area of Interest (AOI) and the robot's starting position, a coverage path planner generates a path that ensures complete area or volume coverage, taking into account the robot's sensor or actuator footprints. Galceran and Carreras (2013) cite Cao, Yuyu Huang, and Hall (1988) when listing the following requirements for robots performing coverage tasks:

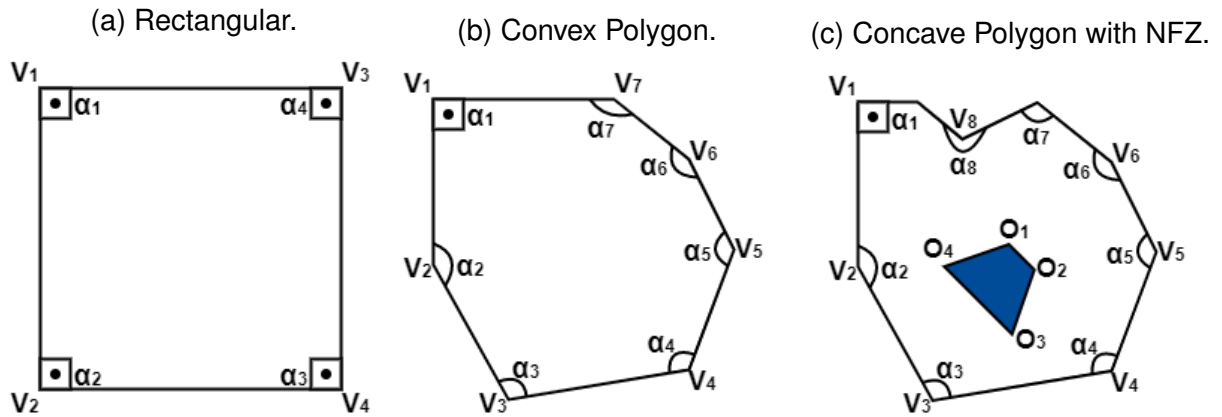
1. The robot must navigate through all the points within the AOI;
2. The robot must fill the AOI without overlapping paths;
3. The robot must operate continuously and sequentially, avoiding path repetition;
4. The robot must avoid all obstacles;
5. Simple motion trajectories such as straight lines or circles are preferred for simplicity in control;
6. The paths should be optimized whenever possible.

Requirements 2 and 3 may appear equivalent, but address different aspects of coverage. Requirement 2 ensures that the robot follows parallel paths without overlap, while requirement 3 emphasizes continuous execution. This means the robot should not interrupt the coverage, navigate elsewhere, and then return to resume it. Cao, Yuyu Huang, and Hall (1988) recognized that satisfying all these criteria in complex environments is not always feasible. In practice, the application may require relaxing certain constraints. As noted by Colomina and Molina (2014), in Unmanned Aerial Vehicle (UAV) image surveys, slight overlaps between paths may be necessary to support image mosaicking and compensate for aircraft instability.

In coverage tasks, especially those conducted in outdoor areas, the work area is typically defined by a user-specified polygon. Figure 11 illustrates examples of AOIs

represented in polygonal space. A polygon is formed by a sequence of connected vertices, where each vertex is described by a coordinate pair and an angle between its adjacent edges. When the polygon is non-convex, it includes one or more vertices with internal angles greater than π , as illustrated by vertex V8 in Figure 11 (a). In addition, some applications involve environments with obstacles that the robot must avoid. In UAV operations, these obstacles are known as No-Flight Zones (NFZs).

Figure 11 – Different AOI types in polygonal space.



The simplest way to cover an AOI is to perform a random walk through the area until full coverage is achieved. Tan, Mohd-Mokhtar, and Arshad (2021) identify two types of Random Walk (RW) methods: the fixed linear method and the variable step method. In the fixed linear approach, the robot moves in a straight line until it encounters a wall or obstacle, then randomly turns by an angle. In the variable step method, the robot repeatedly moves a step of random length, based on a given distribution, and turns in a random direction. However, both approaches result in high energy consumption, excessive travel distance, and increased time due to redundant coverage. For this reason, even the simple path patterns illustrated in Figure 12 offer greater efficiency.

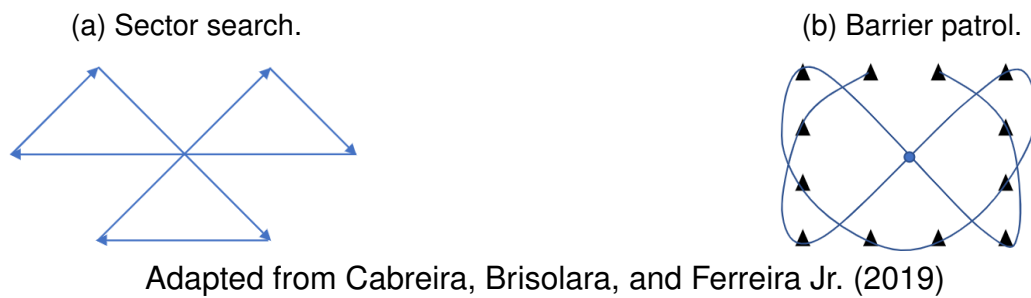
Figure 12 – Simple coverage patterns.



Specific coverage approaches have been designed for disaster response and search and rescue scenarios. Figure 13 illustrates such examples. Andersen (2014) compared the coverage patterns shown in Figures 12 and 13, and found that when

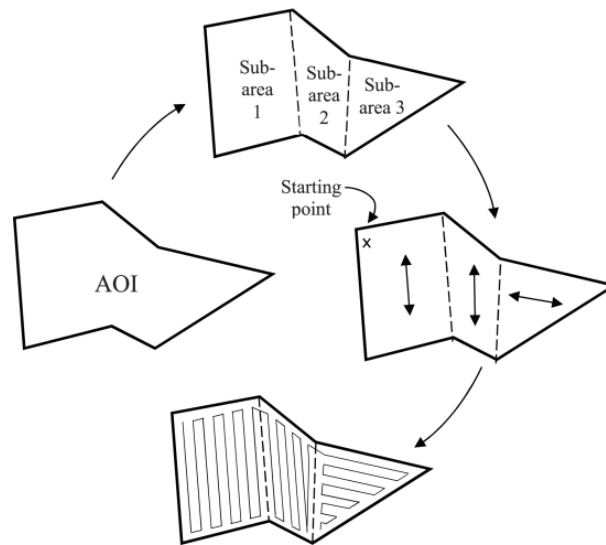
the number of search targets is uncertain and the area is large, sweeping coverage is the preferred strategy. The spiral pattern (also known as the square pattern) is suitable when uniform coverage of the search area is required. The sector search is applied in situations where the target is difficult to detect, offering the advantage of repeatedly crossing the center from multiple directions. The study also indicated that the barrier patrol pattern is more effective when the target is fast-moving or located in water with strong currents. Additional coverage patterns are explored in Artemenko *et al.* (2016).

Figure 13 – Patterns for search and rescue coverage.



In environments with complex shapes, applying basic coverage patterns alone is not sufficient. Figure 14 illustrates one of the most widely used approaches to CPP. According to Tan, Mohd-Mokhtar, and Arshad (2021), the process begins with a decomposition method that divides the AOI into a set of sub-areas. For each sub-area, the approach defines the coverage direction and the robot's starting position. An optimization step may also determine the optimal initial position within each sub-area and the sequence in which they should be visited. Once these parameters are established, the robot covers each sub-area using one of the previously discussed coverage patterns. Due to its simplicity, sweeping coverage is the most frequently used pattern in optimizations that focus on minimizing energy consumption, path length, or completion time.

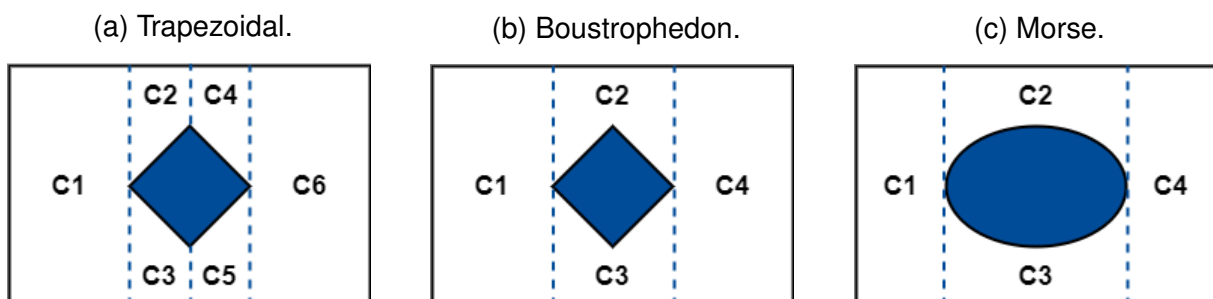
Figure 14 – Typical approach for CPP.



Source: Tan, Mohd-Mokhtar, and Arshad (2021)

There are three widely used methods for cellular decomposition: trapezoidal, boustrophedon, and morse-based decomposition. Figure 15 shows examples of areas divided using these methods. The trapezoidal approach divides the free space into convex trapezoidal cells based on the critical points (edges) of obstacles. Non-convex cells can still be covered using simple back-and-forth motions. As noted by Choset (2001), many of these cells can be merged to form larger ones that do not intersect any obstacles. This limitation led to the development of the boustrophedon decomposition by Choset and Pignon (1998), which merges all intermediate cells between two critical points into a single, non-convex cell, resulting in fewer and larger coverage regions. However, both the trapezoidal and boustrophedon methods are limited to two-dimensional polygonal environments. According to Galceran and Carreras (2013), morse-based decomposition can handle non-polygonal obstacles and is applicable in spaces of any number of dimensions. Nonetheless, Galceran and Carreras (2013) *apud* Milnor (1963), report that this method is unsuitable for rectilinear environments. In such cases, it becomes impossible to define critical points that correspond to topological changes in the space.

Figure 15 – Decomposition methods.



Valente *et al.* (2011) proposed a grid-based approach, as exact cell decompositions such as trapezoidal decomposition are inefficient for imaging surveys. In this context, the environment is discretized into a grid map, where the sensor footprint determines the size of each cell. Most approaches define square or rectangular cells, which align naturally with data provided by cameras. The grid structure enables a homogeneous distribution of the area, allowing each cell center to be treated as a target position for data acquisition. Figure 16 illustrates a typical CPP method based on grid decomposition. For a given polygonal area, the system performs discretization according to the sensor or actuator footprint. Following this step, Galceran and Carreras (2013) describe how methods such as wavefront, spanning tree, or artificial intelligence are used to define the path through the cells. Figure 17 shows an example of each method.

Zelinsky *et al.* (1993) was the first to propose a CPP solution using the wavefront algorithm. This approach requires both start and goal positions. Figure 17 (a) illustrates the process. The method begins by assigning a value of 0 to the goal cell, followed by assigning a 1 to all neighboring cells. Next, all unmarked cells adjacent to those labeled 1 are marked with a 2, and this incremental labeling continues until the wavefront reaches the starting cell. To generate the path, the robot iteratively selects an adjacent unvisited cell with the same label. If no such cell exists, it selects a neighboring cell with a label that is one unit smaller.

Gabriely and Rimon (2002) developed the Spiral Spanning Tree (ST) coverage method. Figure 17 (b) presents an example of this approach. In this method, each cell is treated as a mega cell, where the robot's footprint covers only one-quarter of the cell's area. The free cells in the grid map induce a graph structure, with each node representing the center point of a cell. This graph is then used to generate a ST of the environment. The robot performs the coverage by following a spiral path that skirts the generated ST.

Artificial Intelligence (AI) approaches, such as the Artificial Neural Network (ANN) proposed by S.X. Yang and C. Luo (2004) and the Reinforcement Learning (RL) method presented by Theile *et al.* (2020), typically learn the appropriate behavior based on the robot's and environment's current state. After the training phase, the robot can analyze sensor readings and decide on the next action accordingly. These methods enable online CPP. Furthermore, as shown in the work of Zanliang Dong and Chunhui Liu (2021), once an agent has learned a control policy, other agents with the same characteristics can operate cooperatively in the same environment by leveraging the acquired knowledge.

Figure 16 – Typical approach for CPP using grid decomposition.

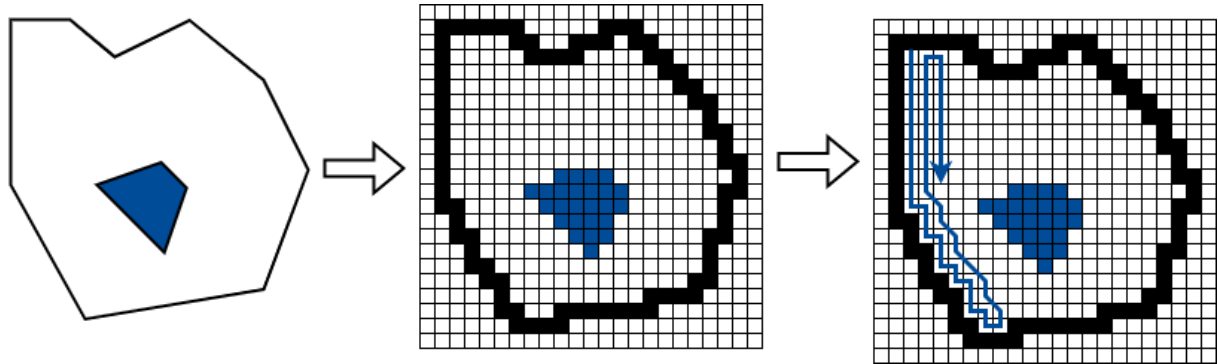
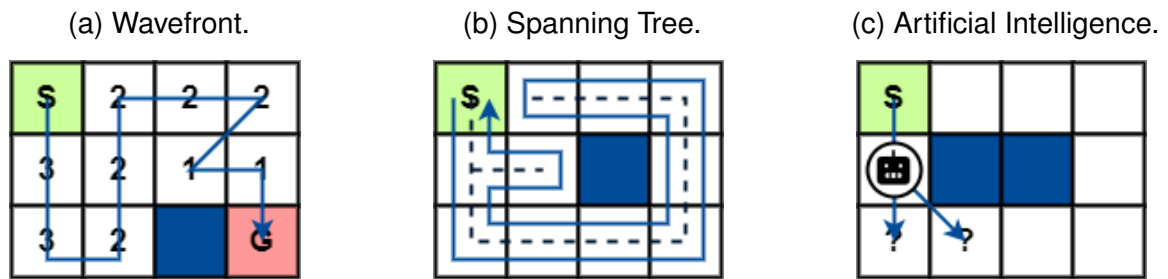


Figure 17 – CPP methods for grid.



Designing solutions for CPP is challenging because multiple criteria can influence an optimal CPP, including travel time, path overlap rate, energy consumption, and percentage of covered area. The latter is related to online approaches, which can have situations where covered areas surround the robot, but uncovered areas remain in the environment. Among the major challenges to optimal path planning listed by Aggarwal and Kumar (2020) are path length, time efficiency, energy efficiency, and robustness. Defining the route with the shortest path length is the usual approach in path planning. Typically, time and energy efficiency are linked to this factor. However, terrain or weather factors can impact the time of operation or the energy required to complete the task. Therefore, the optimization also depends on real-time decisions for dynamic scenarios. The path planner also has to be robust, which implies in tolerate position and driving errors during path planning.

Designing solutions for CPP is challenging because multiple criteria can influence what is considered optimal. These include travel time, path overlap rate, energy consumption, and the percentage of the area covered. The latter criterion is particularly relevant to online approaches, where the robot may be surrounded by already covered areas, even though some regions of the environment remain unexplored. According to Aggarwal and Kumar (2020), the main challenges in achieving optimal path planning involve minimizing path length, improving time and energy efficiency, and ensuring robustness. Route definition often focuses on minimizing the total path length, which is commonly associated to both time and energy efficiency. However, factors

such as terrain and weather can significantly affect the duration of the operation or the energy required to complete it. As a result, effective optimization also requires real-time decision-making in dynamic environments. Robustness is another essential aspect. The path planner must tolerate positioning errors and control inaccuracies that may occur during execution.

Concerning aerial applications, Yan Li *et al.* (2011) demonstrated that coverage paths with fewer turning motions are more efficient for UAVs in terms of energy consumption, travel distance, and time. As a result, a longer path with fewer turns may be preferred over the shortest possible path. This preference is justified because turning introduces acceleration changes, that increases the UAV's energy usage. Jiadong Li, Xueqi Li, and Lijuan Yu (2018) also observed that choosing direction parallel to the longest side of a polygon typically results in fewer turns. Therefore, a useful preprocessing step in CPP may involve identifying the longest side of the AOI and aligning the coverage rows accordingly. Weather conditions also influence efficiency. According to Xu, Viriyasuthee, and Rekleitis (2014), selecting the direction of coverage rows based on wind direction can reduce path cost. Flying in a crosswind may destabilize a fixed-wing UAV and increase the energy required for flight stabilization.

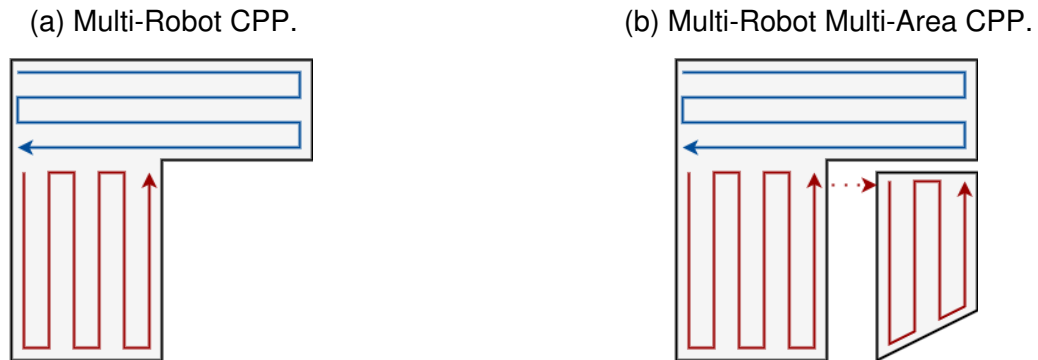
3.2 SURVEY ON MULTI-UAV COVERAGE PATH PLANNING

Using a single robot is not sufficient in certain applications. For example, employing UAVs in agricultural, surveillance and monitoring tasks presents several challenges. UAVs face technical limitations, including low battery efficiency, limited flight time, short communication range, and small payload capacity (Bueren *et al.*, 2015; Kim, J. *et al.*, 2019). Battery and payload constraints directly affect the choice of onboard sensors. According to Tsouros, Bibi, and Sarigiannidis (2019), these sensors must be lightweight and energy-efficient. These limitations also impact the overall feasibility of the application. A single UAV may be unable to complete a monitoring task over large areas due to its limited energy resources. One approach to mitigate this issue is to divide the coverage among multiple robots. In such cases, each robot is responsible for a smaller portion of the task, which can reduce hardware requirements. Using several robots with simpler hardware may be more cost-effective than deploying a single, highly robust unit. Additionally, this strategy can reduce the total time needed to complete the task.

The Multiple Coverage Path Planning (MCP) problem is modeled similarly to the CPP. Typically, a user defines the area of interest as a polygonal region, which might be converted into a grid or a graph representing the sensing points based on the sensor footprint. Then, the solver assigns a portion of the area and a corresponding path for each robot. Figure 18 shows an extension to the MCP, known as the Multiple Areas MCP. The solver must divide multiple areas among robots when a single robot cannot feasibly cover them. The objective is to minimize both the travel path between

areas and the coverage path within each area. Rekleitis *et al.* (2008), states these problems are at least NP-HARD, since even the simplest variation is equivalent to the Multiple Traveling Salesman Problem (MTSP).

Figure 18 – CPP sub-problems.



This section presents the results of a systematic review conducted to define the state of the art in Multi-UAV CPP. Table 4 summarizes general information about the studies identified. Since most works formulate the problem as an optimization task, it is relevant to examine the associated constraints and objectives. The table includes the type of environment and path representation used, the constraints and parameters considered during path planning, and the studies selected for comparison.

Table 5 presents detailed information about each study, examining whether the authors considered dynamics, kinematics, or variations in acceleration during any part of the path planning process. Variations in acceleration are important because they increase energy consumption and can be estimated using the dynamic or kinematic models of the robots. The table also specifies whether the studies applied decentralized control, addressed battery recharging, incorporated some level of fault tolerance, or considered the presence of NFZs. Readers can use the summary tables along with the accompanying paragraphs, which provide a concise explanation of each approach reviewed.

In Avellar *et al.* (2015), the MCPP process begins by defining the optimal sweep direction, which is parallel to the smallest linear dimension of the area. Waypoints are then placed at the beginning and end of each row to be covered. The spacing between rows depends on the footprint of the onboard cameras projected onto the ground. The authors formulated an optimization model using Mixed Integer Linear Programming (MILP) to determine the required number of UAVs and their corresponding routes. The optimization problem was solved using the Yalmip toolbox as the front end and the Gurobi solver as the back end.

Modares *et al.* (2017) conducted experiments demonstrating that a UAV's energy consumption depends on both the distance traveled and the number and sharpness of turns in its path. Based on these results, the authors proposed a linear model to estimate

Table 4 – General information summary of the works found.

Reference	Environment	Path	Solution found	Constraints	Optimization	Work comparison
Avellar <i>et al.</i> (2015)	Polygon to graph	Waypoints	Offline	Maximum time of flight, flight speed, altitude, setup time, and number of operators.	Minimum number of UAVs to cover the area, minimum time of the longest route.	None
Modares <i>et al.</i> (2017)	Grid to graph	Waypoints	Offline	None	Minimize the maximum energy incurred, based on distance and turns.	Simple rastering, brute force, and Barrientos <i>et al.</i> (2011)
Nedjati <i>et al.</i> (2016)	Grid	Cells	Offline	Number of UAVs, maximum time of flight, potential location and number of UAV bases.	Number and location of UAV bases, minimum number of UAVs to each open base, minimum time required for covering.	None
Azpuruu <i>et al.</i> (2018)	Hexagonal grid	Waypoints	Offline	Maximum path length, battery consumption to visit a hexagon, and the location of home position.	Internal paths and the visit order of the paths are optimized depending on the type of survey.	Maza and Ollero (2007)
Choi <i>et al.</i> (2020)	Graph	Waypoints	Offline	Maximum energy.	Minimum number of UAVs to perform coverage and total energy required.	None
Jiadong Li, Xueqi Li, and Lijuan Yu (2018)	Grid 3D	Waypoints	Offline	Average speed, maximum battery life, minimum turning radius, climbing angle, safe distance, maintenance time during two operations, maximum detection distance, maximum perspective, and maximum flight altitude.	Number of UAVs are dispatched from the base every two hours.	None
Yinghui Wang <i>et al.</i> (2018)	Graph	Waypoints	Offline	Speed of travel, altitude, maximum flight time, and a minimum and a maximum downtime between two successive flights, each waypoint is to be covered within a given time interval.	Maximize the number of waypoints covered making full use of UAVs' flight time.	None
Valente <i>et al.</i> (2011)	Grid	Cells	Offline & Online	None	Minimizes the time to competition, considering the number of turns and revisited cells.	None
Sung Won Cho <i>et al.</i> (2021)	Grid	Cells	Offline	Maximum time of flight.	Minimizes the completion time.	Maza and Ollero (2007)
Sawhney <i>et al.</i> (2008)	Grid 3D	Waypoints	Online	Minimum height which the UAVs can be spotted.	Time visibility for surveillance without being spotted.	None
Santin <i>et al.</i> (2021)	Graph	Waypoints	Offline	Maximum time of flight, speed, the recharge time ratio, and the costs to traverse the edges.	Minimizes the longest vehicle route and the number of recharge stations.	None
Kwon and Thangavelautham (2020)	Grid	Cells	Online	None	Area covered (can be partially covered).	None
Kumar Gupta <i>et al.</i> (2017)	Grid	Cells	Offline & Online	None	Balancing of the paths	None
Collins <i>et al.</i> (2021)	Graph	Waypoints	Offline	Speed of travel	Minimizes the worst completion time (work balance indirectly)	Guastella <i>et al.</i> (2019)
Skorobogatov <i>et al.</i> (2021)	Polygon	Waypoints	Offline	Number of UAVs, cruise speeds, field of view of the camera, altitude above ground level, cross-track overlapping.	None	None
Mickey Li, Richards, and Sooriyabandara (2021b)	Graph	Waypoints	Offline	Deadline	Maximizes the probability of mission completion.	Mickey Li, Richards, and Sooriyabandara (2020)
Mickey Li, Richards, and Sooriyabandara (2021a)	Graph	Waypoints	Offline & Online	None	Minimizes the completion time.	BNNB, NB-MSTC, and B-MSTC in Hong Liu, Jayao Ma, and Weibo Huang (2018)
Hassan, Mustafic, and Dikai Liu (2020)	Graph	Waypoints	Offline & Online	None	Minimizes the completion time.	BNNB, NB-MSTC, and B-MSTC in Hong Liu, Jayao Ma, and Weibo Huang (2018)
Zanliang Dong and Chunhui Liu (2021)	Grid	Cell direction	Online	Battery capacity	Incentivizes the coverage and punish solutions that have to expend all the battery.	None
Zhe Zhang, Jian Wu, and He (2019)	Grid	Cells	Offline	None	Ensure time interval between two patrols in the same area not more than 3 hours, balanced flight time, shortest route and time.	None
Sung-Won Cho <i>et al.</i> (2022)	Hexagonal grid	Cells	Offline	Cruising speed, rotational speed, maximum flight duration.	Minimizes the longest travel time to complete the coverage.	Gomez, Lozada, and Carbajal (2018), Na Lin <i>et al.</i> (2019) and Xie, Garcia Carrillo, and Jin (2020)
Shao <i>et al.</i> (2022)	Polygon	Rectangles and turns	Offline	Battery capacity.	Minimizes the length and number of turns.	None
Luna <i>et al.</i> (2022)	Polygon	Waypoints	Offline & Online	None	Optimize the distribution of tasks between the agents.	None
Xiaoxiao Yu <i>et al.</i> (2020)	Polygon	Waypoints	Offline	None	Balance of area coverage and path length.	Short distance first, and Jinchao Chen <i>et al.</i> (2019)
Sanna, Godio, and Guglieri (2021)	Tighten grid	Cell direction	Offline & Online	None	Learn how to minimize the number of time-steps, reduce the number of turnings, and avoid revisit cells.	None
Yao, Huixia Wang, and Ruiguang Hu (2021)	Grid	Cells	Offline	None	Minimizes the longest route among the UAVs.	None
Xiaolin Zhou <i>et al.</i> (2021)	Grid	Cells	Online	Speed of flight.	Minimize the repetitive coverage area, energy consumption, and flight distance (indirectly).	Konda and Tsitsiklis (1999), Osband <i>et al.</i> (2016), and Behzadan and Hsu (2019)
Huan Wang <i>et al.</i> (2021)	Grid	Cells	Offline	Number of cells to visit (Energy).	Balance of energy.	None
Apostolidis <i>et al.</i> (2022)	Grid	Cells	Offline	None	Equally divides the space of coverage.	Bähnemann <i>et al.</i> (2021)
Bouras, Bouzid, and Guaitni (2022)	Polygon	Waypoints	Offline	Number of cells to visit.	Minimizes individual total distance, and balances the efforts between UAVs.	None
Day and Salmon (2021)	Polygon	Waypoints	Offline	Number of trash collectors, battery chargers, UAVs, and battery capacity.	Positions of the recharge stations and trash collectors.	None
Second Search Round - 2025						
Wenjian Hu <i>et al.</i> (2023)	Grid	Cells	Online	Fixed flight height and speed.	Minimize task completion time by avoiding overlaps, uncovered grids, and collisions via cooperative Q-learning.	Xiao Liu <i>et al.</i> (2019) and Jiaxin Dong, Xinde Li, and Yihai Liu (2020).
Lin Li <i>et al.</i> (2023)	Graph	Waypoints	Offline	Dubins vehicle (fixed speed and minimum turning radius).	Minimize the maximum Dubins route length (makespan) with balanced loads and connected partitions.	Xing Zhou <i>et al.</i> (2019) and Karapetyan <i>et al.</i> (2018).
Govindaraju <i>et al.</i> (2023)	Grid	Cells	Offline & Online	Number of robots $\leq$ number of columns	Minimize distance, avoiding overlaps, balancing robot loads.	None
Tang and Hang Ma (2024)	Grid to graph	Waypoints	Offline	None	Minimize the longest tour (makespan) while ensuring full coverage and load balance.	Zheng <i>et al.</i> (2010), Tang and Hang Ma (2023) Tang, Sun, and Xinyu Zhang (2021)
Wan, He Yang, and Meng (2023)	Polygon	3D Waypoints	Offline	None	Minimize the greatest path length aiming load balance.	None
Choton and Prabhakar (2023)	Polygon	Waypoints	Offline	Dubins vehicle (fixed speed and minimum turning radius).	Minimize mission time via balanced partitions and optimal coverage paths.	None
Nair, Dileep, and Guruprasad (2024)	Grid	Cells	Online	None	None	None
Zeng, Tong, and Xia (2024)	Grid	Cells	Online	None	Maximize coverage rate and minimize number of turns.	None
Zikai Wang <i>et al.</i> (2025)	Grid	Cells	Online	None	Minimize the maximum coverage cost among all robots.	Zikai Wang <i>et al.</i> (2024), Hassan, Mustafic, and Dikai Liu (2020) and Hong Liu, Jayao Ma, and Weibo Huang (2018)
Gong, Hyunbin Kim, and Lee (2024)	Polygon	Cells	Offline & Online	None	None	None
Mengyang Wang <i>et al.</i> (2025)	Grid 3D	Waypoints	Offline & Online	Dubins vehicle (fixed speed and minimum turning radius).	Minimize overall mission time.	None
Mo and Zhiyun Lin (2024)	Grid	Graph	Offline & Online	None	Minimize the total area coverage time and balance workload.	Kapoutsis, Chatzichristofis, and Kosmatopoulos (2017)
Boquan Zhang <i>et al.</i> (2024)	Grid	Cells	Online	Maximum mission time.	Maximize the coverage of the mission area.	Ackermann <i>et al.</i> (2019) and Lowe <i>et al.</i> (2017)

energy consumption during flight. Using this model, they formulated the Energy-Efficient CPP problem and showed that it is NP-hard, with complexity increasing rapidly even with a small number of UAVs. To address this limitation, they decomposed the problem into two sub-problems: a load-balancing problem that fairly distributes the area among the UAVs and an energy-efficient path planning problem for each UAV. The load-balancing

Table 5 – Details summary of the works found.

Reference	Dynamics or kinematics	Acceleration	Decentralization	Battery recharge	Fault tolerant	No flight zone
Avellar <i>et al.</i> (2015)	No	No	None	None	No	No
Modares <i>et al.</i> (2017)	No	Yes	None	None	No	Yes
Nedjati <i>et al.</i> (2016)	No	No	None	None	No	No
Azpuruu <i>et al.</i> (2018)	No	No	None	Static recharge station	No	No
Choi <i>et al.</i> (2020)	Yes	Yes	None	None	No	No
Jiadong Li, Xueqi Li, and Lijuan Yu (2018)	No	No	None	Static recharge station	No	No
Yinghui Wang <i>et al.</i> (2018)	No	No	None	Static recharge station	No	No
Valente <i>et al.</i> (2011)	No	No	None	None	No	No
Sung Won Cho <i>et al.</i> (2021)	No	No	None	None	No	No
Sawhney <i>et al.</i> (2008)	No	No	None	None	Yes	Yes
Santin <i>et al.</i> (2021)	No	No	None	Multiple static recharge station	No	No
Kwon and Thangavelautham (2020)	No	No	Multi-agent	None	No	No
Kumar Gupta <i>et al.</i> (2017)	No	No	None	None	Yes	Yes
Collins <i>et al.</i> (2021)	No	No	None	None	No	Yes
Skorobogatov <i>et al.</i> (2021)	No	No	None	None	No	Yes
Mickey Li, Richards, and Sooriyabandara (2021b)	No	No	None	None	Yes	No
Mickey Li, Richards, and Sooriyabandara (2021a)	No	No	None	None	Yes	No
Hassan, Mustafic, and Dikai Liu (2020)	No	Yes	Multi-agent	None	Yes	No
Zanliang Dong and Chunhui Liu (2021)	No	No	None	None	No	No
Zhe Zhang, Jian Wu, and He (2019)	No	No	None	None	No	No
Sung-Won Cho <i>et al.</i> (2022)	No	No	None	None	No	No
Shao <i>et al.</i> (2022)	No	Yes	None	None	No	No
Luna <i>et al.</i> (2022)	No	No	None	None	Yes	No
Xiaoxiao Yu <i>et al.</i> (2020)	No	No	None	None	No	No
Sanna, Godio, and Guglieri (2021)	No	No	Multi-agent	None	No	Yes
Yao, Huixia Wang, and Ruiguang Hu (2021)	No	No	None	None	No	No
Xiaolin Zhou <i>et al.</i> (2021)	No	No	Multi-agent	None	No	Yes
Huan Wang <i>et al.</i> (2021)	No	No	None	Static recharge station	No	Yes
Apostolidis <i>et al.</i> (2022)	No	Yes	None	None	No	Yes
Bouras, Bouzid, and Guiatni (2022)	No	No	None	None	No	No
Day and Salmon (2021)	No	No	None	Static recharge station	No	No
Second Search Round - 2025						
Wenjian Hu <i>et al.</i> (2023)	No	No	Multi-agent	None	Yes	Yes
Lin Li <i>et al.</i> (2023)	Yes	No	None	None	No	No
Govindaraju <i>et al.</i> (2023)	No	No	Multi-agent	None	No	No
Tang and Hang Ma (2024)	No	No	None	None	No	No
Wan, He Yang, and Meng (2023)	No	No	None	None	No	Yes
Choton and Prabhakar (2023)	Yes	No	None	None	No	No
Nair, Dileep, and Guruprasad (2024)	No	No	Multi-agent	None	Yes	No
Zeng, Tong, and Xia (2024)	No	No	Multi-agent	None	No	No
Zikai Wang <i>et al.</i> (2025)	No	No	None	None	No	No
Gong, Hyunbin Kim, and Lee (2024)	No	No	None	None	Yes	No
Mengyang Wang <i>et al.</i> (2025)	Yes	No	None	None	Yes	No
Mo and Zhiyun Lin (2024)	No	No	Multi-agent	None	No	Yes
Boquan Zhang <i>et al.</i> (2024)	Yes	Yes	Multi-agent	None	No	Yes

stage was solved as a Vehicle Routing Problem (VRP) using a MILP model. For the path planning step, the authors adapted the Lin-Kernighan Heuristic (LKH) algorithm to account for the energy cost of turns and generate more efficient paths.

Nedjati *et al.* (2016) investigated MCPP for assessment in earthquake-stricken regions. The area was divided into equal-sized squares based on the UAV's camera footprint, and the UAVs began their routes from a base and returned there either for refueling or upon mission completion. The authors formulated the problem as a MILP to determine which bases should be activated, the number of UAVs assigned to each base, the path for each UAV, and the minimum time required to complete the coverage. The MILP model was implemented in GAMS and solved using the CPLEX solver.

Azpurua *et al.* (2018) presented an approach for aerial magnetometric surveys that incorporates height and orientation constraints. The process begins with a user-defined polygonal region, which is converted into a hexagonal grid. The hexagons are then evenly distributed among robot teams, of up to three robots, using the k-means algorithm. Each hexagon is assigned a predefined coverage path following a lawnmower, square, or hexagon centroid patterns. The algorithm optimizes the entry and exit points of each cell by rotating the predefined paths to minimize the distance between adjacent cells. Depending on the number of robots, cells may be covered multiple times using different patterns to improve data quality. The system also includes a synchronization mechanism based on a finite-state machine to prevent collisions and ensure that UAVs operate within the same time window without interference.

Choi *et al.* (2020) analyzed the energy consumption of aerial imagery missions by segmenting the mission profile into take-off, cruise, hovering, turning, and landing phases. The authors argued that incorporating these profiles allows a more accurate estimation of the total energy required. They proposed an arc-based optimization model for energy-constrained CPP and later converted it into a route-based model. This conversion accounts for turning phases, which are typically excluded from arc-based formulations and may lead to energy underestimation and infeasible routes. By modeling all mission segments during optimization, the proposed method constructs routes with fewer turns, resulting in more energy-efficient paths for multi-UAV imagery missions. To solve the optimization problem, the authors applied a column generation framework combined with a label correcting algorithm to accelerate the process.

Jiadong Li, Xueqi Li, and Lijuan Yu (2018) proposed a Multi-UAV CPP approach for exploration in plateau and mountain environments. A key feature of this scenario is the altitude variation caused by terrain elevation. To address this, the authors used a 3D grid representation, that allows flight altitude to vary according sensor coverage requirements. The problem was divided into two stages. First, a global planner employs a nonlinear programming model to determine the shortest-time flight path for a single UAV, using scan lines aligned with the longest side of the polygonal area. Second, a local planner applies a multiobjective integer programming model to divide the area according to each UAV's initial position and remaining lifetime. Since the mission involves continuous 24-hour exploration, the system also determines how many UAVs should be dispatched to the search area each hour. The local planner's model is solved using a Genetic Algorithm (GA).

Yinghui Wang *et al.* (2018) explored the surveillance of disjoint areas, where multiple AOIs must be periodically covered. Base stations are located outside these areas, and their positions are known a priori. Each station has one or more assigned aircraft that begin and end their missions at the same location. The problem cannot be solved by simply maximizing travel time within each AOI and minimizing travel time

between them, as every region inside an area of interest must be revisited within a specific time window. The approach begins by dividing each AOI into sensing points based on sensor requirements. To cover an AOI, the system generates one or more coverage paths, referred to as single paths. Using a multiobjective evolutionary algorithm, the method constructs missions composed of single paths, inter-area connections, and take-off times for each flight, while ensuring that all points of interest are revisited within their time constraints.

Valente *et al.* (2011) proposed a hybrid online/offline approach. The process begins with a pre-processing stage in which user divides the area into sub-areas using a set of continuous lines. Because the system relies on a grid representation, a modified version of Bresenham's line algorithm is applied to convert the continuous lines into discrete grid-aligned lines. The flood-fill algorithm is then used to identify the grid cells contained within each sub-area. In the online stage, UAVs negotiate among themselves, based on their individual capabilities, to determine which sub-area each will cover. Once all sub-areas are assigned, an offline heuristic based on a wavefront planner computes the coverage path for each robot.

Sung Won Cho *et al.* (2021) developed a method to decompose a polygonal area into a grid, where the cell size is defined according to sensor requirements. A key advantage of this approach is its ability to minimize the number of grid cells by rotating the polygonal area before decomposition. Once the grid is generated, it is converted into a graph. The authors also formulated a MILP model using this graph, framing the CPP as a variant of the VRP. Because solving this model is computationally expensive, they developed a Randomized Search Heuristic (RSH) algorithm to produce feasible solutions for large-scale instances. Experimental results show that the RSH achieves a solution with an optimality gap of approximately 0.7%, while requiring significantly less computation time than the commercial solver tested. As the system is centralized, a base station is responsible for computing and distributing the coverage paths to each UAV.

Sawhney *et al.* (2008) investigated Multi-UAV CPP in hostile terrain, where certain regions must be avoided because UAVs risk detection by hostile observers. If detected, they may be downed. To mitigate this, the authors introduced a metric based on per-time visibility, which combines the visibility gained at a terrain point with the time required to reach it. This metric aims to reduce total path duration by maximizing visibility gained per unit of time. The environment is modeled as a 3D grid. The Multi-UAV CPP system begins by partitioning it into a second 3D grid that identifies safe regions where navigation can occur without detection. In some cases, the number of safe partitions exceeds the number of available UAVs, resulting in unvisited partitions and uncovered terrain points. A greedy algorithm, guided by the proposed metric, determines the sequence of waypoints that each UAV follows within its assigned

sub-area.

Santin *et al.* (2021) proposed and compared four metaheuristics for the multiobjective location-routing problem in complete area coverage missions using heterogeneous UAV fleets. The study explored interactions between an exact method for solving the MILP model and several Multiobjective Variable Neighborhood Search (MOVNS) metaheuristics. The solution process begins by clustering rows and recharge stations, which represent the environment. The exact method is first applied to define initial paths within each cluster, after which, a metaheuristic refines the solution and assigns the paths and clusters to the UAVs. The proposed metaheuristics decompose the original problem into simpler subproblems that are easier to solve, and the exact method is reapplied to generate paths within subgraphs or to improve intermediate solutions. Each MOVNS strategy interacts differently with the exact method. In the first approach, MOVNS divides the coverage area among the UAVs and uses the exact method to generate routes, in the second strategy uses the exact method is used only to construct the initial solution, while in the other two it is applied it during specific stages to enhance solution quality.

Kwon and Thangavelautham (2020) developed a simulation to investigate how multiple agents cooperate to cover basic geometric 2D grids representing open areas. Their CPP formulation relaxes the constraint of complete coverage. The agents do not communicate directly, but use pheromones or markers to indicate visited grid cells, while provides a central station information on regarding total area coverage, elapsed time, and global directional references. The authors proposed an Artificial Neural Tissue (ANT) control algorithm as the core of the central station to handle the simulated CPP tasks. Since the ANT is an artificial neural network controller with a sparse architecture, it requires training. The system was trained using a GA, after which the central station could send movement commands to each agent at every simulation step. Their solution produced path plans with linear or quasilinear time complexity, where time was measured based on the number of robot time steps and the number of open grid cells to be covered.

Kumar Gupta *et al.* (2017) proposed a CPP algorithm that generates M possible solutions for surveying a known region with obstacles using M cooperative UAVs. The algorithm employs cell cost and cell selection functions to construct a list of candidate paths. Out of those M paths, the best solution is selected based on an objective function that prioritizes balanced workload distribution among the UAVs. Additionally, the algorithm applies a spiral search strategy to ensure complete coverage of all cells. The approach also accounts for potential UAV failure: when a failure is detected, the algorithm automatically reassigns the remaining uncovered region to the other UAVs. The path to the newly assigned area or to return to the base station is computed using the A* algorithm.

Collins *et al.* (2021) proposed the Scalable Coverage Path Planning (SCoPP) method, designed for efficiently managing a large number of robots or large map environments. The central idea is to apply iterative clustering to partition the discretized AOI into regions of equal mass. The system requires the user to specify the number of robots, their initial positions, and the AOI, represented as a polygon defined by a set of geographic coordinates (latitude and longitude). The polygon is then converted into a grid, where evenly spaced points define the perimeter of each cell. These points are grouped using Lloyd's clustering algorithm, independent of their parent cell, and each cluster is assigned to a robot. This process may generate conflicting cells that do not belong exclusively to a single cluster. Such cells are resolved through an auction mechanism, where assignment is based on the distance between a robot's initial position and its nearest currently assigned cell, ensuring a balanced workload. Finally, the system computes the route through the cells with the objective of minimizing energy consumption. The planning problem is solved using a KD-tree algorithm, which modeling it as a nearest neighbor problem in which the next waypoint is the closest cell to the current position.

Skorobogatov *et al.* (2021) claimed to have developed the first ready-to-use approach for dividing an area of arbitrary complexity into a variable number of parts. Their solution separates an AOI based on the number of UAVs, the coverage requirements assigned to each UAV, and, optionally, the initial position of each vehicle. The system applies a constrained Delaunay triangulation from the *sect* library as the convex division function. Mission parameters and initial positions are used to generate a list of sites, where each site is defined as a tuple containing the coverage autonomy of a UAV and, optionally, its starting position. The division function uses these sites to determine how the input polygon is split. A coverage trajectory is then generated for each sub-polygon according to the mission parameters, and these routes form the final output. The sub-area coverage algorithm does not minimize time or path length but ensure compliance with restriction on course, track separation, and the required entry and exit sides of the polygon.

Mickey Li, Richards, and Sooriyabandara (2020) proposed a solution for Reliability-Aware Multi-Agent Coverage Path Planning (RA-MCPP) using Integer Linear Programming (ILP). The problem was modeled in a similar way to an MTSP, which has previously been solved with GA. Building on this, the authors extended their work in Mickey Li, Richards, and Sooriyabandara (2021b) to solve the same model using a GA. In the proposed RA-MCPP, the coverage planner generates paths while accounting for the probability of failure-prone UAVs, thereby maximizing the likelihood of mission completion within the deadline. A Markov reliability evaluation with a constant failure rate was applied to determine whether agents survived at each step of the path planning process. Experimental results showed a reduction of up to 26.81% in computation time

for achieving solutions of similar quality.

Hassan, Mustafic, and Dikai Liu (2020) developed the Decentralized Predator-Prey-based Approach to Adaptive Coverage Path Planning (Dec-PPCPP). In this method, each agent iteratively determines the next best neighboring cell or waypoint to visit based on environmental readings and information received from other robots. Since the approach performs one-step local planning, it is well suited for real-time coverage and adapts quickly to obstacles. The next waypoint is defined by the UAVs using four reward functions that evaluate uncovered and obstacle-free neighboring targets. The algorithm follows a greedy strategy by selecting the neighbor with the highest reward. The Stationary Predator Avoidance Reward (SPAR) promotes orderly global motion of the prey, preventing it from moving back and forth between regions. In the Dynamic Predator Avoidance Reward (DPAR), dynamic predators (other robots) influence local motion only when they are close to the prey. As a result, the DPAR encourages the prey to move toward uncovered neighbors farther from the predator. Path smoothness is promoted by the *Smoothness Reward*, which assigns higher rewards to paths with fewer turns, thereby reducing accelerations and decelerations and improving energy efficiency. The *Boundary Reward* further supports natural motion by encouraging agents to cover targets located near already visited areas. If a robot reaches a dead-end with no uncovered and obstacle-free neighbors, a point-to-point planner computes a route to the nearest uncovered target. Additionally, when a robot detects a moving obstacle within a certain range, it temporarily halts coverage to avoid collisions and resumes coverage once the obstacle is no longer a threat.

Zanliang Dong and Chunhui Liu (2021) addressed Multi-UAV CPP using an RL method called Deep Double Q-learning Networks (DDQN). Their approach learns a global policy to control a team of UAVs under power constraints in coverage tasks. The method balances the coverage workload while minimizing path length. In addition, the control framework does not require inter-vehicle communication or prior knowledge of the task area to autonomously and cooperatively plan coverage paths. Since the DDQN method requires a reward function, the authors designed one that promotes collaboration by increasing the reward when the coverage rate improves compared to the previous step. Illegal actions and overlapping coverage reduce efficiency by consuming additional battery capacity, which decreases the reward. During coverage, each UAV transmits its Global Positioning System (GPS) location to a dispatch center, which computes the reward function and determines the next move for each agent.

Zhe Zhang, Jian Wu, and He (2019) proposed a method for disaster inspection using UAVs. In their problem, there are key areas that have to be inspected. As the UAVs fly at fixed altitude and speed, using topography data, the system applies A* (A-star) algorithm to define paths between key areas that do not require flight over the UAV ceiling height. The authors used the Gradient Descent optimization algorithm to

define the visiting sequence of the key areas. This step aimed to maximize coverage and flight time. Note that flight time must be the biggest possible in disaster scenarios. Secondary disasters often occur within 72 hours after an earthquake. Considering that, the authors used the Tabu Search to define the schedule of UAVs for continuous patrol in the disaster area. The system obtained departure and return times for the UAVs in these schedules.

In Sung-Won Cho *et al.* (2022), the authors enhanced the method introduced in Sung Won Cho *et al.* (2021) by developing a new MILP model that reduces computational effort through the use of angular relationships between adjacent grid cells. Unlike the earlier work, the environment is represented with a hexagonal grid. However, in this version the polygon discretization does not aim to minimize the number of cells. The optimization is performed by considering all UAVs operating simultaneously rather than partitioning the environment and planning coverage for each sub-region separately. Because the new MILP model requires less computational time, it can be solved with a commercial solver (Gurobi 8.0.1) even in large scenarios. The authors reported that, for the tested hexagonal grid maps, the search times of the deployed UAVs were reduced by an average of 10.59%, a result attributed to the increased flexibility of hexagonal decomposition, which provides more movable nodes and enables smoother paths.

Shao *et al.* (2022) explored the multi-UAV multi-area coverage path planning problem, in which disjoint areas must be covered, and one or more UAVs can be assigned to each area. The authors developed a collaborative exploration model constrained by the UAVs' energy capacity. Their approach is innovative because it simultaneously considers the inter-area visiting order of multiple UAVs and the intra-area coverage paths. This combined planning strategy minimizes the overall execution cost while respecting battery limitations. Each area is divided into a series of rectangular subregions according to the UAV's battery capacity, which simplifies the CPP and reduces computational effort. The authors proposed the Bipartite Cooperative Coevolution (BiCC) algorithm to solve the problem, which includes an inter-area component that determines the visiting order between areas and an intra-area component that plans the coverage path within each area. These components are integrated through cooperative coevolution to ensure global minimization. In addition, a heuristic solution based on the 2-opt algorithm was developed to provide a reliable result in a short time for emergency scenarios.

Luna *et al.* (2022) developed a framework for multi-UAV CPP based on bin-packing heuristics combined with optimization methods. Given a set of UAVs, their constraints, and a list of waypoints representing the environment, the proposed BINPAT algorithm computes an optimal solution for dividing the coverage task. However, the algorithm does not determine the minimum number of UAVs required. Instead, it allocates work according to input weights assigned to each UAV. These weights are optimized

using Powell's algorithm, which may result in excluding some UAVs from the solution by assigning them a weight of zero. The authors concluded that their algorithm consumes fewer computational resources compared to other approaches in the literature. In addition, they presented a complete system architecture, including communication modules, hardware components, and a user interface for real-time operation.

Xiaoxiao Yu *et al.* (2020) developed two approaches for multi-UAV multi-area coverage path planning. The first approach uses an exact MILP model to assign multiple UAVs to cover scattered regions. In this method, each region is fully assigned to a single UAV. A heuristic initially distributes the regions among the UAVs, taking into account both travel cost and region coverage cost. After the assignment, the Ant Colony (AC) algorithm is applied to minimize the cost of reaching each region. The authors also addressed the issue of imbalanced time consumption in scenarios with super-regions by proposing an alternative method. Instead of the initial heuristic, the Real-Time Reservation Partition algorithm divides large regions into smaller ones and allocates them to different UAVs. Once assignments are defined, a ST algorithm generates intra region coverage paths for both approaches.

Sanna, Godio, and Guglieri (2021) addressed the CPP problem for a fleet of AI-driven K UAVs, considering congestion, collision avoidance, and path efficiency. Their approach begins by applying the K-means algorithm to divide the grid area into K clusters. These clusters are used to generate sub-areas through Voronoi Diagrams, and each sub-area is assigned to a specific UAV. An ANN is then trained and embedded in each agent. Based on the agent's internal state and environmental readings, the ANN selects the next movement direction from north, south, west, or east. Due to the limited sensing range of each agent, unexplored points can exist outside their field of view. In such cases, the authors introduced a modified A* algorithm that computes short paths and favors routes passing through other unexplored cells. If this algorithm fails to converge, the classical A* algorithm is used as a fallback.

Yao, Huixia Wang, and Ruiguang Hu (2021) modeled the CPP problem as a traveling salesman problem, adapting a classic GA to improve the performance in this context. In their approach, the chromosome encodes both the entire area and the UAVs responsible for each cell in the grid, which limits scalability to larger environments. The main contributions of the authors were the development of the zero-adding and negative-number chromosome coding methods. In the zero-adding method, each UAV is assigned a row the size of the entire grid, where unassigned cells are filled with zeros. The negative-number method offers a more memory-efficient solution by representing each UAV's path, with negative values used to partition the paths. Other components of the GA follow standard techniques, including roulette selection, single-point crossover with partial mapping hybridization to repair invalid chromosomes, and swapping mutation. The algorithm also applies evolutionary reversal to evaluate whether the reversed

chromosome improves the solution compared to the original.

Xiaolin Zhou *et al.* (2021) proposed a framework based on Deep Reinforcement Learning (DRL) for multi-robot CPP. The system is decentralized, and each agent includes modules for map management, communication, motion control, and path generation. Communication among the UAVs enables agents to share their positions, update the map by adding or removing obstacles, avoid collisions, and adjust their actions in real time. The motion control module relies on sensing and communication to resolve movement conflicts and handle obstacle avoidance. Movement decisions are made by a path planner based on Deep Q Network (DQN), allowing agents to learn appropriate actions through reward functions. The system uses a three-part reward structure: the first penalizes agents that leave the work area, the second rewards agents that complete their coverage tasks, and the third encourages movements toward directions with higher expected coverage gain, which helps accelerate algorithm convergence.

Huan Wang *et al.* (2021) presented the Shadow Following (SF) algorithm for energy-constrained single-robot CPP. In this problem, UAVs perform part of the coverage task, return to recharge their batteries, and resume the operation. The SF algorithm uses this recharge requirement to generate return paths that pass through unvisited areas, called shadow paths. This strategy improves both energy utilization and coverage efficiency. The authors also developed an obstacle-surrounding algorithm that reduces overlap between the forward and shadow paths while ensuring obstacle avoidance. Following this, they proposed a load-balancing algorithm for multi-robot scenarios. This algorithm determines the number of UAVs needed to complete the coverage by brute-force search over the routes generated by the SF algorithm, taking into account the lowest energy capacity among the available UAVs.

Apostolidis *et al.* (2022) extended the DARP algorithm from previous work in Kapoutsis, Chatzichristofis, and Kosmatopoulos (2017) and incorporated it into a framework for multi-UAV CPP. The DARP algorithm addresses the MCPP problem by dividing it into multiple single-robot CPP problems. At each iteration, it constructs part of an assignment matrix until all cells are labeled to correspond to a unique UAV. The original version of DARP ensures balanced area division, and it was improved to support proportional allocation of the regions of interest. Once each UAV has an assigned area, an ST algorithm for CPP defines the route for each robot. Since DARP operates in a grid environment, the authors proposed a method to reduce the number of cells representing the area without losing fidelity. This reduction impacts the number of path turns and contributes to energy efficiency. The framework also includes a user interface for mission creation, execution, and management, as well as a communication module for integration with commercial UAVs.

Bouras, Bouzid, and Guiatni (2022) created a multi-UAV CPP architecture to estimate gas distribution in an AOI. First, the authors created a new exact cellular de-

composition to transform the polygonal AOI into a Voronoi Diagram. The method splits the area considering a probabilistic sensor's footprint model. Besides the discretization, the authors proposed two methods to allocate the area among the UAVs. The first method used the K-means algorithm to define balanced groups of Voronoi cells. After that, a Traveling Salesman Problem (TSP) model solved by GA was used to define the intra-region routes. The second approach modeled the entire problem as a VRP and solved using the Simulated Annealing (SA) algorithm. Both approaches presented good results. However, the sum of the traveled distances by all UAVs in all cases is lower in the VRP solution.

Day and Salmon (2021) presented a persistent search and retrieval framework for stochastic target scenarios, comparing four coverage strategies: random bounce method, in which each UAV moves in a straight line until reaching the edge of the environment, then selects a new direction at random to continue toward another edge. The second approach uses a lawnmower pattern, where UAVs systematically cover the entire area. In the third and fourth approaches, the environment is first divided into regions using a Voronoi diagram, and each region is randomly assigned to a UAV. Within each sub-region, coverage is performed using either the random bounce or the lawnmower method. A key performance concern in this application is the average time since the last search. Results showed that the pure lawnmower strategy performed poorly in this regard. The authors also optimized the placement of collectors and recharge stations throughout the environment.

Wenjian Hu *et al.* (2023) explain that most multi-agent approaches suffer from poor cooperation, which can lead to overlapping trajectories, uncovered areas, or even collisions in uncertain and complex environments. These issues result in long task completion times and low coverage efficiency. The authors address this challenge by proposing a distributed online cooperative CPP method for multi-UAV systems, aimed at minimizing task completion time. Their approach enables UAVs to quickly adapt to unknown obstacles and respond to complex emergencies, such as UAV malfunctions or communication interruptions. The UAVs share their knowledge of the environment to achieve real-time global exploration in a cooperative manner. Using a Q-learning algorithm, the method generates coverage paths online by minimizing task duration while avoiding overlap, missed areas, and collisions. Their method was validated in simulations, demonstrating efficient full coverage even under emergency conditions.

Lin Li *et al.* (2023) propose a multi-robot CPP method that accounts for Dubins vehicle constraints, characterized by a minimum turning radius, fixed forward speed, and nonholonomic kinematics. The authors formalize the problem as a combinatorial optimization model and present an exact MILP formulation to compute optimal paths, which becomes computationally expensive for large instances. As a more scalable alternative, they design a heuristic algorithm that partitions the environment into connected

subregions and assigns them to individual robots. This approach ensures that each robot follows a continuous, curvature-constrained Dubins path while maintaining balanced workload distribution across the team. Experiments on benchmark environments demonstrate that the heuristic achieves near-optimal coverage and workload balance compared to the MILP, while scaling efficiently to larger problem sizes.

Govindaraju *et al.* (2023) address the challenge of coordinating multiple weeding robots in agricultural paddy fields. The field is modeled as a set of rows with fixed plant spacing, where each robot can cover a row in a straight pass. The authors introduce a method in which distinct, non-overlapping rows are assigned to each robot. The algorithm organizes the robots into synchronized formations, with synchronization points located at the end of each column where robots can reconfigure, join, or leave the team. This design increases tolerance to failures and enables dynamic team resizing. Although all paths are computed offline, the robots share information for synchronization, which characterizes the system as multi-agent. Their solution minimizes both the total path length and the number of turns, reducing energy consumption while balancing workload across robots. Experimental results show that, with 15 robots, the average path length per robot is only 65% of that in standard multi-robot trapezoidal methods, confirming the approach as both efficient and scalable.

In Tang and Hang Ma (2024), the authors extend the ST coverage method to handle graphs in which the supercells, typically composed of four grid cells, used to compute the Minimum ST can be incomplete. Their approach applies a local search strategy that balances the workload among robots and reduces overall task completion time (makespan). This local search uses three simple operators to adjust robot coverage regions. A robot may slightly expand into an area already assigned to another robot, creating limited overlap that improves workload distribution. Overlapping regions covered by multiple robots are trimmed to reduce redundancy and minimize wasted effort. Small regions can also be transferred from one robot to another, allowing one region to grow while the other shrinks, which helps balance path lengths across the team. Experimental results show that the method consistently outperforms existing approaches, achieving faster completion times and higher efficiency, particularly in large and complex environments.

Wan, He Yang, and Meng (2023) proposed a three-dimensional coverage path planning method for multiple UAVs operating over large areas while avoiding obstacles. The approach begins by applying Boustrophedon cell decomposition to divide the target space into independent free cells. These cells are then covered using a zigzag strategy that is optimized to minimize overlaps and turning costs. The A* algorithm connects different cells by creating the shortest transfer paths that avoid no-fly zones. These two-dimensional paths are then transformed into three-dimensional trajectories through interpolation, maintaining a constant flight altitude relative to the terrain. Navi-

gation between Boustrophedon cells is treated as a multi-traveler optimization task and is solved using both genetic algorithms and simulated annealing. The latter method demonstrates faster convergence and produces shorter, more balanced routes.

Choton and Prabhakar (2023) developed a three-step framework for optimal multi-UAV CPP in agricultural fields that incorporates robot motion dynamics. First, the agricultural field is divided into convex polygonal sub-areas using trapezoidal decomposition. These sub-areas are then optimally assigned to the robots in a balanced manner using a dynamic programming approach. Second, each assigned region is covered using a lawnmower sweep algorithm that ensures complete coverage. Finally, the high-level paths are converted into feasible robot trajectories by modeling their motion with Dubins dynamics, generating control inputs that are consistent with real robot constraints. The authors validate their approach through simulations in various field geometries, demonstrating improvements in mission time and balanced robot utilization.

Nair, Dileep, and Guruprasad (2024) created a robust online framework for multi-robot simultaneous exploration and CPP that ensures complete, non-overlapping workspace coverage. In addition, the method can recover from failures, allowing a new robot to take over any unfinished portion of a task. The approach begins by dividing the environment using Manhattan Voronoi partitioning, which assigns each robot a unique region to explore and cover efficiently using the ST Coverage algorithm. The system handles failures using two adaptive strategies. The first is Voronoi cell reallocation, where the unvisited areas of a failed robot are redistributed among the remaining robots by updating the partition boundaries. The second is robot replacement, where a new robot seamlessly takes over the failed robot's region. Unlike prior methods, this approach dynamically adjusts workspace allocation to maintain uniformity, minimize redundant coverage, and conserve energy, while also supporting scalability when new robots are added.

Zeng, Tong, and Xia (2024) proposed a cooperative multi-UAV coverage search algorithm that integrates Distributed Model Predictive Control with a simplified Differential Evolution optimizer. The global coverage task is divided into multiple short time intervals, enabling each UAV to plan its next move online at each time interval. A dynamic reward function guides behavior by balancing the objectives of maximizing area coverage and minimizing energy usage, while also encouraging re-entry if a UAV leaves the mission area. UAVs exchange and merge their local maps when within communication range, allowing them to replan based on both its own data and information received from nearby units. This decentralized strategy lowers computational demands, enhances robustness, and reduces redundant coverage.

Zikai Wang *et al.* (2025) introduced MAC-Planner, a centralized framework for online multi-robot CPP that integrates task allocation and path planning. The central controller collects the robots' positions and local coverage status, then dynamically as-

signs regions and computes their next moves. Large terrains are initially divided into fine square cells that match the robots' footprint. These are then recursively grouped into a coarse map, where the uncovered areas are subdivided until the number of coarse cells slightly exceeds twice the number of robots. Each coarse cell is represented by its number of uncovered fine cells and its centroid, making coverage estimation computationally efficient. Task allocation is carried out at the coarse level using the K-means algorithm to generate an initial distribution among robots. A pairwise optimization algorithm then refines this assignment by minimizing the maximum coverage cost, ensuring balanced workloads. Path execution is guided by a coverage policy that produces systematic back-and-forth motions within each assigned coarse region. If a robot encounters a dead-end, a recovery mechanism either backtracks it to the nearest uncovered cell or reallocates new regions, ensuring complete coverage of the terrain.

Gong, Hyunbin Kim, and Lee (2024) proposed a multi-robot CPP approach based on Boustrophedon decomposition that is capable of handling robot failures. Instead of reassigning the failed robot's path to a single neighboring unit, the method uses a propagation-based strategy to iteratively redistribute the workload among nearby robots. This approach seeks to maintain task balance across the team. The failed robot's path is initially divided among its closest neighbors, and the additional workload is then gradually propagated through an adjacency tree structure. This prevents a single robot becoming overloaded and ensures complete coverage of the area. The method is also designed to be computationally efficient, with each propagation cycle taking only a fraction of a second, making it well-suited for real-time applications.

Mengyang Wang *et al.* (2025) proposed a collaborative 3D CPP method for multiple fixed-wing UAVs performing complete coverage missions in complex environments. Each UAV's trajectory is divided into three components: operation paths (where coverage occurs), entry paths (to reach the coverage area), and transition paths. The motion constraints of fixed-wing UAVs are addressed using Dubins curves to compute the transition paths. Additionally, a greedy algorithm allocates UAVs to different sub-areas based on factors such as area size, shape, and threat level. The sequence of operation paths is optimized using a dynamic programming algorithm. According to the authors, this method avoids the randomness and inefficiency commonly associated with heuristic approaches. Finally, the optimized 2D coverage paths are transformed into terrain-aware 3D trajectories that account for the UAVs' flight limitations.

Mo and Zhiyun Lin (2024) implemented an online multi-robot CPP approach that adapts in real time to imbalances in each robot's assigned area. The method begins by partitioning the environment into subregions, assigning each robot a specific area to cover. Within each subregion, robots use a variant of the ST Coverage algorithm that incrementally plans paths using a sliding window, enabling flexible and dynamic adjustments. Unlike DARP Kapoutsis, Chatzichristofis, and Kosmatopoulos (2017), which

performs a one-time static division of the area and assigns fixed regions, this approach not only partitions initially but also continuously adjusts subregions boundaries during execution. This allows the system to reallocate tasks in response to differences in robot speed or workload.

Boquan Zhang *et al.* (2024) developed a collaborative multi-UAV CPP method using multi-agent deep reinforcement learning. The problem is formulated as a multi-objective optimization task, where each UAV operates based on partial local observations while exchanging limited information with neighboring agents. The authors propose an enhanced algorithm, ε -MATD3, an extension of the method by Ackermann *et al.* (2019). This approach employs two critics to mitigate overestimation bias and introduces a decaying exploration coefficient (ε) to encourage exploration in early training and promote exploitation in later stages. The UAVs learn continuous control policies aimed at maximizing coverage while avoiding collisions, no-fly zones, and redundant overlaps. The proposed method is evaluated against the approaches of Lowe *et al.* (2017) and its improved variant from Ackermann *et al.* (2019), and outperformed both methods across all evaluation metrics.

3.3 CONCLUSION

This chapter introduced the reader to the CPP problem, outlining its relevance in autonomous robotics and the main approaches used to solve it. It also presented a survey of MCP, drawing from 43 works that explored the state of the art in the field. This survey helps highlight how the problem scales in complexity when multiple agents are involved and reveals key trends, common assumptions, and emerging research directions.

The reviewed studies modeled the environment using polygons, graphs, grids, or hybrid representations. A common assumption was the use of constant flight speed and altitude, which is reasonable during the optimization process. Several works addressed energy limitations by introducing constraints such as maximum flight time or a limit on the number of cells or waypoints visited. In addition to minimizing path length or task completion time, some approaches incorporated workload balance, a relevant feature in multi-robot systems. This remains a notable trend, as most recent works emphasize workload balance as a key objective function, with some also performing explicit benchmarking against earlier approaches (Hu, W. *et al.*, 2023; Zhang, B. *et al.*, 2024).

Most of the papers reviewed do not consider the dynamics, kinematics, or acceleration of UAVs. Although this simplification is understandable due to the optimization focus of CPP methods, it introduces limitations. For example, it can result in paths that may be infeasible in real-world scenarios. A classical sweeping coverage strategy based on straight lines is harder to execute and less efficient for UAVs. In contrast, a sweeping

strategy that uses smoother turns is more practical and efficient. Recent studies from 2023 and later have started to include kinematic models and Dubins dynamics (Li, L. *et al.*, 2023; Choton; Prabhakar, 2023; Wang, M. *et al.*, 2025). By accounting for the minimum turning radius, acceleration constraints, and 3D terrain features, these models improve path feasibility and generate trajectories that are more realistic and applicable in practice. However, even straight-line solutions can still produce feasible results when post-processed with a trajectory smoother that accounts for the UAV dynamics.

Some online methods are capable of avoiding obstacles while also performing optimization. However, the reviewed papers did not describe the techniques used for obstacle detection. Additionally, since the actual flight path often deviates from the planned one, it would be beneficial to at least predict potential collisions when paths intersect or come close. It is attainable using a speed planner as presented in Gonçalves *et al.* (2013) and Rezende, Gonçalves, and Pimenta (2021).

From 2015 to 2022, only four works implemented decentralized solutions: three applied AI techniques, and one used a greedy algorithm. In contrast, six papers were produced in just three years (2023–2025), and all of them focused on the use of AI. This sharp growth reflects the increasing availability of computational power and the broader accessibility of advanced models for both researchers and the general public. As Industry 4.0 becomes a reality and 5G connectivity becomes widespread, decentralized approaches are expected to expand further across various domains. These systems typically rely on communication among agents or between agents and a central station. Therefore, advances in internet infrastructure will directly support the deployment of more complex multi-agent frameworks. While only five papers explicitly addressed fault tolerance, decentralized systems are inherently more adaptable to unforeseen situations. Moreover, their distributed decision-making capabilities can produce satisfactory solutions in shorter computation times when compared to exact, centralized methods. A limitation of the reviewed works is that their multi-agent system approaches have been validated only in simulation. It would be valuable to assess whether such systems can achieve comparable results in real environments, where communication and navigation performance may degrade.

A critical area identified in the initial review phase, which concluded at the end of 2022, is the need for robust frameworks capable of detecting UAV failures and dynamically reallocating tasks among the remaining agents.

Some works considered fault tolerance during offline path planning by assigning a probability of failure to the UAVs. Based on this probability, the routes are computed with redundancy. The planning process assumes that a UAV may fail at a certain point, and the system estimates the number of UAVs or battery charges needed to complete the coverage task under such conditions. This approach may require additional UAVs to ensure full area coverage. Since more UAVs are allocated, it is likely that some will

complete their tasks with a significant amount of battery remaining. These UAVs, with higher remaining energy, can then take over the route of a failed one.

More recent multi-agent systems have begun to address critical gaps by enabling real-time monitoring of task progress, vehicle health, battery levels, and energy consumption rates, which supports more resilient and adaptive mission planning. In contrast, earlier studies remained in preliminary stages and often lacked the capability to process environmental data in real time, such as the presence of dynamic obstacles or nearby UAVs. Building on these capabilities, recent studies Nair, Dileep, and Guruprasad (2024) and Wenjian Hu *et al.* (2023) introduce online recovery strategies that reallocate tasks when a UAV fails, often through Voronoi partitioning or adaptive exploration techniques. These frameworks significantly enhance the robustness of UAV teams and help mitigate mission risks, particularly in large-scale deployments.

NFZs are often treated as secondary elements in polygonal, graph, and grid representations. The process of incorporating NFZs is straightforward. In polygonal approaches, they are represented as additional polygons within the AOI. In graph-based representations, NFZs are excluded by omitting their corresponding vertices. In grid-based models, specific values indicate occupied or restricted cells. Accounting for NFZs can be particularly relevant in urban environments, where buildings affect the UAV's flight altitude. In agricultural settings, areas where other machinery is operating may be treated as NFZs. Similarly, in surveillance or monitoring applications, empty regions that do not require coverage can also be designated as NFZs.

Only six works considered battery recharging in their methods, and just one addressed the use of multiple recharge stations. In that study, the authors selected a set of candidate locations and, during path planning, determined both the number of UAVs at each station and their respective starting positions. However, the most common strategy is to assume a single recharge station. Under this assumption, all UAVs start and return to the same location for battery recharging. Although Avellar *et al.* (2015) did not incorporate battery recharging into their method, they proposed including it in future developments. Similarly, Santin *et al.* (2021) plan to consider the station's recharging capacity, drawing inspiration from the system proposed by Ammous *et al.* (2019).

The systematic review also investigated whether wind effects were considered in path planning. Surprisingly, the reviewed works neglected this factor, despite its well-known influence on flight stability, energy consumption, and overall mission reliability. This oversight reveals a relevant gap in the current literature and suggests an opportunity for future research to enhance the realism and robustness of UAV operations.

Additionally, none of the reviewed approaches considered collaboration with other autonomous systems or human agents operating in the same environment. As the field advances toward Industry 5.0, integrating human-robot interaction and cooperation among heterogeneous systems will become increasingly important for achieving

higher levels of autonomy, adaptability, and mission efficiency in complex and dynamic environments.

Overall, the field of MCPP is evolving beyond purely theoretical optimization. Recent trends point toward more resilient, realistic, and domain-adapted solutions that integrate UAV dynamics, online adaptability, and cooperative efficiency. Continued research in these areas, particularly in real-world validation, heterogeneous agent collaboration, and integration with modern infrastructure, will be key to advancing practical applications of multi-agent coverage planning in increasingly complex environments.

4 PROPOSED APPROACH

Unmanned Aerial Vehicles (UAVs) are increasingly employed in environmental monitoring due to their flexibility and ability to cover wide areas, including remote or hazardous environments. However, their effective deployment remains challenging, requiring coordinated strategies for coverage, energy efficiency, and long-term sensing. This chapter introduces a scalable UAV-based monitoring framework that leverages a hierarchical fleet architecture: long-range UAVs perform wide-area surveys and deploy smaller platforms for persistent, fine-grained monitoring beneath forest canopies. The chapter first outlines the system architecture, then presents the Wind-Informed Sweep Planner (WISP), a lightweight, wind-aware coverage path planner that optimizes energy use and task allocation across the fleet. Finally, it describes a vision-based system that enables small UAVs to autonomously land on natural structures using a hook mechanism, supporting persistent, low-power sensing.

4.1 AREA MONITORING APPROACH OVERVIEW

UAVs play a vital role in area monitoring. Despite recent technological advances, employing UAVs remains challenging. Key challenges include mission-specific UAV design, operational weather conditions, legal requirements for UAV operators, human-computer interaction, and supporting infrastructure. These issues are examined in detail by Kramar *et al.* (2022).

Regarding design, UAVs face technical limitations such as battery efficiency, short flight duration, limited communication range, and restricted payload capacity (Bueren *et al.*, 2015; Kim, J. *et al.*, 2019). Battery and payload constraints also impact onboard sensors, which must be lightweight, compact, and energy-efficient (Tsouros; Bibi; Sarigiannidis, 2019). Together, these limitations reduce the operational range of UAVs. Consequently, a single UAV may be unable to monitor an entire mission area due to limited energy resources (Radoglou-Grammatikis *et al.*, 2020a). A common strategy to overcome these constraints is to divide the area of interest among multiple UAVs, thereby making the mission feasible.

Surveillance missions often require permanent or long-term monitoring of extensive regions. Instead of continuous sweeping, UAVs can operate as static sensors when landed at strategic locations. This approach is particularly useful in scenarios such as wildfire management, where UAVs may navigate to positions beneath tree canopies. For example, positioning a UAV upwind allows for smoke sampling without the need to locate plumes during coverage flights. Stationary sensing in such locations conserves battery power and enables extended monitoring over large areas.

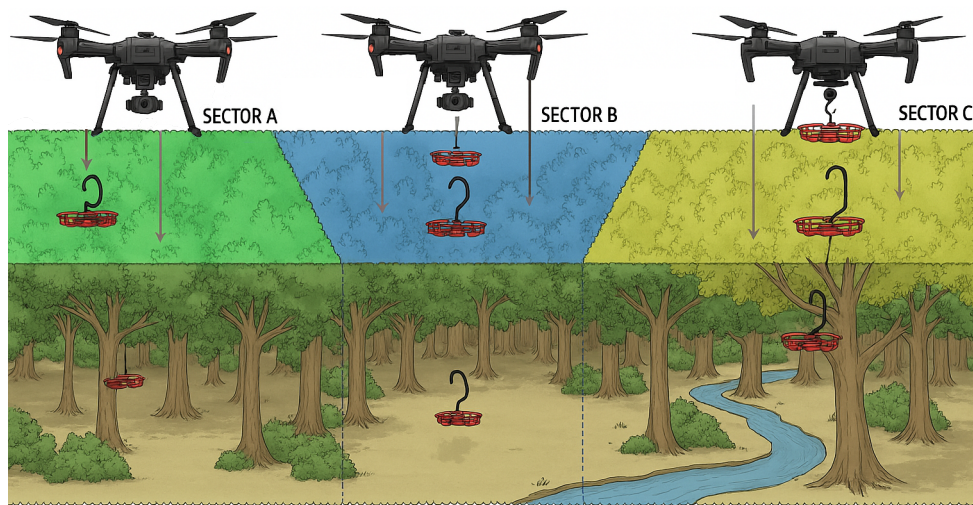
UAVs with perching capability conserve energy by remaining landed while collecting data (Kirchgeorg; Mintchev, 2022; Kitchen *et al.*, 2020; Hang *et al.*, 2019). Such

approaches typically require a grasping mechanism to secure the UAV to the landing site, which increases weight and mechanical complexity, particularly for smaller platforms. An alternative is to use a hook-based mechanism, offering a simpler, lighter solution that achieves the same objective with reduced complexity.

Figure 19 illustrates a conceptual approach for scalable and intelligent environmental monitoring using a hierarchical UAVs fleet. In this architecture, a set of long-range UAVs equipped with large payload capacities are deployed to conduct wide-area surveillance missions over regions of interest such as forests, orchard fields, and commercial timber plantations. These high-endurance UAVs are equipped with multi-modal sensing capabilities, such as multispectral cameras, LiDAR, thermal sensors, or hyperspectral imaging systems to collect comprehensive environmental data.

Since some Area of Interests (AOIs) may be too large to be covered by a single UAV, the proposed approach partitions the area into balanced sub-areas and assigns them across the available vehicles. In Figure 19, this division is illustrated by the green, blue, and yellow sectors. Each UAV is then responsible for covering its assigned sector.

Figure 19 – Hierarchical UAV-based environmental monitoring system overview.



Source: Generated using AI.

Based on sensor inputs, the system can identify regions requiring persistent or high-resolution monitoring. For example, the detection of dry canopies, elevated surface temperatures, abnormal spectral signatures indicating pest infestations, or early-stage wildfire indicators may trigger localized actions. When canopy clearance permits, the larger UAV may deploy a smaller, specialized UAV designed for close-proximity tasks. This child UAV can descend below the tree canopy and navigate among branches, using vision-based localization or LiDAR-based mapping to identify a suitable branch for landing. Once landed, the UAV remains stationary and serves as a sensor node within a distributed environmental sensing network.

To conserve energy, the smaller UAV activates the long-range communication module to transmit data only when critical events are detected (e.g., smoke detection or temperature spikes). In regions where the canopy is too dense for safe descent, the parent UAV records the coordinates for further inspection. Human intervention could then be scheduled, either to manually install a sensor on a tree trunk or to deploy a UAV capable of vertical insertion through dense foliage.

For this vision to become operationally viable, several subsystems must be developed and integrated:

1. **Area Allocation and Coverage Planning System:** This subsystem is responsible for dividing the monitoring area among the available fleet of large UAVs. It must dynamically adapt to real-time environmental conditions and mission goals to ensure efficient and non-redundant coverage.
2. **Autonomous Branch-Landing System for Small UAVs:** The smaller UAVs require a robust mechanism for autonomous landing and stabilization on irregular natural surfaces such as tree branches.
3. **Communication and Coordination Infrastructure:** Communication across the UAV hierarchy is critical. This may include:
 - *LoRaWAN* or *mesh-based radio networks* for reliable, long-range, and low-power data transmission within the UAVs hierarchy.
 - *5G* or *satellite uplinks* for high-throughput communication from central UAVs to ground stations or cloud platforms.

Coordination mechanisms must also manage tasks such as conflict resolution and reallocation of missions in case of UAV failure. They are further responsible for ensuring updates of mission objectives based on real-time sensor feedback and possible human intervention. These functions may be implemented through a cloud-based station or, alternatively, through a decentralized consensus system running embedded directly on the UAVs.

This thesis focuses on the development of two critical subsystems: the *Area Allocation and Coverage Planning System*, and the *Autonomous Branch-Landing System* for small UAVs. The decision to concentrate on these components is driven by both scientific relevance and practical necessity.

First, the *Area Allocation and Coverage Planning System* addresses a key challenge in large-scale UAV operations: efficiently distributing monitoring tasks across a fleet. Its goal is to maximize spatial coverage while minimizing energy consumption. Without a robust allocation and planning mechanism, wide-area monitoring becomes inefficient, leading to excessive resource use and plans that cannot quickly adapt to changing conditions. By investigating novel strategies, this work contributes to more scalable and robust UAV-based environmental monitoring systems.

Second, the *Autonomous Branch-Landing System* tackles a unique and under-explored technological challenge. Persistent monitoring in forested environments often requires sensors to operate beneath the canopy, where conventional aerial platforms cannot sustain stable flight for long periods. This system enables small UAVs to land safely on natural structures such as tree branches, supporting for long-duration sensing, while providing the flexibility to remotely reposition sensors as needed.

4.2 WIND-INFORMED SWEEP PLANNER

The *Area Allocation and Coverage Planning System* referred to in this thesis as WISP, is designed to be computationally lightweight compared with other planning methods. The literature review identified a lack of methods that consider the impact of wind on energy consumption on Coverage Path Planning (CPP). The acronym reflects this design goal. WISP addresses this gap employing a lightweight heuristic to estimate energy costs, which is then used to construct a graph, from which the coverage path is derived.

The method requires as input an AOI and a No-Flight Zone (NFZ), both represented as lists of coordinates (e.g., (0,0), (0,1), (1,1), (1,0), (0,0)), along with the grid size required for discretization. Additional parameters include mission velocity, wind velocity, wind direction, and a set of UAVs defined by their physical characteristics. An altitude function must also be provided to determine elevation at each (x,y) coordinate. For example, the system could retrieve topographic data to determine the altitude Above Mean Sea Level (AMSL) for each grid cell.

As illustrated in Figure 20, the process begins by discretizing the AOI and NFZ using twice the standard grid size. This adjustment accounts for the fact that each cell in the Minimum Spanning Tree (MST) represents four spatial units that must be covered. Each grid cell is represented by its centroid, which serves as a reference point for subsequent operations.

The centroids are then grouped into near-equal clusters using the Constrained K-Means algorithm from Bradley, Bennett, and Demiriz (2000). This clustering ensures a balanced workload across the UAV fleet, with each cluster assigned to one UAV from the provided list.

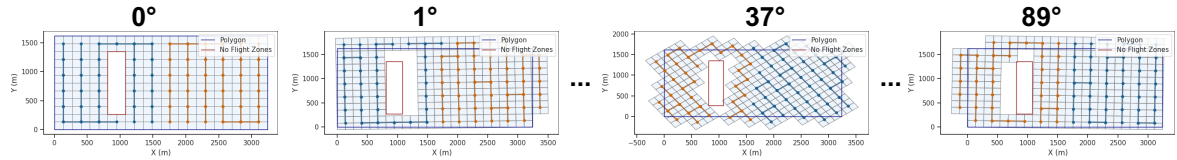
Based on these clustered centroids, a graph is constructed in which each vertex represents a grid cell, and edges connect each cell to its four immediate neighbors to model potential UAV movements. Edge weights are calculated using the wind-aware heuristic, which adjusts the energy cost of traveling between centroids depending on wind directions. The MST of this graph is then computed, and its total cost is recorded.

To optimize coverage, the entire process is repeated with incremental grid rotations from 0° to 89° . A new MST is computed and evaluated for each orientation. In Step 2, the configuration yielding the lowest total cost is selected. Finally, in Step 3, the

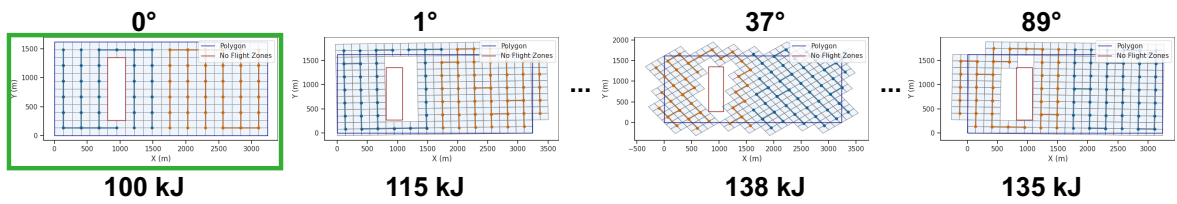
optimal coverage path is generated based on the structure of this MST and the original grid layout.

Figure 20 – WISP's steps overview.

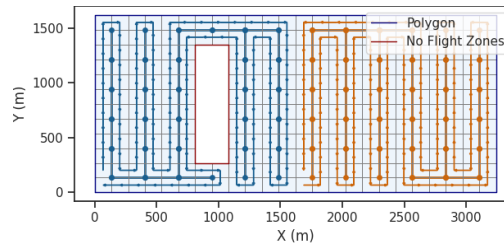
Step 1: Discretization and Minimum Spanning Trees Evaluation



Step 2: Select the Best Minimum Spanning Tree



Step 3: Compute the Coverage Path



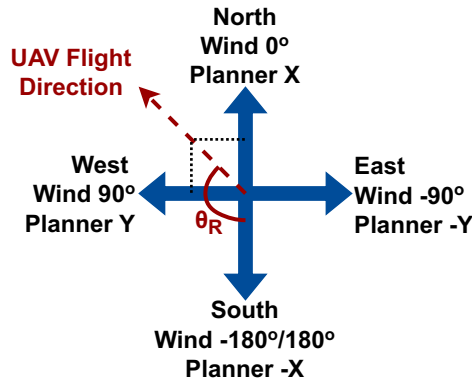
After computing the optimized path, the system transmits it to the module that interfaces the Robot Operating System (ROS) with the UAV's autopilot. This module, known as the Simple Trajectory Controller, acts as a local path planner by sending setpoints to the autopilot and monitoring the UAV's progress to determine when each waypoint has been reached. These setpoints guide the UAV toward the next waypoint while maintaining its heading aligned with the direction of travel.

4.2.1 Energy Heuristic

The heuristic developed in this thesis extends the energy model proposed by Sorbelli *et al.* (2021), while adopting the same relative wind direction framework. In their study, energy consumption was only evaluated at a fixed altitude. This thesis builds upon that research by introducing the capability to plan paths that account for altitude variations. Consequently, terrain features such as mountains, cliffs, and valleys can be incorporated into the planning process, enabling a more realistic estimation of energy consumption.

The first step in estimating the energy required to travel between two grid cells is to compute the relative wind direction. Figure 21 illustrates wind directions in degrees relative to the cardinal points on a compass rose, and provides an example of the relative wind direction θ_R . In this example, the wind originates from 0° , corresponding to a movement toward 180° , and the UAV is flying with a relative wind angle of $\theta_R = 135^\circ$. This indicates that the UAV is flying partially against and partially lateral to the wind direction. For simplicity, this thesis adopts the convention that a wind direction of 180° refers to wind moving toward 180° rather than coming from it. The relative wind direction is computed using Equation (1), which in this case yields $\theta_R = 180^\circ - \text{atan2}(1,1) = 180^\circ - 45^\circ = 135^\circ$.

Figure 21 – Compass rose showing wind directions and an example of relative wind angle θ_R .



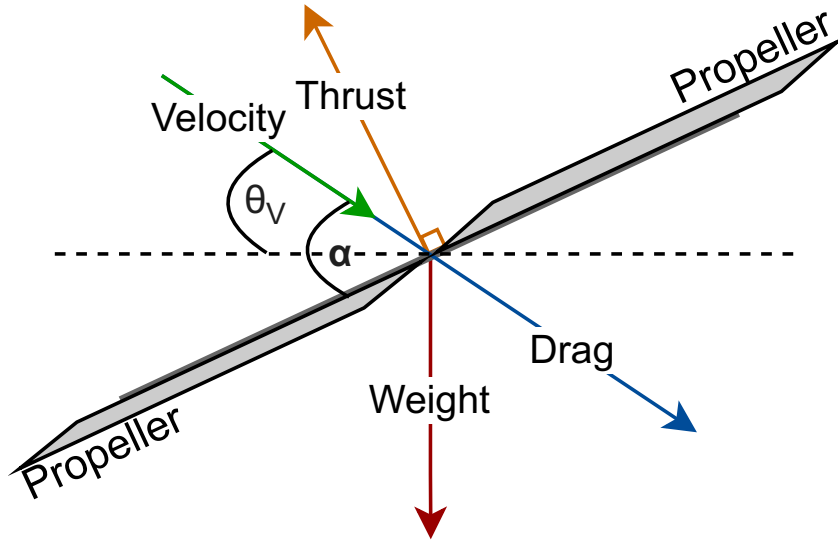
$$\theta_R = \text{Wind Direction} - \text{atan2}(\text{UAV } Y_{\text{Velocity}}, \text{UAV } X_{\text{Velocity}}) \quad (1)$$

where:

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ \frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0 \end{cases} \quad (2)$$

After computing the relative wind direction, the next step in the heuristic is to calculate the required thrust. The approach proposed by Sorbelli *et al.* (2021) considers only the horizontal drag force and the weight of the UAV. In contrast, the version developed in this thesis also incorporates the vertical drag force. Figure 22 illustrates the force and velocity vectors involved in the heuristic computation.

Figure 22 – Heuristic forces and angles demonstration.



First, the horizontal and vertical air velocities are calculated using the UAV's velocity components V_x , V_y , and V_z . The equations $V_{forward}$ and V_{right} represent the velocities relative to the UAV's orientation.

$$V_{horizontal} = \sqrt{V_x^2 + V_y^2} \quad (3)$$

$$V_{vertical} = V_z \quad (4)$$

$$V_{forward} = V_{horizontal} - V_{wind} \cdot \cos(\theta_R) \quad (5)$$

$$V_{right} = -V_{wind} \cdot \sin(\theta_R) \quad (6)$$

$$V_{air_{horizontal}} = \sqrt{V_{forward}^2 + V_{right}^2} \quad (7)$$

$$V_{air_{vertical}} = V_z \quad (8)$$

Then, using the drag force equation, it is computed vertical and horizontal drag forces:

$$F_{drag} = C_D \cdot Area \cdot \frac{\rho \cdot V^2}{2}, \text{ where } \rho \text{ is air density} \quad (9)$$

$$F_{drag_{horizontal}} = 2.2 \cdot \text{Frontal Projected Area from UAV} \cdot \frac{\rho \cdot V_{air_{horizontal}}^2}{2} \quad (10)$$

$$F_{drag_{vertical}} = \begin{cases} 2.2 \cdot \text{Top Projected Area from UAV} \cdot \frac{\rho \cdot V_{air_{vertical}}^2}{2}, & \text{if } V_{air_{vertical}} \geq 0 \\ -2.2 \cdot \text{Top Projected Area from UAV} \cdot \frac{\rho \cdot V_{air_{vertical}}^2}{2}, & \text{if } V_{air_{vertical}} < 0 \end{cases} \quad (11)$$

$$F_{drag} = \sqrt{F_{drag_{horizontal}}^2 + F_{drag_{vertical}}^2} \quad (12)$$

The drag coefficient of 2.2 used in the equations is a typical value for complex shapes and is typically applied to satellites during atmosphere descent (Cook, 1965). Given that the UAV's frame incorporates a variety of shapes, using this coefficient is considered a reasonable approximation for estimating the worst-case drag force.

Then, the required thrust for the flight is given by:

$$T = \sqrt{F_{weight}^2 + F_{drag}^2 - 2 \cdot F_{weight} \cdot F_{drag} \cdot \cos(\frac{\pi}{2} + \theta_V)}, \text{ where:} \quad (13)$$

$$F_{weight} = UAV_{mass} \cdot Gravity \quad (14)$$

$$\theta_V = \arctan2(V_{air_vertical}, V_{forward}) \quad (15)$$

According to Johnson (1994), a reasonable simplification for estimating the power required during flight involving both vertical and horizontal motion is to compute the power for each component separately. This approach involves calculating the thrust T twice: once with $V_{vertical} = 0$ to obtain the horizontal thrust $T_{horizontal}$, and once with $V_{horizontal} = 0$ to obtain the vertical thrust $T_{vertical}$. The total power can then be approximated using the following equation:

$$P = T_{horizontal} \cdot (V_{induced} + V_{horizontal} \cdot \sin(\alpha)) + T_{vertical} \cdot V_{vertical}, \text{ where:} \quad (16)$$

$$\alpha = \arctan2(F_{drag_{horizontal}} \cdot \cos(\theta_V), F_{weight}) \quad (17)$$

$$\alpha = \arctan2(F_{drag_{horizontal}}, F_{weight}), \text{ for forward flight with } \theta_V = 0 \quad (18)$$

The induced velocity $V_{induced}$ represents the air flow passing through the propeller. Johnson (1994) derives the expression for induced velocity during horizontal flight as follows:

$$V_{induced} = \frac{V_{induced_{hovering}}^2}{\sqrt{(V_{horizontal} \cdot \cos(\alpha))^2 + (V_{horizontal} \cdot \sin(\alpha) + V_{induced})^2}} \quad (19)$$

$$V_{induced_{hovering}} = \sqrt{\frac{T_{horizontal}}{2 \cdot \rho \cdot \text{Propeller Disc Area}}} \quad (20)$$

The induced velocity can be obtained by solving the polynomial derived from the implicit equation. Considering $V_{induced} = V_i$ and $V_{induced_{hovering}} = V_h$:

$$\begin{aligned} 0 = & V_i^4 \\ & + V_i^3 \cdot (2 \cdot V_{horizontal} \cdot \sin(\alpha)) \\ & + V_i^2 \cdot [(V_{horizontal} \cdot \cos(\alpha))^2 + (V_{horizontal} \cdot \sin(\alpha))^2] \\ & - \left(\frac{T_{horizontal}}{2 \cdot \rho \cdot \text{Propeller Disc Area}} \right)^2 \end{aligned} \quad (21)$$

One simplification adopted by Sorbelli *et al.* (2021), and also used in this work, is that the total propeller disc area is approximated as the area of a single disc multiplied by the number of propellers on the UAV.

Finally, the energy required to navigate between two cells, is given by:

$$E = P \cdot \text{Time Applying the Power} = P \cdot \frac{\text{Distance Between Cells}}{\text{Mission Speed}} \quad (22)$$

4.2.2 Discretization and Area Allocation

The discretization process begins by rotating the AOI, which includes both the polygon and the NFZ, by an angle of D degrees. After the rotation, the system computes the minimum bounding box of the AOI. Based on the provided grid resolution, two grids are created. The first is a coarse grid that uses twice the specified grid size. The second is a regular grid that uses the original grid size. The coarse grid provides the centroids used for graph construction and MST computation. The regular grid is later used to compute the actual path that follows the structure of the MST.

The method for building both grids is nearly the same. The number of columns and rows is calculated first, and then the lines that define the grid cells are generated. These steps are formalized in Equations (23)–(31). Each grid cell is constructed using four points, which are intersections of the row and column positions, and the cells are stored in a vector.

Each cell in the coarse grid is evaluated to determine whether it meets two criteria. If less than one percent of its area overlaps with the AOI, the cell is discarded. If more than one percent of its area overlaps with the NFZ, it is also discarded. When a coarse cell is removed, all four regular grid cells within it are also removed. As a result, each discarded cell creates an empty spot in the grid vector.

The AOI and both grids are rotated by $-D$ degrees at the end of the process. This restores the AOI to its original orientation while preserving a grid that remains rotated by D degrees relative to the area.

$$\text{Coarse Grid Size} = 2 \cdot \text{Grid Size} \quad (23)$$

$$\text{Coarse Number of Rows} = \left\lceil \frac{\text{Height from AOI'S Bouding Box}}{\text{Coarse Grid Size}} \right\rceil \quad (24)$$

$$\text{Coarse Number of Columns} = \left\lceil \frac{\text{Width from AOI'S Bouding Box}}{\text{Coarse Grid Size}} \right\rceil \quad (25)$$

$$\text{Regular Number of Rows} = \left\lceil \frac{\text{Height from AOI'S Bouding Box}}{\text{Grid Size}} \right\rceil \quad (26)$$

$$\text{Regular Number of Columns} = \left\lceil \frac{\text{Width from AOI'S Bouding Box}}{\text{Grid Size}} \right\rceil \quad (27)$$

$$\begin{aligned} \text{Coarse Row Positions} &= \min_x + i \cdot \text{Coarse Grid Size} \\ &\text{for } i = 0, 1, \dots, \text{Coarse Number of Rows} \end{aligned} \quad (28)$$

$$\begin{aligned} \text{Coarse Column Positions} &= \min_y + i \cdot \text{Coarse Grid Size} \\ &\text{for } i = 0, 1, \dots, \text{Coarse Number of Columns} \end{aligned} \quad (29)$$

$$\begin{aligned} \text{Row Positions} &= \min_x + i \cdot \text{Grid Size} \\ &\text{for } i = 0, 1, \dots, \text{Number of Rows} \end{aligned} \quad (30)$$

$$\begin{aligned} \text{Column Positions} &= \min_y + i \cdot \text{Grid Size} \\ &\text{for } i = 0, 1, \dots, \text{Number of Columns} \end{aligned} \quad (31)$$

Where: $\min_x$ and $\min_y$ represent the minimum coordinates from the bounding box.

As shown in Figure 20, the size of the coarse grid can lead to a reduction in coverage after rotation. This occurs because several grid cells may become partially covered by the NFZ. In such cases, the system could select a MST resulting from a rotation that leaves a significant portion of the area uncovered. The system defines a minimum required area that must be covered by the grid to address this issue. If a candidate grid does not meet this threshold, it is excluded from consideration. WISP allows up to 10% of the area to remain uncovered by default.

Once the discretization is complete, a clustering process is applied to divide the area among the available UAVs. The number of clusters is defined by the number of UAVs, and the Constrained K-Means algorithm from Bradley, Bennett, and Demiriz (2000) is used to ensure a minimum cluster size. This threshold is calculated as the total number of cells divided by the number of UAVs, minus a small percentage. In this thesis, a 15% margin was chosen based on initial empirical evaluations.

In the current implementation, clusters are assigned to UAVs arbitrarily, without accounting for individual platform capabilities. This design choice was initially intended to promote a fair distribution of coverage among the UAVs. However, WISP supports heterogeneous platforms, meaning the UAVs may vary in weight, frontal and top surface area, and propeller configurations. Given these differences, a more advanced allocation strategy would likely improve overall mission efficiency. For instance, it could assign larger regions to UAVs with greater flight endurance or reduce the number of required battery swaps by better matching task load to platform capability.

4.2.3 Coverage Path Planning

The coverage path planning process begins with graph generation. A vertex is created for each valid (non-empty) cell in the coarse grid vector. Up to four edges are created, one for each adjacent cell. In the resulting graph, each vertex will have at

most two edges in the MST. Since each edge in the coarse grid's MST represents movement in both directions, its cost must account for travel both ways. This is handled by assigning a cost equal to the sum of the energy estimates provided by the heuristic for both relative wind directions.

The MST can be constructed using at least three algorithms: Prim's, Borůvka's, and Kruskal's. Prim's and Borůvka's algorithms generally perform better on dense graphs due to how they explore and process edge information. The computational complexity of each algorithm is as follows: Prim's algorithm – $\mathcal{O}(E + V \log V)$ when using a Fibonacci heap, Borůvka's algorithm – $\mathcal{O}(E \log V)$, and Kruskal's algorithm – $\mathcal{O}(E \log V)$. Where V and E represent the number of vertices and edges, respectively.

Equation (32) defines the density of a graph. The graph employed for path computation in this work is not dense.

$$D = \frac{2 \cdot E}{V \cdot (V - 1)} \quad (32)$$

The number of edges is on the order of $4V$, rather than V^2 , as would be expected in a dense graph. Therefore, Equation (33) is obtained by substituting $E = 4V$ into Equation (32).

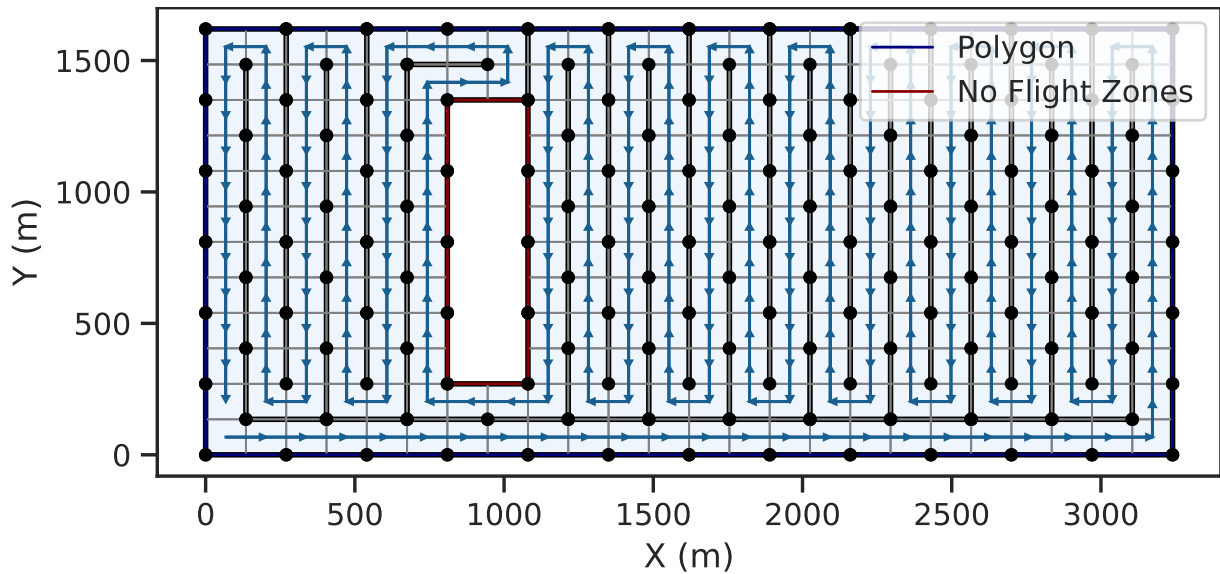
$$D = \frac{2 \cdot 4 \cdot V}{V \cdot (V - 1)} = \frac{8}{V - 1} \quad (33)$$

Finally, the limit of Equation (33) as the number of vertices tends to infinity is zero, thereby formally characterizing the graph as sparse.

$$\lim_{V \rightarrow \infty} D = \lim_{V \rightarrow \infty} \frac{8}{V - 1} = 0 \quad (34)$$

Computing the path that encloses the MST is similar to constructing a maze, through which the UAV can navigate. This maze is formed using the MST and the coarse grid cells as walls. The MST segments are treated as internal walls, while each side of a coarse grid cell is considered a potential wall. A cell side becomes an active wall if its corresponding segment does not intersect the MST. Once the maze structure is defined, the coverage path is computed using the regular grid by identifying a sequence of connected cells that do not cross any walls. Figure 23 illustrates this process: the walls are drawn as black lines and the resulting path shown with blue arrows.

Figure 23 – Walls created to compute the coverage path.



4.2.4 Integration with ROS and PX4 Environment

A major design goal during development was to create a system that could operate with a wide range of UAVs and be accessible to the broader research community. This goal was supported by two key design decisions. First, an open-source autopilot was adopted to enable integration with different UAVs platforms, promoting collaboration and transparency. Second, the system was built on top of ROS, which offers a standardized middleware for communication, modularity, and interoperability across diverse hardware and software components. This choice allows the system to be easily extended, tested, and adopted by researchers and practitioners worldwide.

Two major autopilots are currently adopted by the UAV development community: PX4 and ArduPilot. Both support integration with a wide range of UAV platforms and sensors, and provide Software In The Loop (SITL) simulation environments, making them suitable for research and prototyping. In addition, both ecosystems offer active community support and frequent updates. However, PX4 is released under the BSD license, while ArduPilot uses the GPL license. The BSD license is more permissive, allowing companies to use and modify the autopilot without being required to share the modified source code. Therefore, PX4 is generally more business-friendly and better suited for both academic research and commercial integration. For these reasons, PX4 was chosen for this work. The standard PX4 release v1.15.4 was used in this thesis.

According to Open Source Robotics Foundation (2024), ROS is an open-source software development kit that supports the creation of robotics applications ranging from research and prototyping to deployment and production. With over a decade of development and a global community of contributors, ROS provides a mature ecosystem of tools, libraries, and communication standards that accelerate development while

encouraging collaboration and code reuse. It supports a wide range of application domains, including industrial, consumer, automotive, underwater, and space robotics. ROS is cross-platform, with support for Linux, Windows, macOS, and embedded systems through micro-ROS. It is widely adopted in both academia and industry, serving as a foundation for education, research, and real-world deployments, particularly in autonomous mobile robots. Licensed under permissive terms such as Apache 2.0, ROS allows commercial and proprietary use without requiring source code disclosure. Its modular architecture and compatibility with other software stacks make it a flexible and powerful platform for robotics development.

Since no physical UAV was available to deploy the system, the Gazebo simulator was adopted as the platform to run the SITL environment. PX4 natively supports Gazebo for simulations with its autopilot firmware, making it the most appropriate choice to ensure compatibility with community standards and to facilitate the reproducibility of results.

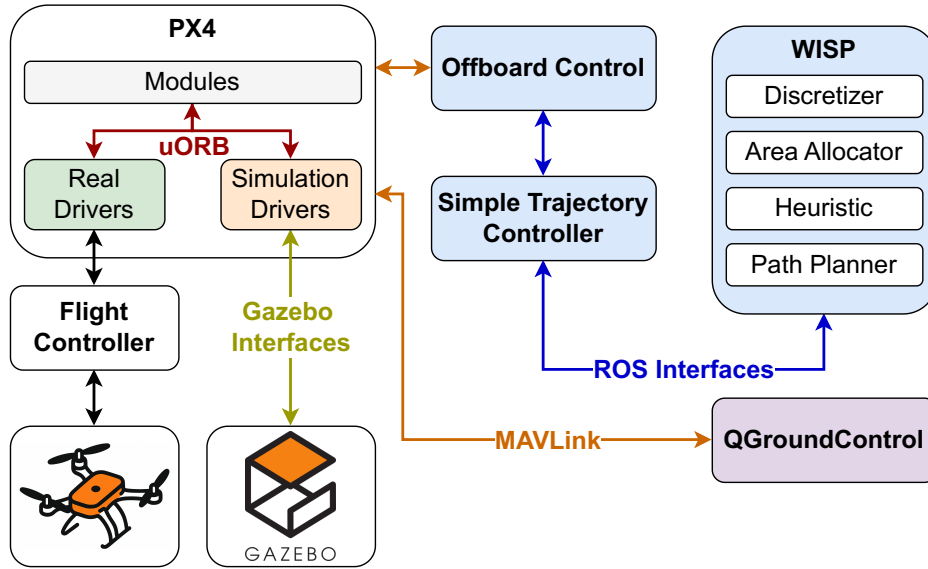
As described by Open Source Robotics Foundation (2025), Gazebo is a collection of open-source, modular libraries designed for developing high-performance robotics applications, with a particular focus on simulation. Its architecture emphasizes modularity, allowing developers to use only the required components without adopting the entire ecosystem. Gazebo is also highly extensible through plugins, enabling users to integrate new features such as custom rendering engines or alternative physics models without modifying the core code. Maintained by Open Robotics under a well-defined development protocol, Gazebo prioritizes stability, backward compatibility, and portability across major operating systems. This design supports a wide range of robotics use cases, offering beginner-friendly defaults while still providing advanced configuration options for experienced users.

The battery discharge model provided by PX4's SITL environment is not realistically modeled. The system assumes an initially full battery that decreases linearly based on a predefined parameter whenever the UAV is armed and ready to fly. To address this limitation, a dedicated Gazebo plugin was designed to simulate instantaneous energy consumption between consecutive simulation steps. The plugin relies on the same equations derived in the previously defined energy heuristic, directly linking energy usage to the UAV's movements and physical characteristics.

Using the wind speed and direction provided by the simulation, the plugin calculates the UAV's flight direction relative to the wind at each step. Based on the heuristic inputs, and considering the distance traveled and velocity, the plugin estimates instantaneous power and energy consumption for each simulation step. This data was then used in simulation-based experiments to evaluate the UAV's trajectory during planned missions, either by integrating the energy values or by averaging the power consumption over time.

Figure 24 illustrates the interconnections among the modules developed in this thesis, namely WISP and the Simple Trajectory Controller, along with other components that form part of the overall architecture.

Figure 24 – Diagram of interconnections among systems.



Internally, PX4 is composed of high-level modules, such as position control, attitude control, and the Extended Kalman Filter (EKF) for sensor fusion, along with low-level drivers such as IMU, battery management, and GPS. The modules implement the core flight control logic, while the drivers manage communication between the firmware and the physical hardware or the simulation environment. PX4 uses an asynchronous messaging system called uORB, which enables inter-thread communication between drivers and modules.

In the architecture, control signals from PX4's drivers are represented as links connecting either to Gazebo or to a physical flight controller. This dual representation highlights the system's ability to operate transparently in both simulated and real UAV environments.

Blue arrows in the figure denote communication through ROS interfaces, whereas orange arrows indicate communication using the MAVLink protocol, a lightweight messaging system widely used in UAV applications. MAVLink protocol is used by the Offboard Control module to send commands to PX4 after receiving setpoints from the Simple Trajectory Controller through ROS. The Offboard Control module is automatically spawned by PX4 when it detects the presence of the uXRCE-DDS middleware. This middleware allows uORB messages to be published and subscribed to on a companion computer as if they were ROS 2 topics, thereby ensuring transparent interoperability between PX4 and ROS. In addition, PX4 interfaces with QGroundControl, an open-source ground station software that enables real-time monitoring and manual control of the UAV during flight.

4.3 AUTONOMOUS LANDING SYSTEM

This section describes the approach for autonomously landing a UAV by suspending it with a hook at a designated landing site. It begins by detailing the UAV's preparation and highlighting limitations encountered during problem analysis, followed by an overview of the control architecture used to develop the system's proof of concept. Next, the integration of YOLOv8 (Jocher; Chaurasia; Qiu, 2023) as a system module is introduced. The subsequent part examines the functionalities of the vision module, and describes the operation of the Landing Site Estimation System (LSES). Finally, the sequence of steps in the proposed landing pipeline is outlined.

An integral component of any autonomous landing is the accurate detection of the landing site. This can be achieved through several methods, including onboard Global Positioning System (GPS) and Inertial Measurement Unit (IMU) data alone (Giernacki *et al.*, 2017), or through computer vision techniques that leverage downward-facing (Gautam *et al.*, 2022; Xuan-Mung *et al.*, 2020; Junaid *et al.*, 2017) or gimbaled (Feng, Y. *et al.*, 2018; Borowczyk *et al.*, 2017) cameras. External sensing systems may also be employed. For example, Rodriguez-Ramos *et al.* (2019) used the OptiTrack motion capture system to precisely estimate the positions of both the UAV and the moving platform, thereby enabling the UAV controller to perform accurate landing site detection.

Another common approach involves capturing images with an onboard camera and applying computer vision techniques to estimate the target's relative position. These methods typically aim to track the target and maintain camera alignment throughout the descent. Common targets include AprilTags (Gautam *et al.*, 2022; Feng, Y. *et al.*, 2018; Borowczyk *et al.*, 2017) and objects defined by specific shapes and colors (blobs) (Junaid *et al.*, 2017). Some systems combine both approaches to enable dual-mode detection, recognizing the landing site as a blob from a distance and as an AprilTag at closer range (Gautam *et al.*, 2022).

GPS sensors and vision based systems can also be combined to improve landing accuracy. In such configurations, the UAV navigates toward a known landing site using GPS for localization and switches to a vision-based approach upon entering the vicinity of the platform (Junaid *et al.*, 2017; Borowczyk *et al.*, 2017). Additionally, equipping the UAV with an Infrared (IR) camera and the landing site with an IR beacon enables reliable landing detection even under low-light or nighttime conditions (Xuan-Mung *et al.*, 2020).

In addition to estimating the landing site's location, it is crucial to accurately detect the moment of contact between the UAV and the platform. Simply stopping the motors once the UAV descends below a predefined altitude threshold may result in toppling or equipment damage (Gautam *et al.*, 2022). Thus, sensors that measure pressure (Rodriguez-Ramos *et al.*, 2019) and distance (Feng, Y. *et al.*, 2018) have been

employed to reliably detect touchdown events.

Distinct from existing approaches found in the literature, the system developed in this work employs computer vision using images captured by a fixed forward-facing camera for landing site detection. Unlike methods that require specific shapes or artificial markers, this system relies on natural features detected using the FAST algorithm (Rosten; Porter; Drummond, 2010), and it performs effectively under good lighting conditions. Although the UAV uses optical flow for localization, which limits overall positioning accuracy, the system has shown reliable performance in detecting landing sites that are visible within the camera's field of view. In addition, the proposed method incorporates IMU-based impact detection to confirm landing, providing an additional layer of reliability and distinguishing it from previous works.

4.3.1 UAV preparation

This study employs the RoboMaster TT (RMTT) UAV to develop a landing system for aerial vehicles equipped with a mechanical hook. Figures 25 (a) and 25 (b) illustrate the UAV's dimensions. The hook was fabricated using 3D printing and securely mounted on the top of the UAV with bolts. After installation, the UAV's center of mass was calibrated to ensure flight stability using the DJI Tello application.

Figure 25 – UAV's views.

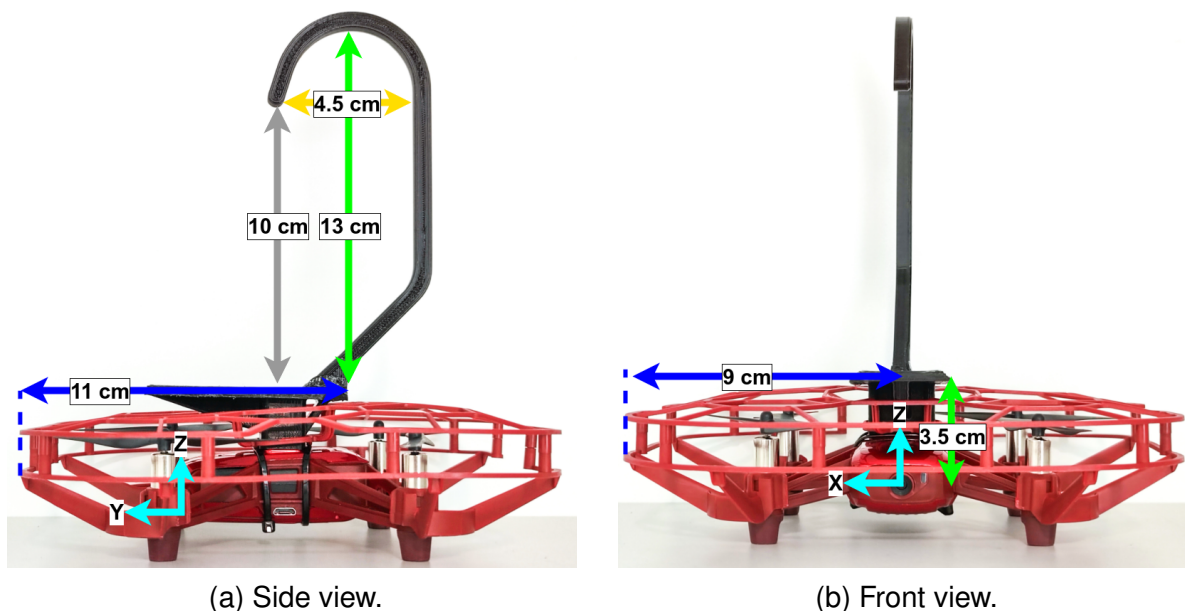


Figure 25 (a) shows that the hook has approximately 10 cm of clearance to reach the landing site, marked by the gray arrow. To ensure proper alignment during the approach, the system maintains a height that places the landing site at the midpoint of this clearance, 5 cm above the protective cage. The protective cage itself extends about 9 cm from its center, as indicated by the blue arrow in Figure 25 (b). According to

Equation (35), the landing site must not exceed a roll angle of 29° . As a safety measure, the system restricts landings to locations where the detected roll angle is below 15° . This constraint accounts for a positional tolerance of ± 2.5 cm, as defined in Equation (36).

$$\arctan\left(\frac{5}{9}\right) \approx 29^\circ \quad (35)$$

$$\arctan\left(\frac{5 - 2.5}{9}\right) \approx 15.5^\circ \quad (36)$$

It is essential that the UAV's hook is oriented forward when reaching the landing site. Yaw deviations do not significantly affect the landing performance. Empirical observations indicate that errors of approximately 10° do not hinder the system's ability to land successfully.

4.3.2 Architecture

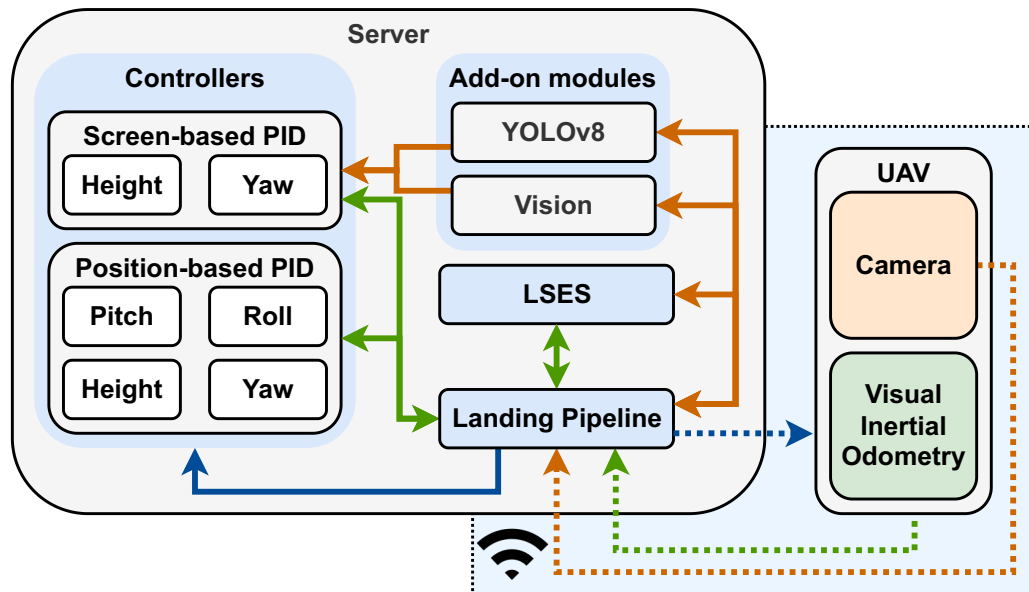
Figure 26 illustrates the architecture employed to implement the proof-of-concept system. The server receives images and odometry data from the UAV via Wi-Fi, indicated by dashed orange and green lines, respectively. Based on this data, the server processes and transmits landing commands to the UAV, represented by the dashed blue line. The architecture assumes a direct connection between the UAV and server. However, the design permits the use of intermediary routers, provided they offer sufficient transmission speed. This condition must be experimentally verified to ensure that the speed is adequate for real-time operation.

The **UAV** system includes the Visual Inertial Odometry (VIO) and camera modules. Operating as a proprietary and closed system, the VIO integrates a camera, barometer, Time-of-Flight (ToF) sensor, and accelerometer to estimate odometry. It provides quaternion data and three-dimensional position estimates. Using the quaternion, the server computes the UAV's yaw, pitch, and roll angles. DJI does not disclose the precision of the VIO system. The camera captures images at a 960x720 resolution and approximately 30 frames per second. In parallel, the VIO system outputs data at a frequency of 10 Hz.

The **Server** system was developed in Python, primarily using three libraries: OpenCV (Bradski, 2000) for image processing, YOLOv8 (Jocher; Chaurasia; Qiu, 2023) for branch detection, and the reverse engineered TelloPY for both data reception and command transmission. The system includes six modules: Vision, YOLOv8, LSES, Screen-based PID, Position-based PID, and the Landing Pipeline. Green and orange lines in the architecture represent the exchange of VIO and image data between these modules. The screen-based PID controllers receive pre-processed data from either the Vision or YOLOv8 modules. This includes the mean position of image features from

the Vision module or the center of a bounding box from YOLOv8. Blue lines denote the transmission of commands, which may also include feedback. For example, when the controller receives sensor data and a desired target position, it returns appropriate control signals to the UAV.

Figure 26 – System’s architecture. Green arrows represent Visual Inertial Odometry data transference. Orange arrows represent image data transference. The blue arrow represents command sending.



The landing process supports two initialization modes. In the first mode, the user manually selects a landing site from the UAV’s camera feed during flight. Alternatively, YOLOv8 automatically detects branches to define a suitable landing site. Afterward, the LSES integrates image features from two altitudes with VIO data to estimate the site’s relative position. Additional implementation details are presented in the following subsections. Relative coordinates are estimated instead of performing landing-site servoing due to the hook and camera configuration on the UAV. During servoing, the landing site may leave the camera’s field of view as the UAV approaches. This constraint makes it necessary to estimate the relative position in advance. Thus, the system calculates the relative position from the beginning of the landing maneuver.

4.3.3 Controllers

The PID modules control UAV movement using data from the camera and VIO systems. Given a pixel coordinate (x, y) and a set of features, the screen-based PID modules adjust this coordinate based on the average position of the matched features in the current UAV image. The yaw PID aligns the features with the x coordinate, while the height PID aligns them with the y coordinate. Each PID module operates independently.

The position-based PID modules facilitate navigation by guiding the UAV from its current position to a designated setpoint, such as the landing site. Similar to the screen-based modules, each position-based PID module operates independently.

During the landing pipeline execution, the server system operated at approximately 25 loops per second to ensure continuous feature detection and matching. This processing rate was essential for maintaining the proper functionality of the screen-based PID modules.

4.3.4 YOLOv8 Module

YOLOv8 stands for "You Only Look Once" version 8, a state-of-the-art object detection system widely applied in computer vision research and practice. The framework uses deep learning methods to detect objects in images or video frames with exceptional accuracy and speed. Ultralytics YOLOv8, an advanced evolution of the YOLO family, improves upon earlier models by adding innovative features that optimize both performance and adaptability. Its architecture emphasizes speed, accuracy, and usability, making it highly effective for applications such as object detection, tracking, instance segmentation, image classification, and pose estimation (Jocher; Chaurasia; Qiu, 2023).

In this study, a dataset was developed containing 238 training images and 61 validation images. The dataset comprised two categories: tree branch and aluminum frame. In the experimental setup, the tree branch was attached to aluminum frames, which occasionally caused YOLO to misidentify the frames as branches and generate false positives. Including aluminum frames as a separated category in the dataset reduced such misclassifications. Since this represented an initial application of YOLO to the problem, the scenario was designed as a simple proof of concept. Figure 27 shows a portion of the dataset included in a YOLOv8 training batch.

The largest YOLOv8 base model (YOLOv8x) was employed for training with 5000 epochs and a patience value of 500 epochs. Figure 28 presents the comprehensive training results obtained from YOLOv8. A detailed interpretation of each graph included in the figure is provided below.

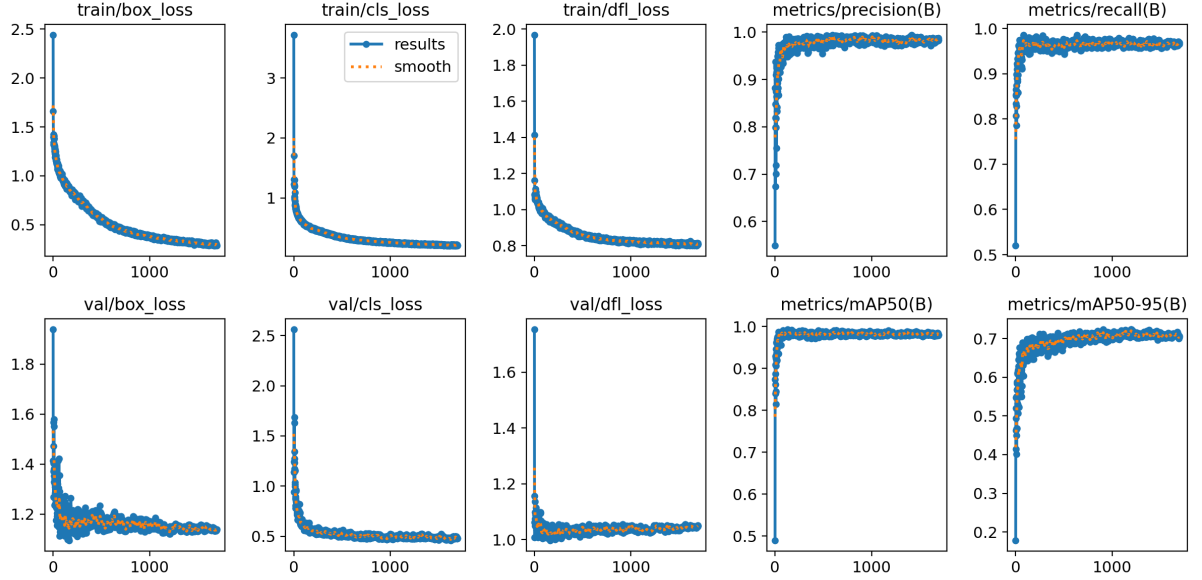
- **Box Loss (box_loss):** Measures how accurately the predicted bounding boxes align with the ground truth bounding boxes.
- **Classification Loss (cls_loss):** Quantifies the model's accuracy in predicting the class labels of detected objects.
- **Detection Fidelity Loss (dfl_loss):** Indicates the presence of false positives and false negatives in predictions.
- **Precision:** Measures the proportion of true positive predictions out of all positive predictions made by the model.

Figure 27 – Example of training data with class annotations. 0 represents a branch, and 1 represents an aluminum frame.



- **Recall:** Also known as sensitivity, measures the proportion of true positive predictions out of all actual positive instances in the dataset. It reflects the model's ability to detect all relevant instances of objects.
- **mAP50:** Mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.5. It calculates the average precision across all classes, counting predicted bounding boxes with an IoU of 0.5 or higher as correct detections.
- **mAP50-95:** mAP computed across IoU thresholds ranging from 0.5 to 0.95. It provides a comprehensive measure of the model's performance by evaluating accuracy across multiple IoU thresholds.

Figure 28 – Graphs illustrating box loss, classification loss, detection fidelity loss, precision, recall, and Mean Average Precision (mAP) across training epochs for both the training and validation sets.



After training, YOLOv8 was able to detect branches or aluminum frames within each received image. It was established a confidence threshold of at least 70% for YOLOv8's detections, ensuring more assertive branch detection. Considering that multiple branches can be detected, a reward criterion was formulated (Equation (37)), enabling the server to select the branch with the highest calculated reward. The reward function prioritized branches with high confidence and greater width-height ratio, thereby favoring elongated horizontal branches.

$$reward = (confidence \cdot 10)^2 \cdot \frac{rectangle_width}{rectangle_height} \quad (37)$$

4.3.5 Vision Module

The Vision module is responsible for five main tasks:

- **Image rectification:** The UAV's camera operates with a pitch angle. Consequently, the system requires rectifying the captured images, correcting this angle to ensure that features move linearly as the UAV ascends. Without rectification, disparity measurements in feature matching would become increasingly inaccurate during vertical motion.
- **Filtering:** For each received image, the Vision module applies the Sharpen Filter followed by a Bilateral Filter. The first enhances object boundaries but increases image noise, while the second reduces noise while preserving these improvements. This process increases the number of detectable features.

- **Feature detection:** Given an area defined by a polygon in an image, the Vision module employs the FAST algorithm (Rosten; Porter; Drummond, 2010) to detect features within the delimited region.
- **Feature matching:** Given a set of features and an image, the system performs brute-force matching to locate the informed features within the image.
- **Area tracking:** Feature detection and brute-force matching may involve high computational cost. Therefore, after the user or YOLOv8 defines the landing site within a rectangular area, the system tracks this region and restricts detection and matching to it. Since this region is considerably smaller than the full image, the computational cost of combined tracking, detection, and matching is reduced. For this purpose, the OpenCV implementation of MOSSE filters (Bolme *et al.*, 2010) is employed.

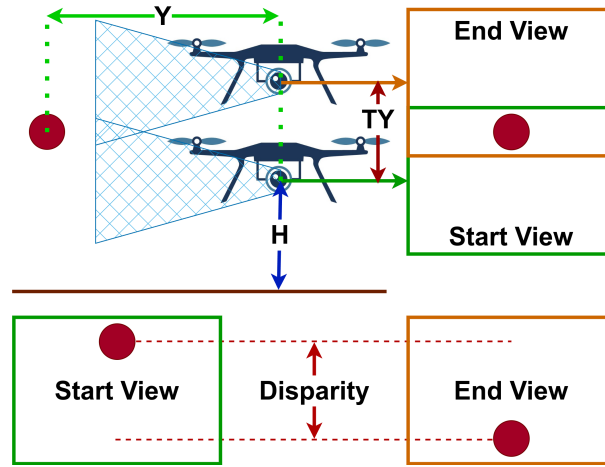
4.3.6 Landing Site Estimation System

After either the user or YOLOv8 defines the landing site in the received images, the server employs the Vision module to extract features at two different heights. These features are then forwarded to the LSES, which applies Stereo Vision (SV) principles to compute the landing site's relative position with respect to the camera. Unlike traditional SV systems that use two parallel cameras, this approach implements SV from motion by capturing images at different known heights with the same camera.

Figure 29 illustrates a simplified representation of the geometry underlying the LSES. Initially, the system detects an object feature in the 'Start View.' After the UAV ascends to the 'End View,' the same feature appears at a different Y coordinate. The difference between these two coordinates is defined as the disparity.

Using the disparity, camera parameters, and the UAV's ascent distance, the LSES estimates the real-world relative position of each feature through Equations (38)-(40). These equations compute the real-world positions of the features at the second picture height. Accuracy requires minimal variation in the features' X coordinate after the UAV's ascent. This study implemented a tolerance of 10 pixels. Within the LSES framework, X denotes left/right, Z represents height, and Y corresponds to forward/backward motion.

Figure 29 – A simplified example of the geometry behind the LSES.



$$Y = \frac{F_Y \cdot T_Y}{I_{Y2} - I_{Y1}} \quad (38)$$

$$X = Y \cdot \frac{I_{X2} - C_X}{F_X} \quad (39)$$

$$Z = Y \cdot \frac{C_Y - (I_{Y2} + I_{Y1}) \cdot 0.5}{F_Y} - \frac{T_Y}{2} \quad (40)$$

Where:

- X , Y , and Z : world coordinates relative to the UAV's camera;
- T_Y : the variation in the UAV's height between the first and second features acquisition;
- F_X and F_Y : focal length for X and Y axis in the images;
- C_X and C_Y : coordinates X and Y in the image representing the position of an object centralized with the camera in the environment;
- I_{X2} : the feature's X coordinate in the second image;
- I_{Y1} and I_{Y2} : the feature's Y coordinate in the first and second images, respectively.

After estimating the real-world relative position of each feature, the system applies a simple segmentation process using K-means clustering. Knee point detection, a well-established technique, was employed to determine the number of clusters for each set of detected points. After clustering, the system selects the cluster nearest to the camera for computing the landing site position. This approach eliminates points belonging to the background, which could otherwise cause inaccurate estimation of the landing site position.

Additionally, the LSES computes the yaw and roll angles of the landing site using a least-squares technique. This information allows the system to determine whether the

landing site is suitable for landing. Equation $y = a \cdot x + b$ defines the yaw angle, employing the features' X and Y coordinates as x and y data to compute the coefficients a and b . The yaw angle is obtained through $\tan^{-1}(a)$. A comparable procedure estimates the roll angle, where the features' X and Z coordinates are treated as x and y data, respectively.

4.3.7 Landing Pipeline

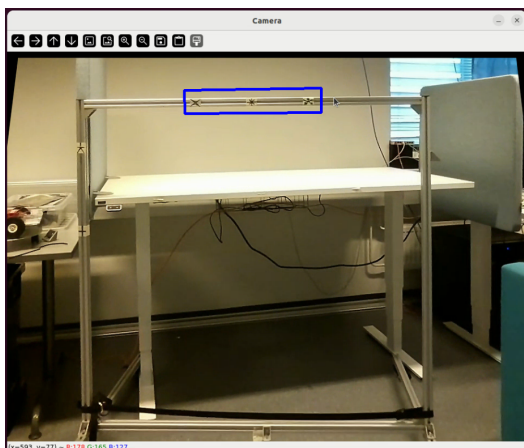
The proposed system follows a landing pipeline consisting of seven sequential steps:

1. The user selects a landing site or initiates automatic detection through YOLOv8 (Jocher; Chaurasia; Qiu, 2023);
2. Height and yaw are adjusted using screen-based PID controllers;
3. The UAV's location and landing site features are recorded;
4. Height and yaw are adjusted using both position-based and screen-based PID controllers;
5. The UAV's location and landing site features are recorded again, followed by estimation of the relative position;
6. Height and X are adjusted using position-based PID controllers;
7. Final navigation toward the landing site is performed using position-based PID controllers;

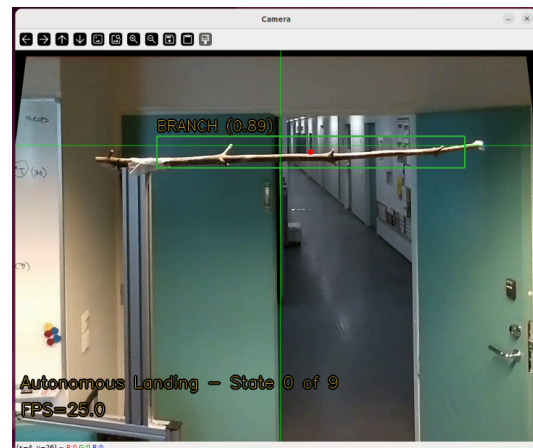
The landing process provides two possible starting points. The user may manually select a landing site from the UAV's camera images during flight (Figure 30 (a)). Alternatively, automatic branch detection can be initiated using YOLOv8, which identifies a branch as the landing site (Figure 30 (a)).

Figure 30 – Possible starting points.

(a) Landing site selected by user.



(b) Branch detected by YOLOv8.



In Step 2, the server employs the vision module to track the region containing the landing site while adjusting the UAV's height and yaw using screen-based PID controllers. When the rectangle becomes centralized, the server records the detected features within the region and the UAV's current height.

Next, the server employs the position-based height controller to raise the UAV by TY cm and the screen-based yaw controller to centralize the matched features. After these adjustments of height and yaw, the system records the most recent matched features along with the UAV's current height.

Once the server obtains the features and the UAV's positions at two different heights, it forwards this information to the LSES, which computes the real-world position of each matched feature, as well as the landing site's roll and yaw angles. The server then calculates the average of the Y coordinates and selects the real-world feature location closest to this average. Using the selected location, the server applies the position-based PID controllers to adjust the UAV's yaw and X position. Finally, the server engages all position-based controllers to guide the UAV toward the chosen real-world location.

Finally, once the server detects the impact on the landing site, it activates the pitch controller to push forward and the height controller to push down for one second. As a result, the UAV lands with the hook contacting the landing site. The server recognizes two possible failure types: a missing landing site and a failed engagement. In the first case, the UAV reaches at the target location but fails to register an impact. In the second case, the UAV registers the impact and begins descending, but if altitude decreases by more than 15 cm, the engagement is marked as failed. In both situations, the UAV returns to its initial position before initiating a new navigation attempt toward the landing site. A demonstration video of the full pipeline is available at <https://youtu.be/-aCFcoKEJI8>.

4.4 CONCLUSION

This chapter presented a scalable UAV-based monitoring framework composed of long-endurance UAVs with large payload capacities and multimodal sensors (e.g., LiDAR, multispectral, or thermal cameras) that perform wide-area surveillance of forests and plantations. Based on sensor data, these UAVs identify areas requiring closer inspection and can deploy smaller units that navigate below the canopy, land on branches, and operate as stationary sensing nodes within a distributed monitoring network.

The framework is composed of three main components: the Area Allocation and Coverage Planning System, the Autonomous Branch-Landing System for Small UAVs, and the Communication and Coordination Infrastructure. Only the first two components were explored in this thesis.

The Area Allocation and Coverage Planning System employed an expanded version of the energetic model proposed by Sorbelli *et al.* (2021), developed in this thesis, to construct graphs for computing optimized paths based on MSTs using the WISP. This energy model was also used to develop a Gazebo simulator plugin for modeling and simulating UAV energy consumption. The WISP addresses the lack of wind-informed planners in the literature and has been integrated with the PX4 autopilot, enabling potential deployment on real UAV platforms.

The Autonomous Branch-Landing System for Small UAVs employs a simple hook-based mechanical design. By integrating a YOLO-based branch detector with a stereo-from-motion technique, the system estimates the depth and orientation of potential landing sites, enabling vision-guided landings that contribute to more energy-efficient and persistent aerial operations.

5 RESULTS AND DISCUSSION

This chapter presents the experimental results that support the proposed methods. The first part evaluates the energy consumption model, comparing theoretical predictions with results from a Gazebo plugin under different flight and wind conditions. A comparison of coverage path planning algorithms is then provided, highlighting the efficiency of Wind-Informed Sweep Planner (WISP) relative to alternative methods. Additional experiments examine K-Means variants for workload allocation and validate theoretical trajectories in simulation, considering wind, terrain, and no-fly zones. Together, these results demonstrate the practical advantages and robustness of the proposed approaches. The second part focuses on the Autonomous Landing System (ALS), including battery consumption analysis, controlled validation of the Landing Site Estimation System (LSES), and flight tests where landings are achieved either by user selection or YOLOv8 detection.

5.1 EXPERIMENTS WITH THE WIND-INFORMED SWEEP PLANNER

This section investigates the proposed energy consumption model by contrasting theoretical estimates with results obtained from a Gazebo plugin across various flight profiles and wind conditions. It then compares multiple coverage path planning algorithms, emphasizing the effectiveness of WISP against alternative methods. Further experiments explore K-Means variants for workload distribution and validate theoretical trajectories through simulation, accounting for wind, terrain, and no-fly zones. Collectively, the findings highlight the robustness and practical benefits of the proposed approach.

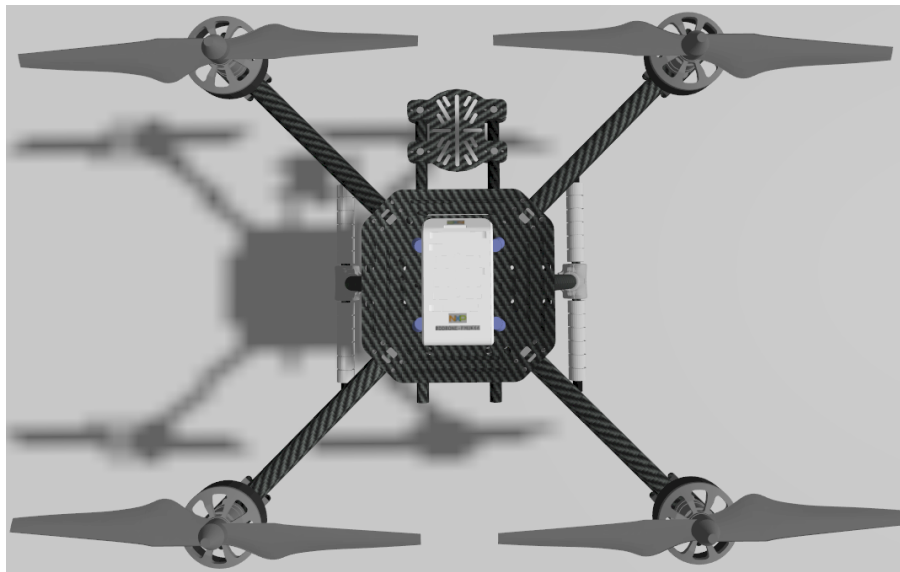
5.1.1 Unmanned Aerial Vehicle Model Used in the Experiments

Figures 31-32 present the frontal and top views of the Holybro X500 Unmanned Aerial Vehicle (UAV) used in the analysis. This UAV, provided in the PX4 models for Gazebo simulation, was used in all the experiments described in this section. The heuristic model developed in this thesis depends on the projected frontal and top surface areas, which are used to estimate aerodynamic forces during flight. These areas were calculated using the *parea* software, created by Rooy (2021), which extracts projected surface areas from 3D models.

Figure 31 – Frontal view of the Holybro X500 UAV.



Figure 32 – Top view of the Holybro X500 UAV.



5.1.2 Heuristic Conceptual Validation

This subsection presents a conceptual validation in which theoretical data are compared with the output data generated by the Gazebo plugin implementing the energy consumption model. The UAV setup considered vertical and horizontal velocities of 5 m/s for a straight-line flight. The average power consumption was evaluated during level, descending, and climbing flights while varying the wind speed and direction. The level flight covered a distance of 1 km, whereas the climbing and descending flights each covered 500 m with an inclination of 45° .

Figures 33-37 present the theoretical results computed using the implemented model, which are consistent with the expected behavior. As expected, a climbing flight consumes more power than a level flight. In turn, a level flight consumes more power

than a descending flight. The UAV also consumes more energy depending on the Relative Wind Direction (RWD), as greater alignment with a headwind increases energy consumption. Another observation is that, since the UAV is flying at 5 m/s, when it flies in the wind direction and the wind speed is lower than the UAV speed, the wind assists the UAV's motion and reduces its power consumption. In contrast, when the wind speed exceeds the UAV velocity, the power consumption increases. This is because, in addition to the power required to remain airborne, the UAV must exert additional power to act as a brake against the wind.

The model was integrated as a Gazebo plugin, enabling the simulation of flights and the estimation of instantaneous energy consumption during operation. By applying a simple summation over the flight data, it is possible to compute the total energy consumed or the average power. In the experiment using this plugin, the same scenario from the theoretical experiment was reproduced. Figures 34-38 present the results of this experiment. The results are quite similar to the theoretical ones, particularly for the level flight. However, because the plugin was executed within a Gazebo simulation, variations in the UAV's trajectory caused by wind effects altered the power consumption curves. This effect was more pronounced in the climbing flight, where the UAV experienced greater instabilities, leading to higher fluctuations in power consumption.

Nevertheless, the results suggest that the model is robust enough to be used as a rough approximation of the actual power consumption, thus serving as a suitable heuristic for coverage planning. In addition, this experiment indicated that performing further tests with a wind speed of 10 m/s would be beneficial, as this condition appears to highlight more clearly the differences in power consumption across all scenarios.

Figure 33 – Theoretical results from the conceptual validation, illustrating the average power consumption of the UAV during level flight.

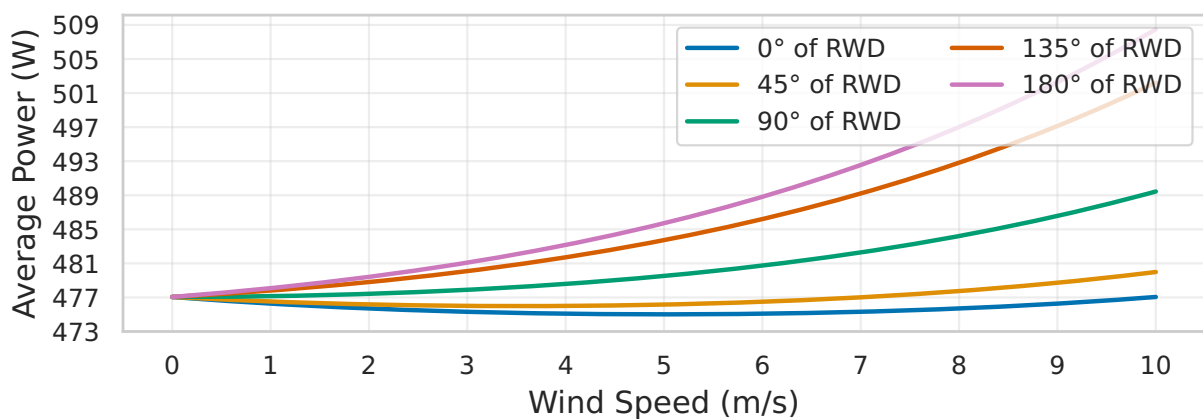


Figure 34 – Experimental results from the conceptual validation, illustrating the average power consumption of the UAV during level flight.

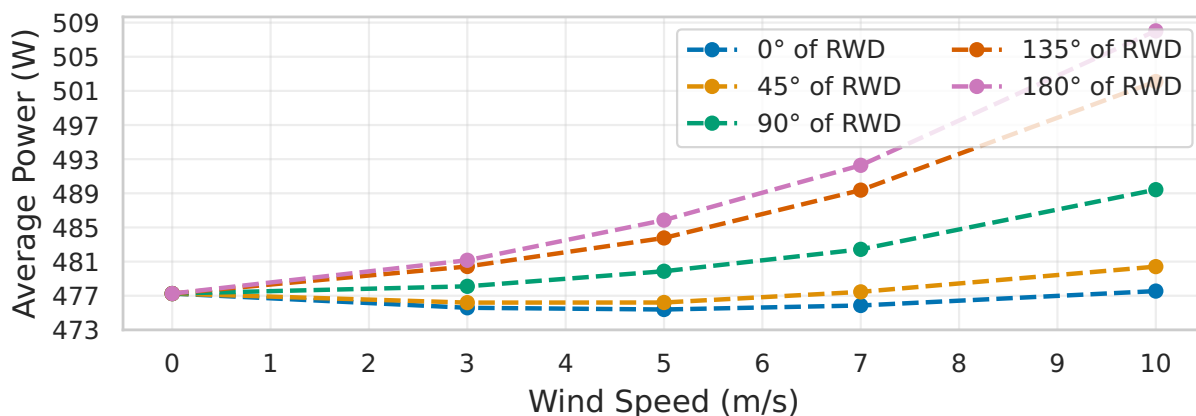


Figure 35 – Theoretical results from the conceptual validation, illustrating the average power consumption of the UAV during descending flight.

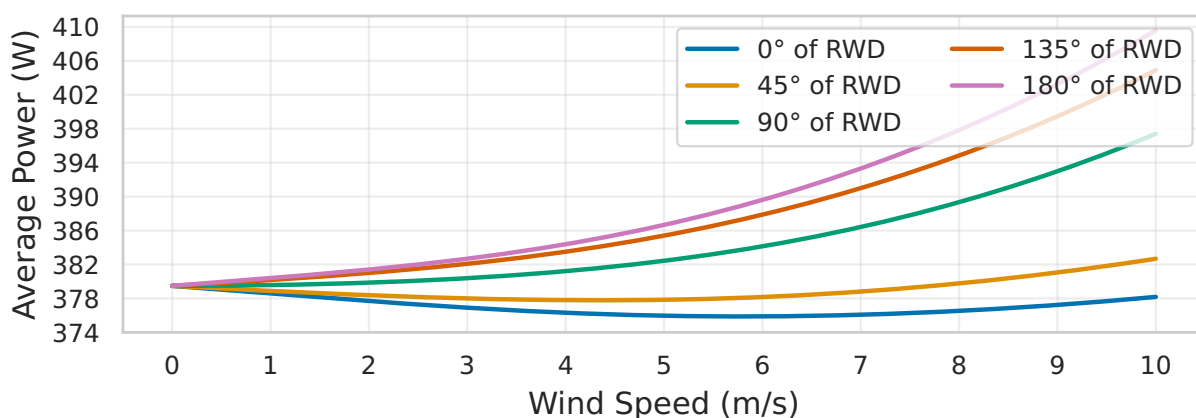


Figure 36 – Experimental results from the conceptual validation, illustrating the average power consumption of the UAV during descending flight.

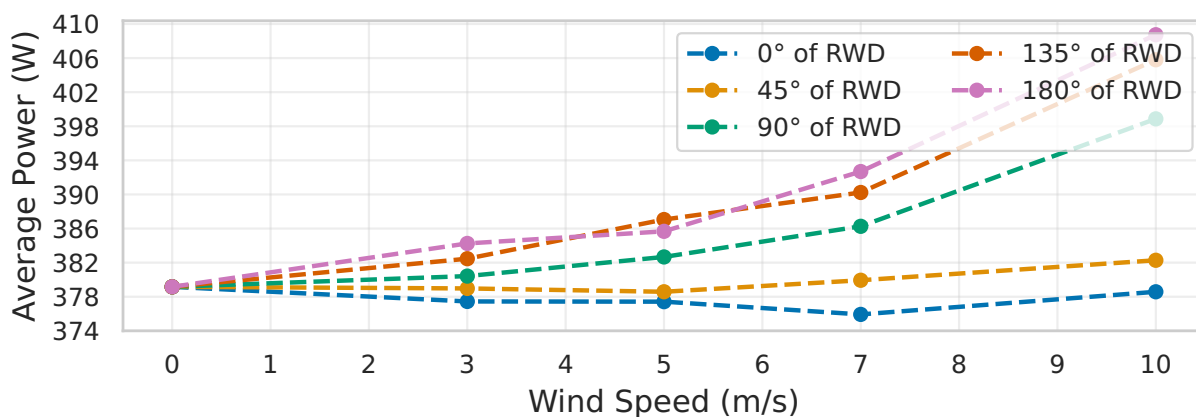


Figure 37 – Theoretical results from the conceptual validation, illustrating the average power consumption of the UAV during climbing flight.

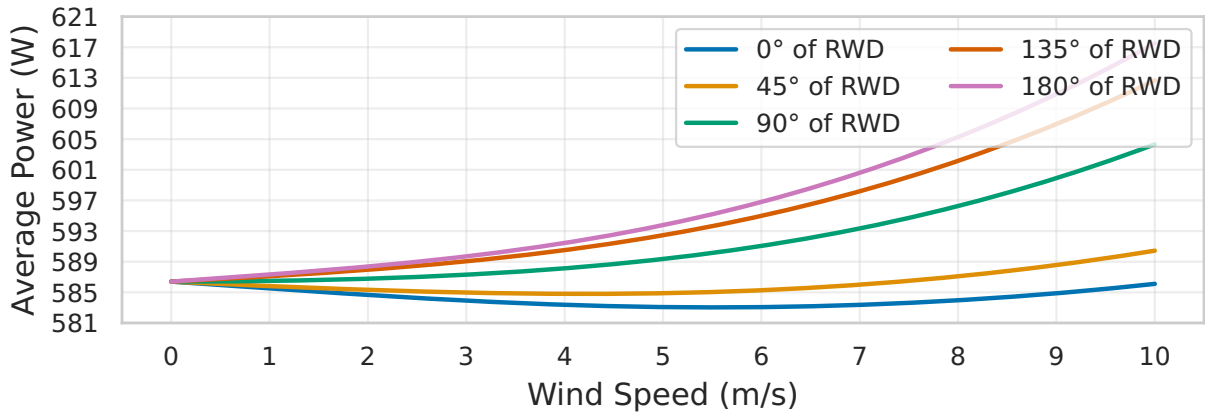
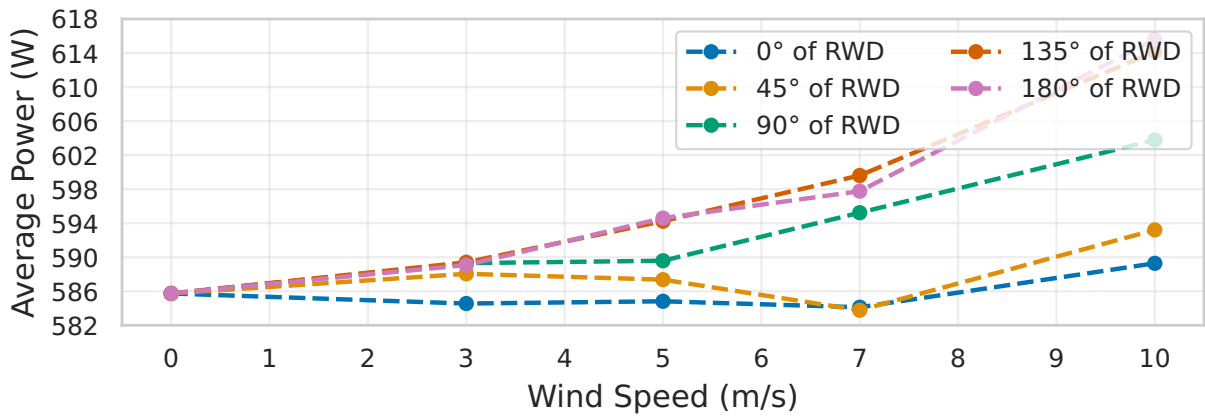


Figure 38 – Experimental results from the conceptual validation, illustrating the average power consumption of the UAV during climbing flight.



5.1.3 Coverage Path Planning Algorithm Comparison

This subsection presents a comparison of six algorithms for Coverage Path Planning (CPP), with an initial focus on computational performance in terms of execution time and memory consumption. This comparison is essential for evaluating their feasibility in real-time or resource-constrained aerial robotic applications.

The primary algorithm under evaluation is WISP, introduced in this thesis as a novel Minimum Spanning Tree (MST)-based planner that constructs coverage paths using a graph weighted by energy estimates. These estimates are derived from the heuristic modified in this work, which accounts for wind effects and the dynamics of the flight.

The WISP's computational efficiency was compared against five baseline algorithms: the energy-based Greedy and K-Nearest Neighbors (KNN) methods, a hybrid Greedy +Simulated Annealing (SA) approach, a standalone SA implementation, and a distance-based KNN variant. Among these, only the distance-based KNN does not rely on the energy-weighted graph. Preliminary experiments showed that optimizations

for Euclidean distance made this algorithm significantly faster when using a purely geometric cost metric.

The energy-based Greedy and KNN algorithms share with WISP the use of an energy-weighted graph, but differ in decision-making granularity. While WISP constructs a globally optimized path using a graph structure, Greedy and KNN make local decisions at each step, prioritizing the next closest or least costly target based on immediate energy estimates. On the other hand, the standalone SA implementation represents a fully metaheuristic approach. Building on this, the hybrid Greedy+SA method introduces a metaheuristic component that probabilistically refines the greedy solution, offering a balance between fast heuristics and global search. In contrast, the distance-based KNN serves as a computational baseline, offering a lightweight alternative that sacrifices model realism for speed. This mix of baselines allows for a comprehensive assessment of WISP's performance across various dimensions, including execution time, memory usage, and algorithmic strategy.

The evaluation scenario consisted of a simple rectangular area measuring 4800 by 1200 meters, totaling 576 hectares. The size of this area was chosen based on the real-world application from Aetha Global (2023). The environment was discretized into grid sizes ranging from 50 to 200, in order to evaluate how each algorithm scales in terms of computation time and memory usage.

Figures 39 and 40 present time and memory usage data that align with the expected behavior. Smaller grid sizes result in a larger number of nodes to be processed. Consequently, the smaller the grid, the greater the time and memory required to compute the path for the environment.

Among all tested scenarios, the energy-based KNN method exhibited the highest computation time, which is consistent with the preliminary experiments. The Greedy, SA, and Greedy+SA approaches showed similar performance, suggesting that the results obtained by the Greedy+SA method are largely driven by the initial greedy solution. Besides that, the proposed WISP method achieved better results than the distance-based KNN, with lower computation time and memory consumption, while also incorporating energy estimation. Thus, WISP showed the best performance among all evaluated algorithms demonstrating its computational efficiency.

Figure 39 – Mean computation time measured across different grid sizes.

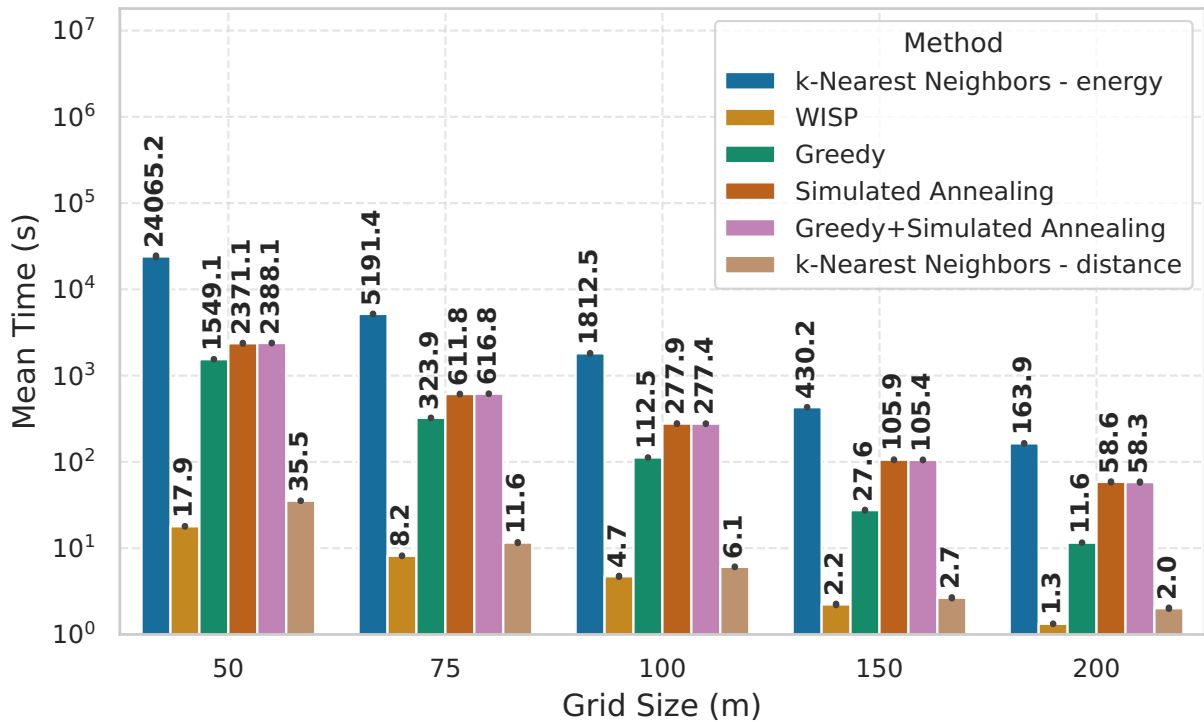
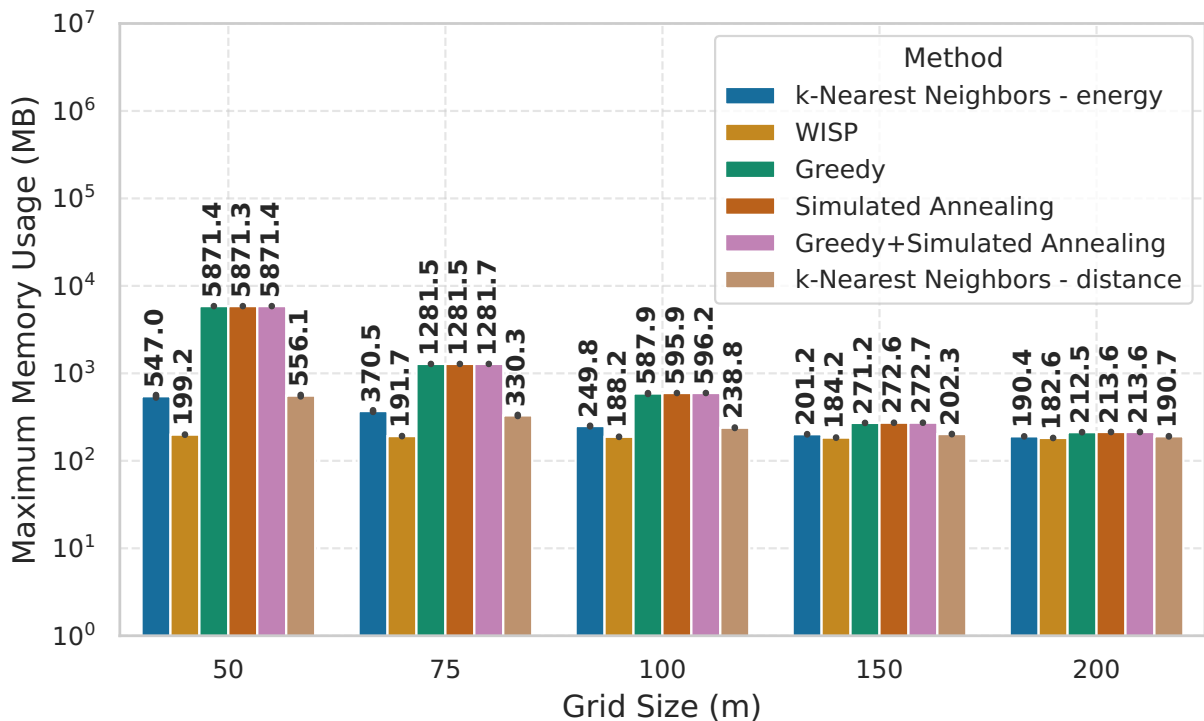


Figure 40 – Mean memory consumption measured across different grid sizes.



In addition to computational efficiency, the algorithms were compared across six distinct environments with respect to the energy consumption of the planned paths: plane, cliff, valley, inclined, concave and convex. Each environment presents unique terrain characteristics that influence the energy required to complete coverage, enabling

a more comprehensive assessment of algorithm performance under varying physical conditions. For each environment at least two wind conditions were evaluated, yielding a total of 14 test cases, each repeated five times. Figures 41–46 show the resulting path for each algorithm in one representative repetition of each test case. Each algorithm used the same discretization procedure as WISP to ensure a fair comparison. The environment was discretized using a grid size of 135 m. Then, it was then rotated from 0° to 89° in 1° increments and the path cost was evaluated for each rotation. Finally, the rotation that yielded the lowest cost path was selected. The characteristics of each environment are described below.

- **Plane:** The total area is 524.88 hectares and the No-Flight Zone (NFZ) covers 29.16 hectares. Elevation is 75 m Above Sea Level (ASL).
- **Inclined:** The total area is 524.88 hectares and there are no NFZs. Elevation varies according to $75 + \frac{x}{10}$ m ASL.
- **Valley:** The total area is 524.88 hectares and there are no NFZs. Elevation is 75 m ASL if $2 \cdot \text{grid_size} \leq y \leq 10 \cdot \text{grid_size}$, otherwise elevation is 275 m ASL.
- **Cliff:** The total area is 524.88 hectares and there are no NFZs. Elevation is 75 m ASL if $x < 8 \cdot \text{grid_size}$, otherwise elevation is 400 m ASL.
- **Convex:** The total area is 500.00 hectares and there are no NFZs. Elevation is 75 m ASL.
- **Concave:** The total area is 612.36 hectares and the NFZ covers 87.48 hectares. Elevation is 75 m ASL.

Visual inspection of Figures 41–46 indicates that the WISP paths are the most regular and visually consistent. For example, in the Concave environment with a wind direction of -45° , visible defects are observed in all methods except WISP. In the KNN-Energy result (Figure 42), repeated cells and a crossing outside the polygon are clearly observed. The KNN-Distance result (Figure 43) exhibits poor quality, with multiple path crossings and incursions into the NFZ. For the Greedy method (Figure 44), a clear repetition of cells appears along the inner right side of the concavity. In the Greedy+SA result (Figure 45), the same repetition persists, and the solution is essentially the one produced by the Greedy algorithm, indicating that SA did not contribute meaningfully. Finally, the worst results are obtained with pure SA, which exhibits the most extensive duplicate coverage. Since WISP also demonstrates superiority in the other environment scenarios, the best visual performance across all cases is achieved by WISP.

Figure 41 – Example of the WISP method across all environments and wind directions.

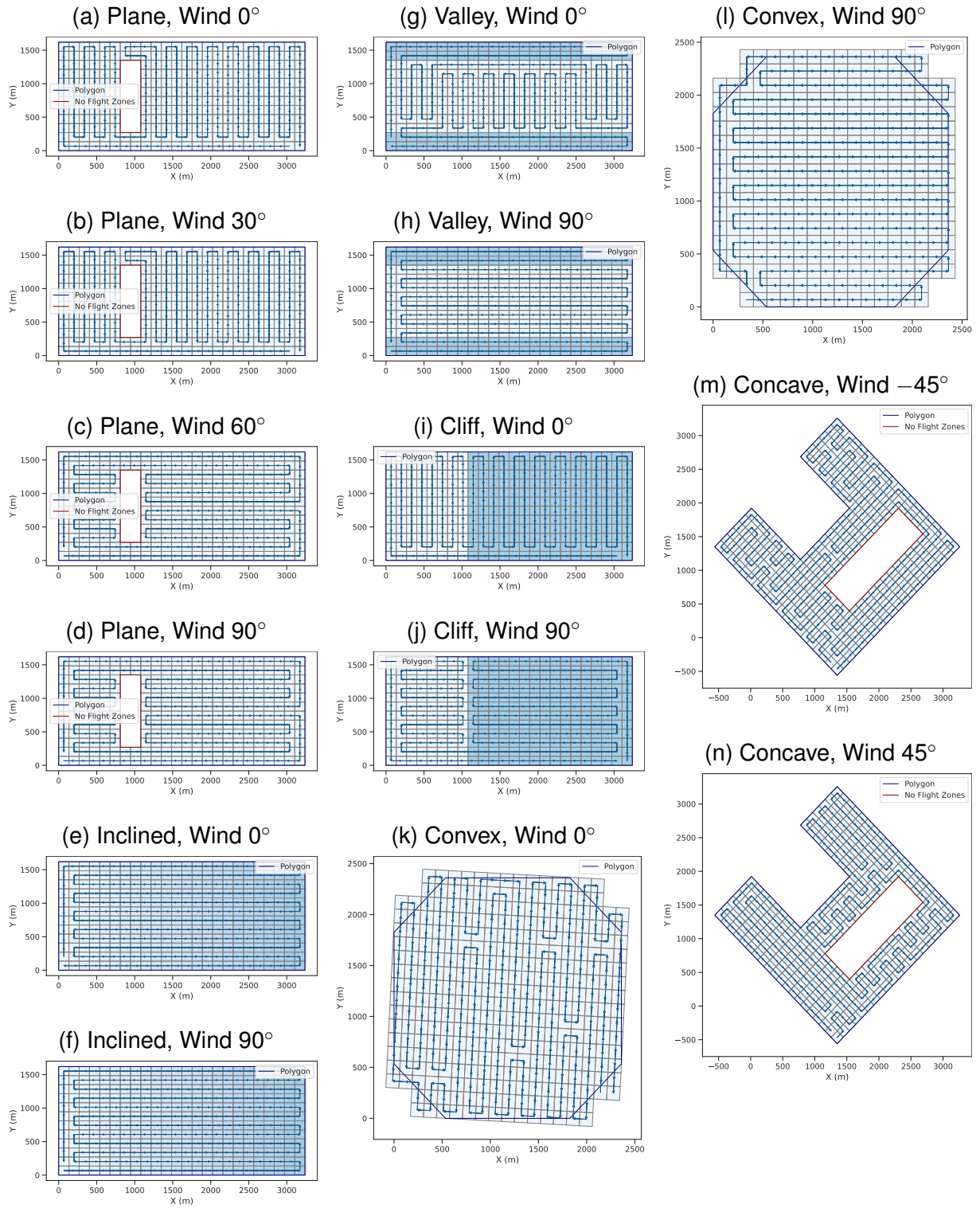


Figure 42 – Example of the KNN-Energy method across all environments and wind directions.

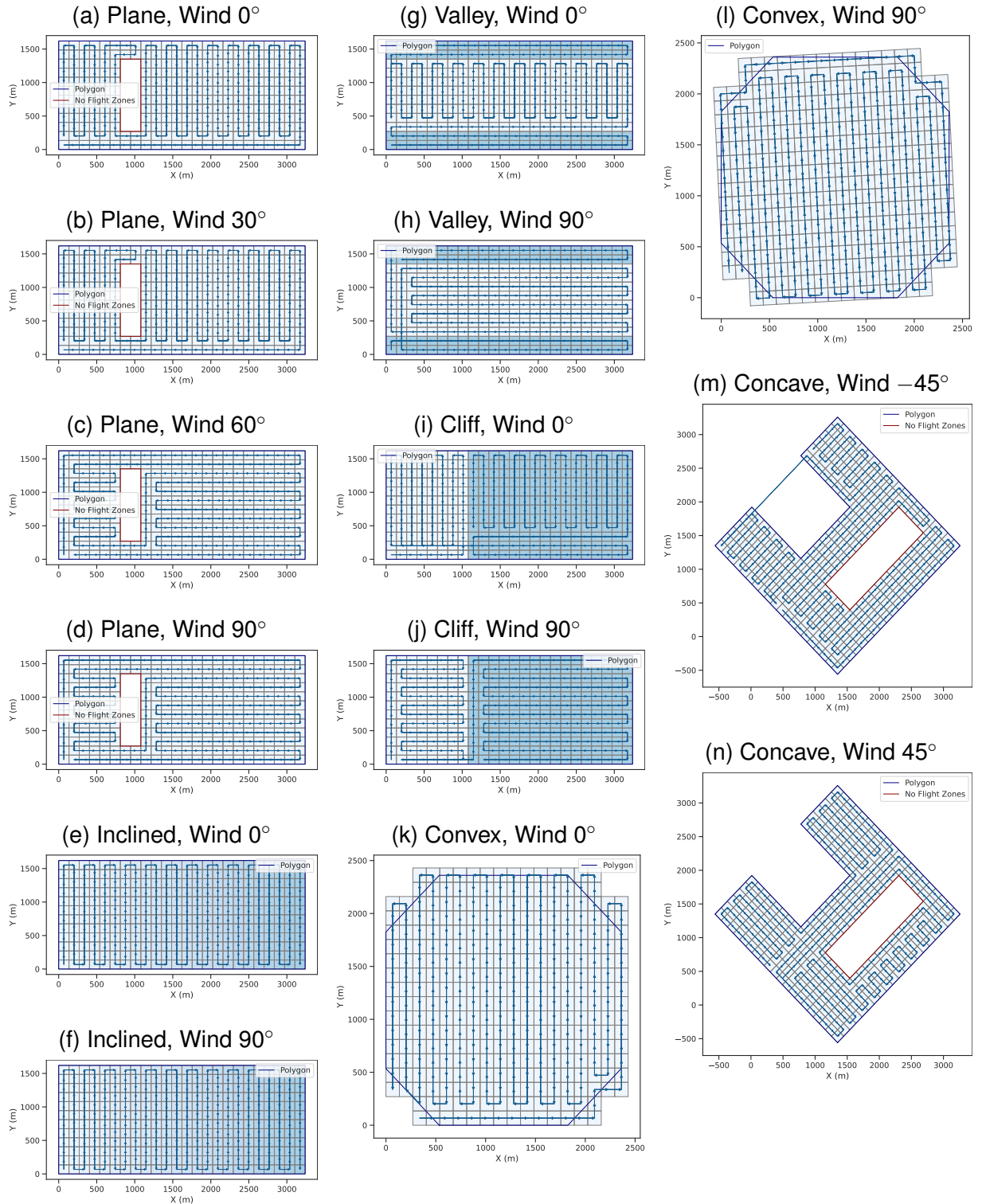


Figure 43 – Example of the KNN-Distance method across all environments and wind directions.

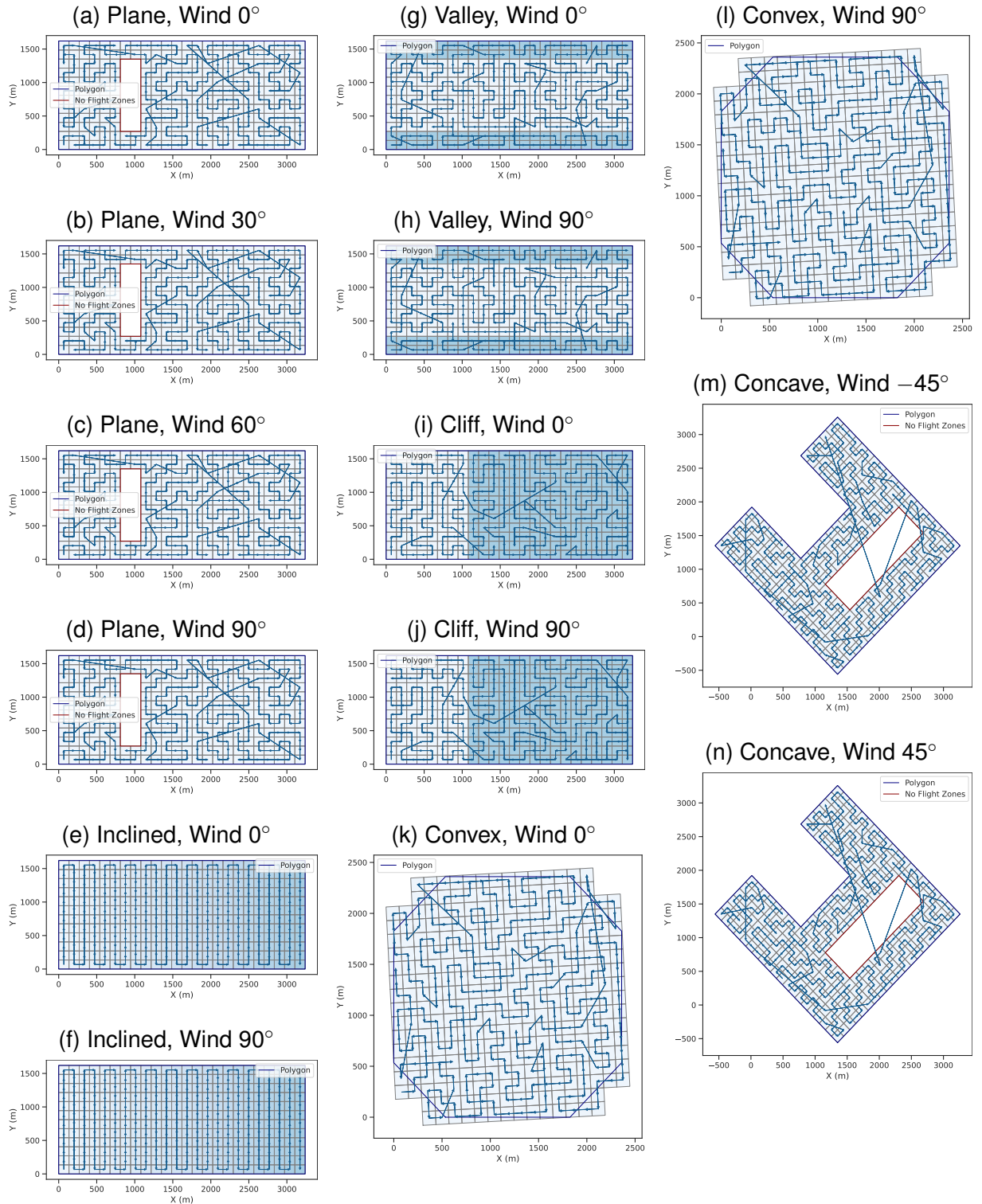


Figure 44 – Example of the Greedy method across all environments and wind directions.

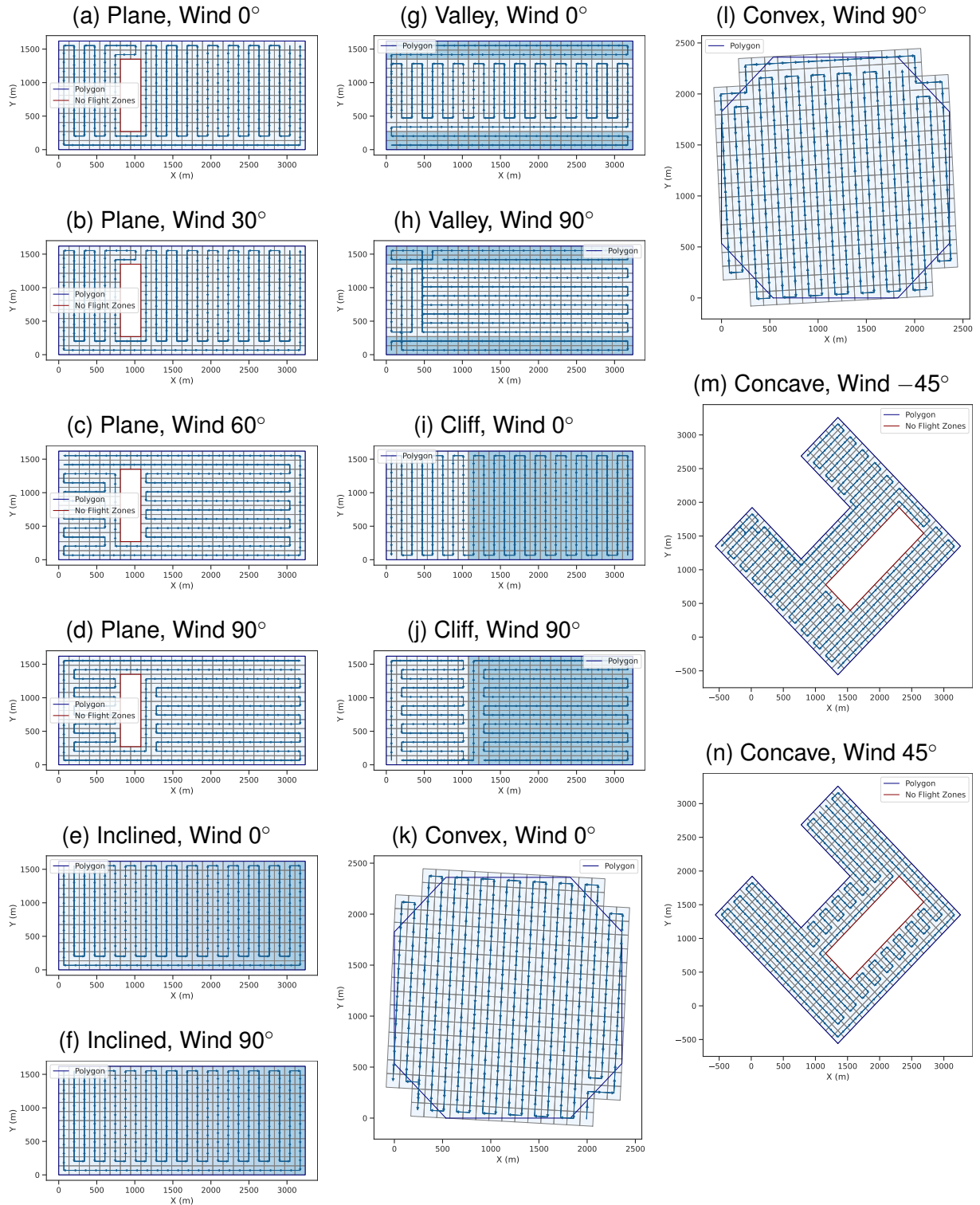


Figure 45 – Example of the Greedy+SA method across all environments and wind directions.

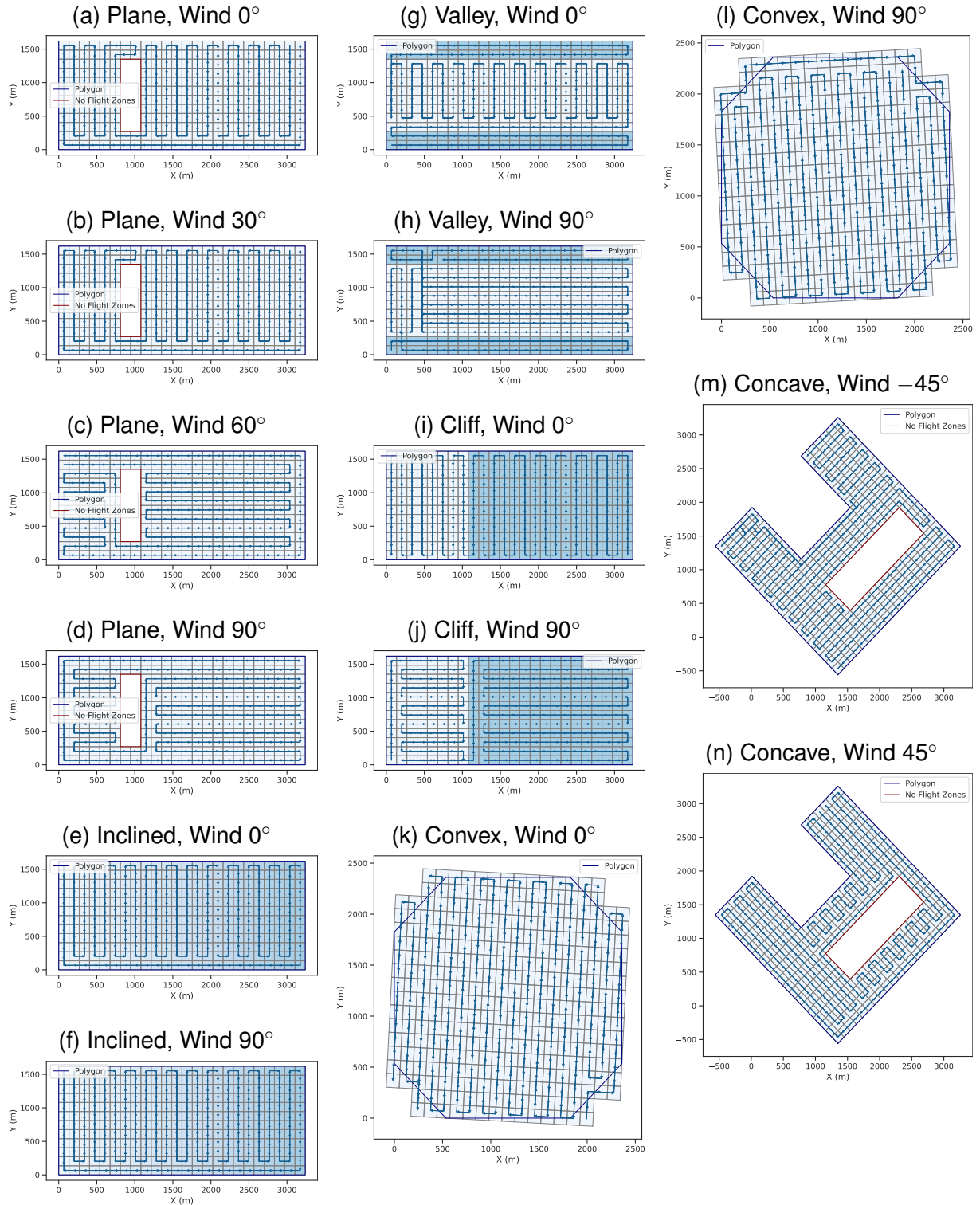
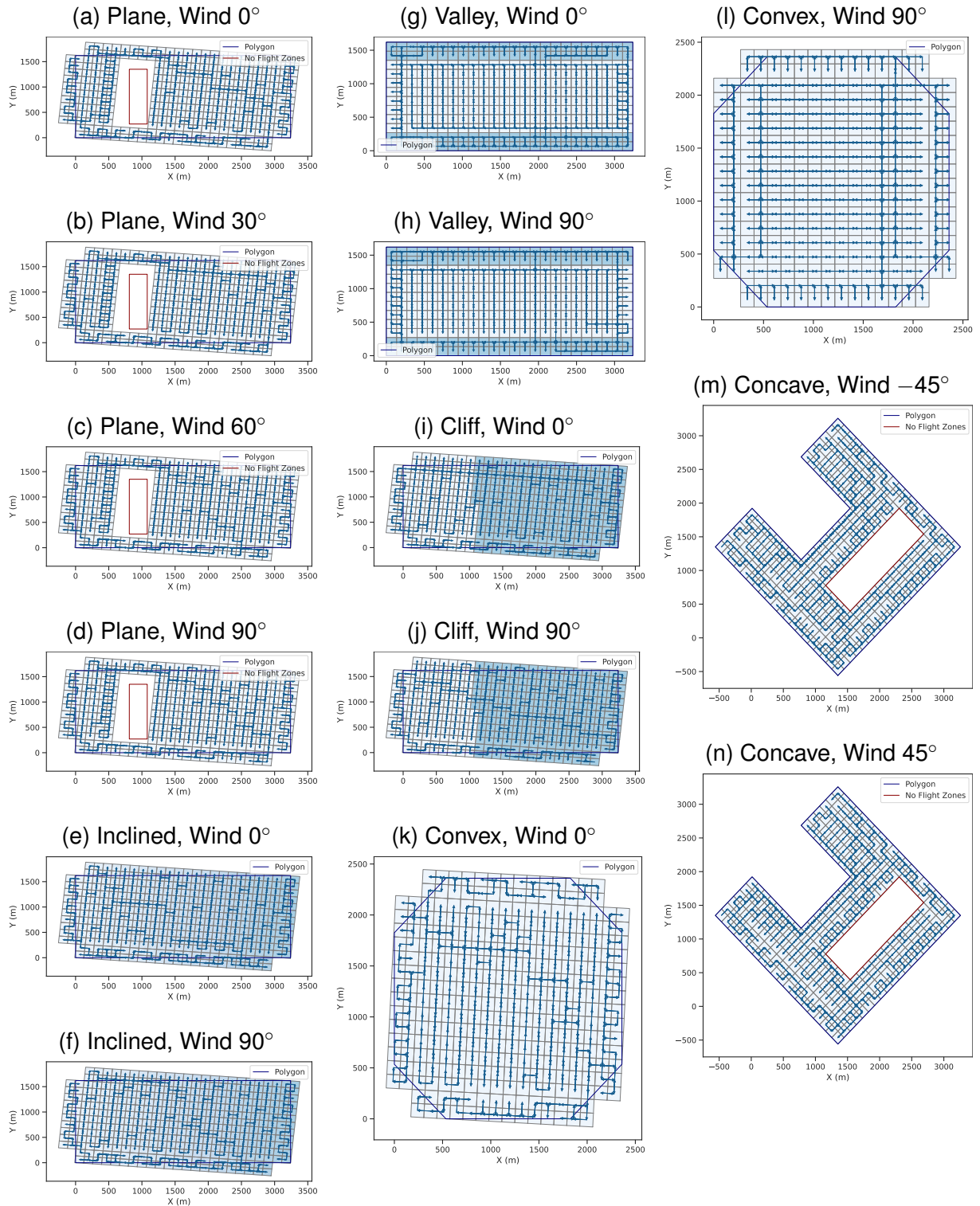


Figure 46 – Example of the SA method across all environments and wind directions.



Following the qualitative assessment based on visual inspection of representative planned trajectories, a detailed quantitative analysis will be conducted. This analysis will evaluate the mean cost of the trajectories, the Standard Deviation (STD) to assess variability, the number of turns made, and the overall energy savings, measured relative to the worst-performing method in the group.

Figure 47 presents the mean total energy cost per environment. It is visually

evident that SA demonstrates the poorest performance, which aligns with the low quality of the planned trajectories it produced. KNN-distance also yields subpar results, as it generated disorganized routes in several test cases, with the Inclined environment being the only exception where it performed reasonably well. Although WISP achieved the best results across all test cases, the performance margin is relatively small when compared to the outcomes from the Greedy and KNN-energy methods.

Figure 47 – Mean total energy cost per environment and method (kJ). Each cell reports the mean total cost (kJ) $\pm$ STD.

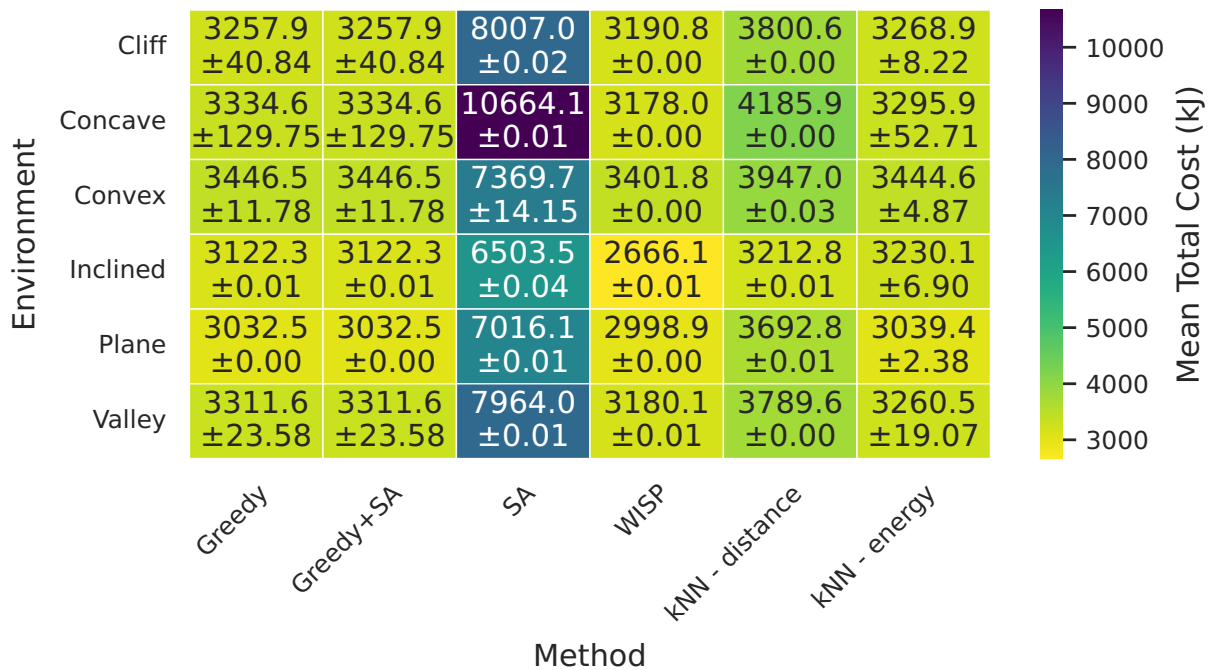


Figure 48 shows the mean number of turns. This variable is frequently cited in the literature as a critical contributor to increased energy consumption, mainly due to the acceleration variations that occur during turning maneuvers.

In the current trajectory planning setup, the number of turns does not directly influence the total cost computed for any of the methods. As a result, the number of turns may not significantly affect the planned trajectory cost in this specific evaluation. However, in the experimental setup described in the next section, where the UAV dynamically adjusts its velocity during curves, this variable may have a more important impact on the overall energy consumption.

In the Inclined and Valley environments, the WISP method achieved both the lowest number of turns and the lowest energy consumption. On the other hand, in scenarios such as the Concave environment, it produced ten more turns than the Greedy method while achieving only a 5% reduction in energy usage. In this case, the path generated by the Greedy method could potentially result in lower real-world energy consumption, even though its theoretical cost was higher.

Figure 48 – Mean number of turns per environment and method.

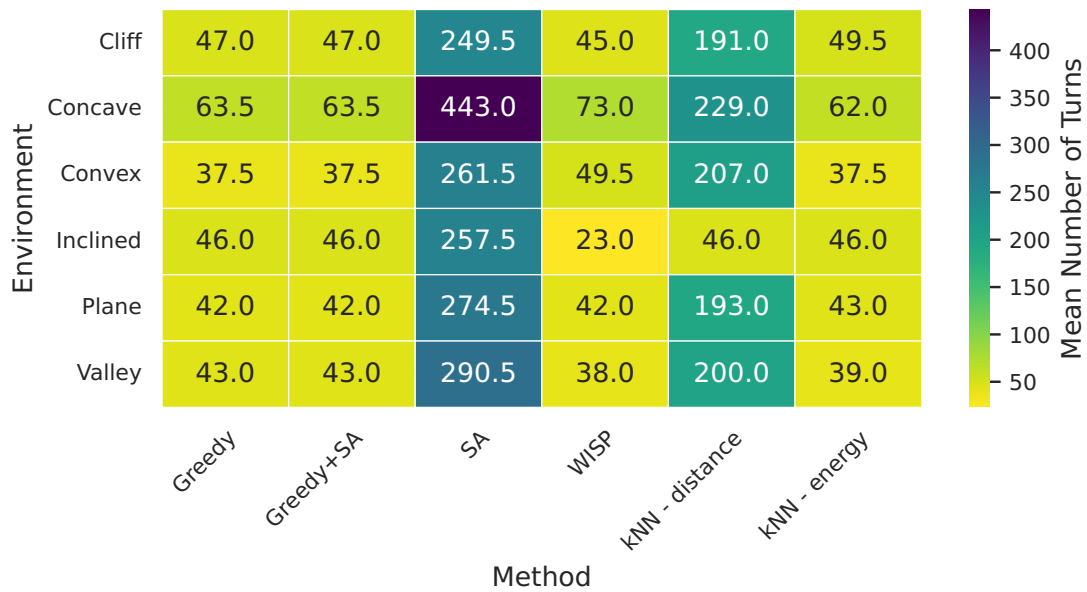


Figure 49 presents the percentage reduction in mean energy consumption for each method. The method with the highest energy cost in each environment is used as the baseline and set to 0 %, while other methods show their relative savings. Since SA consistently produced the worst energy performance, it was excluded from the figure to improve clarity.

Figure 49 – Mean energy saved (%) per method relative to the worst-performing method.

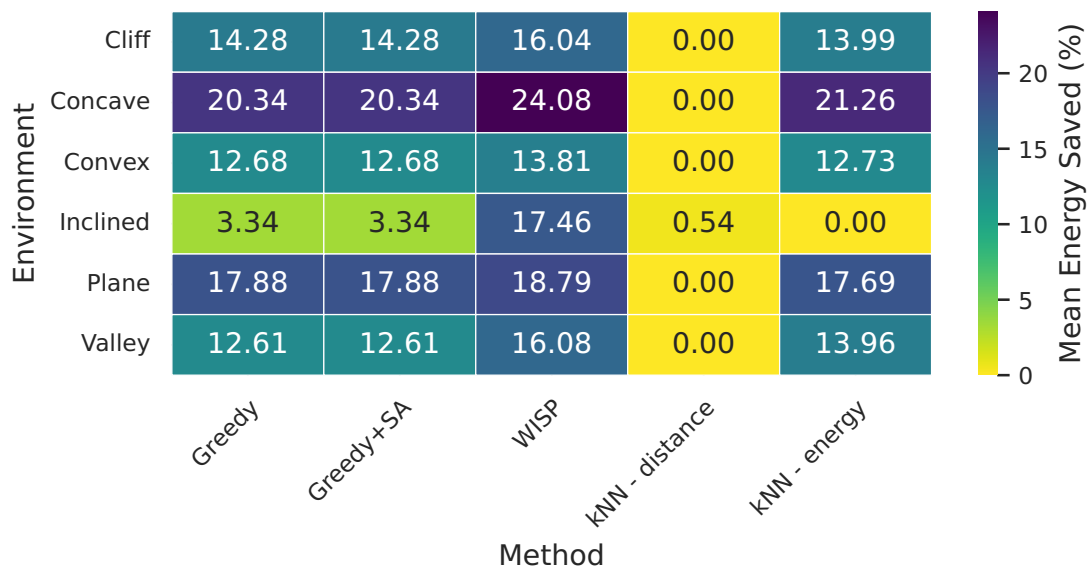
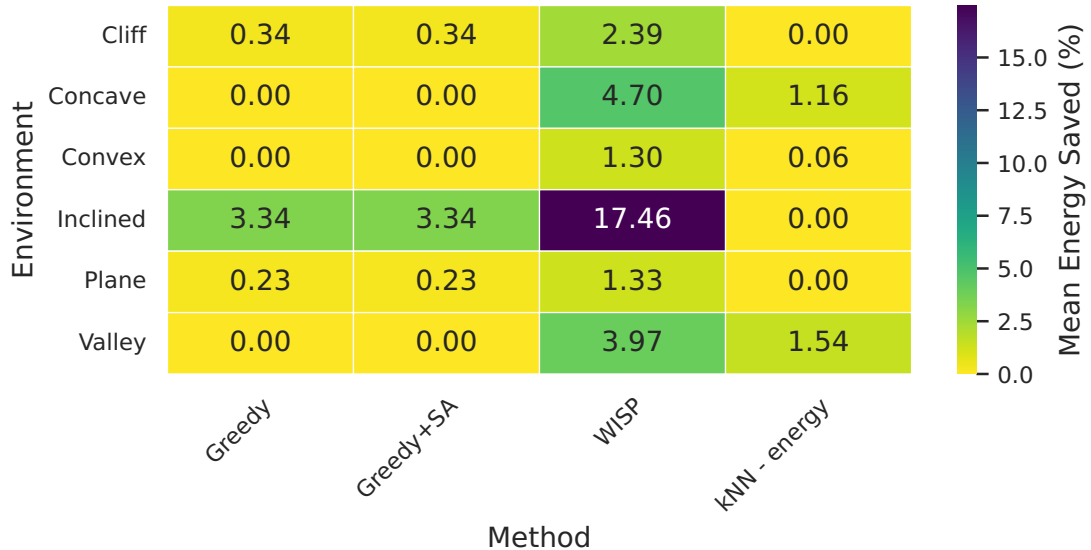


Figure 50 applies the same calculation but also removes KNN-distance, focusing the comparison on methods that explicitly account for wind effects. Across both figures, the WISP method clearly stands out, achieving the highest energy savings in

all environments. These results confirm its effectiveness in optimizing trajectories under realistic conditions and reinforce its suitability for energy-aware navigation.

Figure 50 – Mean energy saved (%) per method excluding SA and KNN-distance.

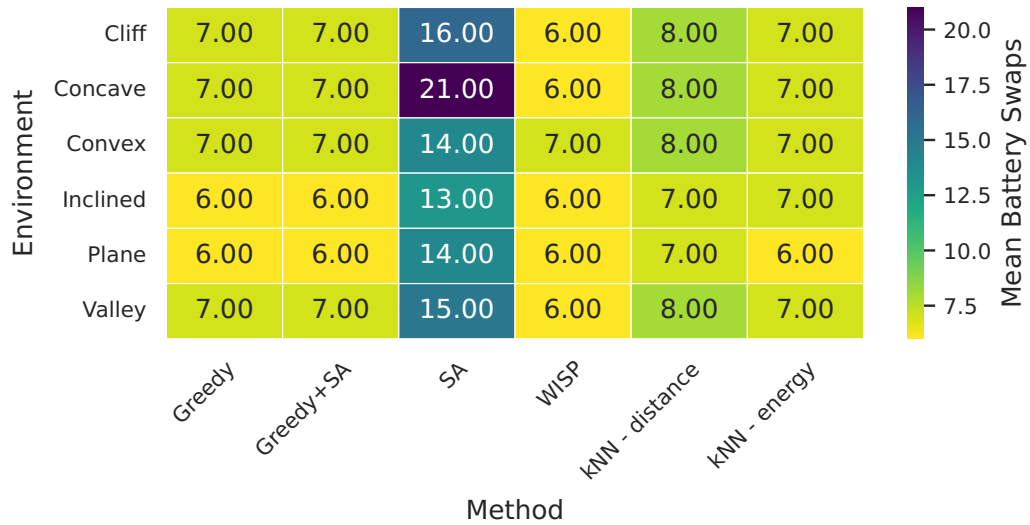


A final and more tangible evaluation metric is the number of battery swaps required per environment and method, as illustrated in Figure 51. A commonly used battery pack for UAVs is a 4S 10 000 mAh unit rated at 14.8 V, which corresponds to an energy capacity of 148 Wh. When converted to joules, this results in approximately 532.8 kJ. By combining this capacity with the mean energy consumption per environment, it is possible to estimate the expected number of battery replacements required for each method.

While some methods lead to a reduction in energy consumption, WISP consistently results in the fewest required battery swaps. This outcome alone represents a significant operational advantage. Fewer battery changes directly translate into reduced downtime, lower logistical complexity, and decreased operational costs. Considering that charging a UAV battery can take several hours and that spare battery packs must be prepared and available for each mission, the reduced number of swaps offered by WISP enhances both mission efficiency and field deployability.

Despite certain limitations and opportunities for further refinement, the robustness and adaptability of the WISP method remain clearly demonstrated. Across diverse and complex environmental conditions, it consistently identified near-optimal trajectories by effectively balancing critical factors such as elevation variation, path feasibility, and turn minimization. This consistent performance across all scenarios reinforces WISP as the most reliable and energy-efficient approach among the evaluated methods, and positions it as a strong candidate for real-world deployment in energy-constrained navigation tasks.

Figure 51 – Mean number of battery swaps per environment and method.



5.1.4 Comparative Analysis of K-Means Variants for Area Allocation

Clustering algorithms such as K-Means are commonly applied for discretized area allocation in CPP approaches. However, standard K-Means often leads to unbalanced area divisions, as it considers only the spatial position of the cells in the discretized space. This limitation can negatively impact workload distribution among UAVs. Three clustering methods were evaluated to investigate which one provides a more balanced division of the discretized area: K-Means (Lloyd, 1957; MacQueen, 1967), Mini Batch K-Means (Sculley, 2010), and Constrained K-Means (Bradley; Bennett; Demiriz, 2000).

The same environments used in the previous assessments were applied in this experiment. The evaluation was conducted using only the WISP since it showed superior to the other methods. For each subarea generated by the clustering algorithms, the total energy required to complete the corresponding path was calculated. The STD of these energy values, normalized by the mean energy consumption, was used to compute the Coefficient of Variation (CV), which serves as an indicator of workload imbalance. Figure 52 presents the results of this analysis. As expected, the Constrained K-Means algorithm outperformed the others, producing the most balanced divisions. Its main advantage lies in its ability to enforce a minimum number of elements per cluster, which results in more evenly distributed subareas and consequently more uniform energy requirements across agents.

Along with the improved uniformity in area division, the use of Constrained K-Means also introduces a higher computational cost. As shown in Figure 53, the WISP solution combined with Constrained K-Means required more computation time in all evaluated scenarios.

Figure 52 – Energy Variation Coefficient for each environment by number of UAVs and method. Where: Km - K-Means, KMC - K-Means Constrained, and MBKM - Mini Batch K-Means

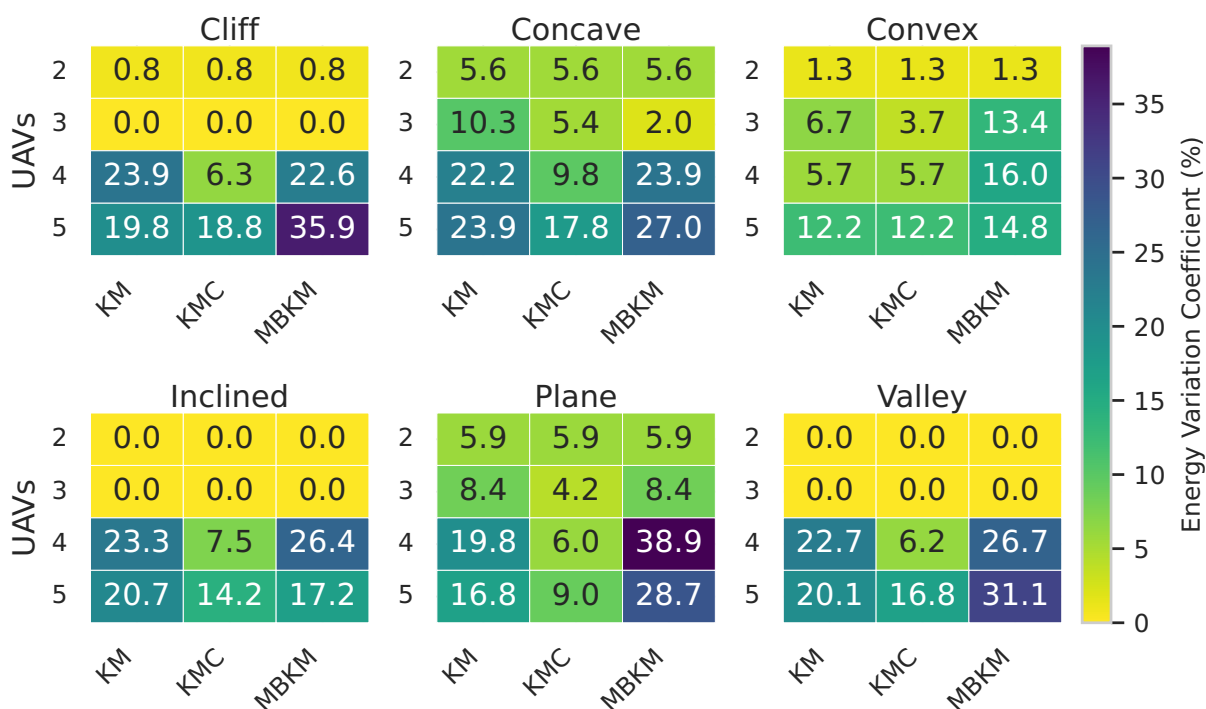
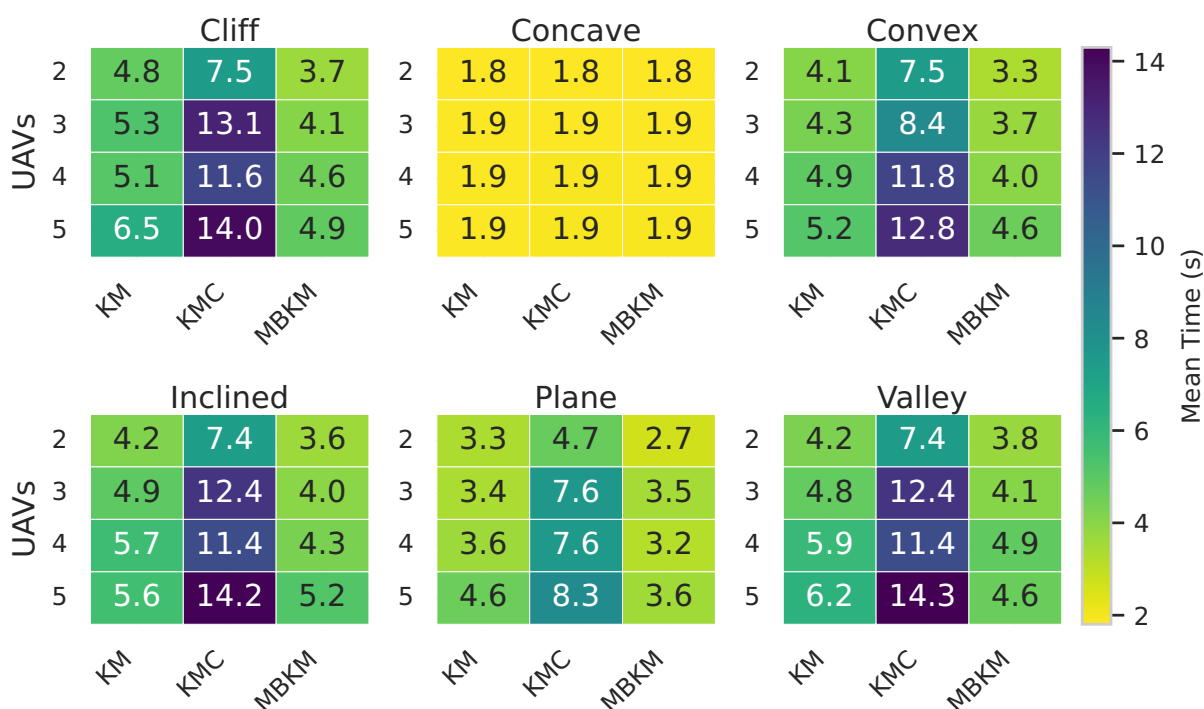


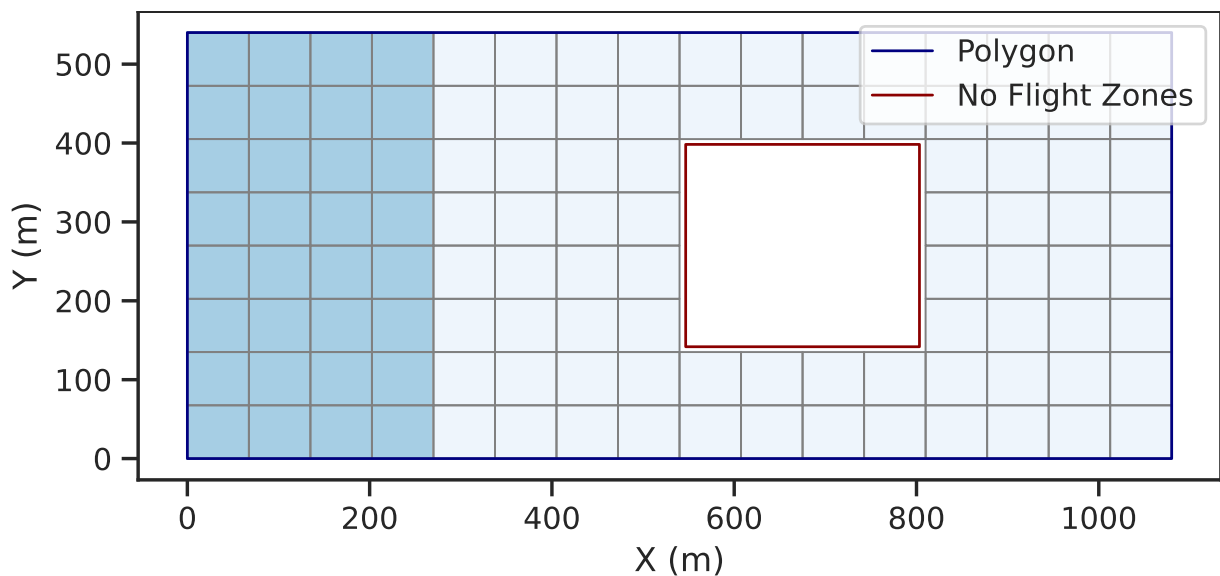
Figure 53 – Mean Time to find a solution for each environment by number of UAVs and method. Where: Km - K-Means, KMC - K-Means Constrained, and MBKM - Mini Batch K-Means



5.1.5 Simulation-Based Validation of Theoretical Trajectories

A simulation environment was designed to validate the theoretical trajectories using the energy consumption plugin developed for Gazebo. Figure 54 illustrates this environment, defined as a rectangular polygon measuring 1080×540 m, with a total area of 58.32 hectares. Within this region, a smaller polygon specifies a no-fly zone of approximately 6.52 hectares. The environment represents a cliff scenario: all points with an x -coordinate less than or equal to 280 m are assigned a height of 150 m, while the remaining area is limited to 30 m. Additionally, wind with a constant speed of 12 m/s is applied along the x -axis.

Figure 54 – Environment used for trajectory validation. Darker blue indicates a height of 280 m, while lighter blue corresponds to 30 m.



The first set of experiments analyzed trajectory planning under three conditions: no wind, wind considered during planning, and planning with a 90° error in wind direction. In all cases, the WISP method was applied to a single UAV. As expected, the absence of wind produced the lowest energy consumption, with the UAV requiring approximately 796.65 kJ. When a 12 m/s wind was introduced and its direction was correctly provided to the planner, energy use increased slightly to 813.05 kJ. Finally, when the planner was given an incorrect wind direction, energy consumption reached its highest value of 830.67 kJ. While the overall variation is about 4%, real-world uncertainties could magnify this difference, resulting in additional battery changes and consequently higher mission costs and longer completion times.

After the initial experiment, WISP, KNN-distance, KNN-energy, Greedy, SA, and Greedy+SA were compared under a 12 m/s wind applied along the x -axis. Energy, time, and distance were recorded and compared against the planned trajectories. Figures 55–57 show that the actual values are consistently greater than the planned ones.

This result is expected, since the UAV varies its position in the simulation due to the underlying flight dynamics and control system's adjustments.

Once again, the WISP method outperformed the others. Figure 55 shows that it consumed less energy than all competing methods, even though Greedy and Greedy+SA produced shorter and faster paths, as indicated in Figures 56–57. As illustrated in Figure 58, these methods do not generate trajectories that return to the initial point, and they still consume more energy than WISP. Including the energy required to return to the base would further increase this difference in consumption.

The SA method once more showed the worst performance, mainly because its paths were not optimized. The Greedy+SA produced results nearly identical to the Greedy method. For these reasons, both methods were executed only once in this experiment, which explains the absence of error bars in their plots. The KNN-distance method yielded the second-worst result, while KNN-energy performed approximately 4.6% worse than the Greedy method. However, both KNN-based methods generated invalid paths: their descent from 280 m to 30 m occurred at invalid positions, which in real-world conditions would likely lead to a crash.

Figure 55 – Planned vs Actual energy by method.

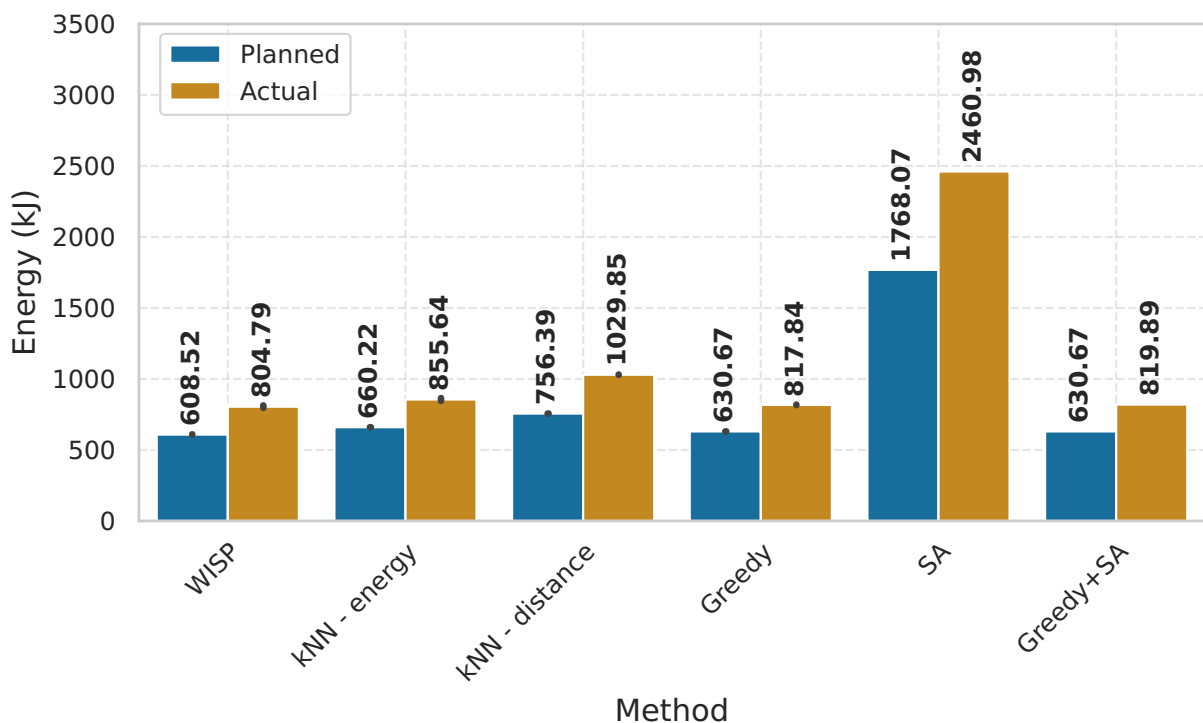


Figure 56 – Planned vs Actual time by method.

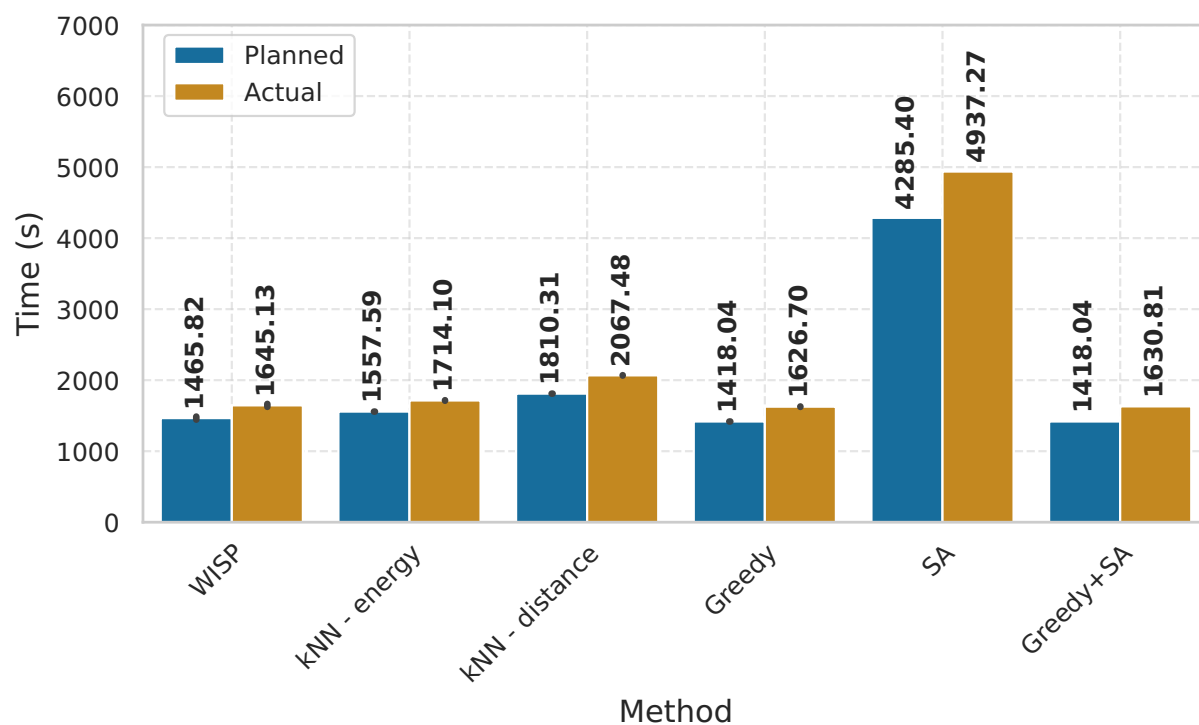


Figure 57 – Planned vs Actual distance by method.

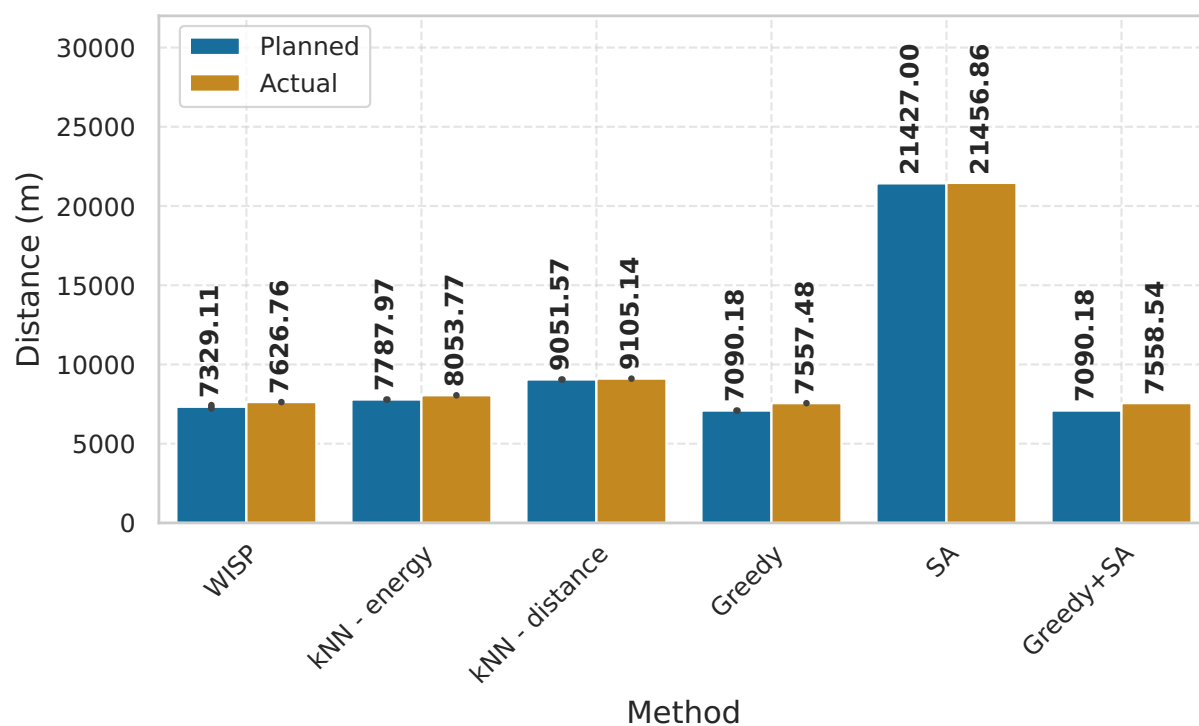
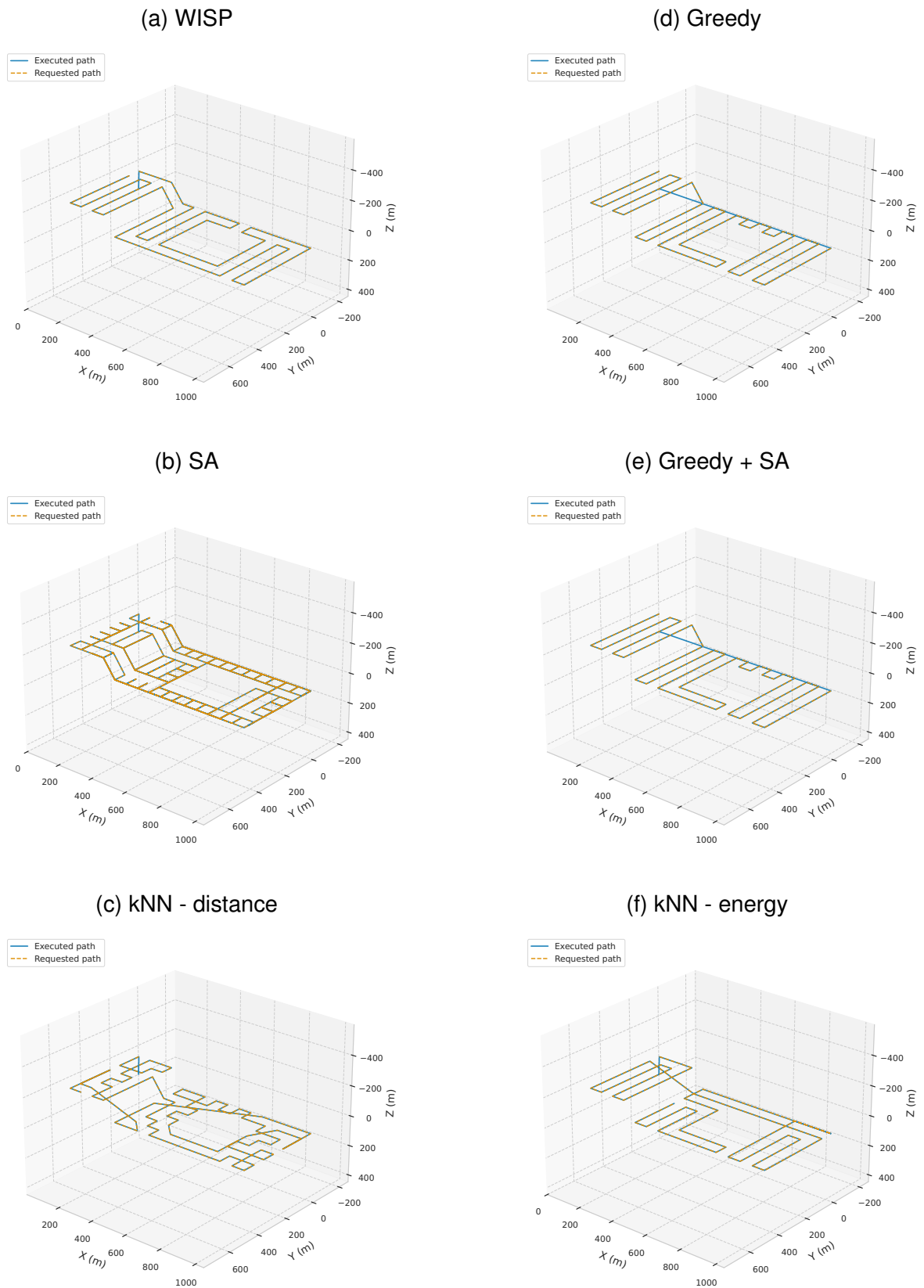


Figure 58 – Planned and executed trajectories for each method in simulation.



5.2 EXPERIMENTS FOR THE AUTONOMOUS LANDING SYSTEM

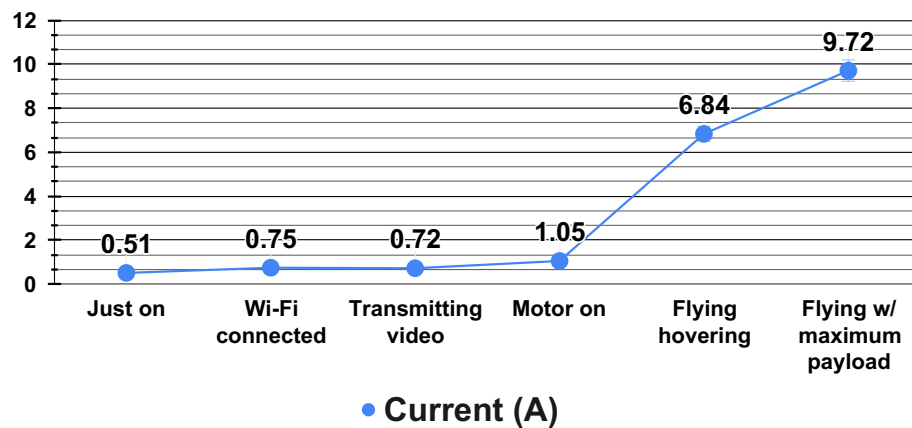
This section presents experiments related to the ALS. It begins by analyzing the battery consumption of the RoboMaster TT (RMTT) under various scenarios, followed by a controlled experiment designed to validate the effectiveness of the LSES. Finally, it describes flight tests in which, despite reduced accuracy from the LSES, the UAV successfully lands on a site either selected by a user or detected using YOLOv8.

5.2.1 Battery Consumption

This experiment evaluated the battery consumption of the RMTT. The UAV was connected to a power supply set to 4.1 V, and the current draw was measured over one minute across six scenarios.

Figure 59 presents the consumption results for each scenario, where each successive case includes the components of the previous ones. As expected, the graph shows a reduction in power consumption when the UAV is not in flight. For instance, operating the motors solely to prevent overheating consumes 84.6% less current compared to hovering. This finding suggests that suspending the UAV from a tree can significantly extend its environmental monitoring capabilities.

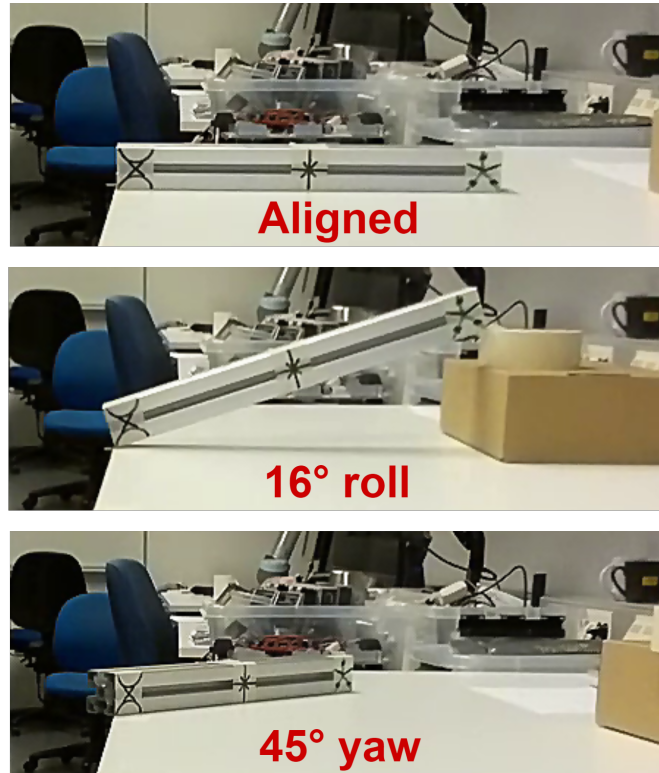
Figure 59 – Current consumption by scenario. (Kunz Cechinel. *et al.*, 2023)



5.2.2 Experiment in Controlled Scenario

This experiment aimed to evaluate the accuracy of the LSES. The landing site was positioned relative to the UAV using known positions defined by (x, y, z) coordinates and $(\text{yaw}, \text{roll})$ angles. Accordingly, the experiment assessed the accuracy of both relative position and angle estimations. Figure 60 displays a selection of images used in the experiment: the first with full alignment, the second with a 16° roll angle, and the third with a 45° yaw angle. The landing site measured $40 \text{ cm} \times 4.5 \text{ cm}$ and included three visual tags to support the feature detection process.

Figure 60 – Examples of roll and yaw variations within the experiment. (Kunz Cechinel. *et al.*, 2023)



The UAV was placed on a height-adjustable table, and two images were captured using its onboard camera. The second image was taken 15 cm above the first. A user selected the landing site in the first image, after which the LSES estimated the real-world position of the detected and matched features at the selected site. Subsequently, it computed the average location of these features in the environment. The yaw and roll angles were estimated from the set of points using the least squares method, yielding absolute values. Finally, the estimated average position and angles were compared with the actual center position of the physical landing site.

Table 6 presents the central position of the physical landing site in the real world for each tested position and angle. The X coordinate denotes the UAV's lateral distance, the Y coordinate indicates the longitudinal distance, and the Z coordinate represents the height relative to the UAV. The distances selected for the Y coordinate were based on the image quality captured by the RMTT's camera. Preliminary tests showed that the system detects and matches features inaccurately for landing sites located beyond 200 cm.

Table 6 – Real-world landing site center location relative to the UAV's camera. (Kunz Cechinel. *et al.*, 2023)

X (cm)	Y (cm)	Z (cm)	Yaw (°)	Roll (°)
15.18	100.00	-29.60	0.00	0.00
15.18	140.00	-29.60	0.00	0.00
15.18	180.00	-29.60	0.00	0.00
14.87	140.00	-26.13	0.00	10.00
14.40	140.00	-24.09	0.00	16.00
13.97	146.84	-29.60	20.00	0.00
9.32	154.14	-29.60	45.00	0.00

Initially, the landing site was positioned at 100 cm, 140 cm, and 180 cm without variation in yaw and roll angles. Subsequently, the landing site remained fixed at 140 cm, while roll angles of 10° and 16° were introduced. In the final phase, the landing site was fixed at 140 cm, and yaw angles of 20° and 45° were applied. Table 7 presents the estimated center location of the landing site relative to the UAV's camera.

Table 7 – Estimated landing site center location relative to the UAV's camera in the second image. (Kunz Cechinel. *et al.*, 2023)

X (cm)	Y (cm)	Z (cm)	Yaw (°)	Roll (°)
17.38	96.85	-29.41	-0.24	-1.39
16.22	137.46	-29.70	0.18	0.18
17.49	177.31	-29.99	2.87	-0.18
16.57	134.47	-24.00	2.60	10.46
12.38	131.71	-22.87	1.48	18.53
12.73	143.00	-29.56	17.99	-0.78
7.96	146.97	-29.03	43.03	0.24

Tables 6 and 7 form the basis of Table 8, which presents the accuracy and angular error with respect to the landing site's center location relative to the UAV's camera. Accuracy was determined using Equation (41), while the roll and yaw errors correspond to the absolute differences between the estimated and actual angles.

$$Accuracy = 1 - \left| \frac{estimated - real}{real} \right| \quad (41)$$

The lowest accuracy result in Table 8 corresponds to the *X* coordinate. This was influenced by the average position of the detected features on the landing site. As the system detected a greater number of features in the middle and right-side tags, the average *X* coordinate was biased toward the right, resulting in reduced accuracy. However, the accuracy for the distance to the landing site (*Y*) and the height (*Z*) exceeded 96%. Although the yaw and roll angle estimations exhibited errors greater than 3°, they remained sufficient for the UAV to approach the landing site while facing forward. These

results demonstrate that the LSES provides sufficient accuracy to effectively detect static landing sites.

Table 8 – Position accuracy and angle error to the landing site center location relative to the UAV's camera. (Kunz Cechinel. *et al.*, 2023)

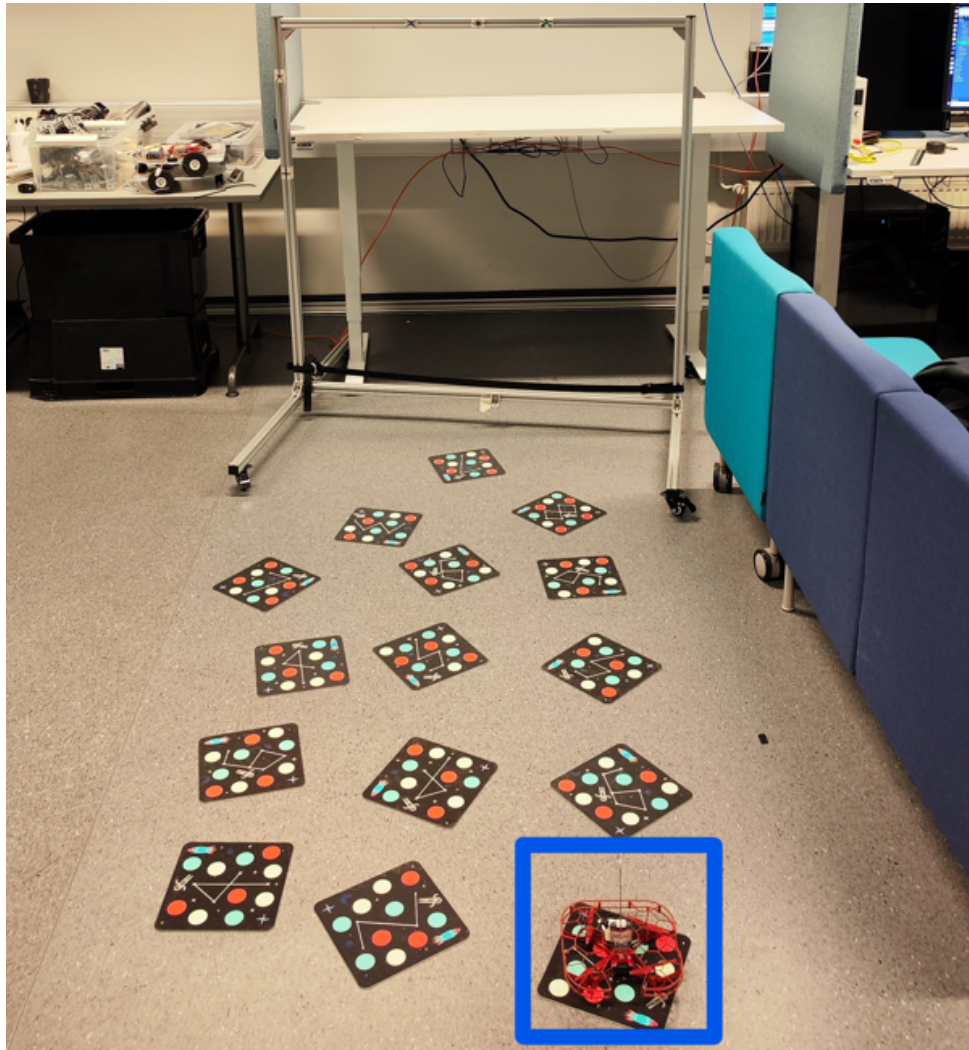
Accuracy (%)			Error (°)	
X	Y	Z	YAW	ROLL
85.44	96.85	99.36	0.24	1.39
93.09	98.19	99.68	0.18	0.18
84.73	98.51	98.68	2.87	0.18
88.60	96.05	91.85	2.60	0.46
85.98	94.08	94.96	1.48	2.53
91.12	97.38	99.85	2.01	0.78
85.45	95.35	98.06	1.97	0.24
Mean				
87.77	96.63	97.49	1.62	0.82

5.2.3 Experiment in Flight – Landing Site Selection by User

Following the evaluation of the LSES in a controlled setting, the system was integrated into the UAV's navigation module, enabling the UAV to perform landings by hooking onto a designated area.

In this experiment, the UAV's initial position was approximately 180 cm to the right of the landing site. The landing site was positioned with a yaw of approximately -20° and a roll of 0° . It is worth noting that, following takeoff, the UAV's movement caused a deviation from the initial 180 cm distance until stabilization was achieved. In each experimental trial, the landing site was selected by the user once the UAV had reached a stable state. The RMTT UAV is equipped with an optical flow system for estimating its relative position from the takeoff point. To enhance the accuracy of this system, reflective pads were placed on the laboratory floor. Figure 61 illustrates the experimental setup.

Figure 61 – Experimental environment. The takeoff region is marked with a blue square. (Kunz Cechinel. *et al.*, 2023)



In this experiment, the success rate of landing attempts was assessed, the routes taken by the UAV were recorded, and the accuracy of the LSES during flight was measured. A total of 20 landing attempts were conducted, of which only one was unsuccessful due to a delay in adjusting the UAV's height to reach the landing site. For each successful landing, the minimum, maximum, and estimated landing points, the actual landing position, and the route taken during landing were recorded. The estimated position of the landing site was determined relative to the UAV's coordinates using Visual Inertial Odometry (VIO) in step 5 of the landing pipeline. The actual landing position corresponds to the coordinates provided by VIO at the moment the UAV made contact with the landing site.

Table 9 presents the error between the actual and estimated landing points for the 19 successful landing attempts. The error was calculated by subtracting the estimated landing point from the actual one. Minor errors were observed in the X, Z, and yaw components. In all cases, the impact point remained within the landing site's

minimum and maximum X coordinates. The most significant error in Z was 2.183 cm in absolute value. Considering that the hook has approximately 10 cm of allowable space to engage with the landing site, and that the system sets the target in the center of this space, an error of up to 4 cm remains acceptable for successful landings. Therefore, the results satisfy the system's requirements.

Table 9 – Error between actual and estimated landing points. Values in red indicate that the UAV landed farther from the expected position. (Kunz Cechinel. *et al.*, 2023)

Error (cm)			Error ($^{\circ}$)
X	Y	Z	YAW
4.45	-5.99	-0.23	0.60
2.90	-13.54	-2.18	-0.20
6.75	-18.60	0.26	-2.67
3.89	-8.32	0.13	-1.23
6.48	-15.79	1.07	0.10
6.57	-32.04	-0.65	-0.18
4.74	-12.66	1.01	-0.39
-1.10	-11.33	0.19	0.54
4.77	-71.87	1.45	-1.29
13.34	-22.81	1.75	-9.15
3.63	-6.80	-0.82	2.58
-2.59	1.66	-0.67	-0.62
-1.37	-1.62	1.74	-2.08
-0.66	11.14	-1.62	-1.00
0.19	-13.78	1.64	-10.30
1.92	-49.42	-0.40	-3.14
5.37	-0.62	-0.43	-0.28
-1.36	3.09	0.53	-1.77
-1.06	-2.14	0.47	-1.14
Average			
2.99	-14.29	0.170	-1.66

The system adds 20 cm to the UAV's absolute movement in the Y direction during the acquisition step to the estimated landing site's Y coordinate to ensure a successful landing. As a result, a negative error in the Y column (Table 9) is expected, indicating that the UAV landed closer in the Y direction than predicted. Significant errors, such as -49 cm, are attributed to UAV position instability during the acquisition step. In such cases, forward movement increases the disparity of detected features, resulting in larger estimated Y distances. Conversely, backward movement reduces disparity, leading to smaller estimated Y distances. Nevertheless, incorporating the UAV's absolute Y movement helps mitigate estimation errors. Even when the landing site is estimated to be farther than its actual position, the system is still able to ensure a successful landing.

Three positive red values in the Y column (Table 9) indicate that the UAV landed farther from the expected position. This outcome is attributed to the system's behavior: upon reaching the landing site or making contact, the Y controller is set to half of the maximum forward force for one second. Subsequently, the height controller is set to half of the maximum downward force, followed by motor shutdown. This sequence is designed to lean the UAV onto the site during landing, ensuring a successful touchdown even if physical contact has not yet occurred, but the UAV is nearby.

Figures 62 and 63 depict the flight path for the best result, while Figures 64 and 65 illustrate the flight path for the worst result. The system adjusts the X and Z coordinates once the relative position of the landing site is determined. Upon reaching a certain threshold, the UAV proceeds toward the intended landing spot. This behavior is observed in both the best and worst results.

Following the initial adjustment in the X and Z coordinates, the UAV navigates toward the landing point while aligning its yaw to ensure a frontal impact. As the landing site presents a yaw angle of approximately -20° relative to the UAV, a clockwise rotation is executed to direct the UAV toward the right side. Subsequently, the roll and pitch controllers stabilize the UAV's position, enabling a successful landing. This behavior appears as a zig-zag pattern in both results.

In the worst result (Figures 64 and 65), the flight path terminates near the desired X and Z coordinates, but the Y coordinate deviates by approximately 70 cm from the landing point. Instability during the acquisition step resulted in an inaccurate Y estimation. However, despite this poor estimation, the accuracy of the X and Z coordinates remained sufficient to enable a successful landing. Thus, even with a larger error in the Y distance, the UAV is still capable of achieving a successful landing. Conversely, an underestimated Y coordinate with a significant error would hinder landing accuracy. Therefore, the safety measure of adding 20 cm plus the absolute Y displacement during the acquisition step contributes to successful landings, particularly in scenarios involving acquisition instability.

Figure 62 – Route in X and Y axes for the best result. (Kunz Cechinel. *et al.*, 2023)

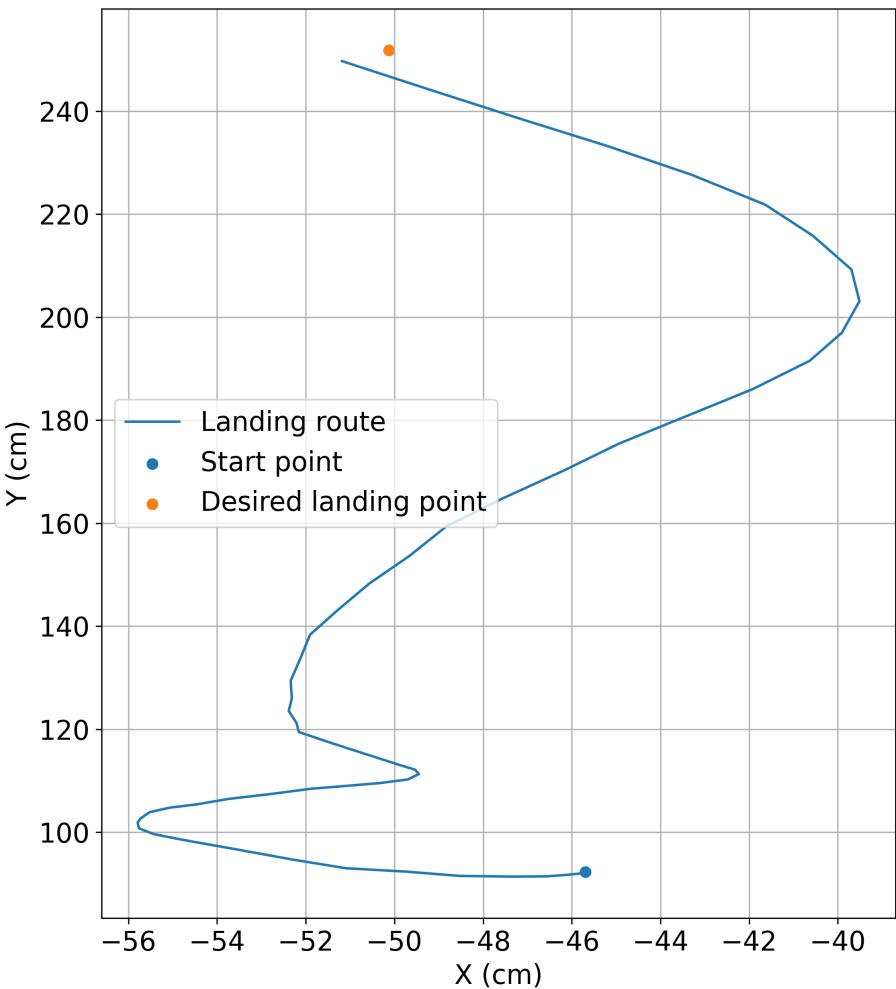


Figure 63 – Route in Y and Z axes for the best result. (Kunz Cechinel. *et al.*, 2023)

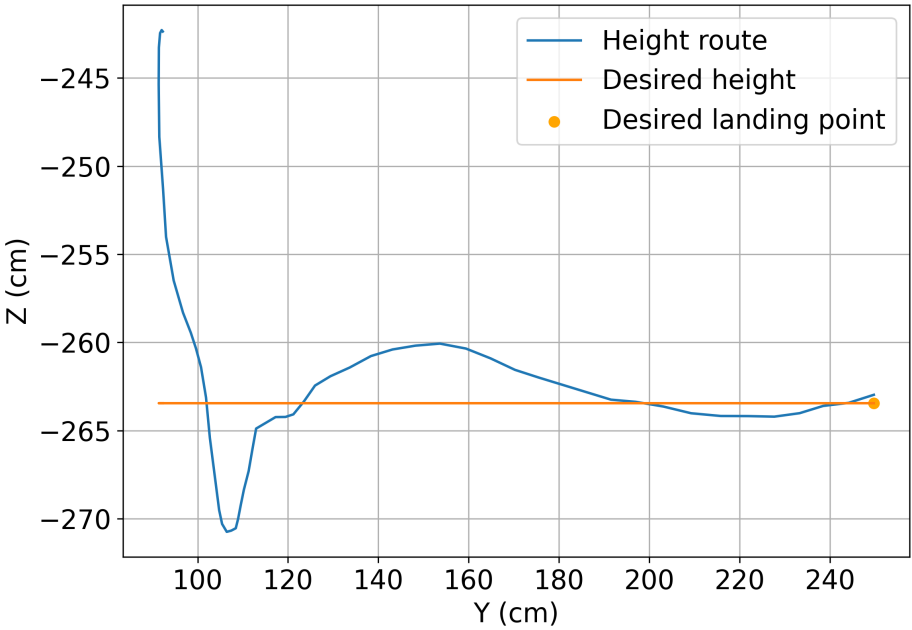


Figure 64 – Route in X and Y axes for the worse result. (Kunz Cechinel. *et al.*, 2023)

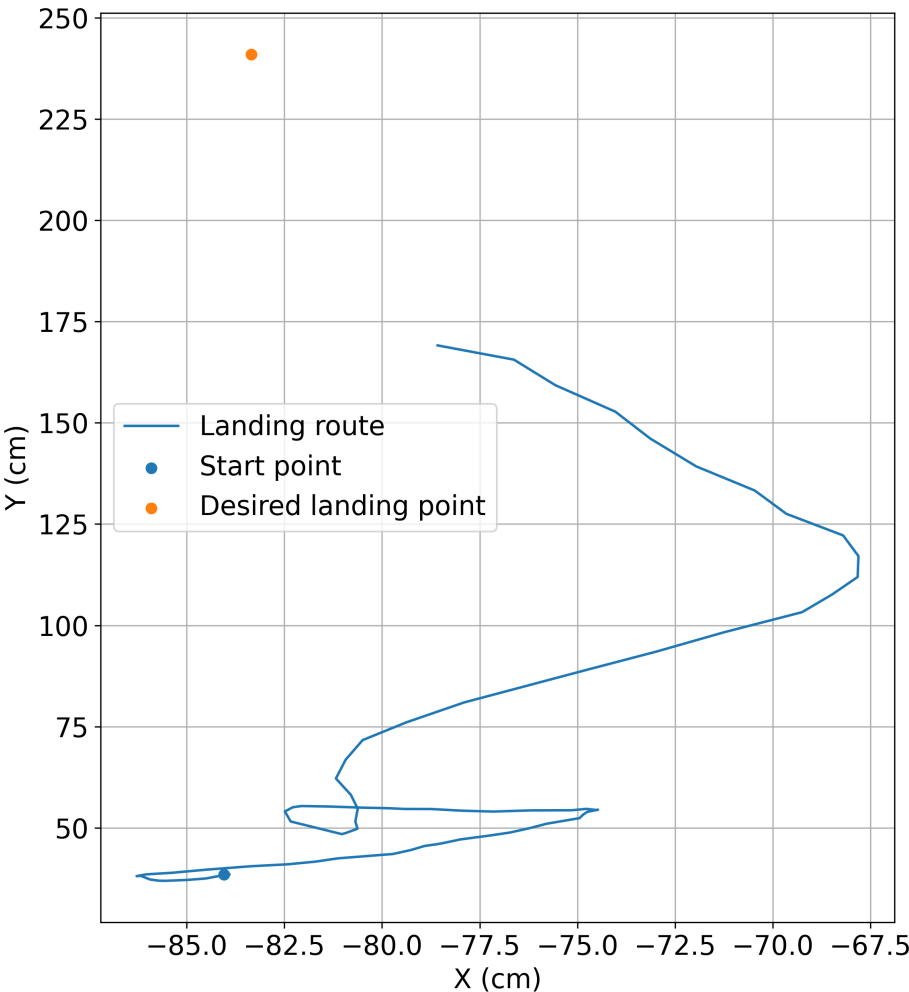
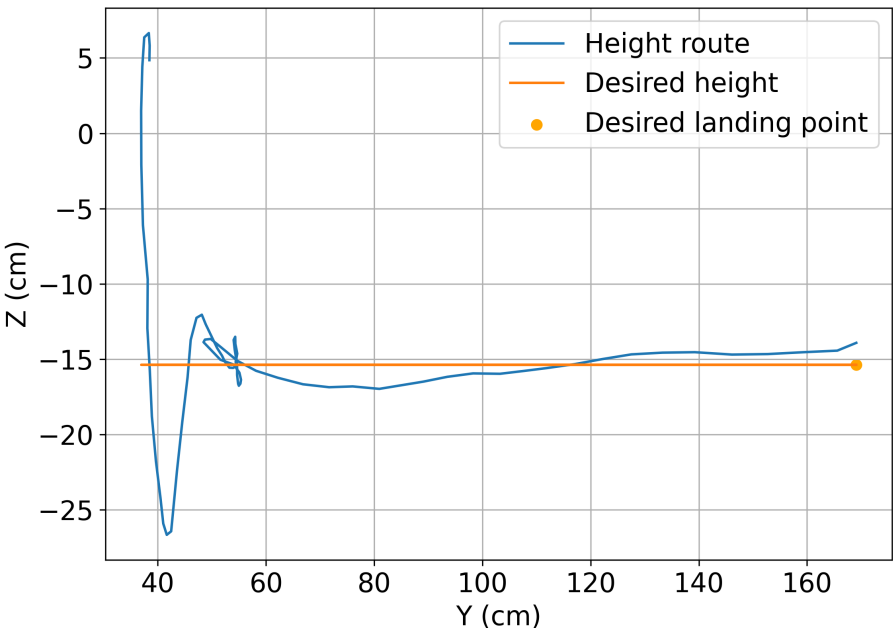


Figure 65 – Route in Y and Z axes for the worse result. (Kunz Cechinel. *et al.*, 2023)



5.2.4 Experiment in Flight – Landing Site Selection by YOLOv8

The experiments described in the previous section were based on the system developed by Kunz Cechinel. *et al.* (2023). In the current experiment, the setup was reproduced with one key modification: the responsibility for selecting the landing site was transferred to YOLOv8. This version also featured enhanced navigation controllers, achieved through careful tuning. Furthermore, a tracking system was introduced to improve the speed of feature detection and matching. These enhancements contributed to greater overall system stability during the experiment. Figure 66 shows the experimental environment, which closely resembles that of the previous setup.

Figure 66 – Experimental environment for YOLO experiments.



Analogously to the previous section, Figures 67 and 68 show the executed route for the best result, while Figures 69 and 70 show the executed route for the worst result. The system adjusts the X and Z coordinates after determining the relative position of the landing site. Once a certain threshold is reached, the UAV navigates toward the desired landing point. This behavior is observed in both the best and worst results.

As shown in Table 10, adjustments made to the system enabled successful landings in 19 out of 20 attempts, without misjudging the distance to the branch. Additionally, the average error in the Y coordinate was approximately -27 cm, corresponding to the adjustment implemented to accommodate the new hook model, which extended the distance by 25 cm in the Y direction. A review of the routes in Figures 67–70 and the results in Table 10 indicates a clear overall improvement.

Figure 67 – Route in X and Y axes for the best result.

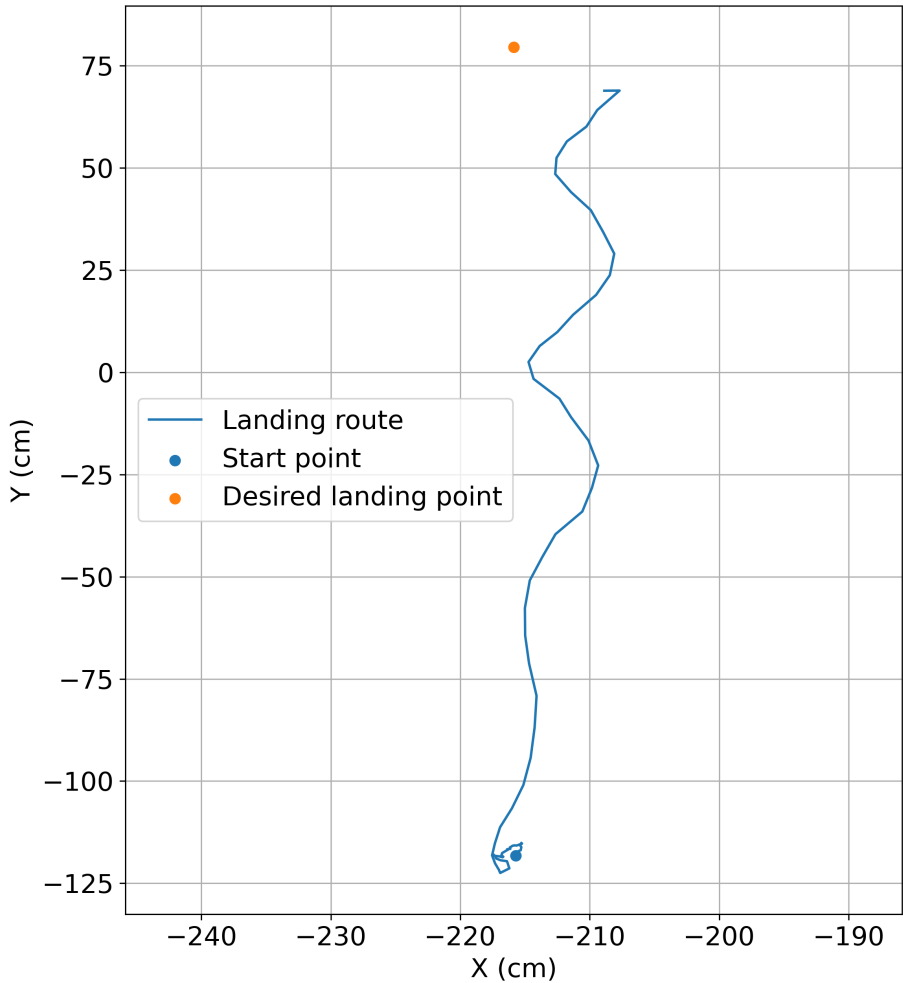


Figure 68 – Route in Y and Z axes for the best result.

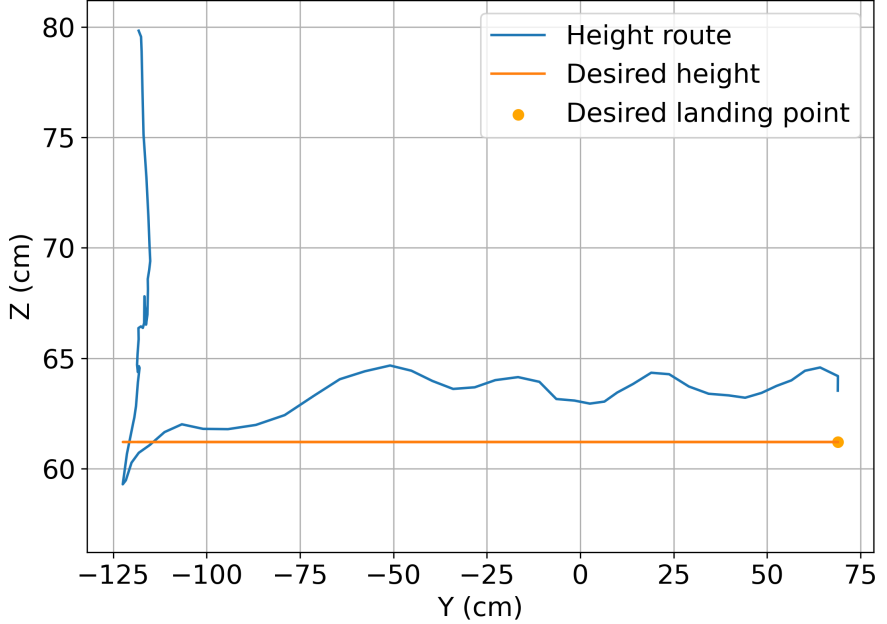


Figure 69 – Route in X and Y axes for the worse result.

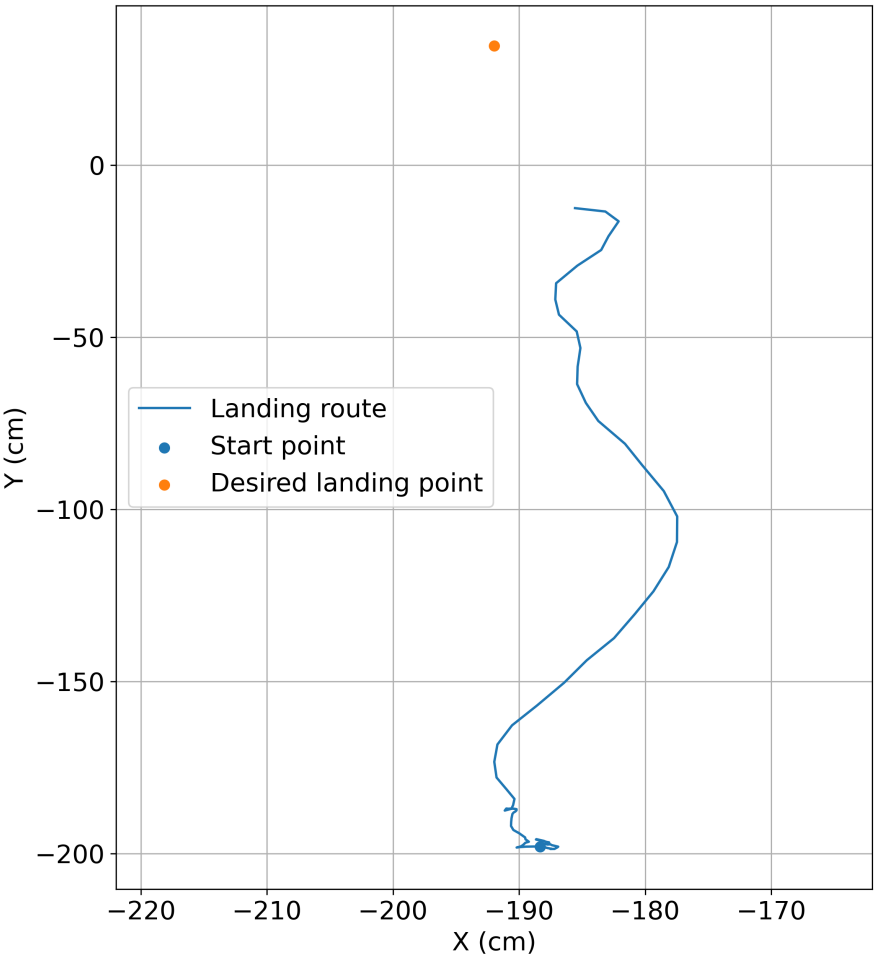


Figure 70 – Route in Y and Z axes for the worse result.

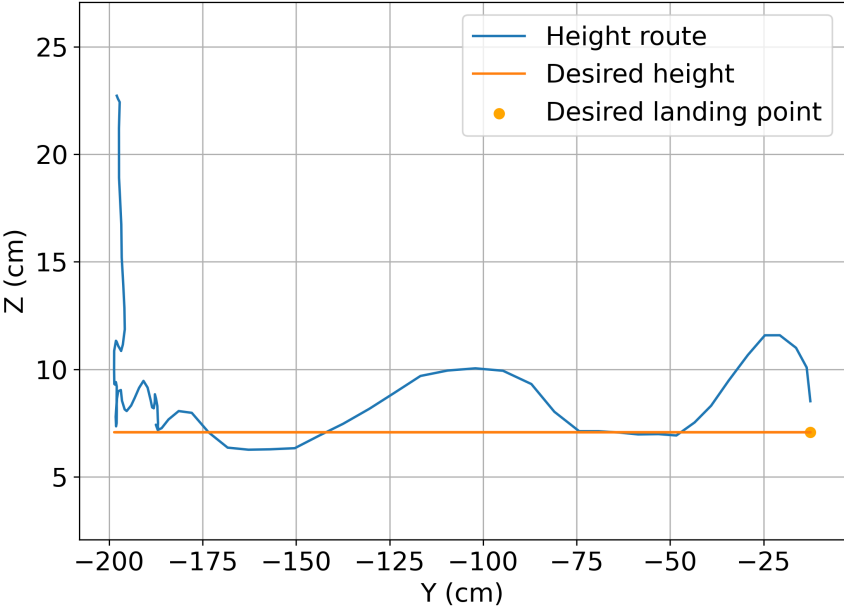


Table 10 – Error between actual and estimated landing points.

Error (cm)			Error ($^{\circ}$)
X	Y	Z	YAW
5.33	-28.44	2.68	-3.2
4.11	-36.82	1.67	0.35
7.83	-40.38	1.74	-0.39
-0.35	-26.6	0.81	3.57
10.48	-25.46	0.62	-0.01
6.42	-47.16	1.44	-1.93
9.94	-16.98	-0.85	1.85
6.76	-25.89	0.32	-0.76
7.00	-10.66	2.32	0.78
0.54	-23.35	0.22	-1.94
3.36	-22.71	2.64	-0.27
5.22	-24.79	2.74	-0.04
0.09	-11.04	2.84	0.58
12.77	-24.94	2.69	2.13
6.15	-45.89	0.7	0.78
6.24	-43.59	2.51	-0.89
10.46	-25.26	1.49	-2.91
1.24	-22.23	-0.95	-0.08
7.14	-18.62	-1.71	1.92
Average			
5.83	-27.41	1.26	-0.02

5.3 CONCLUSION

This chapter presented the experimental evaluation of the proposed framework, showing how this thesis advances the state of the art in energy-aware coverage planning and autonomous landing for small UAVs. The results demonstrated that the heuristic energy consumption model developed in this work provides an accurate approximation of UAV power usage under varying flight profiles and wind conditions. The close agreement between theoretical predictions and the outputs of the Gazebo energy plugin, particularly for level flight, supports its use as a practical and lightweight surrogate for more complex models. Thus, the thesis contributes a reusable, simulation-ready energy model that can be integrated into coverage planners and virtual testbeds, bridging the gap between purely theoretical formulations and realistic mission analysis.

The comparative analysis of coverage path planning algorithms confirmed the effectiveness of WISP as a wind-informed planner and highlighted its contribution beyond existing CPP methods. Across diverse terrain configurations and wind conditions, WISP consistently achieved the lowest energy consumption while maintaining competitive execution time and memory usage. It also produced visually coherent paths, avoiding issues such as repeated coverage, incursions into no-fly zones, and invalid al-

titude transitions found in some of the baseline methods. These results show that WISP fills the gap in the literature, where most planners remain wind-agnostic or treat wind only implicitly. Moreover, the analysis revealed that paths with slightly more turns can still be energetically advantageous when wind and elevation are properly accounted for, challenging the common assumption that minimizing turns alone leads to more energy-efficient paths.

The investigation of K-Means variants for area allocation contributes to the state of the art in multi-UAV workload distribution by quantifying the trade-off between cluster balance and computational cost. Constrained K-Means produced the most uniform energy distribution among UAVs, at the price of increased computation time. These findings offer concrete guidance for future CPP frameworks that rely on clustering for task allocation, indicating when it is beneficial to adopt constrained formulations instead of conventional K-Means variants. Simulation-based validation using the Gazebo plugin further reinforced the practical relevance of WISP, showing that, even under simulations of flight dynamics and controller-induced deviations, WISP preserves its energy advantage. The observed penalties caused by incorrect wind information also highlighted the importance of integrating reliable wind estimates into coverage planners for UAVs.

The second part of the chapter advanced the state of the art in persistent aerial monitoring by demonstrating that landing on natural structures is a viable and effective strategy to extend mission endurance. Battery consumption experiments with the RMTT showed that suspending the vehicle from a branch drastically reduces power usage compared to hovering, providing quantitative evidence for the benefits of branch-landing strategies. In controlled tests, the LSES achieved high accuracy in estimating the relative position and orientation of the landing site, particularly in distance and height, with angular errors still compatible with successful alignment. Flight experiments with user-selected landing sites confirmed that the system can execute hooking maneuvers, achieving a high success rate even in the presence of position estimation noise.

Finally, the integration of YOLOv8 for automatic branch selection further pushed the level of autonomy of the ALS. The improved controllers and tracking pipeline yielded stable trajectories and successful landings in 19 out of 20 attempts, with systematic errors along the Y axis being effectively mitigated by the safety margins incorporated into the landing logic. Taken together, the advances presented in this chapter validate the main hypotheses of the thesis: that a wind-informed planner such as WISP can reduce energy consumption relative to wind-agnostic methods, and that a monocular vision-based system, combined with a hook-based mechanism, can enable reliable and repeatable autonomous landings of small UAVs on natural structures. By jointly addressing energy-aware coverage, multi-UAV workload allocation, and autonomous branch-landing, this work contributes towards scalable and persistent UAV-based environmental monitoring frameworks.

6 CONCLUSION

This thesis presented a hierarchical, multicopter-based Unmanned Aerial Vehicle (UAV) architecture designed to support scalable, energy-efficient, and persistent environmental monitoring. This architecture is composed by three main components: a coordination framework, a Multiple Coverage Path Planning (MCP) method, and an autonomous landing strategy tailored for natural terrain.

The research contributions centered on the development and validation of the Multi-Coverage Path Planning (CPP) method and the autonomous landing approach. Together, these components were designed to enhance the long-term operational capabilities of UAVs in outdoor environments. Through this work, it was demonstrated that incorporating wind direction, magnitude, and altitude variation into CPP planning reduces the estimated energy consumption when compared to traditional wind-agnostic strategies. Additionally, the results showed that a monocular vision-based system, leveraging stereo-from-motion and a hook-based mechanism, can enable reliable and repeatable autonomous landings on natural structures, such as tree branches.

In order to address the first contribution, the energy consumption model from Sorbelli *et al.* (2021) was extended to account for vertical UAV movements. Building on this, the Wind-Informed Sweep Planner (WISP) method was developed. It discretizes the environment, distributes the area among multiple UAVs, and computes individual coverage paths using an approach based on Minimum Spanning Tree (MST) that minimizes energy usage.

Besides energy minimization, a WISP's design goal was workload balancing. In pursuit of this objective, three variations of the K-Means algorithm were evaluated for area allocation: standard K-Means (Lloyd, 1957; MacQueen, 1967), Mini Batch K-Means (Sculley, 2010), and Constrained K-Means (Bradley; Bennett; Demiriz, 2000). As expected, Constrained K-Means outperformed the others, generating more balanced clusters in terms of area size.

In addition to area partitioning, the performance of WISP was benchmarked against five baseline CPP algorithms. Four of these incorporated the energy model: Greedy, K-Nearest Neighbors (KNN), a hybrid Greedy + Simulated Annealing (SA), and a standalone SA method. The fifth was a distance-based KNN variant that did not employ the extended energy model. The evaluation considered solution time, memory usage, and estimated energy consumption across multiple environmental scenarios. The results demonstrated that WISP consistently outperformed all baseline methods in every criterion. These findings corroborate the hypothesis that incorporating wind direction, magnitude, and altitude variation into UAV CPP reduces estimated energy consumption compared to traditional wind-agnostic planning approaches. Moreover, the results revealed that, contrary to what was suggested by Yan Li *et al.* (2011) and

Jiadong Li, Xueqi Li, and Lijuan Yu (2018), coverage paths with fewer turning maneuvers are not always more energy-efficient for UAVs when wind effects are taken into account.

The CPP validation was conducted using a custom Gazebo plugin, which employs the extended energy model to estimate real-time energy consumption at each simulation step. Additionally, the method was integrated with the PX4 autopilot, enabling reproducible simulations and real-time mission control. The simulation results demonstrated that the proposed system functions reliably and shows potential for integration with real-world hardware.

Finally, a proof-of-concept for the autonomous landing system was developed, enabling UAVs to land on natural structures using monocular vision and a hook-based mechanism. The system incorporates a YOLO-based Artificial Intelligence (AI) model trained to detect horizontal tree branches. Using the bounding box that defines the detected branch region, along with image features and the UAV's motion, the system estimates the relative position of the branch and performs a hooking maneuver to land. Experimental results in a controlled environment demonstrated high reliability, achieving a 95% success rate in landing attempts. Therefore, the results support the hypothesis that a monocular vision-based system combining stereo-from-motion and a hook-based mechanism enables reliable and repeatable autonomous landings of UAVs on natural structures.

6.1 LIMITATIONS AND FUTURE WORK

This thesis has made several contributions to the field of multi-UAV systems. However, it did not fully meet all the requirements necessary to develop a complete hierarchical architecture for long-term monitoring. Additionally, the Landing Site Estimation System (LSES) was validated only in a laboratory setting, and the WISP was tested exclusively through simulation. Given these limitations, this section outlines the main challenges encountered and proposes directions for future work.

First of all, the proposed hierarchical architecture lacks a coordination system capable of handling task planning and allocation to the UAVs in an online manner, while also monitoring execution and performing replanning and reassignment of coverage areas in the event of failures. A coordination system integrated with PX4 would represent a solid solution for area coverage, especially when accounting for energy consumption influenced by wind. This would provide a robust approach not only for long-term monitoring but also for applications such as crop field photogrammetry.

The WISP was designed, developed, and validated based on a well-known energy model used to compute the power requirements of helicopters, as described by Johnson (1994). A plugin was developed using this model to enable simulation-based validation in Gazebo. However, despite relying on a validated model, the significance of the planner would reach a much higher level if it were tested on real UAVs. This would

allow for direct comparison with other methods to evaluate whether the differences in energy consumption observed in simulation are consistent with real-world performance.

Another extension to WISP may involve estimating whether the group of UAVs can complete its coverage tasks given the available battery capacity. In addition, it would be valuable to consider different profiles for both UAV and wind velocities, enabling the planner to provide more accurate estimates in regions such as cliffs, where wind conditions may vary with altitude. Furthermore, the area allocation strategy could be enhanced by accounting for the UAVs' initial take-off points and, when necessary, battery replacements, allowing the planner to estimate return locations during mission planning.

An interesting development in this thesis was the autonomous landing system. It relies on the LSES to compute the relative position of a tree branch. Based on this information, it generates commands that guide the UAV toward the branch, allowing it to hang itself. However, the system was validated only in a laboratory setting, which provides a limited perspective on its robustness in real-world conditions. Preliminary experiments indicate that the system is promising. Even so, operating in a forest would require a UAV with better localization capabilities and greater endurance than the RoboMaster TT (RMTT) used in the lab.

The autonomous landing system was also tested using a powerful computer. For a proof-of-concept, this is acceptable, as it offers a controlled environment to evaluate the system without hardware constraints. However, deploying it on a real UAV would require optimizations to make it compatible with the limited resources available onboard. One possible solution is to use a smaller version of the YOLO model, such as YOLOv11n, which can run on embedded platforms like the Raspberry Pi. These lightweight models are capable of real-time inference, even in resource-constrained environments. Nevertheless, to achieve reliable results, it is necessary to use a high-quality dataset and apply proper tuning during training. Otherwise, both detection accuracy and system performance may degrade in real-world scenarios.

A final direction for future work could be to focus on extending the autonomous landing system to also include an autonomous take-off capability. This would allow the small UAV to return to the long-range UAV or enable a user to later retrieve it under manual control.

6.2 LITERATURE PUBLICATIONS

Paper related to multi-robot task allocation:

CECHINEL, Alan Kunz *et al.* Multi-robot Task Allocation Using Island Model Genetic Algorithm. **IFAC-PapersOnLine**, v. 54, n. 1, p. 558–563, 2021. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021. ISSN 2405-8963

Paper related to the autonomous landing system:

KUNZ CECHINEL., Alan *et al.* Hanging Drone: An Approach to UAV Landing for Monitoring. *In*: INSTICC. PROCEEDINGS of the 20th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO. [S. l.]: SciTePress, 2023. p. 363–373. ISBN 978-989-758-670-5

Paper related to CPP for a poultry litter robot:

KUNZ CECHINEL, Alan *et al.* Mobile Robot + IoT: Project of Sustainable Technology for Sanitizing Broiler Poultry Litter. **Sensors**, v. 24, n. 10, 2024. ISSN 1424-8220

Extension to paper related to the autonomous landing system:

CECHINEL, Alan Kunz *et al.* UAV Deployment for Wildfire Monitoring: Introducing the Hanging Drone Landing Technique. *In*: GINI, Giuseppina; NIJMEIJER, Henk; FILEV, Dimitar (eds.). **Informatics in Control, Automation and Robotics**. Cham: Springer Nature Switzerland, 2025. p. 171–198

A final paper is currently under development to report the results and findings related to the WISP. This work is expected to be published in 2026.

REFERENCES

- ACKERMANN, Johannes; GABLER, Volker; OSA, Takayuki; SUGIYAMA, Masashi. **Reducing Overestimation Bias in Multi-Agent Domains Using Double Centralized Critics**. [S. l.: s. n.], 2019. arXiv: 1910.01465 [cs.LG]. Available from: <https://arxiv.org/abs/1910.01465>.
- AETHA GLOBAL. **Using UAV LiDAR for a Reforestation Project in Indonesia**. Accessed: 16/08/2025. 2023. Available from: <https://www.aetha.global/case-study/using-uav-lidar-for-a-reforestation-project-in-indonesia>.
- AGGARWAL, Shubhani; KUMAR, Neeraj. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. **Computer Communications**, v. 149, p. 270–299, 2020. ISSN 0140-3664.
- ALATISE, Mary B.; HANCKE, Gerhard P. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. **IEEE Access**, v. 8, p. 39830–39846, 2020.
- ALSALAM, Bilal Hazim Younus; MORTON, Kye; CAMPBELL, Duncan; GONZALEZ, Felipe. Autonomous UAV with vision based on-board decision making for remote sensing and precision agriculture. *In*: 2017 IEEE Aerospace Conference. [S. l.: s. n.], 2017. p. 1–12.
- ALVARENGA, Jessica; VITZILAIOS, Nikolaos I.; VALAVANIS, Kimon P.; RUTHERFORD, Matthew J. Survey of Unmanned Helicopter Model-Based Navigation and Control Techniques. **Journal of Intelligent & Robotic Systems**, v. 80, n. 1, p. 87–138, Oct. 2015. ISSN 1573-0409.
- AMMOUS, Mustafa; BELAKARIA, Syrine; SOROUR, Sameh; ABDEL-RAHIM, Ahmed. Optimal Cloud-Based Routing With In-Route Charging of Mobility-on-Demand Electric Vehicles. **IEEE Transactions on Intelligent Transportation Systems**, v. 20, n. 7, p. 2510–2522, 2019.
- ANDERSEN, Håvard Lægreid. **Path Planning for Search and Rescue Mission Using Multicopters**. 2014. MA thesis – Institutt for Teknisk Kybernetikk, Trondheim, Norway.
- APOSTOLIDIS, Savvas D.; KAPOUTSIS, Pavlos Ch.; KAPOUTSIS, Athanasios Ch.; KOSMATOPOULOS, Elias B. Cooperative multi-UAV coverage mission planning platform for remote sensing applications. **Autonomous Robots**, v. 46, n. 2, p. 373–400, Feb. 2022. ISSN 1573-7527.
- ARTEMENKO, Oleksandr; DOMINIC, Omachonu Joshua; ANDRYEYEV, Oleksandr; MITSCHLE-THIEL, Andreas. Energy-Aware Trajectory Planning for the Localization of Mobile Devices Using an Unmanned Aerial Vehicle. *In*: 2016 25th International

Conference on Computer Communication and Networks (ICCCN). [S. l.: s. n.], 2016. p. 1–9.

AULINAS, Josep; PETILLOT, Yvan; SALVI, Joaquim; LLADO, Xavier. The SLAM problem: a survey. *In*: v. 184, p. 363–371.

AVELLAR, Gustavo S. C.; PEREIRA, Guilherme A. S.; PIMENTA, Luciano C. A.; ISCOLD, Paulo. Multi-UAV Routing for Area Coverage and Remote Sensing with Minimum Time. **Sensors**, v. 15, n. 11, p. 27783–27803, 2015. ISSN 1424-8220.

AZPURUA, Héctor; FREITAS, Gustavo M.; MACHARET, Douglas G.; CAMPOS, Mario F. M. Multi-robot coverage path planning using hexagonal segmentation for geophysical surveys. **Robotica**, Cambridge University Press, v. 36, n. 8, p. 1144–1166, 2018.

BÄHNEMANN, Rik; LAWRENCE, Nicholas; CHUNG, Jen Jen; PANTIC, Michael; SIEGWART, Roland; NIETO, Juan. Revisiting Boustrophedon Coverage Path Planning as a Generalized Traveling Salesman Problem. *In*: FIELD and Service Robotics. [S. l.]: Springer Singapore, 2021. p. 277–290.

BARRIENTOS, Antonio; COLORADO, Julian; CERRO, Jaime del; MARTINEZ, Alexander; ROSSI, Claudio; SANZ, David; VALENTE, João. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. **Journal of Field Robotics**, v. 28, n. 5, p. 667–689, 2011.

BECHAR, Avital; VIGNEAULT, Clément. Agricultural robots for field operations: Concepts and components. **Biosystems Engineering**, v. 149, p. 94–111, 2016. ISSN 1537-5110.

BEHZADAN, Vahid; HSU, William. RL-Based Method for Benchmarking the Adversarial Resilience and Robustness of Deep Reinforcement Learning Policies. **arXiv e-prints**, arXiv:1906.01110, arxiv:1906.01110, June 2019. arXiv: 1906.01110 [cs.LG].

BOLME, David S.; BEVERIDGE, J. Ross; DRAPER, Bruce A.; LUI, Yui Man. Visual object tracking using adaptive correlation filters. *In*: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. [S. l.: s. n.], 2010. p. 2544–2550.

BORMANN, Richard; JORDAN, Florian; HAMPP, Joshua; HÄGELE, Martin. Indoor Coverage Path Planning: Survey, Implementation, Analysis. *In*: 2018 IEEE International Conference on Robotics and Automation (ICRA). [S. l.: s. n.], 2018. p. 1718–1725.

BOROWCZYK, Alexandre; NGUYEN, Duc-Tien; PHU-VAN NGUYEN, André; NGUYEN, Dang Quang; SAUSSIÉ, David; NY, Jerome Le. Autonomous Landing of a

Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle**This work was partially supported by CFI JELF Award 32848 and a hardware donation from DJI.

IFAC-PapersOnLine, v. 50, n. 1, p. 10488–10494, 2017. 20th IFAC World Congress. ISSN 2405-8963.

BOURAS, Abdelwahhab; BOUZID, Yasser; GUIATNI, Mohamed. Multi-UAV Coverage Path Planning for Gas Distribution Map Applications. **Unmanned Systems**, v. 10, n. 03, p. 289–306, 2022.

BRADLEY, P. S.; BENNETT, K. P.; DEMIRIZ, A. **Constrained K-Means Clustering**. Redmond, WA, May 2000. Available from: <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2000-65.pdf>.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BRUZZONE, L.; QUAGLIA, G. Review article: locomotion systems for ground mobile robots in unstructured environments. **Mechanical Sciences**, v. 3, n. 2, p. 49–62, 2012.

BUDIHARTO, Widodo; CHOWANDA, Andry; GUNAWAN, Alexander Agung Santoso; IRWANSYAH, Edy; SUROSO, Jarot Sembodo. A Review and Progress of Research on Autonomous Drone in Agriculture, Delivering Items and Geographical Information Systems (GIS). *In: 2019 2nd World Symposium on Communication Engineering (WSCE)*. [S. l.: s. n.], 2019. p. 205–209.

BUEREN, S. K. von; BURKART, A.; HUENI, A.; RASCHER, U.; TUOHY, M. P.; YULE, I. J. Deploying four optical UAV-based sensors over grassland: challenges and limitations. **Biogeosciences**, v. 12, n. 1, p. 163–175, 2015.

BURGARD, Wolfram; HEBERT, Martial; BENNEWITZ, Maren. World Modeling. *In: Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, 2016. p. 1135–1152. ISBN 978-3-319-32552-1.

CABREIRA, Tauã M.; BRISOLARA, Lisane B.; FERREIRA JR., Paulo R. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. **Drones**, v. 3, n. 1, 2019. ISSN 2504-446X.

CAO, Zuo; HUANG, Yuyu; HALL, Ernest. Region filling operations with random obstacle avoidance for mobile robots. **Journal of Robotic Systems**, v. 5, p. 87–102, Apr. 1988.

CASTANEDO, Federico. A Review of Data Fusion Techniques. **The Scientific World Journal**, Hindawi Publishing Corporation, v. 2013, p. 704504, Oct. 2013. ISSN 2356-6140.

CECHINEL, Alan Kunz; DE PIERI, Edson Roberto; FERNANDES PEREZ, Anderson Luiz; MÉA PLENTZ, Patricia Della. Multi-robot Task Allocation

Using Island Model Genetic Algorithm. **IFAC-PapersOnLine**, v. 54, n. 1, p. 558–563, 2021. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021. ISSN 2405-8963.

CECHINEL, Alan Kunz; RÖNING, Juha; TIKANMAKI, Antti; PIERI, Edson Roberto De; PLENTZ, Patricia Della Méa. UAV Deployment for Wildfire Monitoring: Introducing the Hanging Drone Landing Technique. *In*: GINI, Giuseppina; NIJMEIJER, Henk; FILEV, Dimitar (eds.). **Informatics in Control, Automation and Robotics**. Cham: Springer Nature Switzerland, 2025. p. 171–198.

CHEN, Jinchao; DU, Chenglie; LU, Xu; CHEN, Keke. Multi-region Coverage Path Planning for Heterogeneous Unmanned Aerial Vehicles Systems. *In*: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). [S. l.: s. n.], 2019. p. 356–3565.

CHO, Sung Won; PARK, Hyun Ji; LEE, Hanseob; SHIM, David Hyunchul; KIM, Sun-Young. Coverage path planning for multiple unmanned aerial vehicles in maritime search and rescue operations. **Computers & Industrial Engineering**, v. 161, p. 107612, 2021. ISSN 0360-8352.

CHO, Sung-Won; PARK, Jin-Hyoung; PARK, Hyun-Ji; KIM, Seongmin. Multi-UAV Coverage Path Planning Based on Hexagonal Grid Decomposition in Maritime Search and Rescue. **Mathematics**, v. 10, n. 1, 2022. ISSN 2227-7390.

CHOI, Younghoon; CHOI, Youngjun; BRICENO, Simon; MAVRIS, Dimitri N. Energy-Constrained Multi-UAV Coverage Path Planning for an Aerial Imagery Mission Using Column Generation. **Journal of Intelligent & Robotic Systems**, v. 97, n. 1, p. 125–139, Jan. 2020. ISSN 1573-0409.

CHOSSET, Howie. Coverage for robotics – A survey of recent results. **Annals of Mathematics and Artificial Intelligence**, v. 31, n. 1, p. 113–126, Oct. 2001. ISSN 1573-7470.

CHOSSET, Howie; PIGNON, Philippe. Coverage Path Planning: The Boustrophedon Cellular Decomposition. *In*: ZELINSKY, Alexander (ed.). **Field and Service Robotics**. London: Springer London, 1998. p. 203–209.

CHOTON, Jahid Chowdhury; PRABHAKAR, Pavithra. Optimal Multi-Robot Coverage Path Planning for Agricultural Fields using Motion Dynamics. *In*: 2023-May, p. 11817–11823.

COLLINS, Leighton; GHASSEMI, Payam; ESFAHANI, Ehsan T.; DOERMANN, David; DANTU, Karthik; CHOWDHURY, Souma. Scalable Coverage Path Planning of Multi-Robot Teams for Monitoring Non-Convex Areas. *In*: 2021 IEEE International Conference on Robotics and Automation (ICRA). [S. l.: s. n.], 2021. p. 7393–7399.

COLOMINA, I.; MOLINA, P. Unmanned aerial systems for photogrammetry and remote sensing: A review. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 92, p. 79–97, 2014. ISSN 0924-2716.

COOK, G.E. Satellite drag coefficients. **Planetary and Space Science**, v. 13, n. 10, p. 929–946, 1965. ISSN 0032-0633.

DAY, Ryan; SALMON, John. A Framework for Multi-UAV Persistent Search and Retrieval with Stochastic Target Appearance in a Continuous Space. **Journal of Intelligent & Robotic Systems**, v. 103, n. 4, p. 65, Nov. 2021. ISSN 1573-0409.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, v. 1, n. 1, p. 269–271, Dec. 1959. ISSN 0945-3245.

DONG, Jiaxin; LI, Xinde; LIU, Yihai. Multi-quadrotor Cooperative Area Coverage Mission Planning Based on DQN. *In*: p. 672–677.

DONG, Zanliang; LIU, Chunhui. Collaborative Coverage Path Planning of UAV Cluster based on Deep Reinforcement Learning. *In*: 2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC). [S. l.: s. n.], 2021. p. 201–207.

EVERETT, H R. **Sensors for Mobile Robots**. [S. l.]: A K Peters, 1995. ISBN 9781568810485.

FENG, Kai; LI, Weixing; GE, Shengyang; PAN, Feng. Packages delivery based on marker detection for UAVs. *In*: 2020 Chinese Control And Decision Conference (CCDC). [S. l.: s. n.], 2020. p. 2094–2099.

FENG, Yi; ZHANG, Cong; BAEK, Stanley; RAWASHDEH, Samir; MOHAMMADI, Alireza. Autonomous Landing of a UAV on a Moving Platform Using Model Predictive Control. **Drones**, v. 2, n. 4, 2018. ISSN 2504-446X.

FOX, D.; BURGARD, W.; THRUN, S. The dynamic window approach to collision avoidance. **IEEE Robotics & Automation Magazine**, v. 4, n. 1, p. 23–33, 1997.

FOX, Dieter. Adapting the Sample Size in Particle Filters Through KLD-Sampling. **The International Journal of Robotics Research**, v. 22, n. 12, p. 985–1003, 2003.

FUNG, Man Lok; CHEN, Michael Z. Q.; CHEN, Yong Hua. Sensor fusion: A review of methods and applications. *In*: 2017 29th Chinese Control And Decision Conference (CCDC). [S. l.: s. n.], 2017. p. 3853–3860.

GABRIELY, Y.; RIMON, E. Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot. *In*: PROCEEDINGS 2002 IEEE International

Conference on Robotics and Automation (Cat. No.02CH37292). [S. l.: s. n.], 2002. v. 1, 954–960 vol.1.

GALCERAN, Enric; CARRERAS, Marc. A survey on coverage path planning for robotics. **Robotics and Autonomous Systems**, v. 61, n. 12, p. 1258–1276, 2013. ISSN 0921-8890.

GAO, Xinyu; LI, Jinhai; FAN, Lifeng; ZHOU, Qiao; YIN, Kaimin; WANG, Jianxu; SONG, Chao; HUANG, Lan; WANG, Zhongyi. Review of Wheeled Mobile Robots' Navigation Problems and Application Prospects in Agriculture. **IEEE Access**, v. 6, p. 49248–49268, 2018.

GARG, P. K. **Unmanned Aerial Vehicles**. [S. l.]: Mercury Learning and Information, Aug. 2021. ISBN 978-1683927099.

GAUTAM, Alvika; SINGH, Mandeep; SUJIT, Pedda Baliyarasimhuni; SARIPALLI, Srikanth. Autonomous Quadcopter Landing on a Moving Target. **Sensors**, v. 22, n. 3, 2022. ISSN 1424-8220.

GIERNACKI, Wojciech; SKWIERCZYŃSKI, Mateusz; WITWICKI, Wojciech; WROŃSKI, Paweł; KOZIERSKI, Piotr. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. *In: 2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*. [S. l.: s. n.], 2017. p. 37–42.

GOMEZ, Juan Irving Vasquez; LOZADA, Juan Carlos Herrera; CARBAJAL, Mauricio Olguin. Coverage Path Planning for Surveying Disjoint Areas. *In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. [S. l.: s. n.], 2018. p. 899–904.

GONÇALVES, Vinicius M.; PIMENTA, Luciano C. A.; MAIA, Carlos A.; PEREIRA, Guilherme A. S. Coordination of multiple fixed-wing UAVs traversing intersecting periodic paths. *In: 2013 IEEE International Conference on Robotics and Automation*. [S. l.: s. n.], 2013. p. 849–854.

GONG, Junghwan; KIM, Hyunbin; LEE, Seunghwan. Resilient Multi-Robot Coverage Path Redistribution Using Boustrophedon Decomposition for Environmental Monitoring. **Sensors**, v. 24, n. 23, 2024.

GOVINDARAJU, Murugaraj; FONTANELLI, Daniele; SELVAKUMAR, S.; PILLAI, Anju S. Optimized Offline-Coverage Path Planning Algorithm for Multi-Robot for Weeding in Paddy Fields. **IEEE Access**, v. 11, p. 109868–109884, 2023.

GRISSETTI, Giorgio; STACHNISS, Cyrill; BURGARD, Wolfram. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. **IEEE Transactions on Robotics**, v. 23, n. 1, p. 34–46, 2007.

GUASTELLA, Dario; CANTELLI, Luciano; LONGO, Domenico; MELITA, Carmelo; MUSCATO, Giovanni. Coverage path planning for a flock of aerial vehicles to support autonomous rovers through traversability analysis. **ACTA IMEKO**, v. 8, p. 9, Dec. 2019.

GUPTA, Raj U.; CONRAD, James M. A Survey on Multi-robot Particle Filter SLAM. *In*: 2019 SoutheastCon. [S. l.: s. n.], 2019. p. 1–5.

HALDER, Srijeet; AFSARI, Kereshmeh. Robots in Inspection and Monitoring of Buildings and Infrastructure: A Systematic Review. **Applied Sciences**, v. 13, n. 4, 2023. ISSN 2076-3417.

HANG, Kaiyu; LYU, Ximin; SONG, Haoran; STORK, Johannes A.; DOLLAR, Aaron M.; KRAGIC, Danica; ZHANG, Fu. Perching and resting—A paradigm for UAV maneuvering with modularized landing gears. **Science Robotics**, v. 4, n. 28, eaau6637, 2019.

HART, Peter E.; NILSSON, Nils J.; RAPHAEL, Bertram. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, 1968.

HASSAN, Mahdi; MUSTAFIC, Daut; LIU, Dikai. Dec-PPCPP: A Decentralized Predator–Prey-based Approach to Adaptive Coverage Path Planning Amid Moving Obstacles. *In*: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S. l.: s. n.], 2020. p. 11732–11739.

HU, Wenjian; YAO, Yu; LIU, Shumei; SHE, Changyang; GUO, Lei; VUCETIC, Branka S.; LI, Yonghui. Multi-UAV Coverage Path Planning: A Distributed Online Cooperation Method. **IEEE Transactions on Vehicular Technology**, v. 72, n. 9, p. 11727–11740, 2023.

IIDA, Michihisa; KANG, Donghyeon; TANIWAKI, Mitsuru; TANAKA, Mutsumi; UMEDA, Mikio. Localization of CO2 source by a hexapod robot equipped with an anemoscope and a gas sensor. **Computers and Electronics in Agriculture**, v. 63, n. 1, p. 73–80, 2008. Special issue on bio-robotics. ISSN 0168-1699.

JOCHER, Glenn; CHAURASIA, Ayush; QIU, Jing. **Ultralytics YOLO (Version 8.0.0) [Computer software]**. [S. l.: s. n.], 2023. Available from: <https://github.com/ultralytics/ultralytics>.

JOHNSON, Wayne. **Helicopter theory**. Rev. ed. New York, N.Y: Dover Publications, 1994. (Dover books on engineering). ISBN 0486682307.

JOINT DIRECTORS OF LABORATORIES. **Data Fusion Lexicon**. Ed. by Franklin E. White. [S. l.], Oct. 1991.

JUNAID, Ali Bin; KONOIKO, Aleksay; ZWEIRI, Yahya; SAHINKAYA, M. Necip; SENEVIRATNE, Lakmal. Autonomous Wireless Self-Charging for Multi-Rotor Unmanned Aerial Vehicles. **Energies**, v. 10, n. 6, 2017. ISSN 1996-1073.

KANELLAKIS, Christoforos; NIKOLAKOPOULOS, George. Survey on Computer Vision for UAVs: Current Developments and Trends. **Journal of Intelligent & Robotic Systems**, v. 87, n. 1, p. 141–168, July 2017. ISSN 1573-0409.

KAPOUTSIS, Athanasios Ch.; CHATZICHRISTOFIS, Savvas A.; KOSMATOPOULOS, Elias B. DARP: Divide Areas Algorithm for Optimal Multi-Robot Coverage Path Planning. **Journal of Intelligent & Robotic Systems**, v. 86, n. 3, p. 663–680, June 2017. ISSN 1573-0409.

KARAPETYAN, Nare; MOULTON, Jason; LEWIS, Jeremy S.; LI, Alberto Quattrini; O'KANE, Jason M.; REKLEITIS, Ioannis M. Multi-robot Dubins Coverage with Autonomous Surface Vehicles. **2018 IEEE International Conference on Robotics and Automation (ICRA)**, p. 2373–2379, 2018.

KAZIM, Muhammed; AZAR, Ahmad Taher; KOUBAA, Anis; IBRAHIM, Zahra Fathy; ZAIDI, Adeel; ZHANG, Lixian. Chapter 21 - Event-driven programming-based path planning and navigation of UAVs around a complex urban environment. *In*: KOUBAA, Anis; AZAR, Ahmad Taher (eds.). **Unmanned Aerial Systems**. [S. l.]: Academic Press, 2021. (Advances in Nonlinear Dynamics and Chaos (ANDC)). p. 531–565. ISBN 978-0-12-820276-0.

KHALASTCHI, Eliahu; KALECH, Meir. Fault Detection and Diagnosis in Multi-Robot Systems: A Survey. **Sensors**, v. 19, n. 18, 2019. ISSN 1424-8220.

KIM, Jeongeun; KIM, Seungwon; JU, Chanyoung; SON, Hyoung Il. Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications. **IEEE Access**, v. 7, p. 105100–105115, 2019.

KIRCHGEORG, Steffen; MINTCHEV, Stefano. HEDGEHOG: Drone Perching on Tree Branches With High-Friction Origami Spines. **IEEE Robotics and Automation Letters**, v. 7, n. 1, p. 602–609, 2022.

KITCHEN, Ryan; BIERWOLF, Nick; HARBERTSON, Sean; PLATT, Brage; OWEN, Dean; GRIESSMANN, Klaus; MINOR, Mark A. Design and Evaluation of a Perching Hexacopter Drone for Energy Harvesting from Power Lines. *In*: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S. l.: s. n.], 2020. p. 1192–1198.

KONDA, Vijay; TSITSIKLIS, John. Actor-Critic Algorithms. *In*: SOLLA, S.; LEEN, T.; MÜLLER, K. (eds.). **Advances in Neural Information Processing Systems**. [S. l.]: MIT Press, 1999. v. 12.

KOUBAA, Anis; BENNACEUR, Hachemi; CHAARI, Imen; TRIGUI, Sahar; AMMAR, Adel; SRITI, Mohamed-Foued; ALAJLAN, Maram; CHEIKHROUHO, Omar; JAVED, Yasir. Introduction to Mobile Robot Path Planning. *In: ROBOT Path Planning and Cooperation: Foundations, Algorithms and Experimentations*. Cham: Springer International Publishing, 2018. p. 3–12. ISBN 978-3-319-77042-0.

KRAMAR, Vadim; RÖNING, Juha; ERKKILÄ, Juha; HINKULA, Henry; KOLLI, Tanja; RAUHALA, Anssi. Unmanned Aircraft Systems and the Nordic Challenges. *In: LIPPING, Tarmo; LINNA, Petri; NARRA, Nathaniel (eds.). New Developments and Environmental Applications of Drones*. Cham: Springer International Publishing, 2022. p. 1–30.

KUMAR GUPTA, Shivam; DUTTA, Pramit; RASTOGI, Naveen; CHATURVEDI, Shashank. A control algorithm for co-operatively aerial survey by using multiple UAVs. *In: 2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*. [S. l.: s. n.], 2017. p. 280–285.

KUNZ CECHINEL, Alan *et al.* Mobile Robot + IoT: Project of Sustainable Technology for Sanitizing Broiler Poultry Litter. **Sensors**, v. 24, n. 10, 2024. ISSN 1424-8220.

KUNZ CECHINEL., Alan; RÖNING., Juha; TIKANMAKI., Antti; DEPIERI., Edson; DELLA MÉA PLENTZ., Patricia. Hanging Drone: An Approach to UAV Landing for Monitoring. *In: INSTICC. PROCEEDINGS of the 20th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO*. [S. l.]: SciTePress, 2023. p. 363–373. ISBN 978-989-758-670-5.

KWON, Byong; THANGAVELAUTHAM, Jekanthan. Autonomous Coverage Path Planning using Artificial Neural Tissue for Aerospace Applications. *In: 2020 IEEE Aerospace Conference*. [S. l.: s. n.], 2020. p. 1–10.

LAGHARI, Asif Ali; JUMANI, Awais Khan; LAGHARI, Rashid Ali; NAWAZ, Haque. Unmanned aerial vehicles: A review. **Cognitive Robotics**, v. 3, p. 8–22, 2023. ISSN 2667-2413.

LEONARD, J.J.; DURRANT-WHITE, H.F. Mobile robot localization by tracking geometric beacons. **IEEE Transactions on Robotics and Automation**, v. 7, n. 3, p. 376–382, 1991.

LI, Jiadong; LI, Xueqi; YU, Lijuan. Multi-UAV cooperative coverage path planning in plateau and mountain environment. *In: 2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. [S. l.: s. n.], 2018. p. 820–824.

LI, Lin; SHI, Dianxi; JIN, Songchang; YANG, Shaowu; ZHOU, Chenlei; LIAN, Yaoning; LIU, Hengzhu. Exact and Heuristic Multi-Robot Dubins Coverage Path Planning for Known Environments. **Sensors**, v. 23, n. 5, 2023.

- LI, Mickey; RICHARDS, Arthur; SOORIYABANDARA, Mahesh. Asynchronous Reliability-Aware Multi-UAV Coverage Path Planning. *In: 2021 IEEE International Conference on Robotics and Automation (ICRA)*. [S. l.: s. n.], 2021. p. 10023–10029.
- LI, Mickey; RICHARDS, Arthur; SOORIYABANDARA, Mahesh. Reliability-Aware Multi-UAV Coverage Path Planning Using a Genetic Algorithm. *In: PROCEEDINGS of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. Virtual Event, United Kingdom: International Foundation for Autonomous Agents and Multiagent Systems, 2021. (AAMAS '21), p. 1584–1586.
- LI, Mickey; RICHARDS, Arthur; SOORIYABANDARA, Mahesh. Reliability-Aware Multi-UAV Coverage Path Planning Using Integer Linear Programming. *In: UKRAS20 Conference: "Robots into the real world"*. [S. l.: s. n.], 2020. p. 15–17.
- LI, Ming; WAKABAYASHI, Kenji Imouand Katsuhiko; YOKOYAMA, Shinya. Review of research on agricultural vehicle autonomous guidance. **International Journal of Agricultural and Biological Engineering**, v. 2, n. 3, 2009.
- LI, Yan; CHEN, Hai; JOO ER, Meng; WANG, Xinmin. Coverage path planning for UAVs based on enhanced exact cellular decomposition method. **Mechatronics**, v. 21, n. 5, p. 876–885, 2011. Special Issue on Development of Autonomous Unmanned Aerial Vehicles. ISSN 0957-4158.
- LIN, Na; SHI, Yanjun; ZHANG, Tongliang; WANG, Xuping. An Effective Order-Aware Hybrid Genetic Algorithm for Capacitated Vehicle Routing Problems in Internet of Things. **IEEE Access**, v. 7, p. 86102–86114, 2019.
- LIU, Hong; MA, Jiayao; HUANG, Weibo. Sensor-based complete coverage path planning in dynamic environment for cleaning robot. **CAAI Transactions on Intelligence Technology**, v. 3, n. 1, p. 65–72, 2018.
- LIU, Xiao; LIU, Yuxin; ZHANG, Ning; WU, Wen; LIU, Anfeng. Optimizing Trajectory of Unmanned Aerial Vehicles for Efficient Data Acquisition: A Matrix Completion Approach. **IEEE Internet of Things Journal**, PP, p. 1–1, Jan. 2019.
- LLOYD, Stuart P. **Least Squares Quantization in PCM**. Murray Hill, NJ, Sept. 1957. Technical Report.
- LOWE, Ryan; WU, Yi; TAMAR, Aviv; HARB, Jean; ABBEEL, Pieter; MORDATCH, Igor. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, June 2017.
- LUNA, Marco Andrés; ALE ISAAC, Mohammad Sadeq; RAGAB, Ahmed Refaat; CAMPOY, Pascual; FLORES PEÑA, Pablo; MOLINA, Martin. Fast Multi-UAV Path Planning for Optimal Area Coverage in Aerial Sensing Applications. **Sensors**, v. 22, n. 6, 2022. ISSN 1424-8220.

- LUO, R.C.; KAY, M.G. A tutorial on multisensor integration and fusion. *In*: [PROCEEDINGS] IECON '90: 16th Annual Conference of IEEE Industrial Electronics Society. [S. l.: s. n.], 1990. 707–722 vol.1.
- MACQUEEN, J. B. Some Methods for Classification and Analysis of Multivariate Observations. *In*: LE CAM, Lucien M.; NEYMAN, Jerzy (eds.). **Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability**. Berkeley, CA: University of California Press, 1967. v. 1, p. 281–297.
- MARTINS, O.O.; ADEKUNLE, Adefemi; ADEJUYIGBE, S.B.; ADEYEMI, Oluwole; AROWOLO, Michael. Wheeled Mobile Robot Path Planning and Path Tracking Controller Algorithms: A Review. **Journal of Engineering Science and Technology Review**, v. 13, p. 152–164, Jan. 2020.
- MAZA, Ivan; OLLERO, Anibal. Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. *In*: ALAMI, Rachid; CHATILA, Raja; ASAMA, Hajime (eds.). **Distributed Autonomous Robotic Systems 6**. Tokyo: Springer Japan, 2007. p. 221–230.
- MILNOR, J. **Morse Theory**. [S. l.]: Princeton University Press, 1963.
- MITCHELL, H.B. Introduction. *In*: MULTI-SENSOR Data Fusion: An Introduction. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 3–13. ISBN 978-3-540-71559-7.
- MO, Weimin; LIN, Zhiyun. MDSTC: A Dynamic Approach to Multi-Robot Coverage Path Planning. *In*: p. 1100–1105.
- MODARES, Jalil; GHANEI, Farshad; MASTRONARDE, Nicholas; DANTU, Karthik. UB-ANC planner: Energy efficient coverage path planning with multiple drones. *In*: 2017 IEEE International Conference on Robotics and Automation (ICRA). [S. l.: s. n.], 2017. p. 6182–6189.
- MOHSAN, Syed Agha Hassnain; OTHMAN, Nawaf Qasem Hamood; LI, Yanlong; ALSHARIF, Mohammed H; KHAN, Muhammad Asghar. Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends. **Intelligent Service Robotics**, v. 16, n. 1, p. 109–137, Mar. 2023.
- MONTEMERLO, Michael; THRUN, Sebastian; KOLLER, Daphne; WEGBREIT, Ben. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. *In*:
- MUR-ARTAL, Raúl; MONTIEL, J. M. M.; TARDÓS, Juan D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. **IEEE Transactions on Robotics**, v. 31, n. 5, p. 1147–1163, 2015.

NAGLA, K.; UDDIN, Moinuddin; SINGH, Dilbag. Multisensor Data Fusion and Integration for Mobile Robots: A Review. **IAES International Journal of Robotics and Automation (IJRA)**, v. 3, Sept. 2014.

NAIR, Vishnu G.; DILEEP, M. V.; GURUPRASAD, K. R. Robust Online Multi-Robot Simultaneous Exploration and Coverage Path Planning. **IEEE Access**, v. 12, p. 72990–73003, 2024.

NEDJATI, Arman; IZBIRAK, Gokhan; VIZVARI, Bela; ARKAT, Jamal. Complete Coverage Path Planning for a Multi-UAV Response System in Post-Earthquake Assessment. **Robotics**, v. 5, n. 4, 2016. ISSN 2218-6581.

NEX, Francesco; REMONDINO, Fabio. UAV for 3D mapping applications: a review. **Applied Geomatics**, v. 6, n. 1, p. 1–15, Mar. 2014. ISSN 1866-928X.

OLIVEIRA, Luiz F. P.; MOREIRA, António P.; SILVA, Manuel F. Advances in Agriculture Robotics: A State-of-the-Art Review and Challenges Ahead. **Robotics**, v. 10, n. 2, 2021. ISSN 2218-6581.

OPEN SOURCE ROBOTICS FOUNDATION. **About Gazebo**. Accessed: 04/09/2025. 2025. Available from: <https://gazebo-sim.org/about>.

OPEN SOURCE ROBOTICS FOUNDATION. **Why ROS? It's the fastest way to build a robot!** Accessed: 04/09/2025. 2024. Available from: <https://www.ros.org/blog/why-ros/>.

OSBAND, Ian; BLUNDELL, Charles; PRITZEL, Alexander; ROY, Benjamin Van. Deep Exploration via Bootstrapped DQN. *In: PROCEEDINGS of the 30th International Conference on Neural Information Processing Systems*. Barcelona, Spain: Curran Associates Inc., 2016. (NIPS'16), p. 4033–4041.

PANIGRAHI, Prabin Kumar; BISOY, Sukant Kishoro. Localization strategies for autonomous mobile robots: A review. **Journal of King Saud University - Computer and Information Sciences**, 2021. ISSN 1319-1578.

PATHAN, Misbah; PATEL, Nivedita; YAGNIK, Hiteshri; SHAH, Manan. Artificial cognition for applications in smart agriculture: A comprehensive review. **Artificial Intelligence in Agriculture**, v. 4, p. 81–95, 2020. ISSN 2589-7217.

PATLE, B.K.; BABU L, Ganesh; PANDEY, Anish; PARHI, D.R.K.; JAGADEESH, A. A review: On path planning strategies for navigation of mobile robot. **Defence Technology**, v. 15, n. 4, p. 582–606, 2019. ISSN 2214-9147.

PERSSON, Linnea; WAHLBERG, Bo. Model Predictive Control for Autonomous Ship Landing in a Search and Rescue Scenario. *In: AIAA Scitech 2019 Forum*. [S. l.: s. n.], 2019.

R. KHOT, L.; TANG, L.; B. BLACKMORE, S.; NØRREMARK, M. Navigational Context Recognition for an Autonomous Robot in a Simulated Tree Plantation. **Transactions of the ASABE**, ASABE, St. Joseph, MI, v. 49, n. 5, p. 1579–1588, 2006. ISSN 2151-0032.

RADOGLOU-GRAMMATIKIS, Panagiotis; SARIGIANNIDIS, Panagiotis; LAGKAS, Thomas; MOSCHOLIOS, Ioannis. A compilation of UAV applications for precision agriculture. **Computer Networks**, v. 172, p. 107148, 2020. ISSN 1389-1286.

RADOGLOU-GRAMMATIKIS, Panagiotis; SARIGIANNIDIS, Panagiotis; LAGKAS, Thomas; MOSCHOLIOS, Ioannis. A compilation of UAV applications for precision agriculture. **Computer Networks**, v. 172, p. 107148, 2020. ISSN 1389-1286.

RAVANKAR, Abhijeet; RAVANKAR, Ankit A.; RAWANKAR, Arpit; HOSHINO, Yohei. Autonomous and Safe Navigation of Mobile Robots in Vineyard with Smooth Collision Avoidance. **Agriculture**, v. 11, n. 10, 2021. ISSN 2077-0472.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. *In*: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S. l.: s. n.], 2016. p. 779–788.

REKLEITIS, Ioannis; NEW, Ai Peng; RANKIN, Edward Samuel; CHOSSET, Howie. Efficient Boustrophedon Multi-Robot Coverage: an algorithmic approach. **Annals of Mathematics and Artificial Intelligence**, v. 52, n. 2, p. 109–142, Apr. 2008. ISSN 1573-7470.

REZENDE, Adriano M.C.; GONÇALVES, Vinicius M.; PIMENTA, Luciano C.A. Safe coordination of robots in cyclic paths. **ISA Transactions**, v. 109, p. 126–140, 2021. ISSN 0019-0578.

RODRIGUEZ-RAMOS, Alejandro; SAMPEDRO, Carlos; BAVLE, Hriday; PUENTE, Paloma de la; CAMPOY, Pascual. A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform. **Journal of Intelligent & Robotic Systems**, v. 93, n. 1, p. 351–366, Feb. 2019. ISSN 1573-0409.

ROOY, Nathan A. **p-area: The easiest way to compute the frontal/projected area of an STL geometry file**. [S. l.: s. n.], June 2021.

<https://github.com/nathanrooy/p-area>. Accessed: 16/08/2025.

ROSSMANN, Jürgen; KRAHWINKLER, Petra; BÜCKEN, Arno. Mapping and Navigation of Mobile Robots in Natural Environments. *In*: **Advances in Robotics Research: Theory, Implementation, Application**. Ed. by Torsten Kröger and Friedrich M. Wahl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 43–52.

ROSTEN, Edward; PORTER, Reid; DRUMMOND, Tom. Faster and Better: A Machine Learning Approach to Corner Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 1, p. 105–119, 2010.

RUANGWISSET, Annop. The application of unmanned aerial vehicle to precision agriculture: Verification experiments of the power consumption. *In: 2014 International Conference on Information Science, Electronics and Electrical Engineering. [S. l.: s. n.], 2014. v. 2, p. 968–971.*

RUBIO, Francisco; VALERO, Francisco; LLOPIS-ALBERT, Carlos. A review of mobile robots: Concepts, methods, theoretical framework, and applications. **International Journal of Advanced Robotic Systems**, v. 16, n. 2, p. 1729881419839596, 2019.

SÁNCHEZ-IBÁÑEZ, José Ricardo; PÉREZ-DEL-PULGAR, Carlos J.; GARCÍA-CEREZO, Alfonso. Path Planning for Autonomous Mobile Robots: A Review. **Sensors**, v. 21, n. 23, 2021. ISSN 1424-8220.

SANNA, Giovanni; GODIO, Simone; GUGLIERI, Giorgio. Neural Network Based Algorithm for Multi-UAV Coverage Path Planning. *In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS). [S. l.: s. n.], 2021. p. 1210–1217.*

SANTIN, Rafael; ASSIS, Luciana; VIVAS, Alessandro; PIMENTA, Luciano C. A. Matheuristics for Multi-UAV Routing and Recharge Station Location for Complete Area Coverage. **Sensors**, v. 21, n. 5, 2021. ISSN 1424-8220.

SAWHNEY, Rahul; KRISHNA, K. Madhava; SRINATHAN, Kannan; MOHAN, Mahesh. On Reduced Time Fault Tolerant Paths for Multiple UAVs Covering a Hostile Terrain. *In: PROCEEDINGS of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems, 2008. (AAMAS '08), p. 1171–1174.*

SCULLEY, David. Web-scale k-means clustering. *In: PROCEEDINGS of the 19th International Conference on World Wide Web (WWW 2010). Raleigh, North Carolina, USA: ACM, 2010. p. 1177–1178.*

SHAO, Xian-Xin; GONG, Yue-Jiao; ZHAN, Zhi-Hui; ZHANG, Jun. Bipartite Cooperative Coevolution for Energy-Aware Coverage Path Planning of UAVs. **IEEE Transactions on Artificial Intelligence**, v. 3, n. 1, p. 29–42, 2022.

SIEGWART, Roland; NOURBAKHSH, Illah R.; SCARAMUZZA, Davide. **Introduction to Autonomous Mobile Robots**. 2. ed. [S. l.]: The MIT Press, 2011. (Intelligent Robotics and Autonomous Agents series). ISBN 0262015358.

SKOROBOGATOV, Georgy; BARRADO, Cristina; SALAMÍ, Esther; PASTOR, Enric. Flight planning in multi-unmanned aerial vehicle systems: Nonconvex polygon area

decomposition and trajectory assignment. **International Journal of Advanced Robotic Systems**, v. 18, n. 1, p. 1729881421989551, 2021.

SORBELLI, Francesco Betti; CORÒ, Federico; DAS, Sajal K.; PINOTTI, Cristina M. Energy-Constrained Delivery of Goods With Drones Under Varying Wind Conditions. **IEEE Transactions on Intelligent Transportation Systems**, v. 22, n. 9, p. 6048–6060, 2021.

SRINIVASAN, Nandagopal; PRABHU, Prithviraj; SMRUTHI, S Sanjana; SIVARAMAN, N Vivek; GLADWIN, S Joseph; RAJAVEL, R; NATARAJAN, Abeshek Ram. Design of an autonomous seed planting robot. *In*: 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC). [S. l.: s. n.], 2016. p. 1–4.

STACHNISS, Cyrill; LEONARD, John J.; THRUN, Sebastian. Simultaneous Localization and Mapping. *In*: **Springer Handbook of Robotics**. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, 2016. p. 1153–1176.

TAN, Chee Sheng; MOHD-MOKHTAR, Rosmiwati; ARSHAD, Mohd Rizal. A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms. **IEEE Access**, v. 9, p. 119310–119342, 2021.

TANG, Jingtao; MA, Hang. Large-Scale Multi-Robot Coverage Path Planning via Local Search*. *In*: 16. v. 38, p. 17567–17574.

TANG, Jingtao; MA, Hang. Mixed Integer Programming for Time-Optimal Multi-Robot Coverage Path Planning With Efficient Heuristics. **IEEE Robotics and Automation Letters**, v. 8, n. 10, p. 6491–6498, 2023.

TANG, Jingtao; SUN, Chun; ZHANG, Xinyu. MSTC*:Multi-robot Coverage Path Planning under Physical Constrains. *In*: 2021 IEEE International Conference on Robotics and Automation (ICRA). [S. l.: s. n.], 2021. p. 2518–2524.

THEILE, Mirco; BAYERLEIN, Harald; NAI, Richard; GESBERT, David; CACCAMO, Marco. UAV Coverage Path Planning under Varying Power Constraints using Deep Reinforcement Learning. *In*: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S. l.: s. n.], 2020. p. 1444–1449.

THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. [S. l.]: The MIT Press, 2005. ISBN 0262201623.

TITTERTON, David; WESTON, John. **Strapdown Inertial Navigation Technology**. 2. ed. Stevenage, England: Institution of Engineering and Technology, Oct. 2004. (Electromagnetics and Radar).

TSOUROS, Dimosthenis C.; BIBI, Stamatia; SARIGIANNIDIS, Panagiotis G. A Review on UAV-Based Applications for Precision Agriculture. **Information**, v. 10, n. 11, 2019. ISSN 2078-2489.

VALENTE, João; BARRIENTOS, Antonio; CERRO, Jaime del; ROSSI, Claudio; COLORADO, Julian; SANZ, David; GARZÓN, Mario. Multi-robot Visual Coverage Path Planning: Geometrical Metamorphosis of the Workspace through Raster Graphics Based Approaches. In: MURGANTE, Beniamino; GERVASI, Osvaldo; IGLESIAS, Andrés; TANIAR, David; APDUHAN, Bernady O. (eds.). **Computational Science and Its Applications - ICCSA 2011**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 58–73.

WAN, Fan; YANG, He; MENG, Qingxiang. A three-dimensional coverage path planning method for multi-UAV collaboration. In: 2023-July.

WANG, Huan; SHI, Dianxi; WU, Yunlong; LI, Lin; LI, Ning; XU, Jiachi. A Balanced Shadow-Following Coverage Path Planning Approach under Energy Constraints. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). [S. l.: s. n.], 2021. p. 1599–1606.

WANG, Mengyang; ZHANG, Dong; LI, Chaoyue; ZHANG, Zhaohua. Multiple fixed-wing UAVs collaborative coverage 3D path planning method for complex areas. **Defence Technology**, v. 47, p. 197–215, 2025.

WANG, Yinghui; KIRUBARAJAN, Thiagalingam; THARMARASA, Ratnasingham; JASSEMI-ZARGANI, Rahim; KASHYAP, Nathan. Multiperiod Coverage Path Planning and Scheduling for Airborne Surveillance. **IEEE Transactions on Aerospace and Electronic Systems**, v. 54, n. 5, p. 2257–2273, 2018.

WANG, Zikai; LYU, Xiaoxu; ZHANG, Jiekai; WANG, Pengyu; ZHONG, Yuxing; SHI, Ling. MAC-Planner: A Novel Task Allocation and Path Planning Framework for Multi-Robot Online Coverage Processes. **IEEE Robotics and Automation Letters**, v. 10, n. 5, p. 4404–4411, 2025.

WANG, Zikai; ZHAO, Xiaoqi; ZHANG, Jiekai; YANG, Nachuan; WANG, Pengyu; TANG, Jiawei; ZHANG, Jiuzhou; SHI, Ling. APF-CPP: An Artificial Potential Field Based Multi-Robot Online Coverage Path Planning Approach. **IEEE Robotics and Automation Letters**, v. 9, n. 11, p. 9199–9206, 2024.

WU, Wenxin; GUO, Liang; GAO, Hongli; YOU, Zhichao; LIU, Yuekai; CHEN, Zhiqiang. YOLO-SLAM: A semantic SLAM system towards dynamic environment with geometric constraint. **Neural Computing and Applications**, v. 34, n. 8, p. 6011–6026, Apr. 2022.

XIAO, Xuesu; LIU, Bo; WARNELL, Garrett; STONE, Peter. Motion planning and control for mobile robot navigation using machine learning: a survey. **Autonomous Robots**, v. 46, n. 5, p. 569–597, July 2022. ISSN 1573-7527.

XIE, Junfei; GARCIA CARRILLO, Luis Rodolfo; JIN, Lei. Path Planning for UAV to Cover Multiple Separated Convex Polygonal Regions. **IEEE Access**, v. 8, p. 51770–51785, 2020.

XIONG, Ya; GE, Yuanyue; GRIMSTAD, Lars; FROM, Pål J. An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. **Journal of Field Robotics**, v. 37, n. 2, p. 202–224, 2020.

XMBOOTS. **NAURU 500C ISR**. Accessed: 16/09/2025. 2025. Available from: <https://xmobots.com.br/produto/nauru-500c-isr/>.

XU, Anqi; VIRIYASUTHEE, Chatavut; REKLEITIS, Ioannis. Efficient complete coverage of a known arbitrary environment with applications to aerial operations. **Autonomous Robots**, v. 36, n. 4, p. 365–381, Apr. 2014. ISSN 1573-7527.

XUAN-MUNG, Nguyen; HONG, Sung Kyung; NGUYEN, Ngoc Phi; HA, Le Nhu Ngoc Thanh; LE, Tien-Loc. Autonomous Quadcopter Precision Landing Onto a Heaving Platform: New Method and Experiment. **IEEE Access**, v. 8, p. 167192–167202, 2020.

YANG, S.X.; LUO, C. A neural network approach to complete coverage path planning. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 34, n. 1, p. 718–724, 2004.

YAO, Zheng; WANG, Huixia; HU, Ruiguang. Improved Genetic Algorithm and Coding Method for Cooperative Search of UAV Group. *In: 2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. [S. l.: s. n.], 2021. p. 141–144.

YILMAZ, Abdurrahman; TEMELTAS, Hakan. Self-adaptive Monte Carlo method for indoor localization of smart AGVs using LIDAR data. **Robotics and Autonomous Systems**, v. 122, p. 103285, 2019. ISSN 0921-8890.

YU, Xiaoxiao; JIN, Songchang; SHI, Dianxi; LI, Lin; KANG, Ying; ZOU, Junbo. Balanced Multi-Region Coverage Path Planning for Unmanned Aerial Vehicles. *In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S. l.: s. n.], 2020. p. 3499–3506.

ZELINSKY, Alexander; JARVIS, Ray A.; BYRNE, Julian; YUTA, Shin'ichi. Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot. *In: INTERNATIONAL Conference on Advanced Robotics*. [S. l.: s. n.], 1993. p. 533–538.

ZENG, Hui; TONG, Lei; XIA, Xuewen. Multi-UAV Cooperative Coverage Search for Various Regions Based on Differential Evolution Algorithm. **Biomimetics**, v. 9, n. 7, 2024.

ZHANG, Boquan; TIAN, Jing; LIN, Xiang; CUI, Yanru; ZHU, Yifan; ZHU, Zhi. Deep Reinforcement Learning-based Collaborative Multi-UAV Coverage Path Planning. *In: 1. v. 2833.*

ZHANG, Zhe; WU, Jian; HE, Cheng. Search Method of disaster inspection coordinated by Multi-UAV. *In: 2019 Chinese Control Conference (CCC). [S. l.: s. n.], 2019. p. 2144–2148.*

ZHENG, Xiaoming; KOENIG, Sven; KEMPE, David; JAIN, Sonal. Multirobot Forest Coverage for Weighted and Unweighted Terrain. **IEEE Transactions on Robotics**, v. 26, n. 6, p. 1018–1031, 2010.

ZHOU, Xiaolin; LIU, Xiaojie; WANG, Xingwei; WU, Shiguang; SUN, Mingyang. Multi-Robot Coverage Path Planning based on Deep Reinforcement Learning. *In: 2021 IEEE 24th International Conference on Computational Science and Engineering (CSE). [S. l.: s. n.], 2021. p. 35–42.*

ZHOU, Xing; WANG, Huaimin; DING, Bo; HU, Tianjiang; SHANG, Suning. Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. **Expert Systems with Applications**, v. 116, p. 10–20, 2019. ISSN 0957-4174.

APPENDIX A – SYSTEMATIC REVIEW PROTOCOL

The Systematic Review (SR) aimed to get the state-of-art of the Multi-UAV coverage path planning problem. Therefore, the main question to be answered was:

What are the techniques, algorithms, and approaches to solve the Multi-UAV coverage path planning problem?

The secondary questions were:

- How the environment is modeled?
- What parameters are usually optimized?
- What variables are considered for the optimization and what are the restrictions?
- How the path is represented?
- What is the role of the wind in the problem?
- What is the influence of the terrain irregularities in the path?
- How the UAV dynamics/kinematics are considered?
- Does the problem consider acceleration and deceleration?
- Are the solutions found online, offline or hybrid?
- Is there a replanning of failed paths?
- Are there decentralized approaches? (Algorithm or Multi-agent proposal)
- Are there approaches considering battery exchange? Or recharge station?

The SR considered journal articles, conference articles, and book chapters to answer the research questions. It gathered works written only in English. They were found using the following web engines: Scopus, Web of Science, IEEE Explore, ACM Library, and Science Direct. The following string was adapted for each web engine to search works by title, abstract, and keywords containing:

("mult* uav" OR "mult* aerial" OR "mult* aircraft" OR "mult* drone" OR "mult* robot")
AND ("cover* path" OR "cover* search")

The * symbol represents any suffix in the word, for instance, "mult*", multi, multiple, multiples, etc. This feature had to be adapted for each web engine. In Science Direct, it was not possible to use it, but according to them, their engine automatically detects suffixes and similar words.

After gathering the works in each web engine, they were selected considering the following criteria:

- **Inclusion:**

- Explain a Multi-UAV Coverage Path Planning (CPP) method;
- Explain a Multi-robot CPP method;

- **Exclusion:**

- No mention of using UAVs in the text;
- No access;
- Explain a Multi-robot CPP for cleaning;
- Not related to Multi-UAV CPP at all;
- CPP for single UAV;
- Trajectory planning or control theory;

The software StArt (http://lapes.dc.ufscar.br/tools/start_tool) was used to organize the protocol, selection, and data extraction from the works. The file containing the SR in StArt format will be shared after finishing the survey paper.