



ARCHITECTURE FOR HYBRID MULTI-ROBOT FOR COVERAGE PATH PLANNING IN PARTIALLY KNOWN DYNAMIC ENVIRONMENTS

Gabriel Gelard Reis de Castro

Thesis submitted to the Postgraduate Program in Instrumentation and Applied Optics of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of doctor.

Advisor: Milena Faria Pinto
Co-advisor: José Lima

Rio de Janeiro,

January, 2026

ARCHITECTURE FOR HYBRID MULTI-ROBOT FOR COVERAGE PATH PLANNING IN PARTIALLY KNOWN DYNAMIC ENVIRONMENTS

Thesis submitted to the Postgraduate Program in Instrumentation and Applied Optics of the Federal Center for Technological Education of Rio de Janeiro, CEFET/RJ, as partial fulfillment of the requirements for the degree of doctor.

Gabriel Gelard Reis de Castro

Examining jury:

President, Milena Faria Pinto, D.Sc. (CEFET/RJ) (Advisor)

José Lima, D.Sc. (IPB) (Co-advisor)

Amaro Azevedo Lima, Ph.D. (CEFET-RJ)

Ana Isabel Pereira, D.Sc. (IPB)

Andre Luis Marques Marcato, D.Sc. (UFJF)

Mario Antônio Ribeiro Dantas, D.Sc. (UFJF)

Rio de Janeiro,

January, 2026

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

C355 Castro, Gabriel Gelard Reis de
Architecture for hybrid multi-robot for coverage path planning in partially known dynamic environments / Gabriel Gelard Reis de Castro. — 2026.
114f. : il. color. , enc.

Tese (Doutorado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2026.
Bibliografia : f. 105-114
Orientadora: Milena Faria Pinto
Coorientador: José Lima

1. Robôs. 2. Drone. 3. Mapeamento ambiental. 4. Planejamento – Trajetória. 5. Inteligência artificial. I. Pinto, Milena Faria. (Orient.). II. Lima, José. (Coorient.). III. Título.

CDD 629.892

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

DEDICATION

To my mother, Marize, who has always supported me throughout all these years.

To my father, Elcio, who is my example in life.

To all my family and friends for their continuous support during my years of study.

A special thanks to my girlfriend, who saw something in me long ago and encouraged me to pursue a degree, without her I would never have considered an academic path.

And also thanks to my advisor, Milena, who showed me the passion for research and teaching at a moment when I felt lost along this journey.

ACKNOWLEDGMENTS

This work was developed with the support of the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Finance Code 001.

The author also acknowledges the support of CEFET/RJ, CNPq, and FAPERJ.

RESUMO

Architecture for Heterogeneous Multi-Robots for Coverage Path Planning in Partially Known Dynamic Environments

Esta pesquisa de doutorado introduz uma arquitetura de cooperação para equipes heterogêneas de robôs, formadas por Veículos Aéreos Não Tripulados (UAVs) e Veículos Terrestres Não Tripulados (UGVs), atuando em ambientes parcialmente conhecidos e sujeitos a mudanças dinâmicas. Dois cenários foram utilizados para validação: uma subestação elétrica e uma área de agricultura de precisão. Na solução proposta, os UAVs são responsáveis pelas tarefas de inspeção e mapeamento aéreo, enquanto os UGVs executam inspeções em solo e podem ainda realizar o transporte dos UAVs. Para o planejamento inicial das rotas nas áreas de interesse, emprega-se um algoritmo híbrido de Coverage Path Planning (CPP) offline, baseado na combinação dos métodos Wavefront e Bidirectional $A^{\star}$, utilizando mapas de ocupação previamente gerados. Embora esse planejamento inicial seja importante, o diferencial do sistema está no planejamento de rota online local, capaz de adaptar-se continuamente às mudanças do ambiente durante a missão. Essa camada integra técnicas de Redes Neurais e Aprendizado por Reforço Profundo (Deep Reinforcement Learning – DRL) para viabilizar o ajuste dinâmico das rotas em tempo real. No cerne da arquitetura, foi implementada uma abordagem de planejamento tridimensional que combina árvores de exploração rápida (Rapidly-Exploring Random Trees – RRT) com DRL, permitindo que os robôs reajam de forma autônoma a obstáculos inesperados e alterações nas condições do ambiente. A solução também contempla rotinas de contingência voltadas à detecção e resposta a falhas críticas, como esgotamento de bateria, perda de comunicação ou problemas de localização. Os testes realizados em simulações com o Robot Operating System (ROS) e o simulador Gazebo demonstraram a eficácia da abordagem híbrida frente aos desafios da cooperação entre múltiplos robôs em ambientes dinâmicos e com informações limitadas.

Palavras-chave: Robôs heterogêneos cooperativos; Robôs móveis; Sistemas multi-robôs; Planejamento de trajetória; Ambientes dinâmicos.

ABSTRACT

Architecture for Heterogeneous Multi-Robots for Coverage Path Planning in Partially Known Dynamic Environments

This doctoral research presents a cooperation strategy for heterogeneous teams of robots composed of multiple Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) operating in environments with dynamic obstacles. Two scenarios were evaluated: an electrical substation and a precision agriculture environment. To address challenges such as partially known terrains and moving obstacles, the architecture assigns specific roles to UAVs and UGVs. UAVs are responsible for aerial inspection and mapping, while UGVs handle ground-level inspection and can also transport UAVs. An offline Coverage Path Planning (CPP) algorithm, developed by integrating Wavefront and Bidirectional A* methods, is employed to generate initial obstacle-free paths for both UAVs and UGVs, using a precomputed occupancy grid map as a reference. While this step provides a baseline coverage strategy, the system's main innovation lies in its online decision-making layer, which is designed to adapt in real-time to dynamic environmental changes. This layer integrates advanced techniques such as Neural Networks and Deep Reinforcement Learning (DRL) for reactive path planning. At its core, the system implements an adaptive 3D trajectory planning strategy that combines Rapidly-Exploring Random Trees (RRT) with DRL, enabling the robots to respond to unforeseen obstacles and changes in the environment dynamically. The architecture also includes contingency subroutines for UAVs and UGVs to handle critical issues such as battery depletion, communication loss, and localization failures. Extensive simulations using the Robot Operating System (ROS) and the Gazebo simulator were conducted to validate the proposed hybrid approach, demonstrating its effectiveness in addressing the challenges of autonomous multi-robot cooperation in partially known and dynamic scenarios.

Keywords: Heterogeneous Robot Cooperation; Robots; Multiple Robots; Path Planning; Dynamic Environment.

LIST OF FIGURES

Figure 1 – Proposed methodology applied in an electrical substation scenario (Castro, G. G. R. de et al., 2024). 32

Figure 2 – Proposed methodology applied on an agriculture scenario. 33

Figure 3 – Overview of the proposed cooperation strategy (Castro, G. G. R. de et al., 2024). 34

Figure 4 – High-level representation of the cooperative architecture's pipeline, illustrating the data flow from environmental perception to physical execution through the three core planning layers. 35

Figure 5 – Wavefront propagation in a simulated environment.(Castro, G. G. R. de et al., 2025) (a) The simulated environment, where the red square represents the goal; (b) The simulated environment displaying the wavefront weights of each cell, propagated outward from the goal grid. 39

Figure 6 – Coverage path combining Wavefront and Bidirectional A* algorithms (Castro, G. G. R. de et al., 2025). 42

Figure 7 – RRT path generation implemented over an occupancy 2D grid. 45

Figure 8 – Overall idea of the DRL–RRT methodology proposed in (Castro, G. G. d. et al., 2023). (a) The agent starts trajectory planning using RRT to explore multiple path options. (b) Once RRT determines feasible routes, the DQN refines the motion and adapts it in real time as the environment changes. 46

Figure 9 – Gazebo training environment for the DQN agent (Castro, G. G. d. et al., 2023). (a) Simplified world for accelerating DQN training. (b) Pickup model used as a dynamic obstacle. (c) Olive-tree models used as static obstacles. 48

Figure 10 – UGV mission initiation and verification (Castro, G. G. R. de et al., 2024). 52

Figure 11 – Routine of UAV 1 (Castro, G. G. R. de et al., 2024). 53
Figure 12 – Routine of UAV 2 (Castro, G. G. R. de et al., 2024). 54
Figure 13 – UAV quadrotor structure (Castro, G. G. d. et al., 2023). 56
Figure 14 – UGV's mecanum wheel model. 58
Figure 15 – Four-mecanum wheeled UGV state description. 58
Figure 16 – $A^\star$ pathfinding algorithm evaluated in a 10×10 occupancy grid. (a) Environment representation, highlighting the free space $\mathcal{A}_{\text{free}}$ (white) and obstacles $\mathcal{A}_{\text{obs}}$ (light blue), with start and goal poses marked in green and red. (b) The resulting optimal trajectory $\mathcal{P}_{A^\star}$ (dashed line) successfully avoiding local minima. 61
Figure 17 – Bidirectional $A^\star$ pathfinding algorithm implemented in a 2D environment. (a) Grid map segmentation highlighting free space $\mathcal{A}_{\text{free}}$ (white) and obstacles $\mathcal{A}_{\text{obs}}$ (light blue). (b) Optimal bridging path generated by the meeting of the forward frontier Q_{start} (red dashed line) and the backward frontier Q_{goal} (green dashed line). 64
Figure 18 – Comparison between (a) standard RRT and (b) RRT optimized using a Genetic Algorithm. 66
Figure 19 – Dijkstra-based geodesic coverage implemented on a 2D occupancy grid $\mathcal{G}$, where the marker "S" denotes the source reference cell p_{ref}. (a) The initial geodesic distance field $D(p)$ computed over the navigable free space $\mathcal{A}_{\text{free}}$. (b) The resulting coverage trajectory $\mathcal{P}$ generated by combining the distance field gradient with the greedy policy. 69
Figure 20 – Flood Fill expansion implemented over a 2D occupancy grid $\mathcal{G}$, demonstrating the identification of the reachable free space $\mathcal{A}_{\text{free}}$ from a given seed cell p_{seed}. 70
Figure 21 – Standard Wavefront algorithm execution over an occupancy grid $\mathcal{G}$. (a) Basic distance transform wave propagating from the goal p_{goal} (G) towards the start p_{start} (S) through the free space $\mathcal{A}_{\text{free}}$. (b) The resulting trajectory $\mathcal{P}_{\text{Wavefront}}$, highlighting the high curvature and numerous sharp turns generated by the unmodified cost function. 74

Figure 22 – Wavefront algorithm modified with the path transform cost function (Castro, G. G. R. de et al., 2024). (a) Path transform wave propagating from p_{goal}, incorporating obstacle penalties. (b) The resulting smoothed trajectory $\mathcal{P}_{\text{Wavefront}}$. 75

Figure 23 – IRIS UAV quadrotor model utilized in the Gazebo software simulation. 76

Figure 24 – Olive-growing simulation environment. The first test scenario featuring an unstructured layout with natural, sparsely distributed obstacles. 77

Figure 25 – Electrical Substation simulation environment. The second test scenario featuring a highly structured industrial topology with strict geometric constraints. 77

Figure 26 – Software framework overview. Gazebo simulates the physical environment and interacts with the UAV's FCU PX4. The DQN algorithm controls the agent in a closed-loop via ROS and MAVROS bridges (Castro, G. G. d. et al., 2023). 78

Figure 27 – Communication hierarchy of the robotic systems, detailing the network bridges between the SITL autopilots, the central ROS Master, and the UGV companion computer (Castro, G. G. de et al., 2024). 79

Figure 28 – ROS node structure isolating the Deep Reinforcement Learning closed-loop data flow (Castro, G. G. d. et al., 2023). 80

Figure 29 – Active ROS nodes and topics detailing the multi-UAV coordination (UAV1 and UAV2) and shared mission parameters. 81

Figure 30 – Active ROS nodes and topics detailing the UGV internal structure, highlighting the mapping and serial control bridges. 82

Figure 31 – Mean reward returns for the evaluated DQN models across 2000 training events (Castro, G. G. d. et al., 2023). 85

Figure 32 – Milestone comparison between the three best models. The green line marks 1/8th of training, the yellow line indicates the end of the strict exploration phase, and the red line shows exploration variance during the exploitation phase (Castro, G. G. d. et al., 2023). 86

Figure 33 – Top-down perspective of the agricultural Gazebo environment. The image details the spatial distribution of the insect traps (A, B, C, D) and the dotted piecewise linear paths generated offline by the global RRT algorithm. 87

Figure 34 – Boxplot analysis of the angular tracking deviation. The y-axis represents the error between the global RRT path angle (θ_{RRT}) and the actual UAV heading (ψ_{UAV}), segmented by mission legs (Paths AB, BC, and CD). 88

Figure 35 – A 2D occupancy grid representation of the substation environment generated by the UGV, highlighting the operational boundaries and structural footprints (Castro, G. G. de et al., 2024). 89

Figure 36 – A volumetric 3D representation of the substation environment generated via the OctoMap framework, detailing the vertical constraints and elevated structures (Castro, G. G. de et al., 2024). 90

Figure 37 – Wavefront tuning benchmark. The dashed lines represent the frequency of route overlaps, while the red lines quantify the required sharp maneuvers across different penalty factors (Castro, G. G. de et al., 2024). 91

Figure 38 – Systematic map coverage generated by the tuned Wavefront CPP. (a) UGV continuous path. (b) Multi-UAV parallel paths showcasing spatial partitioning (Castro, G. G. de et al., 2024). 92

Figure 39 – Mean performance analysis benchmarking Wavefront+BA*: (a) Overlaps, (b) Direction changes, (c) Time consumption (Castro, G. G. R. de et al., 2025). 93

Figure 40 – 3D trajectory plot of the battery scarcity routine. The graph captures the exact spatial handover where UAV 1 (green) descends for a hot swap, allowing UAV 2 (blue) to resume the coverage mission (Castro, G. G. de et al., 2024). 95

Figure 41 – Trajectory plot of the unmapped object avoidance routine. The black trajectory segments represent the DQN actively overriding the nominal path to detour around unmapped obstacles (red markers) (Castro, G. G. de et al., 2024). 96

Figure 42 – Simulation of a total communication loss. The green trajectory displays the flying UAV instantly aborting its mission and executing a direct Return to Home (RTH) descent arc (Castro, G. G. de et al., 2024). 97

LIST OF TABLES

Table 1 – Evolution of Cooperative Robotic Architectures. 21

Table 2 – Coordination and MRTA Approaches in Cooperative Robotics 22

Table 3 – Comparison of Coverage Path Planning (CPP) Algorithms. 24

Table 4 – Advantages and Drawbacks of Contemporary Path Planning Algorithms. 28

Table 5 – Recent Hybrid UAV–UGV Frameworks (2020–2025). 29

Table 6 – DQN's models and hyperparameters set during the training phase. 83

Table 7 – Performance and convergence comparison of Models 1, 2, and 3. 85

Table 8 – Percentage improvement of the proposed Wavefront+BA* method against baseline algorithms (Castro, G. G. R. de et al., 2025). 94

Table 9 – Compilation of performance metrics and success rates during simulated contingency scenarios. 98

LIST OF ALGORITHMS

Algorithm 1 – Wavefront and Bidirectional A* coverage path planning. 41

Algorithm 2 – Rapidly-Exploring Random Tree (RRT) applied to local waypoints. 44

Algorithm 3 – DQN training and experience collection process. 49

Algorithm 4 – Coordination and execution pipeline for the hybrid UAV-UGV framework. 51

Algorithm 5 – $A^\star$ algorithm applied to the discrete free space. 62

Algorithm 6 – Bidirectional $A^\star$ Search Algorithm 65

Algorithm 7 – Dijkstra's Algorithm (Geodesic Distance Field Computation) 68

Algorithm 8 – Flood Fill Region Expansion Algorithm 71

Algorithm 9 – Coverage Path Planning Wavefront Algorithm 73

LIST OF ABBREVIATIONS

DQN	Deep Q-Network
FCU	Flight Controller Unit
GA	Genetic Algorithm
MARL	Multi-agent Reinforcement Learning
MAVLINK	Micro Air Vehicle Link
POI	Points Of Interest
QOS	Quality-of-Service
RRT	Rapidly-Exploring Random Tree
RTAB-MAP	Real-Time Appearance-Based Mapping
RTH	Return To Home
SITL	Software-in-the-loop
SLAM	Simultaneous Localization And Mapping
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicle

CONTENTS

Introduction 14

1 Introduction 14
1.1 Context and Motivation 14
1.2 Cooperative Heterogeneous Multi-Robot Systems 14
1.3 Motion and Path Planning in Dynamic Environments 16
1.4 Advances and Challenges in Autonomous Planning 17
1.5 Main Contributions and Motivations 18
1.6 Thesis Organization 19

2 Related Works and Background 20
2.1 From Cooperation to Hybrid Architectures 20
2.2 The Challenge of Coordination and Task Allocation 21
2.3 From Task Allocation to Cooperative Coverage. 22
2.4 Navigating Dynamic Environments: Planning Beyond Geometry 25
2.5 Insights from Recent UAV–UGV Collaboration Reviews 28

3 Proposed Framework 30

Part I 32
3.1 Mission Overview and System Assumptions 32
3.1.1 Multi-Layer Hybrid Architecture 35
3.2 Global Coverage Planning 37
3.2.1 Wavefront Propagation for Global Gradient Construction 38
3.2.2 Hybridization with Bidirectional A* 38
3.2.3 Role in the Hybrid Architecture 42
3.3 Local Dynamic Planning Layer 43

3.3.1 Rapidly-Exploring Random Tree Algorithm 43
3.3.2 Deep Reinforcement Learning and Deep Q-Network 44
3.3.3 Training Environment Setup 47
3.4 Coordination and Task Reallocation Layer 50
3.5 Ground Inspection 51
3.5.1 Aerial Inspection 53

Part II 56
3.6 UAV quadrotor model 56
3.7 UGV model 57
3.8 Problem Description 59
3.9 Path Planning Algorithms 60
3.9.1 A* 60
3.9.2 Bidirectional $A^\star$ 62
3.9.3 Genetic Algorithm 65
3.10 Coverage Path Planning 67
3.10.1 Dijkstra-Based Geodesic Coverage 67
3.10.2 Flood Fill 69
3.10.3 Wavefront 71

4 Results and Discussions 76
4.1 Hardware and Environment Setup 76
4.2 Integration 79
4.3 Simulation Results 83
4.3.1 Agriculture Scenario: Local Obstacle Avoidance 83
4.3.2 Electrical Substation: 3D Mapping and Representation 88
4.3.3 Coverage Path Planning (CPP) Evaluation 90
4.3.4 System Resilience and Contingency Routines 94
4.4 Discussions 98

5 Conclusions and Future Works 99

6 References 104

1- Introduction

1.1- Context and Motivation

In recent years, the integration of autonomous systems has revolutionized several industrial and scientific domains, ranging from logistics and infrastructure inspection to precision agriculture and environmental monitoring. Among these systems, Unmanned aerial vehicles (UAV)s and Unmanned Ground Vehicle (UGV)s have emerged as two complementary robotic platforms with distinct yet synergistic capabilities (Shi et al., 2020; Shakhatreh et al., 2019; Luque Vega et al., 2020).

UAVs provide fast and flexible coverage of large areas, offering high-resolution perception from elevated viewpoints (Lyu2022; Stek, 2016; Biundini et al., 2022). Their agility enables operations in complex or hazardous environments, such as industrial plants or disaster zones. However, UAVs has inherent limitations in terms of payload capacity, endurance, and continuous operation. On the other hand, UGVs offers higher energy autonomy, stronger payload capacity, and the ability to transport sensing or actuation modules over extended periods (Narváez et al., 2020).

The combination of these two robotic agents enables hybrid systems that use the advantages of both aerial and ground mobility. Through cooperation, UAVs can extend their operational range, while UGVs benefit from the aerial perspective and situational awareness provided by UAVs. This synergy motivates the development of cooperative architectures that enhance autonomy, coverage, and decision-making under dynamic and uncertain environments (Berger; Scherer; Lozano, 2023; Qiu; Lin; Wang, 2024).

1.2- Cooperative Heterogeneous Multi-Robot Systems

Cooperative multi-robot systems have become a central topic in modern robotics, as collaboration enables task distribution, parallelization of operations, and fault toler-

ance (Madridano et al., 2021). Within this field, heterogeneous teams, composed of UAVs and UGVs, represent a particularly promising approach. By combining different locomotion modalities and sensing capabilities, such teams can achieve levels of autonomy and efficiency unattainable by homogeneous fleets (Kim; Kim; Park, 2022).

Typical cooperative missions include:

- Aerial mapping and ground verification: UAVs generate aerial maps while UGVs perform detailed ground-level inspections and interventions (Almadhoun et al., 2019);
- Energy-aware logistics: UGVs act as mobile charging platforms or transport bases for UAVs during long-term missions (Narváez et al., 2020);
- Cooperative coverage: UAVs provide global situational awareness, while UGVs ensure local accuracy and physical interaction with the environment.

Achieving such coordinated behavior requires more than task division. It demands a unifying control architecture that integrates communication and decision-making across heterogeneous agents. In this context, motion planning plays a fundamental yet integrated role: it is not treated as an isolated problem, but as a central mechanism that governs how cooperation unfolds in real-time. Through shared situational awareness and adaptive planning, UAVs and UGVs can continuously synchronize their actions, ensuring safety, efficiency, and robustness in dynamically changing environments.

A fundamental aspect of cooperation in heterogeneous teams lies in the problem of task allocation, determining which agent should execute a given task, when, and under what constraints. This challenge, known as Multi-Robot Task Allocation (MRTA), directly affects mission efficiency, coverage overlap, and overall system robustness. In the proposed framework, UAVs and UGVs dynamically share responsibilities based on their sensory capabilities, energy availability, and environmental updates. The theoretical foundation for this concept follows the taxonomy proposed by Korsah et al. (Korsah; Stentz; Dias, 2013), which classifies MRTA problems according to task type, agent heterogeneity, and communication topology.

1.3- Motion and Path Planning in Dynamic Environments

Path planning constitutes the core of autonomous navigation, defining the sequence of feasible and optimal movements required for a robot to reach its objective while avoiding obstacles (Ali et al., 2023). In dynamic and partially known environments, robots must continuously adapt to changes such as moving obstacles, incomplete maps, and sensor uncertainties. This challenge is particularly critical for UAV–UGV teams, which must synchronize aerial and ground trajectories while maintaining global mission coherence.

Traditionally, motion planning is divided into two complementary levels. The global planner operates over a known or previously mapped environment, generating an optimal route that minimizes a given cost function, such as distance, energy consumption, or exposure to risk. The local planner, in contrast, focuses on short-term adaptation, modifying trajectories in real time in response to unforeseen obstacles, updated sensor data, or communication delays. The cooperation between these two layers ensures that the system maintains both long-term optimality and short-term reactivity, which is essential for autonomous operation in unstructured or time-varying environments (Zhang; Liu; Wang, 2022).

Among the various forms of navigation, Coverage Path Planning (CPP) plays a particularly relevant role in heterogeneous missions. Unlike conventional point-to-point planning, CPP aims to ensure that the entire operational area is efficiently explored or monitored by the robotic team (Galceran; Carreras, 2013). In UAV–UGV cooperation, this leads to minimizing overlap, maximizing coverage completeness, and adapting scanning trajectories to environmental dynamics and energy limitations. Efficient coverage is fundamental for applications such as precision agriculture, search and rescue, and infrastructure inspection, where both global and local observations are required.

In the proposed framework, CPP functions as the high-level coordination layer that governs the division of the workspace among agents. UAVs perform global exploration and map generation, while UGVs handle detailed ground-level inspection and interaction. This hierarchical cooperation allows the team to dynamically balance its workload and maintain operational efficiency even under conditions of partial perception or environmental change.

Recent advances in motion planning combine traditional deterministic algorithms with learning-based models to enhance adaptability and generalization (Gao; Sun; Li, 2023;

Sun; Li; Zhang, 2024). Sampling-based approaches such as Rapidly-Exploring Random Trees (RRT) provide probabilistic completeness and scalability, while Deep Reinforcement Learning (DRL) offers policy optimization in high-dimensional, uncertain domains (Ma; Chen; Zhao, 2023). The integration of these paradigms enables reactive, intelligent motion planning that maintains performance in dynamic, partially known environments, a capability particularly valuable for cooperative UAV–UGV systems.

However, achieving reliable navigation in dynamic, partially known environments requires more than optimizing isolated algorithms. It demands a systemic perspective in which planning, perception, and communication operate as interdependent layers of a unified architecture. Within this context, coordination between UAVs and UGVs is a key enabler of true autonomy, enabling the team to continuously share situational awareness, adapt motion strategies, and reconfigure tasks based on environmental feedback (Pinto et al., 2017).

The integration of learning-based adaptation with classical planning offers a powerful means to achieve this cooperation in practice. By embedding path planning within a broader hybrid control architecture, the robotic system can reason about its actions collectively rather than individually. This paradigm shift transforms motion planning from a local optimization process into a global decision mechanism, one that aligns perception, action, and cooperation toward a common mission objective.

1.4- Advances and Challenges in Autonomous Planning

Although classical path planners such as $A^\star$, $D^\star$, and their variants remain foundational due to their reliability and interpretability, they often fail to meet real-time constraints in high-dimensional dynamic environments (Aggarwal; Kumar, 2020). Sampling-based algorithms such as Rapidly-Exploring Random Trees (RRT) have been widely adopted for their ability to handle complex constraints, yet they lack adaptability once a path is generated.

Recent works have explored integrating data-driven models with classical planners to overcome these limitations (Sung; Choi; Nielsen, 2021; Cui; Wang, 2021). Neural policies can accelerate search, predict collision-free regions, or refine sampling distribu-

tions, enabling hybrid systems that combine deterministic guarantees with learning-based adaptability. Nonetheless, the cooperation among multiple agents adds another layer of complexity, requiring dynamic task reallocation, decentralized communication, and adaptive coordination under changing mission contexts (Ramos; Pinto; Haddad, 2025).

This research addresses these challenges by proposing a cooperative hybrid planning architecture that merges the generalization power of learning-based models with the deterministic safety of classical planners. The proposed method allows UAVs and UGVs to plan and adapt jointly in environments that evolve over time, ensuring continuity of operations even under partial perception or temporary communication failures.

1.5- Main Contributions and Motivations

This thesis introduces a novel framework for heterogeneous multi-robot cooperation focused on coverage and path planning in dynamic and partially known environments. The proposed architecture integrates UAVs and UGVs in a system with UAVs performing aerial mapping and UGVs handling ground-level inspection, transportation, and energy support.

The scientific contributions of this work can be summarized as follows:

1. Hybrid path planning framework: Development of an adaptive navigation strategy that fuses RRT with DRL to achieve 3D motion planning in dynamic environments;
2. Coverage Path Planning: Integration of a sensor-based CPP algorithm to minimize overlap, energy consumption, and coverage time under UAV flight constraints;
3. Cooperative architecture: Design of a UAV–UGV coordination model with shared mapping, task adaptation, and fault-tolerant behavior under communication or energy disruptions;
4. Simulation-based validation: Implementation and testing in ROS and Gazebo environments, replicating realistic physical and computational constraints for heterogeneous robotic teams.

By combining classical motion planning and modern learning-based adaptation, this research advances the state of the art in cooperative robotics, contributing toward more autonomous, resilient, and efficient multi-robot systems.

1.6- Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews the state of the art on heterogeneous multi-robot cooperation, coverage path planning, and reinforcement learning for adaptive navigation. Chapter 3 describes the proposed hybrid architecture, detailing the mathematical formulation and control strategies for UAV and UGV systems. Chapter 4 presents the experimental setup and simulation results, validating the proposed methodology in ROS/Gazebo environments. Finally, Chapter 5 summarizes the main conclusions and outlines perspectives for future research.

2- Related Works and Background

2.1- From Cooperation to Hybrid Architectures

The evolution of cooperative robotics represents a gradual shift from simple interaction paradigms to complex architectures capable of autonomous decision-making under uncertainty. Early systems such as ALLIANCE (Parker, 1998) and ACTRESS (Cao; Fukunaga; Kahng, 1997) introduced the foundations of multi-robot cooperation, emphasizing robustness and fault tolerance through behavior-based control. These architectures relied on local sensing and implicit coordination, enabling emergent cooperation without centralized supervision.

However, as robotic missions became more complex, facing exploration, mapping, and inspection, only reactive systems proved to be insufficient. The field progressed toward structured frameworks capable of reasoning about goals, dependencies, and shared resources (Matarić, 2007; Michael; Fink; Kumar, 2011). This transition from emergent behavior to hierarchical cooperation marked the birth of hybrid architectures, where global decision layers coexist with local autonomy.

Recent reviews confirm that modern cooperative systems have evolved into heterogeneous networks, communication-driven (Madridano et al., 2021; Kim; Kim; Park, 2022). UAV–UGV teams, in particular, exemplify this trend by utilizing complementary capabilities: aerial agents contribute global perception and mobility, while ground agents provide endurance, manipulation, and computational stability (Berger; Scherer; Lozano, 2023; Qiu; Lin; Wang, 2024). Despite these advances, scalability and adaptability remain open challenges, motivating the development of hybrid frameworks that integrate perception, communication, and motion planning across heterogeneous agents.

To contextualize this evolution, Table 1 summarizes the chronological development of cooperative architectures, highlighting their main contributions and limitations across the past three decades.

Table 1 – Evolution of Cooperative Robotic Architectures.

Period	Architecture Method	Main Contributions	Limitations	Reference
1990 2000	ACTRESS, ALLIANCE	Introduced behavior-based cooperation emphasizing robustness and fault tolerance	Lack of global planning and explicit coordination	(Cao; Fukunaga; Kahng, 1997; Parker, 1998)
2000 2010	Hierarchical Architectures	Combined high-level decision and low-level autonomy for structured tasks	Limited scalability and dependence on precise models	(Matarić, 2007; Michael; Fink; Kumar, 2011)
2010 2020	UAV–UGV Cooperation	Integration of communication and sensor sharing for joint tasks	Complex synchronization and planning among heterogeneous agents	(Madridano et al., 2021; Kim; Kim; Park, 2022)
2020 Present	Hybrid Architectures with Learning	Integration of perception, coordination, and learning through DRL	High computational cost and limited real-world datasets	(Berger; Scherer; Lozano, 2023; Qiu; Lin; Wang, 2024)

2.2- The Challenge of Coordination and Task Allocation

At the core of cooperation lies the problem of coordination, i.e., deciding which agent performs which task, and when. The MRTA problem formalizes this challenge as the assignment of tasks to agents to maximize collective efficiency (Gerkey; Matarić, 2004). The taxonomy proposed by Korsah et al. (Korsah; Stentz; Dias, 2013) provided a rigorous classification of MRTA problems by agent-task multiplicity and temporal structure, becoming a theoretical pillar of multi-robot coordination.

Initially, MRTA was dominated by market-based frameworks (Dias et al., 2006), which drew inspiration from economics to allocate tasks via distributed auction mechanisms. While effective in structured environments, such approaches often assumed perfect communication and complete knowledge of the world. As robotic systems entered more uncertain and dynamic contexts, new paradigms emerged, including graph-theoretic and learning-based methods that dynamically infer task assignments from local feedback (Liu; Zhang; Meng, 2022; Tang et al., 2023).

For heterogeneous teams, MRTA extends beyond simple task division. It must

consider cross-domain trade-offs, energy efficiency, mobility constraints, and sensing ranges that are particularly relevant for UAV–UGV cooperation. Recent advances use DRL to couple task allocation with motion planning and resource management (Mondal et al., 2025). In these frameworks, coordination becomes not only a matter of logic but also of continuous adaptation, where agents learn to negotiate and reassign roles in real time.

To provide a structured comparison, Table 2 classifies the principal MRTA approaches, highlighting their main advantages and limitations in the context of cooperative robotics.

Table 2 – Coordination and MRTA Approaches in Cooperative Robotics

Category	Method	Key Advantages	Challenges	Reference
Distributed	Market- and Auction-Based	Decentralized allocation; scalable coordination	Requires reliable communication and global awareness	(Dias et al., 2006)
Graph-Theoretic	Matching and Flow Optimization	Formal representation of inter-task dependencies	Limited adaptability to dynamic and uncertain environments	(Liu; Zhang; Meng, 2022; Tang et al., 2023)
Learning-Based	Deep and Multi-Agent Reinforcement Learning (DRL/MARL)	Continuous adaptation; implicit cooperation and negotiation	Convergence instability; heavy computational training demand	(Mondal et al., 2025; Li et al., 2024)

2.3- From Task Allocation to Cooperative Coverage.

Task allocation defines who acts, but not necessarily how space is explored. In many multi-robot missions, such as surveillance, agriculture, or environmental monitoring, the logical outcome of MRTA is a coverage problem: how to partition and explore a region efficiently. This connection between task assignment and spatial planning gave rise to the field of CPP (Galceran; Carreras, 2013; Choset, 2001).

CPP aims to ensure full area coverage while minimizing redundant traversal, energy consumption, and execution time. Classical methods such as cellular decomposition and boustrophedon trajectories offer deterministic guarantees for single-robot coverage but struggle to scale in multi-agent and dynamic scenarios. Consequently, CPP has evolved into a multi-layered decision problem that must integrate area segmentation, trajectory generation, and coordination between heterogeneous agents.

Effective CPP begins with the spatial segmentation of the environment into discrete regions or cells, enabling each robot to operate on a subdomain while minimizing overlap. The literature identifies several segmentation paradigms:

- Exact and Boustrophedon Cellular Decomposition (ECD/BCD) — partitions the workspace along critical points of obstacle boundaries, ensuring mathematical completeness but high computational cost (Choset, 2001).
- Voronoi- and Graph-Based Partitioning — produces balanced regions by distance metrics or connectivity, widely used for heterogeneous multi-robot systems (Cabreira; Brisolara; Ferreira, 2019; Cabreira; Almeida; Brisolara, 2023).
- Wavefront and Floodfill Propagation — deterministic grid-based approaches that simulate wave expansion to determine the next unvisited region; efficient and easily parallelizable for multi-robot coverage (Galceran; Carreras, 2013).

Segmentation not only defines spatial responsibility but also dictates how communication and task allocation are coupled: global methods, such as Voronoi or BCD, require centralized computation, whereas local propagation (Wavefront or Floodfill) enables distributed decision-making, an essential property for UAV–UGV teams operating under intermittent connectivity.

To overcome the limitations of purely geometric planners, recent works have introduced hybrid algorithms that couple deterministic propagation with heuristic search. Among these, the family of algorithms that integrate Wavefront or Floodfill expansion with informed search methods, such as $A^\star$ or $D^\star$, have proven particularly effective in structured environments with obstacles and non-uniform cost maps (Zhang; Liu; Wang, 2022; Gao; Sun; Li, 2023).

In this context, (Castro, G. G. R. de et al., 2025) proposed a hybrid algorithm that combines Wavefront Propagation and Bidirectional $A^\star$ ($BA^\star$) to optimize coverage in grid-based environments. The Wavefront layer guarantees complete area accessibility by propagating the coverage frontier uniformly, while $BA^\star$ defines shorter paths than the greedy wavefront policy when the first algorithm gets stuck in local-minima, accelerating convergence between start and goal points by simultaneously expanding search trees in both directions. This approach achieved substantial gains in computational efficiency and trajectory smoothness compared to classical BCD and single-direction $A^\star$ planners, particularly in cluttered domains (Castro, G. G. R. de et al., 2025).

Complementarily, in (Carvalho et al., 2025) introduced a Floodfill + $BA^{\star}$ algorithm tailored for heterogeneous UAV–UGV systems, emphasizing robust coverage under obstacle density and dynamic mission constraints. This algorithm uses deterministic propagation for region identification and bidirectional heuristic planning for local optimization, achieving a balance between exploration completeness and responsiveness to dynamic changes. The results demonstrated improved coverage uniformity and adaptability when integrating UAV global maps with UGV local execution.

These contributions situate hybrid propagation–search algorithms as a promising intermediary between classical deterministic CPP and learning-based planners, forming a natural bridge to the adaptive architectures explored later in this thesis.

To illustrate the position of these hybrid methods in the broader CPP landscape, Table 3 compares representative approaches, emphasizing their computational nature, advantages, and limitations.

Table 3 – Comparison of Coverage Path Planning (CPP) Algorithms.

Algorithm / Method	Type	Advantages	Limitations	Reference
Cellular / Boustrophedon Decomposition	Deterministic Geometric	Guarantees full coverage; mathematically complete	High computational cost; poor scalability to dynamic environments	(Choset, 2001; Galceran; Carreras, 2013)
Voronoi and Graph-Based Partitioning	Topological / Distributed	Balanced area segmentation; suitable for multi-robot teams	Requires accurate map and global knowledge; communication overhead	(Cabreira; Brisolara; Ferreira, 2019; Cabreira; Almeida; Brisolara, 2023)
Wavefront Propagation	Grid-Based Deterministic	Simple and robust; efficient in static structured environments	Suboptimal paths and higher overlap without heuristics	(Galceran; Carreras, 2013)
Floodfill Expansion	Deterministic / Local	Ensures high coverage in obstacle-rich domains	Sensitive to map resolution; limited path optimization	(Cabreira; Almeida; Brisolara, 2023)
Wavefront + Bidirectional A* (BA*)	Hybrid Heuristic–Deterministic	Reduces overlap; improves trajectory smoothness and computation time	Requires grid connectivity and tuning of heuristic weights	(Castro, G. G. R. de et al., 2025)
Floodfill + Bidirectional A* (BA*)	Hybrid Local–Heuristic	Enhances coverage completeness and responsiveness under dynamic conditions	Slightly higher computational demand; sensitive to dynamic map updates	(Carvalho et al., 2025)
Learning-Based (DRL, MARL)	Adaptive / Data-Driven	Online adaptation to dynamic environments; scalable to heterogeneous teams	Requires large datasets and extensive training; low interpretability	(Liu2021; Wen2022; Mondal et al., 2025)

As shown, CPP has evolved from deterministic geometric coverage to hybrid and adaptive formulations that combine propagation, heuristic search, and machine learning. The hybrid approaches proposed in (Castro, G. G. R. de et al., 2025; Carvalho et al., 2025) demonstrate that integrating global and local decision layers—Wavefront for exploration scope and Bidirectional $A^\star$ for local optimization, enables efficient and adaptable coverage, especially when UAVs and UGVs cooperate hierarchically.

This perspective aligns directly with the hybrid architecture proposed in this thesis, where coverage planning operates as an intermediate layer between task allocation (Section 2.2) and dynamic motion adaptation (Section 2.4). By utilizing these hybrid CPP methods, UAV–UGV teams can synchronize global perception with ground-level execution, bridging the gap between classical CPP algorithms and reinforcement learning–based cooperation frameworks.

2.4- Navigating Dynamic Environments: Planning Beyond Geometry

While MRTA and CPP address coordination and spatial partitioning at a global scale, real-world environments are rarely static or predictable. Unexpected obstacles, moving agents, sensor noise, and discrepancies between the static map and real perception require continuous local replanning. This local replanning capability is essential to hybrid cooperative architectures, enabling agents to react to environmental changes in real-time while maintaining mission-level goals.

In the proposed framework, local path planning operates as a reactive and safety-oriented layer. It ensures collision-free navigation when discrepancies arise between the preplanned global map and online sensor readings. Moreover, it improves exploration efficiency by prioritizing unvisited areas while coupling safety and task completion within the same decision cycle. This design philosophy aligns with the hybrid perception–planning pipeline described in (Yang; Sukkarieh, 2008; Yan; Liu; Xiao, 2013; Debnath et al., 2020; Schøler; Cour-Harbo; Bisgaard, 2012), where local planners act as the adaptive component linking perception and control.

Over the past few decades, numerous path-planning algorithms have been proposed for navigation in both static and dynamic environments. Traditional graph-based methods

such as Dijkstra's algorithm (Dijkstra et al., 1959), $A^\star$ (Hart; Nilsson; Raphael, 1968), and $D^\star$ (Carsten; Ferguson; Stentz, 2006) provide deterministic solutions with strong optimality guarantees, but have scalability issues and assume static maps. Sampling-based methods like RRT (Yang; Sukkarieh, 2008) and Probabilistic Roadmaps (PRM) (Yan; Liu; Xiao, 2013) enable fast search in high-dimensional spaces but produce non-optimal paths and require post-smoothing. Bio-inspired and metaheuristic planners such as Genetic Algorithms (GA) (Tsai; Chou; Liu, 2006) and Artificial Potential Fields (APF) (Chen; Zhang, 2013) offer adaptability but often converge to local minima or suffer from poor scalability.

Despite their limitations, these methods remain valuable for structured environments and as fallback planners. In hybrid architectures, they are often embedded as low-level planners triggered whenever the agent detects an inconsistency between the expected and observed environment.

With the increasing complexity of dynamic missions, neural network (NN)–based approaches have emerged as powerful alternatives for online adaptation. Early implementations used NNs to approximate trajectory optimization (Horn et al., 2012) or to model system dynamics and cost functions (Horn; Geiger; Schmidt, 2009), thereby reducing the dimensionality of non-linear programming problems. More recent works integrate NNs with traditional algorithms—such as APF, GA, or Particle Swarm Optimization (PSO)—to improve convergence and robustness (Zhao; Zheng; Liu, 2018).

A significant milestone came with the introduction of DRL in navigation. Chen et al. (Chen; Zhang, 2013) applied neural control to overcome APF local minima, while Motlagh et al. (Motlagh et al., 2014) used RL to allow mobile robots to learn navigation policies autonomously. Tong et al. (Tong et al., 2021) later proposed a DRL-based approach for UAVs, decomposing navigation into sub-tasks using a Deep Q-Network (DQN), achieving robustness in highly dynamic environments. Similarly, Lei et al. (Lei; Zhang; Dong, 2018) implemented a Double DQN architecture to accelerate convergence and improve stability in unknown terrains, and Yang et al. (Yang; Juntao; Lingling, 2020a) combined RRT and DQN to use the exploratory efficiency of RRT with the adaptivity of deep learning. These works demonstrate that RL-based local planners can adapt not only to static obstacles but also to time-varying constraints, sensor drift, and partial observability.

In the context of cooperative UAV–UGV systems, this adaptability is crucial: aerial agents must continuously re-evaluate their paths to maintain line-of-sight with ground

robots, while UGVs must avoid dynamic obstacles while respecting communication and energy constraints. Integrating DRL-based local planning modules within hybrid architectures allows the entire team to reconfigure routes collaboratively in real time.

In this thesis, the local path planning layer follows a hybrid paradigm that merges deterministic and learning-based reasoning. When discrepancies between the global static map and online sensor data are detected, a reactive module activates, executing a local replan that combines two decision objectives: (1) ensuring agent safety through obstacle avoidance; and (2) prioritizing unvisited frontier zones to maintain mission progress.

This dual-objective planner extends the classical frontier-based exploration concept (Yamauchi, 1997), integrating it with neural-based predictive models to identify navigable regions before committing to motion commands. The planner computes short-term trajectories using modified $D^\star$ Lite or incremental $A^\star$ search for deterministic reactivity, and augments them with DRL-based policies for adaptive prioritization in uncertain areas. This synergy allows the system to maintain operational continuity even in rapidly changing or partially mapped environments.

Table 4 summarizes the advantages and drawbacks of contemporary path planning algorithms, from classical geometric methods to modern reinforcement learning–based strategies, providing an overview of how each contributes to safety, adaptability, and computational efficiency in hybrid cooperative frameworks.

Local path planning thus represents more than obstacle avoidance, it embodies the agent's capacity to interpret discrepancies between model and reality, preserving safety while promoting exploration. By integrating deterministic planners (e.g., D* Lite, A*) with learning-based adaptation (e.g., DRL or DQN), the proposed architecture ensures that each robot maintains situational awareness and mission continuity, even under partial observability. This continuous interplay between global and local reasoning defines the adaptability and robustness of the hybrid cooperative framework, bridging perception, motion, and learning as a unified decision process.

Table 4 – Advantages and Drawbacks of Contemporary Path Planning Algorithms.

Algorithm / Family	Advantages	Drawbacks	References
RRT / PRM	Simple implementation; scalable to high-dimensional spaces	Non-optimal paths; sensitive to randomness; slow in cluttered environments	(Yang; Sukkarieh, 2008; Yan; Liu; Xiao, 2013)
Dijkstra / $A^\star$ / $D^\star$	Optimality and completeness in static environments	High time complexity; poor reactivity to dynamic changes	(Dijkstra et al., 1959; Hart; Nilsson; Raphael, 1968; Carsten; Ferguson; Stentz, 2006)
APF	Fast convergence; intuitive force-field metaphor	Local minima; unstable under dynamic obstacles	(Chen; Zhang, 2013)
Genetic / PSO	Solves NP-hard multiobjective optimization	High computational cost; premature convergence	(Tsai; Chou; Liu, 2006; Zhao; Zheng; Liu, 2018)
CNN-based	Learns from perception data; generalizes across environments	Requires large datasets; high hyperparameter tuning effort	(Wu et al., 2019; Qureshi et al., 2019)
DQN / DRL	Adapts to dynamic and unknown conditions; decentralized learning	Non-optimal paths; complex training; limited interpretability	(Tong et al., 2021; Lei; Zhang; Dong, 2018; Yang; Juntao; Lingling, 2020a)
Hybrid ($D^\star$ Lite + DRL, Frontier-Based)	Balances reactivity and learning; ensures safety and coverage continuity	Requires sensor reliability and computational resources	(Yamauchi, 1997; Gao; Sun; Li, 2023; Mondal et al., 2025)

2.5- Insights from Recent UAV–UGV Collaboration Reviews

Recent surveys, such as the comprehensive review by Munasinghe, Perera, and Deo (2024) (Munasinghe; Perera; Deo, 2024), highlighted a paradigm shift in heterogeneous robotics toward tighter UAV–UGV cooperation. Their study categorized collaboration frameworks according to the functional roles each platform may assume, namely as sensors, actuators, decision-makers, or auxiliary facilitators, establishing a useful taxonomy for analyzing multi-domain missions. Such a taxonomy not only clarifies the division of

responsibilities in cooperative systems but also provides a foundation for understanding the trade-offs between autonomy, communication bandwidth, and energy efficiency. The review further identifies that reliable communication and coordination mechanisms remain major bottlenecks, despite advances in relay pods, millimeter-wave links, and 5G-enabled Quality-of-Service (QoS) architectures. Besides, sensor fusion and 3D mapping techniques have become critical enablers for achieving cooperative localization in GPS-denied or dynamic environments, a recurring challenge addressed by recent UAV-UGV systems.

In terms of applications, the review consolidates significant progress across domains such as search and rescue, precision agriculture, and infrastructure inspection, each demonstrating the complementary nature of aerial and ground assets. These works reinforce the motivation of this thesis, which proposes a hybrid cooperative architecture that integrates CPP, task allocation, and local motion control within a unified UAV–UGV framework, thereby addressing several of the open issues identified in contemporary literature.

To illustrate these advances, Table 5 compares representative UAV–UGV hybrid frameworks published between 2020 and 2025, detailing their objectives, algorithms, and main findings.

Table 5 – Recent Hybrid UAV–UGV Frameworks (2020–2025).

Study / Year	Main Objective	Approach / Algorithm	Key Findings / Limitations	Reference
Berger et al. (2023)	Cooperative exploration and mapping	Hierarchical control integrating UAV perception with UGV mapping	Effective in static environments; limited reactivity to dynamics	(Berger; Scherer; Lozano, 2023)
Wang et al. (2024)	Multi-layered coverage planning	UAV global scanning guiding UGV ground coverage	Improved efficiency; communication latency impacts coordination	(Wang; Chen; Huang, 2024)
Zhang et al. (2025)	Air-ground hierarchical CPP	Joint aerial-ground coverage optimization with dynamic replanning	High coverage efficiency; sensitive to sensor drift	(Zhang; Zhao; Lin, 2025)
Mondal et al. (2025)	Energy-aware MRTA integration	Deep reinforcement learning coupled with task allocation	Adaptive allocation; requires extensive training	(Mondal et al., 2025)
Li et al. (2024)	Coalition formation in cluttered terrain	Multi-agent reinforcement learning (MARL) for adaptive path coordination	High adaptability; computationally intensive in real-time missions	(Li et al., 2024)

3- Proposed Framework

The methodology and mathematical foundations presented in this chapter are built upon the author's previously published contributions. The first work, Heterogeneous Multi-Robot Collaboration for Coverage Path Planning in Partially Known Dynamic Environments, published in Machines (Vol. 12, Issue 3, 2024) (Castro, G. G. de et al., 2024), introduced the core concepts of hierarchical cooperation between UAVs and UGVs under partial map knowledge. The second work, Adaptive Path Planning for Fusing Rapidly Exploring Random Trees and Deep Reinforcement Learning in an Agriculture Dynamic Environment UAVs, published in Agriculture (Vol. 13, Issue 2, 2023) (Castro, G. G. de et al., 2023), established the foundations for the local dynamic path planning module employed in this thesis.

More recently, the hybrid CPP frameworks proposed by the author in (Castro, G. G. R. de et al., 2025; Carvalho et al., 2025) demonstrated the efficacy of combining classical propagation-based methods with heuristic search to produce globally complete and locally optimized coverage trajectories. These works show that integrating global and local decision layers, namely Wavefront propagation for exploration scope definition and Bidirectional $A^\star$ for short-range optimization, enables fast, robust, and adaptive coverage, particularly in missions where UAVs and UGVs operate cooperatively under a hierarchical architecture.

Building on these foundations, the present chapter details the complete hybrid framework developed in this thesis. The chapter is divided into two complementary parts:

- Part I — Proposed Hybrid Architecture: This part introduces the mission model, the multi-layer decision structure, the global coverage planner (Wavefront + Bidirectional $A^\star$), the local dynamic path planner (DRL–RRT hybrid), and the task reallocation logic that links the UAV–UGV cooperation loop.

- Part II — Baseline Algorithms for Comparison: This part presents the classical and learning-based algorithms selected as baselines, which are used to benchmark the performance of the proposed architecture in the results chapter.

Part I

3.1- Mission Overview and System Assumptions

The mission addressed in this thesis involves the cooperative operation of a heterogeneous robotic team composed of two Unmanned Aerial Vehicles (UAVs) and one Unmanned Ground Vehicle (UGV) deployed in a partially known and dynamic environment. As established in the author's previous works (Castro, G. G. de et al., 2024, 2023), the operational scenario is characterized by three fundamental conditions: (i) an initial global map is available but contains uncertain or outdated information, (ii) new obstacles may appear during execution due to the dynamic nature of the environment, and (iii) the robots possess complementary sensing and mobility capabilities that must be exploited cooperatively.

The objective of the mission is to ensure full and efficient coverage of the environment while maintaining the agents' safety and enabling real-time adaptation to unforeseen changes. The UAVs provide high-level situational awareness by capturing aerial observations, identifying candidate frontier regions, and broadcasting global updates. In contrast, the UGV performs detailed ground inspection and executes refined coverage of areas that are either inaccessible to the aerial platforms or inefficient for them to inspect. The cooperation between these platforms is crucial for handling large-scale or highly cluttered regions, mitigating occlusions, and reducing overall mission duration through parallelism and complementary capabilities. Figure 1 gives a general idea of the proposed architecture during an inspection of an electrical substation and Figure 2 the same but in agriculture scenario.

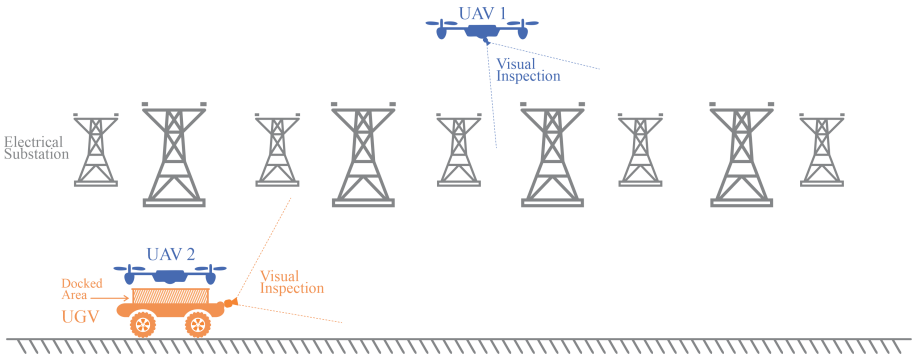


Figure 1 – Proposed methodology applied in an electrical substation scenario (Castro, G. G. R. de et al., 2024).

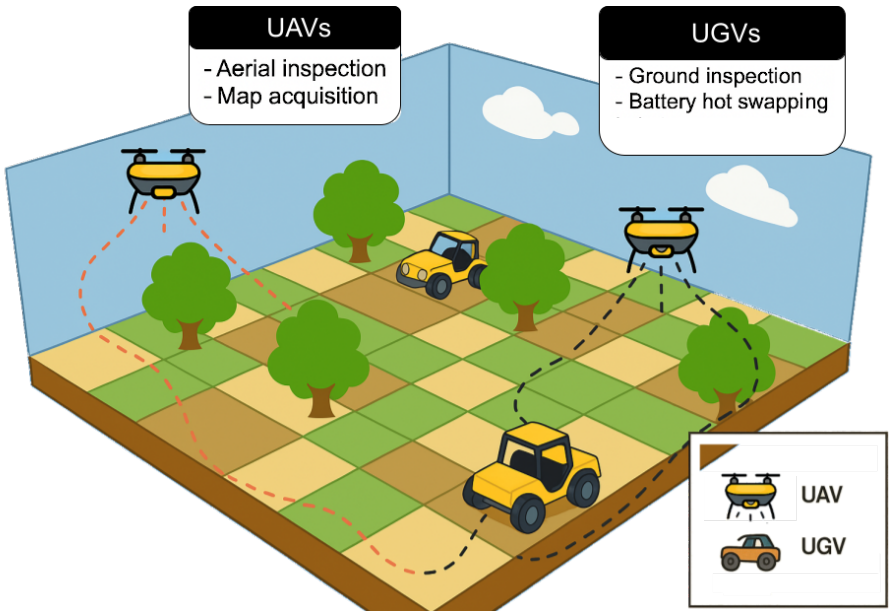


Figure 2 – Proposed methodology applied on an agriculture scenario.

The three-dimensional physical environment is projected into a two-dimensional occupancy grid used for planning purposes. The workspace is discretized into a uniform grid divided into layers, where each cell encodes occupancy and exploration status (visited or unvisited). Although an initial map is assumed to be fully available, either from previous missions or from pre-flight aerial scans, the information it contains may be inaccurate, incomplete, or outdated. Unmapped or uncertain regions are conservatively treated as occupied. The state of each grid cell may change at any moment according to sensor readings obtained by the robots.

To address these environmental constraints, the proposed architecture combines two complementary planning layers. The first is a global coverage planning phase, responsible for generating an initial collision-free route that visits all free cells at least once. This global route is computed exclusively from the initial map and represents a deterministic baseline for mission execution. The second is an online local planning module, which continuously adjusts each agent's trajectory when discrepancies arise between the static map and the observed environment. This module ensures collision avoidance, maintains safety margins, and enables the system to dynamically adapt when new obstacles appear or when previously free regions become obstructed.

Communication between the robots is assumed to be available but not always reliable. Following the assumptions discussed in (Castro, G. G. de et al., 2024), the UAVs

maintain intermittent line-of-sight communication with the UGV and relay updates when possible. This communication model reflects realistic indoor and cluttered scenarios where wireless links may degrade due to occlusions or electromagnetic interference. Despite these limitations, the robots are assumed to exchange essential information, such as local map patches, coverage progress, task completion signals, and reallocation triggers.

The mission relies fundamentally on the complementarity of the agents. The UAVs offer high mobility, global perception, and rapid traversal but are constrained by battery capacity and payload limitations. The UGV provides reliable locomotion, extended operational duration, and high-fidelity ground sensing but is restricted by terrain topology and obstacle density. These characteristics naturally shape the cooperative architecture, in which global situational awareness and map exploration are primarily performed by the aerial agents, while local refinement and detailed area coverage are executed by the ground agent. This hybrid interaction forms the conceptual basis for the planning and coordination framework described in the following sections. Figure 3 provides a comprehensive overview of the proposed cooperation strategy among heterogeneous robots (i.e., UAVs and UGVs), in a dynamic environment partially mapped with potential dynamic obstacles.

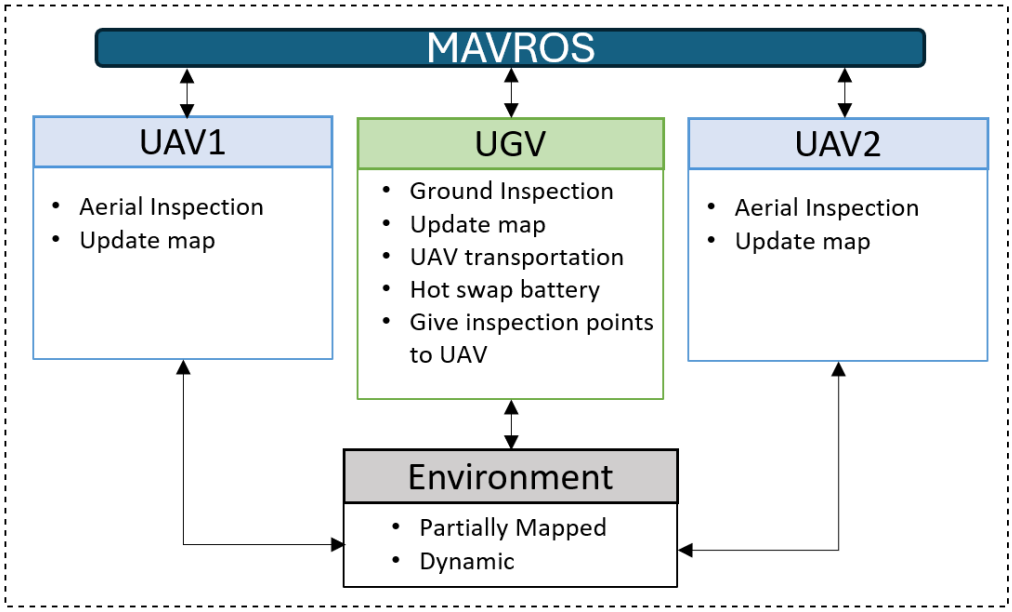


Figure 3 – Overview of the proposed cooperation strategy (Castro, G. G. R. de et al., 2024).

3.1.1 Multi-Layer Hybrid Architecture

The cooperative framework proposed in this thesis is structured as a multi-layer hybrid architecture that integrates global coverage planning, local dynamic path optimization, and adaptive task allocation among heterogeneous agents. This architecture is grounded in the author's previous contributions on hierarchical UAV-UGV cooperation (castro2024heterogeneous), adaptive local planners (Castro, G. G. d. et al., 2023), and hybrid coverage formulations combining Wavefront propagation and Bidirectional $A^{\star}$ search (castro2025hybrid; castro2025b).

To contextualize the system's execution pipeline, Figure 4 provides a high-level operational flowchart. The process begins with the agents interacting with the dynamic environment through the perception and mapping modules. This sensory data feeds directly into the "Architecture Core", which sequentially processes the mission through three distinct decision-making levels: macroscopic route generation (Global Coverage), real-time collision avoidance (Online Adaptive Planning), and team management (Task Reallocation and Coordination). Ultimately, these decisions are translated into physical commands at the UAV/UGV execution level, forming a continuous closed-loop system.

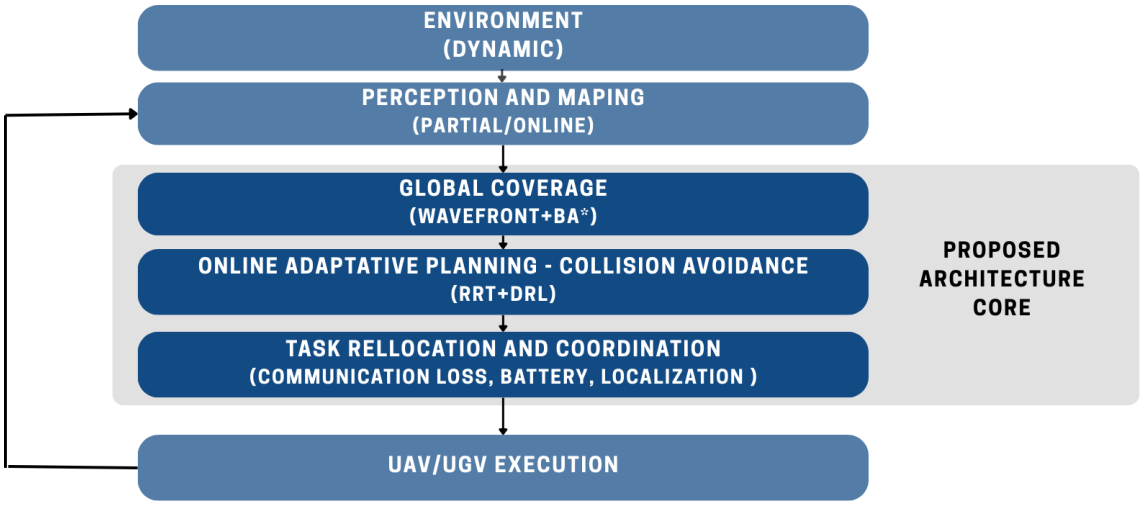


Figure 4 – High-level representation of the cooperative architecture's pipeline, illustrating the data flow from environmental perception to physical execution through the three core planning layers.

The core architecture is strictly composed of these three interconnected layers:

- (i) Global Coverage Layer — responsible for generating a complete, collision-free

route that ensures full exploration of the environment under an initially known static map. This layer employs Wavefront propagation to estimate exploration fronts and assigns reachable regions to each agent. The Bidirectional $A^\star$ refines short-horizon transitions between local subregions, improving smoothness and reducing unnecessary maneuvers.

- (ii) Local Dynamic Planning Layer — responsible for real-time adaptation when discrepancies between the static map and new sensor readings arise. This module fuses a DRL-based navigation policy with an RRT-inspired sampling strategy, adjusting trajectories to avoid emergent obstacles, maintain safety margins, and preserve the continuity of the mission.

- (iii) Coordination and Task Reallocation Layer — the supervisory module that manages cooperation between UAVs and the UGV. It handles region assignment, reallocation triggers, communication of map patches, battery-related constraints, and dynamic adjustments to maintain mission coherence even under partial map invalidation or delayed communication.

These layers operate asynchronously but exchange structured information through the mission communication protocol introduced in (Castro, G. G. de et al., 2024). The UAVs provide fast global perception and identify large-scale environmental changes, whereas the UGV offers long-term autonomy and detailed ground sensing. Their cooperation enables a hierarchical synergy: the UAV accelerates global decision-making while the UGV performs precise local inspection.

The interplay between global and local planning is the core of this architecture. Global planning provides a deterministic high-level skeleton for coverage, while local planning ensures responsiveness and safety in dynamic environments. The author's previous works (Castro, G. G. R. de et al., 2025; Carvalho et al., 2025) demonstrated that coupling Wavefront propagation with Bidirectional $A^\star$ improves coverage efficiency by reducing redundant traversal and avoiding dead-ends. Meanwhile, the DRL–RRT hybrid strategy (Castro, G. G. de et al., 2023) provides robust adaptability when unexpected obstacles emerge during execution.

This multi-layered structure forms the foundation of the methodology developed in this thesis. The following sections detail each layer individually: Section 3.3 presents the global coverage planner; Section 3.4 describes the dynamic local planner; Section 3.5

introduces the cooperative task allocation and reallocation logic; and Section 3.6 consolidates the complete operational pipeline.

3.2- Global Coverage Planning

The global planner is responsible for generating a deterministic, collision-free coverage route that guarantees that all free cells of the environment are visited at least once. The environment is represented as a discrete occupancy grid $\mathcal{G}$, where each cell p is classified as either free ($\mathcal{A}_{\text{free}}$) or occupied ($\mathcal{A}_{\text{obs}}$). The planner assumes that $\mathcal{G}$ is static during route generation, following the hierarchical planning separation adopted in this thesis and previously evaluated in (Castro, G. G. R. de et al., 2025; Carvalho et al., 2025; Castro, G. G. de et al., 2024).

The objective is to generate a coverage path:

$$\mathcal{P} = \{p_1, p_2, \ldots, p_n\}, \tag{1}$$

such that:

$$\mathcal{P} \cap \mathcal{A}_{\text{obs}} = \emptyset,$$

ensuring complete, collision-free traversal.

To meet this objective efficiently, the global planner combines:

- a Wavefront propagation module, which generates a global gradient field guiding large-scale exploration;
- a Bidirectional $A^\star$ module, used as a fallback mechanism when Wavefront becomes trapped or when transitions between isolated free-space components are required.

This hybridization leverages the global completeness of Wavefront and the optimality guarantees of heuristic search, ensuring robustness against topological irregularities in the map.

3.2.1 Wavefront Propagation for Global Gradient Construction

Wavefront propagation starts at a designated goal or frontier cell p_{goal} and expands outward through $\mathcal{A}_{\text{free}}$. Each cell receives a cost proportional to the number of discrete moves required to reach the goal.

Let $w(p)$ denote the weight of a grid cell p. The update rule is:

$$w(p) = \min\{w(p_{\text{adj}}) + 1\},\ p_{\text{adj}} \in \text{neighbors}(p), \tag{2}$$

where p_{adj} represents adjacent cells and neighbors(p) is described as:

$$\text{neighbors}(p) = \{(x \pm 1, y), (x, y \pm 1), (x \pm 1, y \pm 1)\}. \tag{3}$$

This produces a discrete potential field over $\mathcal{G}$, enabling the robot to move greedily from p_{start} to p_{goal} by repetitively selecting the lowest-weight adjacent cell.

As demonstrated in (Castro, G. G. R. de et al., 2025), this method:

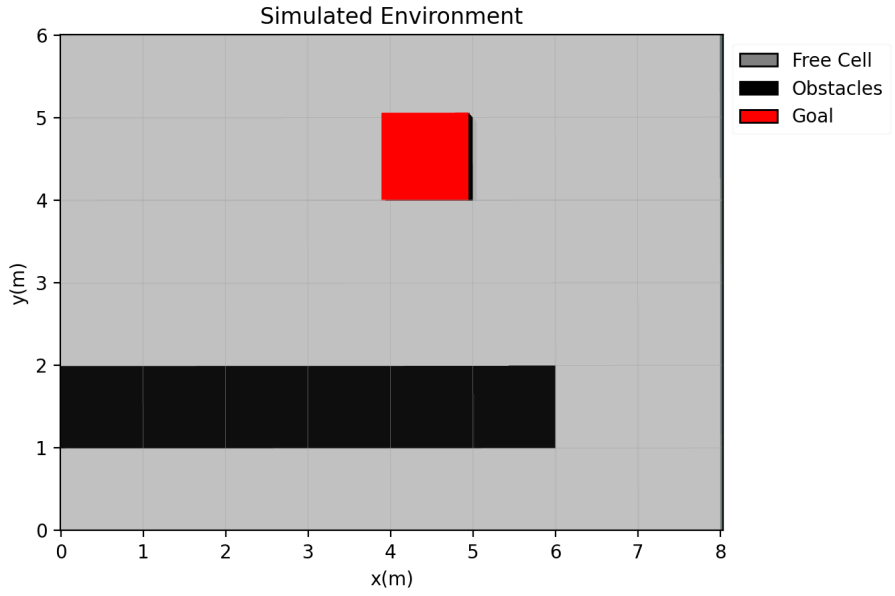
- ensures uniform expansion over the entire free space;
- avoids metric computations (cheaper than Euclidean potentials);
- naturally identifies frontiers and isolated pockets of free cells;
- produces predictable, fully connected coverage routes.

Figure 5 illustrates typical weight propagation.

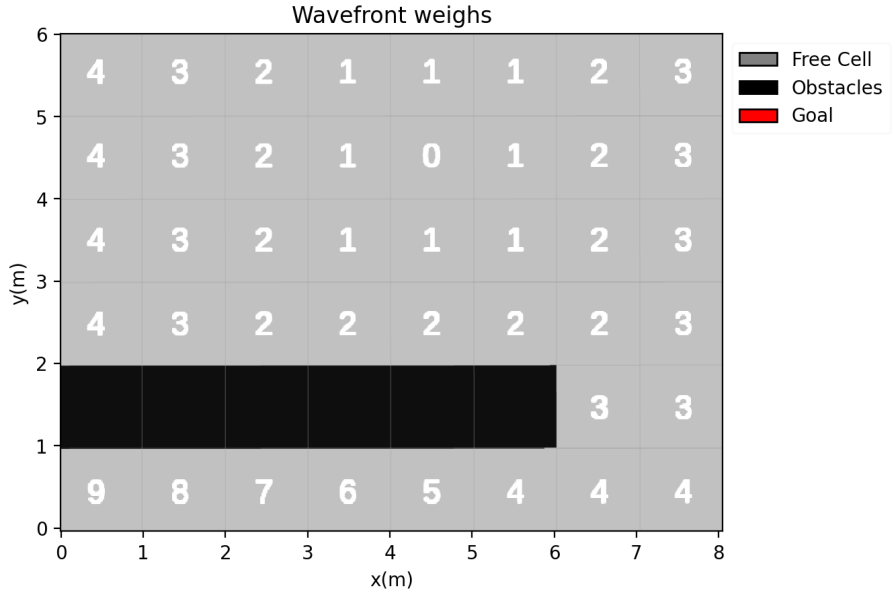
3.2.2 Hybridization with Bidirectional A*

Wavefront propagation may fail in two situations:

1. when it reaches a dead-end with no valid free neighbor, trapped in local minima;
2. when regions of $\mathcal{A}_{\text{free}}$ are not directly connected, requiring transitions across complex topology.



(a)



(b)

Figure 5 – Wavefront propagation in a simulated environment.(Castro, G. G. R. de et al., 2025) (a) The simulated environment, where the red square represents the goal; (b) The simulated environment displaying the wavefront weights of each cell, propagated outward from the goal grid.

To overcome this, whenever Wavefront halts at a cell p_{halt}, the planner identifies the nearest unvisited free cell:

$$p_{\text{nearest}} = \arg\min_{p \in \mathcal{A}_{\text{nv}}} d(p_{\text{halt}}, p), \tag{4}$$

where d is the Manhattan distance and $\mathcal{A}_{\text{nv}}$ is the set of not-visited free cells.

A Bidirectional $A^\star$ is then invoked to compute a transition path:

$$\mathcal{P}_{A^\star} = \{p_1, \ldots, p_k\}, \qquad p_1 = p_{\text{halt}},\ p_k = p_{\text{nearest}}.$$

The algorithm searches simultaneously from two directions: forward from the start point (p_{halt}) and backward from the goal point (p_{nearest}). Each search maintains its own priority queue, Q_{start} and Q_{goal}, ordered by a cost function $f(p)$, where:

$$f(p) = g(p) + h(p), \tag{5}$$

where g is accumulated cost and h the admissible heuristic.

The termination condition is:

$$Q_{\text{start}} \cap Q_{\text{goal}} \neq \emptyset.$$

Once the meeting point is found, paths from both searches are merged:

$$\mathcal{P}_{A^\star} = \mathcal{P}_{\text{start}} \ + \ \mathcal{P}^{\text{rev}}_{\text{goal}}\ , \tag{6}$$

where $\mathcal{P}_{\text{start}}$ is the path from p_{halt} to the meeting point, and $\mathcal{P}^{\text{rev}}_{\text{goal}}$ is the reversed path from p_{nearest} to the meeting point.

As shown experimentally in (Castro, G. G. R. de et al., 2025), this strategy mechanism:

- guarantees coverage completeness even in fragmented maps;
- drastically reduces Wavefront dead-ends;
- lowers redundant traversal;
- preserves optimality within transition segments.

A representative execution of the proposed strategy is illustrated in Figure 6. Starting from $p = (0, 2)$, the path is generated by performing a gradient ascent on the wavefront potential field (see Figure 5). The algorithm iteratively selects adjacent cells with decreasing weights—moving from four to three, and subsequently to two. At cell $p = (6, 2)$, a stochastic branching occurs towards $p = (6, 1)$; this choice is arbitrary, as the neighbors

$p = (7, 1)$ and others share identical weights and unvisited status.A similar scenario occurs at $p = (0, 6)$, where the path continues until reaching a local minimum at $p = (0, 0)$. At this point, no unvisited neighbors are available, causing the agent to halt. To resolve this, two distinct strategies are compared in Figure 6. A pure backtracking approach (shown in red) simply retraces the path through previously visited cells until a free neighbor is found, which results in a redundant and complex path with significant overlaps. In contrast, the proposed Bidirectional $A^\star$ search (shown in blue) identifies the $p_{nearest}$ free cell in the unvisited space $\mathcal{A}_{nv}$ and computes an optimal, smooth path using the heuristic from Equation 5. This approach effectively minimizes revisitations and coverage overlap. Once the agent reaches $p_{nearest}$, the wavefront-driven strategy resumes. The complete integration is formalized in Algorithm 1.

Algorithm 1 – Wavefront and Bidirectional A* coverage path planning.

map, Start p_{start}, Goal p_{goal};
Path $\mathcal{P}$;
Step 1: Convert *map* to a graph representation G;
Step 2: Wavefront Propagation
Initialize weights $w(p) = \infty$ for all cells $p \in G$;
Set $w(p_{\text{goal}}) = 0$;
Propagate wave from p_{goal} to p_{start} using Equation 2 Set $p_{\text{actual}} = p_{\text{start}}$;
Initialize the set of non-visited nodes $\mathcal{A}_{\text{nv}}$ with all navigable cells in G;
while $\mathcal{A}_{\text{nv}} \neq \emptyset$ do
    Step 3: Initial Path Planning
    Generate an initial path $\mathcal{P}_{\text{wave}}$ from p_{actual} to p_{goal} using Wavefront;
    if Wavefront fails or $\mathcal{P}_{\text{wave}}$ is suboptimal then
        Step 4: Search for Next Nearest Node
        Find the nearest unvisited node using Equation 4
        Step 5: Bidirectional A* Fallback
        Initialize two search frontiers: one at p_{actual} and another at p_{nearest};
        Expand both frontiers simultaneously until they meet;
        Combine paths from both sides to generate $\mathcal{P}_{\text{A}*}$;
        Mark cells in $\mathcal{P}_{\text{A}*}$ as visited;
        Update $p_{\text{actual}} = p_{\text{nearest}}$;
    end
    Remove visited nodes from $\mathcal{A}_{\text{nv}}$;
end
Step 6: Path Publishing
Publish $\mathcal{P}$ as a ROS Path message;

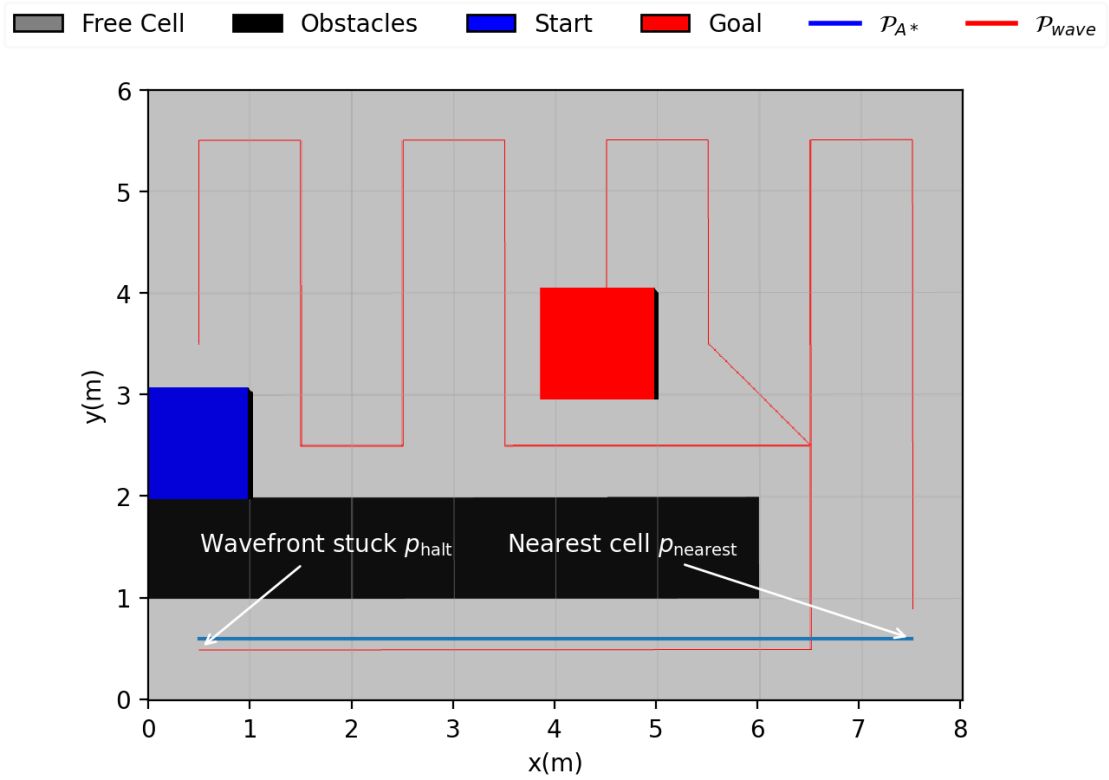


Figure 6 – Coverage path combining Wavefront and Bidirectional A* algorithms (Castro, G. G. R. de et al., 2025).

3.2.3 Role in the Hybrid Architecture

The global planner provides the hierarchical framework's deterministic backbone. It is responsible for ensuring full area coverage, assigning exploration regions consistently, and minimizing redundant traversal across robots. Furthermore, it structures the mission into coherent subsequences of waypoints and operates exclusively on the initial static map.

Whenever dynamic obstacles or unexpected changes are detected, the system transitions fluidly to the local dynamic planner, ensuring safe navigation while preserving global coverage guarantees.

Together, Wavefront and Bidirectional A^* provide a powerful hybrid global planner that is computationally efficient, provably complete, and robust, forming one of the core pillars of the cooperative methodology proposed in this thesis.

3.3- Local Dynamic Planning Layer

Although the global planner provides a deterministic coverage route derived from the initially known map, real deployments inevitably introduce discrepancies between the static representation and the perception of the robots' onboard sensors. Dynamic obstacles, partial occlusions, unmapped structures, and sensor uncertainty require a mechanism capable of reacting immediately to local disturbances. To address this, the proposed architecture incorporates a Local Dynamic Planning Layer that operates in parallel with the global coverage planner and ensures safe motion at an acceptable temporal resolution.

This layer adopts the hybrid DRL–RRT formulation introduced by the author in (Castro, G. G. de et al., 2023), where a Deep Reinforcement Learning (DRL) policy provides fast reactive behavior, while a Rapidly Exploring Random Tree (RRT) module acts as a corrective mechanism in challenging or highly cluttered configurations.

3.3.1 Rapidly-Exploring Random Tree Algorithm

The Rapidly-Exploring Random Tree (RRT) algorithm is a sampling-based motion planner that incrementally builds a search tree over a continuous state space. To maintain consistency with the global planner formulation, let the continuous search domain be denoted by $\mathcal{C} \subset \mathbb{R}^3$. Within this space, the occupied obstacle region is defined as $\mathcal{A}_{\text{obs}}$, and the collision-free region as $\mathcal{A}_{\text{free}}$, both representing continuous equivalents of the discrete grid $\mathcal{G}$.

The central objective is to construct a collision-free path $\mathcal{P}$ between a starting pose $p_{\text{start}} \in \mathcal{A}_{\text{free}}$ and a target pose $p_{\text{goal}} \in \mathcal{A}_{\text{free}}$, while minimizing traversal time and geometric length. The RRT algorithm expands a tree of feasible motions by repeatedly sampling points in $\mathcal{A}_{\text{free}}$, growing a structure that inherently biases its extension toward unexplored regions.

At each iteration, a random configuration, denoted as $p_{\text{rand}} \in \mathcal{A}_{\text{free}}$, is sampled according to a specified probability distribution over the free space. The nearest node currently in the tree, p_{near}, is determined using a metric such as the Euclidean distance.

The tree then attempts to expand toward p_{rand} by generating a new candidate node in that direction. If the straight-line segment connecting p_{near} to p_{rand} does not intersect the obstacle region $\mathcal{A}_{\text{obs}}$, the new node is added to the tree. Over time, this repeated sampling explores the state space with dense coverage, ensuring probabilistic completeness toward the target region.

Since RRT is a probabilistic algorithm, it does not guarantee optimality and may require significant computational time to find a path in highly cluttered regions. Thus, the implementation used in this work employs two stopping criteria: (i) a maximum number of iterations N, and (ii) a minimum distance threshold, J, between the current extension node and p_{goal}.

Algorithm 2 summarizes the procedure as implemented in this research. Figure 7 illustrates a standard RRT expansion in a 2D occupancy grid map.

Algorithm 2 – Rapidly-Exploring Random Tree (RRT) applied to local waypoints.

$\mathcal{P} \leftarrow \emptyset$;
for each segment $(p_{\text{start}}, p_{\text{goal}})$ in Waypoints do
  $\mathcal{T} \leftarrow \text{InitializeTree}(p_{\text{start}})$;
  for $i = 1$ to N do
    $p_{\text{rand}} = \text{Sample}(\mathcal{A}_{\text{free}})$;
    $p_{\text{near}} \leftarrow \text{NearestNode}(\mathcal{T}, p_{\text{rand}})$;
    if $\text{CollisionFree}(p_{\text{near}}, p_{\text{rand}}, \mathcal{A}_{\text{obs}})$ then
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{p_{\text{rand}}\}$;
      $\text{AddEdge}(\mathcal{T}, p_{\text{near}}, p_{\text{rand}})$;
    end
    if $\|p_{\text{rand}} - p_{\text{goal}}\| \leq J$ then
      $\mathcal{P} \leftarrow \mathcal{P} \cup \text{ExtractPath}(\mathcal{T}, p_{\text{start}}, p_{\text{rand}})$;
      Break;
    end
  end
end

3.3.2 Deep Reinforcement Learning and Deep Q-Network

The Deep Q-Network (DQN) algorithm is adopted due to its ability to combine neural networks with the Q-learning framework, enabling the approximation of the action–

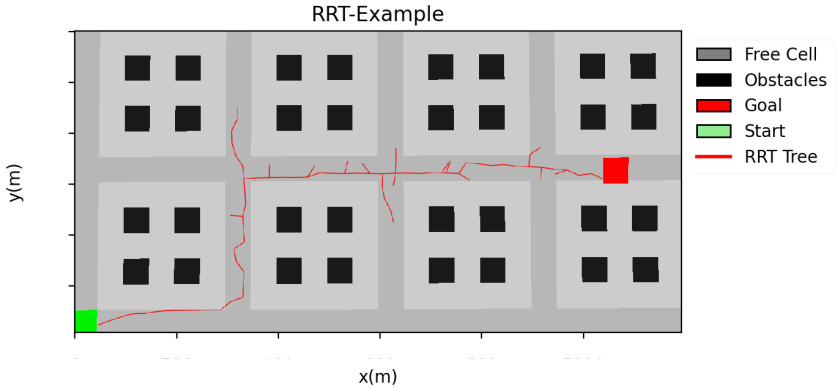


Figure 7 – RRT path generation implemented over an occupancy 2D grid.

value function. DQN addresses classical challenges in reinforcement learning, such as high-dimensional state spaces and unstable learning, by incorporating experience replay and neural network-based target estimation (Yang; Juntao; Lingling, 2020b).

Similar to other deep learning approaches, the training process involves three stages: (1) training, (2) validation, and (3) performance testing. In the proposed navigation system, the DQN is responsible for dynamically adapting the UAV or UGV trajectory whenever the agents' sensors detect discrepancies in the prior map, i.e., non-mapped obstacles appear along the previously planned route.

In this context, the navigation problem is formulated as a Markov Decision Process (MDP), in which the agent receives sensor information as the current state and must select an action that maximizes long-term expected reward. DRL-based navigation is advantageous because it requires less dependence on precise sensor readings and can directly associate raw perceptual inputs with collision-free actions (Zhang et al., 2019).

The proposed method consists of two sequential stages. First, the RRT algorithm runs and find a non-optimal feasible path between two points in $\mathcal{A}_{\text{free}}$. In the second step, the DQN policy continuously adjusts the trajectory, avoiding collisions. If at any moment the current path becomes unsafe, the DRL avoidance controller is activated to prevent the agent from entering a collision course with static or dynamic obstacles.

Figure 8 illustrates the methodology. In scenario A, the agent begins by exploring the environment using RRT. Once the candidate routes are found, the DQN refines the motion and adapts to new environmental changes, as shown in scenario B.

The objective of this methodology is to implement a lightweight system with rapid convergence and sufficient reactivity to operate in highly dynamic environments, while

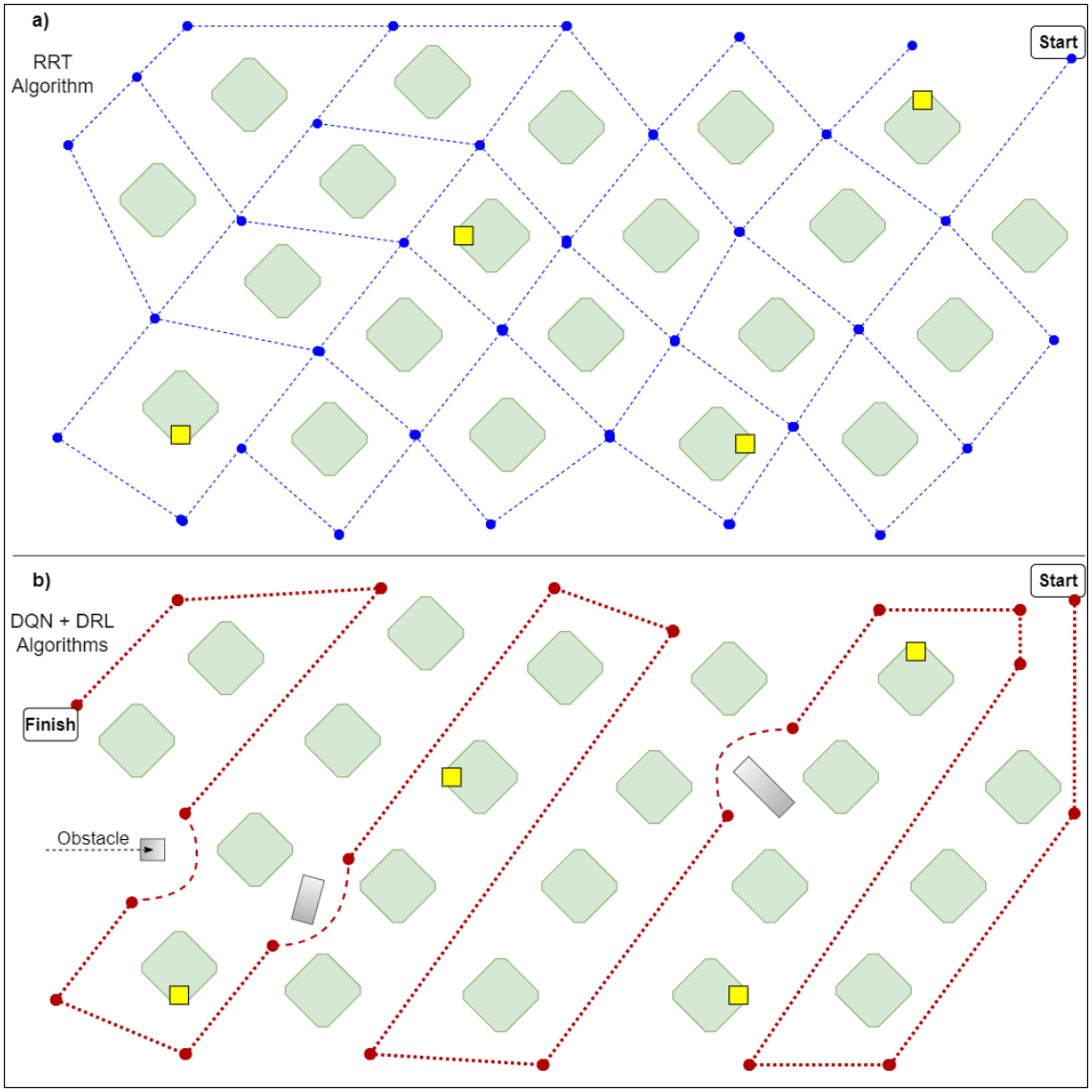


Figure 8 – Overall idea of the DRL–RRT methodology proposed in (Castro, G. G. d. et al., 2023). (a) The agent starts trajectory planning using RRT to explore multiple path options. (b) Once RRT determines feasible routes, the DQN refines the motion and adapts it in real time as the environment changes.

maintaining short and safe trajectories.

3.3.3 Training Environment Setup

Properly training the DQN is crucial for the robustness of the overall system. Although reinforcement learning provides strong generalization abilities, training a Deep Q-Network is notoriously challenging. Unlike supervised learning, where the desired output is known, DRL agents must discover successful behaviors through trial and error. This introduces several training difficulties: (i) reward sparsity, where meaningful feedback occurs only when approaching the goal or avoiding a collision; (ii) instability caused by temporal correlations in the agent's experience; (iii) slow convergence due to high-dimensional state-action spaces; and (iv) the high computational cost of simulating realistic physics while keeping the learning loop fast enough. For these reasons, the training phase is typically the most time-consuming stage in DRL-based navigation systems and requires careful environment design to accelerate experience collection.

To tackle these challenges, this research developed two tailored simulation environments in Gazebo (Koenig; Howard, 2004). The first is a simplified scenario used for training the DQN, reducing physics computation to accelerate learning. The second resembles an olive-farm environment and is used to validate the trained model under realistic conditions (Figures 9–24).

The simplified training world contains only 22 models, enabling faster simulation steps. The pickup vehicle (Figure 9b) moves within the free region $\mathcal{A}_{\text{free}}$ with random acceleration and velocity vectors defined as:

$$\begin{aligned}
&\text{Acceleration direction:} && \hat{P}_a = \begin{bmatrix} \cos\Theta \\ \sin\Theta \end{bmatrix}, && \Theta \sim \mathcal{U}(0,\pi), \\
&\text{Velocity direction:} && \hat{P}_v = \begin{bmatrix} \cos\Phi \\ \sin\Phi \end{bmatrix}, && \Phi \sim \mathcal{U}(0,\pi), \\
&\text{Acceleration magnitude:} && \|P_a\| = \sqrt{X_1^2 + Y_1^2}, \\
&\text{Velocity magnitude:} && \|P_v\| = \sqrt{X_2^2 + Y_2^2},
\end{aligned} \tag{7}$$

where $X_1, Y_1, X_2, Y_2 \sim \mathcal{U}(-c, c)$ are independently sampled scalars from a uniform distribution bounded by the vehicle's dynamic limits c. These dynamic variations create

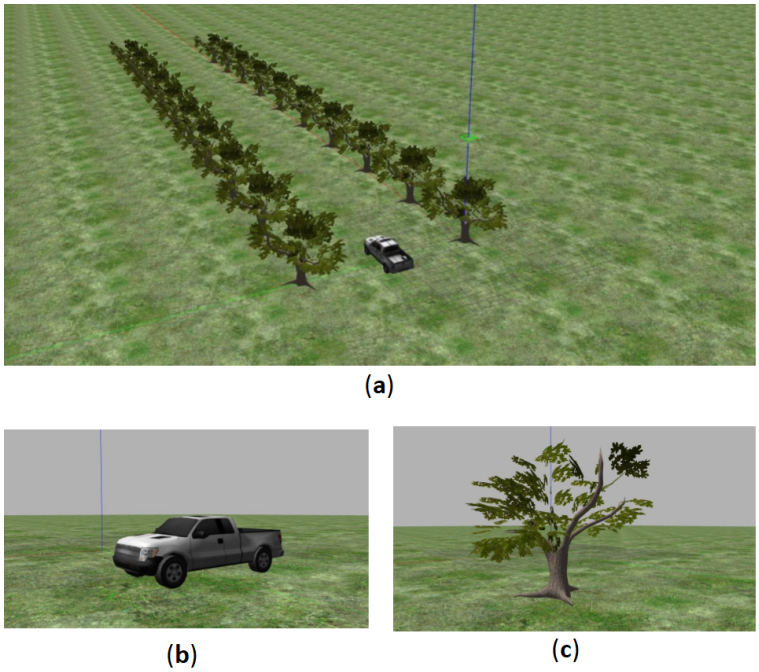


Figure 9 – Gazebo training environment for the DQN agent (Castro, G. G. d. et al., 2023). (a) Simplified world for accelerating DQN training. (b) Pickup model used as a dynamic obstacle. (c) Olive-tree models used as static obstacles.

non-deterministic scenarios that improve policy generalization and expose the DQN to diverse obstacle interactions.

In the simplified simulation, the continuous environment is discretized into a grid $\mathcal{G}$ with $1\,\mathrm{m}^3$ volumetric cells. The UAV (or UGV) occupies a single grid cell, while obstacles may fill multiple cells. To maintain mathematical consistency with the global planner, the set of occupied cells is denoted by $\mathcal{A}_{\text{obs}}$.

The state space is defined as a vector:

$$S_{\text{state}} = \begin{bmatrix} O_{\text{local}} \\ D_{\alpha} \\ \Delta p \end{bmatrix}^{T}, \tag{8}$$

where O_{local} encodes the occupancy of the surrounding grid cells obtained from onboard sensors, D_{α} is the angular deviation between the robot's heading and the next waypoint generated by the RRT module, p_{robot} represents the current instantaneous position of the vehicle and $\Delta p = p_{\text{goal}} - p_{\text{robot}}$ denotes the relative position of the current target waypoint expressed in the robot-centered reference frame.

The DQN outputs the next action to be executed, corresponding to the discrete motion commands available to each platform. For the UAV, the action space contains eight possible moves (representing an 8-connected grid), while for the UGV it contains four moves (4-connected grid). Although the DQN provides a Q-value for each possible action, a safety filter is applied before execution: any action that would lead the agent into an occupied cell in $\mathcal{A}_{\text{obs}}$ is suppressed, ensuring that unsafe maneuvers have zero probability of being selected. This guarantees that the learned policy remains compliant with local obstacle constraints even under uncertainty or partial observability.

Algorithm 3 summarizes the training process of the DQN.

Algorithm 3 – DQN training and experience collection process.

$E_{\max} \leftarrow$ maximum number of episodes;
$S_{\max} \leftarrow$ maximum steps per episode;
Initialize obstacle set $\mathcal{A}_{\text{obs}}$ using `InitializeObjs(N)`;
Initialize DQN model and experience replay memory $\mathcal{M}$;
for $e = 1$ to $E_{\max}$ do
    $p_{\text{robot}} \leftarrow$ `SampleFreePose`$(\mathcal{A}_{\text{free}})$;
    $p_{\text{goal}} \leftarrow$ `SampleFreePose`$(\mathcal{A}_{\text{free}})$;
    for $s = 1$ to $S_{\max}$ do
        $O_{\text{local}} \leftarrow$ `GetOccupancyReadings`();
        $S_{\text{state}} \leftarrow$ `DefineState`$(O_{\text{local}}, D_\alpha, \Delta p)$;
        $Action \leftarrow$ DQN.output(S_{state});
        $p_{\text{new}} \leftarrow p_{\text{robot}} + Action$;
        $Reward \leftarrow$ `GetReward`$(S_{\text{state}}, p_{\text{new}})$;
        `StoreExperience`$(\mathcal{M}, S_{\text{state}}, Action, Reward, S_{\text{next}})$;
        $p_{\text{robot}} \leftarrow p_{\text{new}}$;
        if size$(\mathcal{M})$ > batch_threshold then
            `SampleBatchAndTrain`$(\mathcal{M})$;
        end
    end
end
Save trained DQN model;

The associated functions in the training loop are defined as follows:

- `InitializeObjs(N)`: Randomly selects N free grid cells and marks them as occupied in $\mathcal{A}_{\text{obs}}$.
- `SampleFreePose(`$\mathcal{A}_{\text{free}}$`)`: Generates valid random initial and goal positions ensuring $p \in \mathcal{A}_{\text{free}}$.
- `GetOccupancyReadings()`: Returns the local grid occupancy matrix O_{local} (1 if occupied, 0 if free).

- `DefineState()`: Concatenates the observation vectors into the state space array S_{state}.
- `GetReward()`: Computes the scalar reward based on the distance to p_{goal} and the collision risk penalty.
- `StoreExperience()`: Saves the transition tuple into the replay buffer $\mathcal{M}$ for off-policy learning.

3.4- Coordination and Task Reallocation Layer

The cooperative execution of the global and local planners described in Sections 3.3 and 3.4 requires a supervisory mechanism capable of synchronizing the heterogeneous agents, monitoring mission progress, and reallocating tasks when necessary. This layer follows the hierarchical cooperation model presented in (Castro, G. G. R. de et al., 2024), in which UAVs provide high-level situational awareness and rapid map updates, while the UGV performs refined ground-level coverage and structural inspection.

The Coordination Layer operates asynchronously and interacts with both planning modules through well-defined communication messages. Its main responsibilities are: (i) monitoring regional coverage progress by updating the set of unvisited cells $\mathcal{A}_{\text{nv}}$, (ii) synchronizing local occupancy grid updates O_{local} among agents, (iii) detecting inconsistencies between the initially known grid $\mathcal{G}$ and the updated local observations, and (iv) triggering task redistribution or global replan requests when required.

Task redistribution follows the rules defined in (Castro, G. G. R. de et al., 2024). A reallocation request is triggered whenever any of the following conditions is met:

1. Detection of new obstacles: Updated sensor data indicates the presence of dynamic or previously unmapped objects, resulting in an update to the obstacle set $\mathcal{A}_{\text{obs}}$ that invalidates part of the global planned route $\mathcal{P}_{\text{global}}$.
2. Communication failure: Loss of communication between agents or between any agent and the ground station.
3. UAV battery constraint: When a UAV reaches the predefined low-battery threshold,

it initiates a return-to-UGV maneuver. The second UAV immediately takes over the remaining coverage from the point where the first UAV stopped, while the low-battery UAV proceeds to the UGV for a battery hot-swap.

Algorithm 4 presents the pseudocode governing the full execution pipeline, demonstrating the data flow between the variables formalized in the previous sections.

Algorithm 4 – Coordination and execution pipeline for the hybrid UAV-UGV framework.

Input: Initial discrete grid $\mathcal{G}$, robot team $\mathcal{R} = \{R_{\text{UGV}}, R_{\text{UAV}_1}, R_{\text{UAV}_2}\}$;
Output: Full coverage path execution;
Initialize: Inflate obstacles into $\mathcal{A}_{\text{obs}}$, compute initial free space $\mathcal{A}_{\text{free}}$, and set unvisited cells $\mathcal{A}_{\text{nv}}$;
Global Planning:
$\mathcal{P}_{\text{global}} \leftarrow \text{Wavefront}(\mathcal{A}_{\text{free}}) + \text{Bidirectional } A^{\star}(\mathcal{G})$;
$\mathcal{W} \leftarrow \text{ExtractWaypoints}(\mathcal{P}_{\text{global}})$;
foreach waypoint $p_{\text{goal}} \in \mathcal{W}$ do
    Local Planning:
    if CollisionRisk($p_{\text{robot}}, \mathcal{A}_{\text{obs}}$) or Stuck() then
        $\mathcal{P}_{\text{local}} \leftarrow \text{RRT-Recovery}(p_{\text{robot}}, p_{\text{goal}}, \mathcal{A}_{\text{free}})$;
    else
        $Action \leftarrow \text{DQN-Policy}(S_{\text{state}})$;
    end
    Execute($Action$ or $\mathcal{P}_{\text{local}}$);
    Coordination Updates:
    UpdateVisitedCells($\mathcal{A}_{\text{nv}}$);
    SyncOccupancyGrid(O_{local});
    if MapChanged($\mathcal{A}_{\text{obs}}$) or AgentFailure() or RegionBlocked() then
        Reallocation or Replanning:
        ReassignTasks($\mathcal{R}$);
        if GlobalMapInvalidated() then
            $\mathcal{P}_{\text{global}} \leftarrow \text{ReplanGlobalRoute}(\mathcal{A}_{\text{free}}, \mathcal{A}_{\text{nv}})$;
            $\mathcal{W} \leftarrow \text{ExtractWaypoints}(\mathcal{P}_{\text{global}})$;
        end
    end
end

3.5- Ground Inspection

In the context of coverage path planning for heterogeneous robot teams, it's essential to establish a clear mission initiation protocol. Typically, the UGV takes the lead by

initiating the mission. As it sets out on its path, the UGV's first task is to ensure the path ahead is clear and free of obstacles. If the UGV is not currently engaged in hot swapping (i.e., the seamless battery replacement or recharging mechanism), it proceeds with the mission. This mission involves systematically traversing the environment, collecting data, and contributing to the coverage task. This aligns with the coordinated efforts of the heterogeneous robot team. This approach reflects a coordinated workflow in which the UGV assumes initial responsibility, assesses the path's feasibility, and, when conditions permit, initiates the mission. Systematic coordination and path verification are essential to ensuring a successful, efficient coverage path-planning strategy in dynamic and challenging environments. Figure 10 summarizes this process.

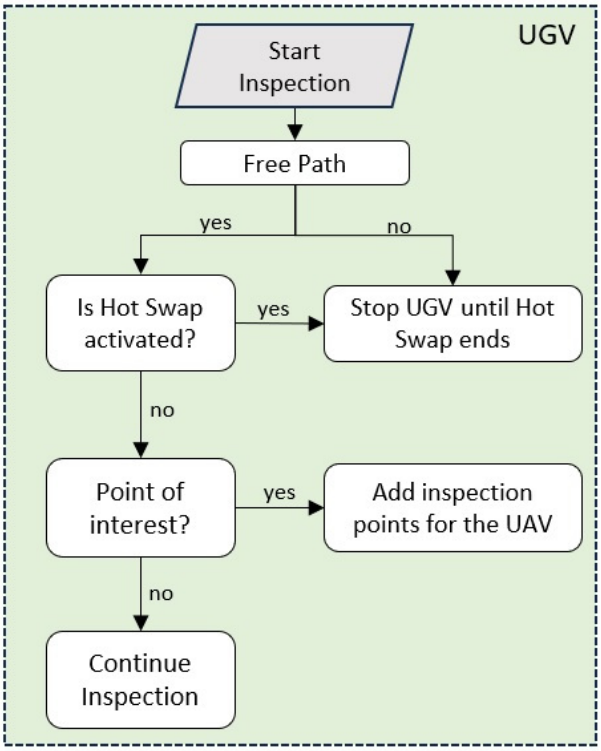


Figure 10 – UGV mission initiation and verification (Castro, G. G. R. de et al., 2024).

3.5.1 Aerial Inspection

In this specific scenario, the multi-robot collaboration involves a team of two UAVs and one UGV. As previously mentioned, the UGV takes on the initial path assessment and mission initiation role, providing the conditions for the UAVs. Additionally, it assumes responsibility for data collection and updating the map. Furthermore, if one of the UAVs experiences a low battery, the UGV facilitates hot swapping.

Both UAVs are primarily tasked with aerial inspections. However, their operations must be carefully orchestrated to ensure coordination and prevent redundant visits to the same Points of Interest (PoI). It's worth noting that if the UGV encounters a PoI during its ground-level mission, it can request the UAV currently on patrol to adjust its routine and investigate the identified PoI. This collaborative approach ensures efficient coverage and thorough inspection of the target area. Figures 11 and 12 illustrate the routines of UAV 1 and UAV 2, respectively.

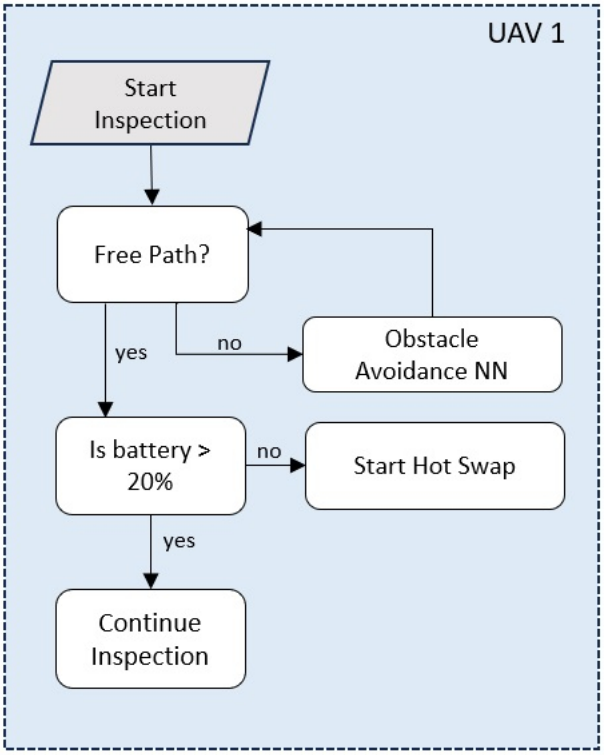


Figure 11 – Routine of UAV 1 (Castro, G. G. R. de et al., 2024).

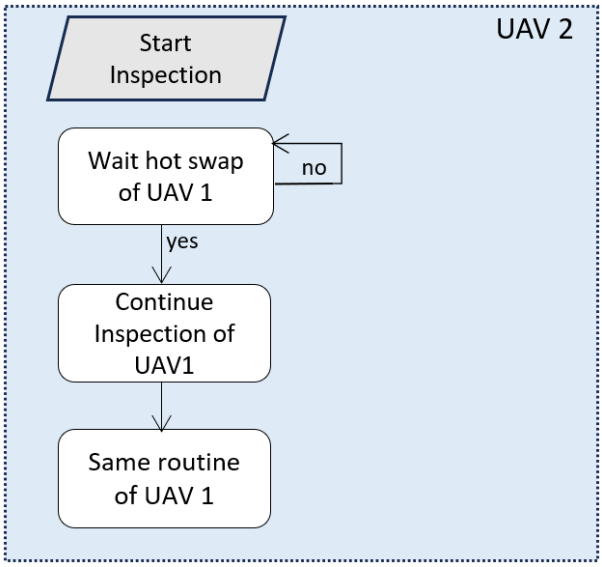


Figure 12 – Routine of UAV 2 (Castro, G. G. R. de et al., 2024).

Part II

3.6- UAV quadrotor model

The dynamics of the UAV are provided by the PX4 package and are detailed and described in Andrade et al. (Andrade et al., 2021). The UAV system considered in this work has four rotors, as presented in Figure 13.

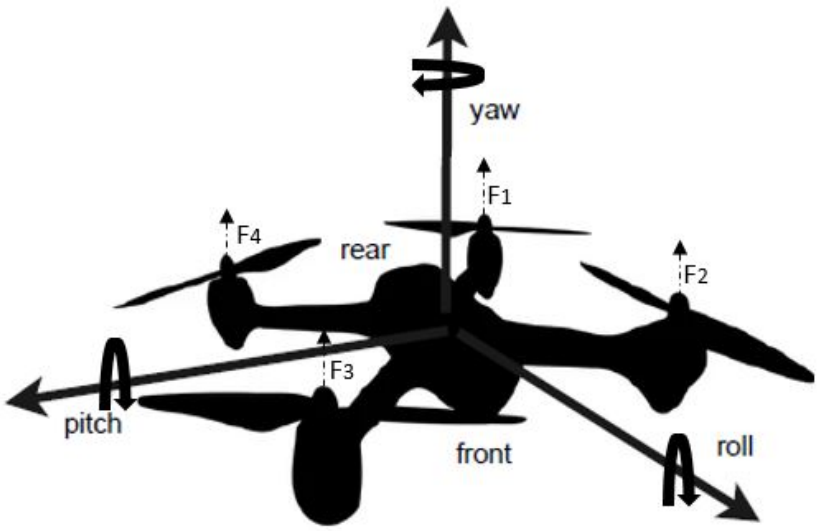


Figure 13 – UAV quadrotor structure (Castro, G. G. d. et al., 2023).

The Euler angles presented in this figure describe the orientation of the system about the local level surface, as well as about the true north (azimuth reference). The roll (ϕ) and pitch (θ) angles give the tilt of the UAV with respect to the gravity vector, while the yaw angle (ψ) describes the rotation around the local vertical direction. These angles, as shown in Equation 9, are used as inputs on the forward kinematics equation to calculate the world-to-frame conversion matrix to estimate the actual UAV position,

$$R_{\psi,\theta,\phi} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}. \quad (9)$$

Equation 10 gives the UAV body frame axes a mathematical model,

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \\ \ddot{z} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{zz}}\dot{\theta}\dot{\psi} + \theta\frac{J_r}{I_{xx}}\Omega_r + \frac{l}{I_{xx}}U_2 \\ \frac{I_{zz}-I_{zz}}{I_{yy}}\dot{\phi}\dot{\psi} + \phi\frac{J_r}{I_{yy}}\Omega_r + \frac{l}{I_{yy}}U_3 \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{l}{I_{zz}}U_4 \\ g - \frac{\cos(\phi)\cos(\theta)U_1}{m} \\ \frac{U_1}{m}(\Delta + \sin(\phi)\sin(\psi)) \\ \frac{U_1}{m}(\Gamma + \sin(\phi)\cos(\psi)) \end{bmatrix} = \begin{bmatrix} \xi_\phi(t) \\ \xi_\theta(t) \\ \xi_\psi(t) \\ \xi_h(t) \\ 0 \\ 0 \end{bmatrix}. \tag{10}$$

Note that $\Delta = \cos(\phi)\sin(\theta)\cos(\psi)$ and $\Gamma = \cos(\phi)\sin(\theta)\sin(\psi)$ are auxiliary variables, $U_i|(i = 1, \cdots, 4)$ represents the thrust and the actuation torques for the UAV movements, Ω_r is the engines residual angular speed, and finally, $\xi_{(\phi,\theta,\psi,h)}$ are the dynamic disturbances affecting the angles ϕ, θ, and ψ. Note that these variables are used in the Motor Mixing (i.e., Equation 10), which specifies the magnitude of forces that should be applied to each motor of your multirotor. U_1 is the thrust equivalence force, U_2 represents a rotational force on roll axis, U_3 represents a rotational force on pitch axis, and U_4 represents a rotational force on yaw axis. Thus, it is possible, through the Motor Mixing Algorithm, to analyze the U_i inputs in relation to angular speed ($\Omega_{(1,2,3,4)}$). The $b(Ns^2)$ and in Equation 11 are the Impulse coefficient and the Drag coefficient, respectively,

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ \Omega_r \end{bmatrix} = \begin{bmatrix} b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ b(-\Omega_2^2 + \Omega_4^2) \\ b(\Omega_1^2 - \Omega_3^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{bmatrix}. \tag{11}$$

3.7- UGV model

UGVs come in a wide variety of models, each with its own skills. Due to its greater flexibility in congested environments (West; Asada, 1997) and its ability to change its motion without changing orientation, an omnidirectional four-mecanum wheeled UGV model was implemented during experiments through this work. The Mecanum wheels, Figure 14, are special wheel designs that are based on a concept that achieves traction in

one direction and allows passive motion in another (Tlale; Villiers, 2008). Figure 15 shows some of the UGV variables that will later compose the UGV vector state.

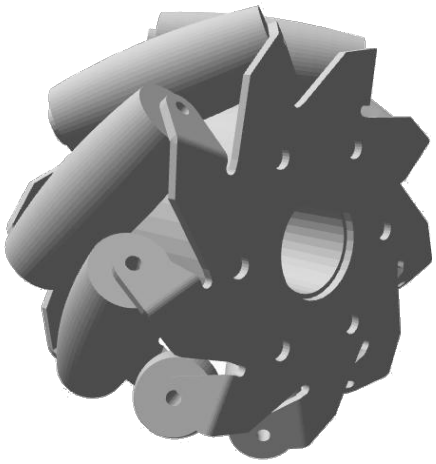


Figure 14 – UGV's mecanum wheel model.

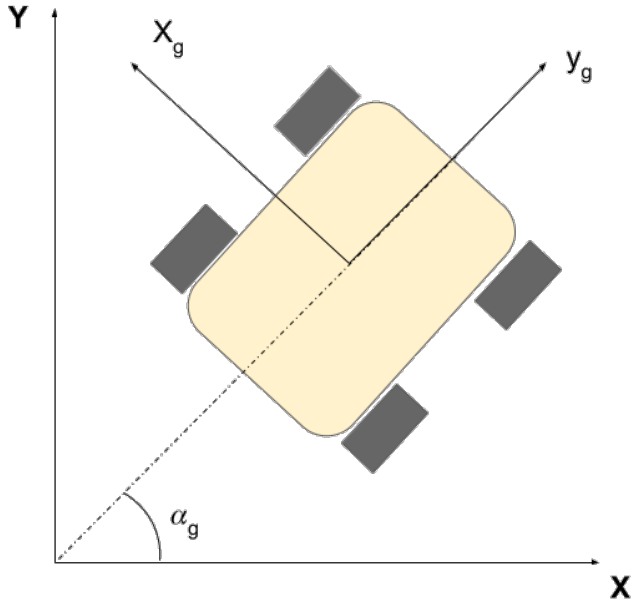


Figure 15 – Four-mecanum wheeled UGV state description.

3.8- Problem Description

Let the physical workspace be defined as $\mathcal{W} \subset \mathbb{R}^3$. To maintain consistency with the planning architecture, the obstacle region is denoted by $\mathcal{A}_{\text{obs}} \subset \mathcal{W}$, and the collision-free space by $\mathcal{A}_{\text{free}} \subset \mathcal{W}$.

The heterogeneous multi-robot system operates within a continuous state space $\mathcal{X}$, where the state of each agent is defined by its pose, local sensor readings, and available energy. For any given agent, the state vector is structured as:

$$s = (p,\, \theta,\, O_{\text{local}},\, E),$$

where $p = (x, y, z) \in \mathbb{R}^3$ is the spatial position, $\theta \in [-\pi, \pi)$ is the heading angle, $O_{\text{local}} \in \mathbb{R}^k$ represents the onboard sensor readings (as defined in the local planning layer), and $E \in [0, 100]$ indicates the remaining battery energy percentage.

We define the following specific subsets and spaces for the team:

- $\mathcal{C}_{\text{UAV}} \subset \mathcal{A}_{\text{free}}$: the specific 3D region to be covered by the aerial agents;
- $\mathcal{C}_{\text{UGV}} \subset \mathcal{A}_{\text{free}}$: the 2D planar region to be covered by the ground vehicle;
- $\mathcal{X}_{\text{UAV}_1}, \mathcal{X}_{\text{UAV}_2} \subset \mathcal{X}$: the individual state spaces of the two UAVs;
- $\mathcal{X}_{\text{UGV}} \subset \mathcal{X}$: the individual state space of the UGV.

The coverage regions are mathematically obtained by restricting the obstacle-free space $\mathcal{A}_{\text{free}}$ to the operational altitudes of the respective platforms:

$$\mathcal{C}_{\text{UAV}} = \{p \in \mathcal{A}_{\text{free}} \mid z = z_{\text{inspect}}\},$$

$$\mathcal{C}_{\text{UGV}} = \{p \in \mathcal{A}_{\text{free}} \mid z = 0\}.$$

Each agent must generate a continuous coverage path $\mathcal{P}$ visiting all states within its designated region. The aerial and ground paths are defined as sequences of waypoints:

$$\mathcal{P}_{\text{UAV}} = \{p_1, p_2, \ldots, p_N\}, \qquad p_i \in \mathcal{C}_{\text{UAV}},$$

$$\mathcal{P}_{\text{UGV}} = \{p_{g1}, p_{g2}, \ldots, p_{gM}\}, \qquad p_{gi} \in \mathcal{C}_{\text{UGV}}.$$

Let J_{UAV} and J_{UGV} be the cost functions encoding overlap penalties, maneuvering costs, and energy constraints. The multi-agent optimization problem seeks the optimal paths $\mathcal{P}^{\star}_{\text{UAV}}$ and $\mathcal{P}^{\star}_{\text{UGV}}$:

$$\mathcal{P}^{\star}_{\text{UAV}} = \arg\min_{\mathcal{P}_{\text{UAV}}} \sum_{i=1}^{N} J_{\text{UAV}}(p_i, E_{\text{UAV}_1}, E_{\text{UAV}_2}),$$

$$\mathcal{P}^{\star}_{\text{UGV}} = \arg\min_{\mathcal{P}_{\text{UGV}}} \sum_{i=1}^{M} J_{\text{UGV}}(p_{gi}),$$

3.9- Path Planning Algorithms

This section presents the path-planning algorithms considered in this work. Some of these methods are introduced as baseline approaches for validation and comparison, while others constitute key components of the proposed hybrid coverage architecture. The objective is to provide a clear conceptual and mathematical foundation for each algorithm, enabling a structured understanding of how they contribute to global coverage, local navigation, or dynamic obstacle avoidance within the multi-robot system.

3.9.1 A*

The $A^{\star}$ algorithm is an efficient path-finding and graph traversal algorithm. Developed by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968 (Hart; Nilsson; Raphael, 1968), it elegantly balances the dual objectives of path cost minimization and goal proximity. It accomplishes this by employing a heuristic approach that estimates the cost to reach the goal from each node, guiding the search toward the most promising paths. This characteristic makes $A^{\star}$ particularly suitable for applications in robotics, where optimizing route selection is essential for operational efficiency and mission success.

The $A^{\star}$ algorithm pseudocode presented in Algorithm 5 outlines the method for finding the optimal collision-free path between a start point p_{start} and a target point

p_{goal} within the discrete occupancy grid $\mathcal{G}$. It initializes by placing p_{start} into a priority queue Q_{open} (the open set) and maintaining an initially empty Q_{closed} (the closed set) to track evaluated cells. The algorithm iteratively processes the current cell p_{curr} with the lowest estimated total cost $f(p)$ until p_{goal} is reached. For each cell, it examines its navigable neighbors $p_{\text{adj}} \in \mathcal{A}_{\text{free}}$, updating their accumulated costs $g(p)$ based on the distance from the start, plus an estimated heuristic distance $h(p)$ to the goal. This process is repeated, expanding paths based on their promise indicated by $f(p)$, until the target is found returning the optimal sequence of waypoints $\mathcal{P}_{A^\star}$ or until Q_{open} is emptied, indicating that no valid path exists. Figure 16 illustrates the execution of the $A^\star$ algorithm within a discretized 2D environment. Panel (a) presents the occupancy grid $\mathcal{G}$, where the light blue cells represent the obstacle region $\mathcal{A}_{\text{obs}}$ and the white cells denote the navigable free space $\mathcal{A}_{\text{free}}$. The green and red markers indicate the starting cell p_{start} and the target goal p_{goal}, respectively. Panel (b) demonstrates the optimal, collision-free path $\mathcal{P}_{A^\star}$ (dashed black line) computed by the algorithm. By continuously evaluating the heuristic cost, the search effectively navigates around the complex obstacle formations, ensuring the shortest geometric route without intersecting $\mathcal{A}_{\text{obs}}$.

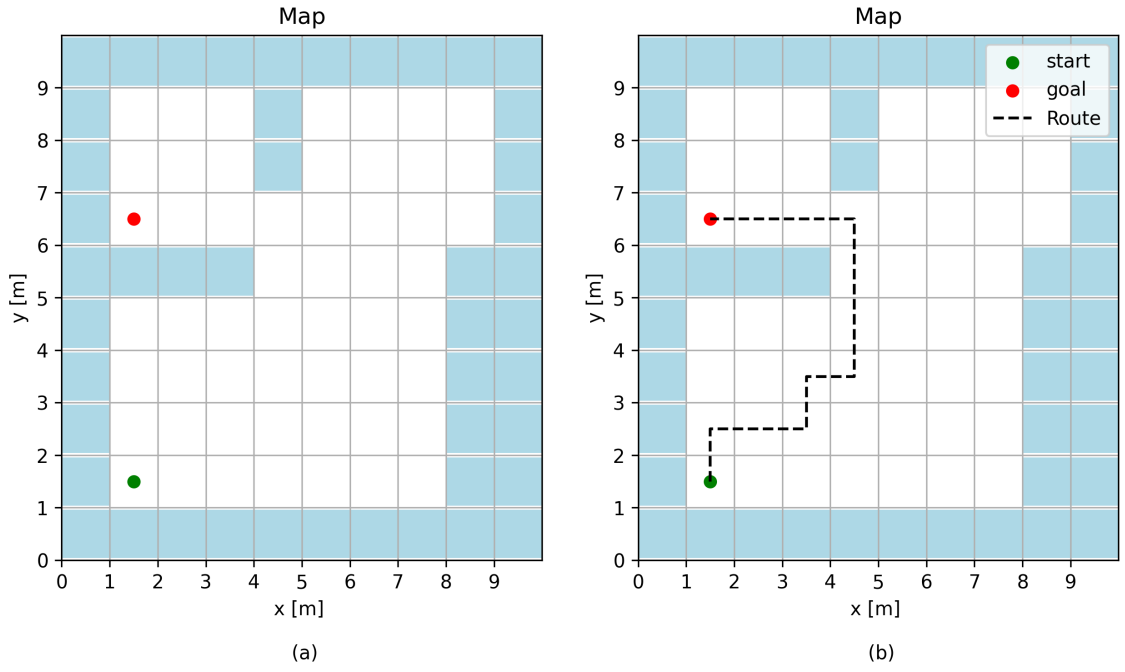


Figure 16 – $A^\star$ pathfinding algorithm evaluated in a 10×10 occupancy grid. (a) Environment representation, highlighting the free space $\mathcal{A}_{\text{free}}$ (white) and obstacles $\mathcal{A}_{\text{obs}}$ (light blue), with start and goal poses marked in green and red. (b) The resulting optimal trajectory $\mathcal{P}_{A^\star}$ (dashed line) successfully avoiding local minima.

Algorithm 5 – $A^\star$ algorithm applied to the discrete free space.

Input: Occupancy grid $\mathcal{G}$, Start cell p_{start}, Goal cell p_{goal}
Output: Path $\mathcal{P}_{A^\star}$ (or "No path found")
$Q_{\text{open}} \leftarrow$ priority queue containing only p_{start};
$Q_{\text{closed}} \leftarrow \emptyset$;
$g(p_{\text{start}}) \leftarrow 0$;
$f(p_{\text{start}}) \leftarrow h(p_{\text{start}}, p_{\text{goal}})$;
while $Q_{\text{open}} \neq \emptyset$ do
    $p_{\text{curr}} \leftarrow$ extract cell with lowest $f(p)$ from Q_{open};
    if $p_{\text{curr}} = p_{\text{goal}}$ then
        return `reconstructPath`(p_{curr}) ; `/* Path found */`
    Q_{open}.remove(p_{curr});
    Q_{closed}.add(p_{curr});
    for each valid neighbor $p_{\text{adj}} \in \text{neighbors}(p_{\text{curr}}) \cap \mathcal{A}_{\text{free}}$ do
        if $p_{\text{adj}} \in Q_{\text{closed}}$ then
            continue;
            ; `/* Skip explored neighbors */`
        $g_{\text{tentative}} \leftarrow g(p_{\text{curr}}) + \text{cost}(p_{\text{curr}}, p_{\text{adj}})$;
        if $p_{\text{adj}} \notin Q_{\text{open}}$ or $g_{\text{tentative}} < g(p_{\text{adj}})$ then
            p_{adj}.parent $\leftarrow p_{\text{curr}}$;
            $g(p_{\text{adj}}) \leftarrow g_{\text{tentative}}$;
            $f(p_{\text{adj}}) \leftarrow g(p_{\text{adj}}) + h(p_{\text{adj}}, p_{\text{goal}})$;
            if $p_{\text{adj}} \notin Q_{\text{open}}$ then
                Q_{open}.add(p_{adj});
            else
                Q_{open}.updatePriority(p_{adj});
                ; `/* Update priority */`
return "No path found" ;

Function *`reconstructPath`*(p_{curr})
    $\mathcal{P}_{A^\star} \leftarrow \emptyset$;
    while p_{curr} is not None do
        $\mathcal{P}_{A^\star}$.append(p_{curr});
        $p_{\text{curr}} \leftarrow p_{\text{curr}}$.parent;
    $\mathcal{P}_{A^\star}$.reverse();
    ; `/* Start → goal order */`
    return $\mathcal{P}_{A^\star}$;

3.9.2 Bidirectional $A^\star$

Bidirectional $A^\star$ is a graph-search algorithm that extends the classical $A^\star$ planning strategy by performing two simultaneous searches: one starting from the initial cell toward

the target, and another starting from the target toward the initial cell. The two expanding search frontiers eventually meet, significantly reducing the explored area when compared to the unidirectional version.

In the context of this work, Bidirectional $A^\star$ is incorporated as a support mechanism to the Wavefront global planner (Section 3.2). Whenever the Wavefront propagation becomes trapped due to obstacle-induced dead ends or disconnectivity in the free-space region at a cell p_{halt}, Bidirectional $A^\star$ provides a bridging path to the nearest unvisited cell p_{nearest}. This ensures that the global coverage route remains complete, connected, and deterministic, even in maps with narrow passages or compartmentalized structures.

The algorithm operates over the discrete grid representation $\mathcal{G}$, using two priority queues Q_{start} and Q_{goal}, each guided by its own cost function:

$$f(p) = g(p) + h(p),$$

where $g(p)$ is the accumulated cost from the respective starting point and $h(p)$ is the heuristic estimate, normally Manhattan or Euclidean distance, to the opposite target.

Figure 17 illustrates the execution of the Bidirectional $A^\star$ algorithm to resolve a local minimum within a discretized 2D environment. Panel (a) displays the occupancy grid $\mathcal{G}$, distinguishing the obstacle region $\mathcal{A}_{\text{obs}}$ (light blue cells) from the navigable free space $\mathcal{A}_{\text{free}}$ (white cells). Panel (b) details the bridging process. The red square represents the cell where the global planner halted, p_{halt}, while the green square indicates the closest unvisited target cell, p_{nearest}. The algorithm expands simultaneously from both ends: the red dashed line traces the forward search frontier managed by Q_{start}, and the green dashed line traces the backward search frontier managed by Q_{goal}. The optimal, collision-free bridging path $\mathcal{P}_{A^\star}$ is finalized at the exact point where these two frontiers intersect, successfully bypassing the central obstacle formation while exploring a fraction of the state space required by a traditional unidirectional search.

Algorithm 6 summarizes the complete Bidirectional $A^\star$ search procedure customized for the hybrid global planner.Algorithm 6 details the step-by-step execution of this bridging mechanism. The procedure begins by initializing two independent priority queues, Q_{start} and Q_{goal}, seeded with the deadlock cell p_{halt} and the target cell p_{nearest}, respectively. Two corresponding closed sets, C_{start} and C_{goal}, are kept empty to track evaluated cells.

The algorithm then enters an iterative loop, alternating between forward and back-

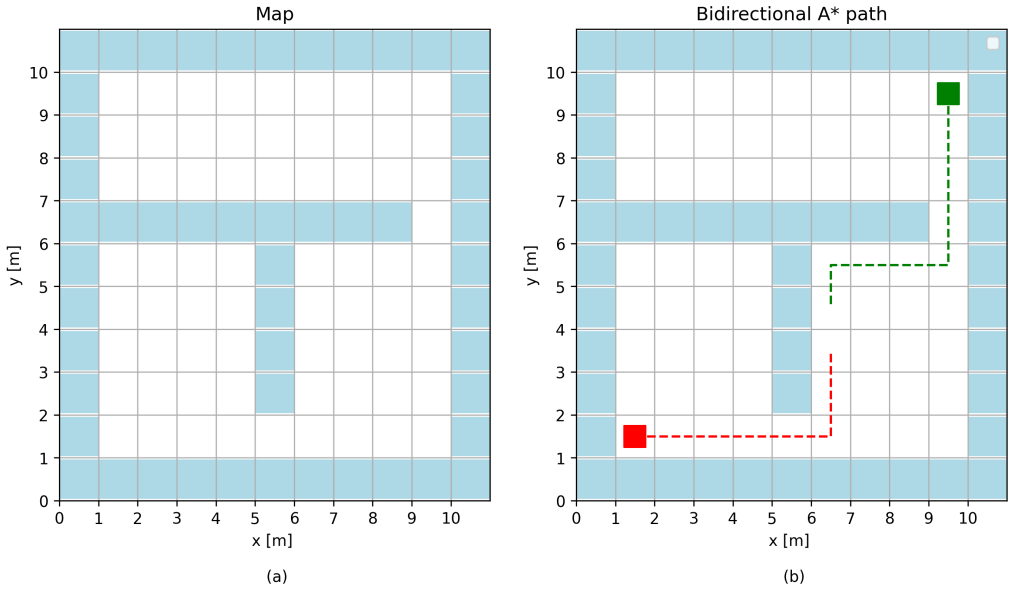


Figure 17 – Bidirectional $A^\star$ pathfinding algorithm implemented in a 2D environment. (a) Grid map segmentation highlighting free space $\mathcal{A}_{\text{free}}$ (white) and obstacles $\mathcal{A}_{\text{obs}}$ (light blue). (b) Optimal bridging path generated by the meeting of the forward frontier Q_{start} (red dashed line) and the backward frontier Q_{goal} (green dashed line).

ward expansions. In each phase, the cell with the lowest total heuristic cost $f(p)$ is extracted from its respective queue. For every extracted cell, the algorithm explores its valid, collision-free neighbors $p_{\text{adj}} \in \mathcal{A}_{\text{free}}$. If a cheaper route to a neighbor is discovered, its accumulated cost $g(p)$ is updated, and the neighbor is inserted into the queue.

The critical termination condition occurs when the two search frontiers intersect—specifically, when a cell extracted from one queue has already been evaluated and placed into the opposite closed set (for instance, if $p_{\text{curr, start}} \in C_{\text{goal}}$). At this exact intersection point, the algorithm halts the expansion and smoothly merges the parent pointers from both sides. This ensures the output is a continuous, optimal bridging path $\mathcal{P}_{A^\star}$ connecting the previously isolated regions.

Bidirectional $A^\star$ reduces search time by approximately a quadratic factor in many grid configurations, making it a suitable backup planner for global coverage operations where bridging between disconnected free-space components $\mathcal{A}_{\text{free}}$ is required.

Algorithm 6 – Bidirectional $A^\star$ Search Algorithm

Input: Start cell p_{halt}, Target cell p_{nearest}, Occupancy grid $\mathcal{G}$
Output: Bridging path $\mathcal{P}_{A^\star}$

$Q_{\text{start}} \leftarrow \{p_{\text{halt}}\}, \quad Q_{\text{goal}} \leftarrow \{p_{\text{nearest}}\}$;
$C_{\text{start}} \leftarrow \emptyset, \quad C_{\text{goal}} \leftarrow \emptyset$;
Initialize $g_{\text{start}}(p_{\text{halt}}) = 0$, $g_{\text{goal}}(p_{\text{nearest}}) = 0$;
Initialize priority queues using $f(p) = g(p) + h(p)$;
while $Q_{\text{start}} \neq \emptyset$ and $Q_{\text{goal}} \neq \emptyset$ do
    // Forward expansion from $p_{\texttt{halt}}$
    $p_{\text{curr, start}} \leftarrow \text{ExtractMin}(Q_{\text{start}})$;
    if $p_{\text{curr, start}} \in C_{\text{goal}}$ then
        return `MergePaths`$(p_{\text{curr, start}})$;
    end
    Move $p_{\text{curr, start}}$ to C_{start};
    foreach valid neighbor $p_{\text{adj}} \in \text{neighbors}(p_{\text{curr, start}}) \cap \mathcal{A}_{\text{free}}$ do
        $g_{\text{tentative}} \leftarrow g_{\text{start}}(p_{\text{curr, start}}) + \text{cost}(p_{\text{curr, start}}, p_{\text{adj}})$;
        if $p_{\text{adj}} \notin C_{\text{start}}$ and $g_{\text{tentative}} < g_{\text{start}}(p_{\text{adj}})$ then
            $p_{\text{adj}}.\text{parent}_{\text{start}} \leftarrow p_{\text{curr, start}}$;
            $g_{\text{start}}(p_{\text{adj}}) \leftarrow g_{\text{tentative}}$;
            `InsertOrUpdate`$(Q_{\text{start}}, p_{\text{adj}})$;
        end
    end
    // Backward expansion from $p_{\texttt{nearest}}$
    $p_{\text{curr, goal}} \leftarrow \text{ExtractMin}(Q_{\text{goal}})$;
    if $p_{\text{curr, goal}} \in C_{\text{start}}$ then
        return `MergePaths`$(p_{\text{curr, goal}})$;
    end
    Move $p_{\text{curr, goal}}$ to C_{goal};
    foreach valid neighbor $p_{\text{adj}} \in \text{neighbors}(p_{\text{curr, goal}}) \cap \mathcal{A}_{\text{free}}$ do
        $g_{\text{tentative}} \leftarrow g_{\text{goal}}(p_{\text{curr, goal}}) + \text{cost}(p_{\text{curr, goal}}, p_{\text{adj}})$;
        if $p_{\text{adj}} \notin C_{\text{goal}}$ and $g_{\text{tentative}} < g_{\text{goal}}(p_{\text{adj}})$ then
            $p_{\text{adj}}.\text{parent}_{\text{goal}} \leftarrow p_{\text{curr, goal}}$;
            $g_{\text{goal}}(p_{\text{adj}}) \leftarrow g_{\text{tentative}}$;
            `InsertOrUpdate`$(Q_{\text{goal}}, p_{\text{adj}})$;
        end
    end
end
return “No path found”;

3.9.3 Genetic Algorithm

Genetic Algorithm (GA) represents a class of evolutionary algorithms that mimic the process of natural selection, embodying the principle of ”survival of the fittest” to solve optimization and search problems. They generate solutions to optimization problems using

techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover (Holland, 1992). This approach is particularly effective in complex search spaces where traditional methods might struggle to find optimal solutions due to the problem's combinatorial nature or multiple local optima. In this thesis, the Genetic Algorithm (GA) is applied together with some path-planning algorithms to optimize their resulting trajectories. Figure 18 illustrates a comparison between a raw RRT solution (a) and an RRT trajectory optimized with GA (b).

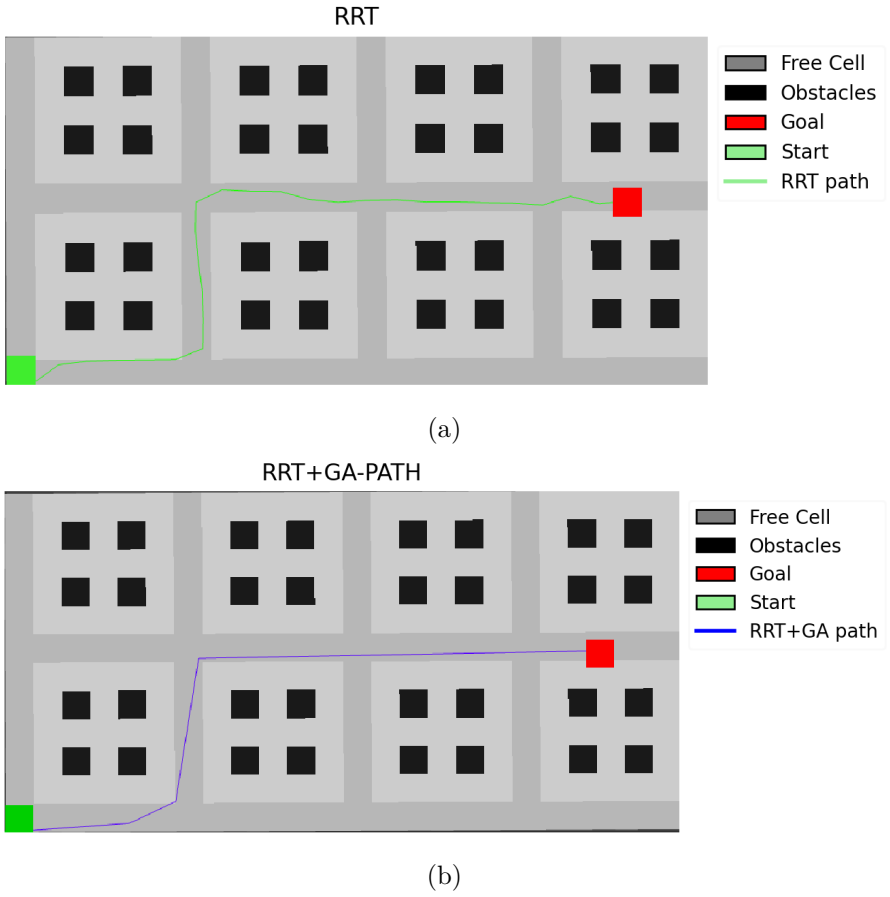


Figure 18 – Comparison between (a) standard RRT and (b) RRT optimized using a Genetic Algorithm.

3.10- Coverage Path Planning

3.10.1 Dijkstra-Based Geodesic Coverage

The Dijkstra algorithm is a classical graph-search method used to find the shortest path between nodes in a weighted graph. This characteristic makes it particularly useful in path planning for autonomous navigation, where the goal is to find the most efficient route between two points in a known environment. It operates by iteratively selecting the node with the minimum distance from the start node, updating the distances of its neighbors, and repeating the process until all reachable nodes have been processed (Wang; Yu; Yuan, 2011).

In the context of this thesis, the algorithm operates over the discrete occupancy grid $\mathcal{G}$, where each navigable cell $p \in \mathcal{A}_{\text{free}}$ corresponds to a node, and the weights represent edge traversal costs between adjacent cells.

Algorithm 7 formalizes the step-by-step computation of the geodesic distance field $D(p)$. The procedure initializes by assigning an infinite traversal cost to all navigable cells within the free space $\mathcal{A}_{\text{free}}$, with the exception of the source reference cell p_{ref}, which is set to zero. All free cells are initially placed into a priority queue Q_{open}.

During the main iterative phase, the algorithm continuously extracts the cell p_{curr} featuring the minimum accumulated distance from Q_{open}, transferring it to the evaluated set C_{closed}. For every valid, collision-free neighbor p_{adj} adjacent to p_{curr}, the algorithm computes a tentative path cost D_{alt}, representing the sum of the current cell's distance and the transition weight between them.

If this newly computed cost D_{alt} is strictly less than the neighbor's previously recorded distance $D(p_{\text{adj}})$, a relaxation step occurs: the neighbor's distance is updated, and its priority within Q_{open} is adjusted accordingly. This expansion systematically ripples outward until all reachable cells in $\mathcal{A}_{\text{free}}$ are processed, yielding a complete scalar field D that mathematically reflects the true topological cost of reaching any free space cell from p_{ref}, effectively bypassing local obstacles.

In this work, Dijkstra's algorithm is not directly employed as a single point-to-point coverage strategy. Instead, it is used to generate a geodesic scalar field D over the

Algorithm 7 – Dijkstra's Algorithm (Geodesic Distance Field Computation)

Input: Occupancy grid $\mathcal{G}$, Source cell p_{ref}
Output: Geodesic distance field D
Data: Q_{open}: Priority queue of unprocessed cells in $\mathcal{A}_{\text{free}}$
Data: C_{closed}: Set of evaluated cells ($\emptyset$ initially)
$D(p) \leftarrow \infty, \quad \forall p \in \mathcal{A}_{\text{free}}$;
$D(p_{\text{ref}}) \leftarrow 0$;
$Q_{\text{open}} \leftarrow \mathcal{A}_{\text{free}}$;
while $Q_{\text{open}} \neq \emptyset$ do
    $p_{\text{curr}} \leftarrow$ extract cell from Q_{open} with minimum $D(p)$;
    Move p_{curr} from Q_{open} to C_{closed};
    for each valid neighbor $p_{\text{adj}} \in \text{neighbors}(p_{\text{curr}}) \cap \mathcal{A}_{\text{free}}$ do
        $D_{\text{alt}} \leftarrow D(p_{\text{curr}}) + \text{cost}(p_{\text{curr}}, p_{\text{adj}})$;
        if $D_{\text{alt}} < D(p_{\text{adj}})$ then
            $D(p_{\text{adj}}) \leftarrow D_{\text{alt}}$;
            Update priority of p_{adj} in Q_{open};
return D;

grid map. This distance field represents the accumulated traversal cost from a reference point and serves as a global guidance mechanism for the coverage route $\mathcal{P}$.

The coverage path is then iteratively constructed using a greedy policy over the computed scalar field. At each step, the robot selects the adjacent cell $p_{\text{adj}} \in \mathcal{A}_{\text{nv}}$ (the set of unvisited free cells) that possesses the highest geodesic value, effectively performing a gradient ascent. After a cell is visited, it is removed from the set $\mathcal{A}_{\text{nv}}$. When the agent encounters a local minimum, a dead-end configuration p_{halt} with no unvisited neighbors available, Dijkstra's algorithm is re-invoked from the current position toward the nearest unvisited cell $p_{\text{nearest}} \in \mathcal{A}_{\text{nv}}$. The resulting shortest bridging path is appended to the main coverage trajectory $\mathcal{P}$, and the greedy exploration resumes.

Therefore, rather than applying classical Dijkstra for full coverage mapping, the proposed approach can be interpreted as an iterative geodesic replanning strategy guided by dynamic distance fields.

Figure 19 provides a detailed visual representation of the Dijkstra-based geodesic coverage strategy. Panel (a) illustrates the occupancy grid $\mathcal{G}$, distinguishing the obstacle region $\mathcal{A}_{\text{obs}}$ (light blue cells) from the navigable free space $\mathcal{A}_{\text{free}}$ (white cells). The cell marked with "S" serves as the source reference point p_{ref}. The numeric values within the free cells represent the computed geodesic distance field $D(p)$, demonstrating how the traversal cost correctly propagates outward from p_{ref} and naturally wraps around the central obstacle. Panel (b) depicts the final coverage trajectory $\mathcal{P}$ as a dashed black

line. By applying the greedy exploration policy over the $D(p)$ scalar field, the agent systematically sweeps the available free space, smoothly avoiding $\mathcal{A}_{\text{obs}}$ while ensuring that all navigable cells are visited without the need for continuous global replanning.

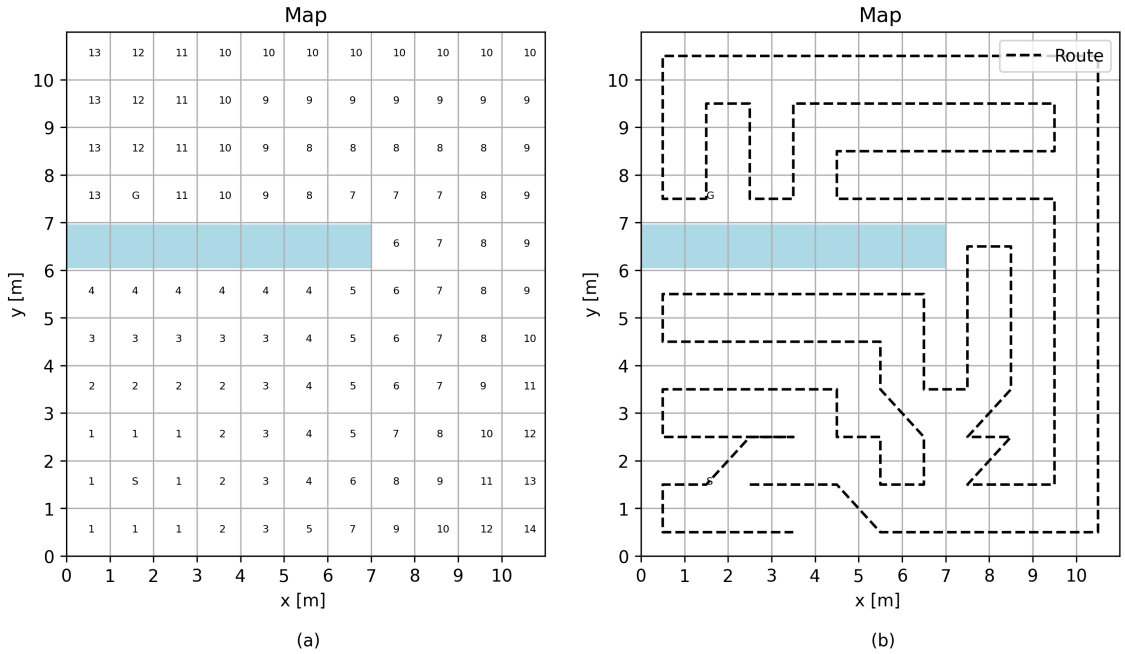


Figure 19 – Dijkstra-based geodesic coverage implemented on a 2D occupancy grid $\mathcal{G}$, where the marker "S" denotes the source reference cell p_{ref}. (a) The initial geodesic distance field $D(p)$ computed over the navigable free space $\mathcal{A}_{\text{free}}$. (b) The resulting coverage trajectory $\mathcal{P}$ generated by combining the distance field gradient with the greedy policy.

3.10.2 Flood Fill

The Flood Fill algorithm is a classical region-expansion method that operates by recursively or iteratively visiting all connected free cells starting from a seed location. Although originally conceived for image segmentation and region labeling, its grid-based propagation mechanism makes it suitable for coverage and exploration tasks in robotics.

In the context of this work, Flood Fill is used as a supporting analysis tool to validate the global coverage map and to verify the topological connectivity of the navigable free space $\mathcal{A}_{\text{free}}$. By propagating from the robot's initial position, the algorithm identifies reachable regions and detects enclosed or disconnected areas that may require special handling by the global planner. Flood Fill, therefore, plays a structural role in ensuring

that the coverage planner is aware of topological discontinuities in the workspace.

Given the discrete occupancy grid $\mathcal{G}$, where cells are partitioned into the free space $\mathcal{A}_{\text{free}}$ and the obstacle region $\mathcal{A}_{\text{obs}}$, Flood Fill operates by expanding a queue of frontier cells. Each visited cell is marked, and its valid, collision-free neighbors are added to the frontier until the reachable free region has been fully explored. Figure 20 provides a visual demonstration of the Flood Fill algorithm applied to a 2D occupancy grid $\mathcal{G}$. Panel (a) presents the initial environment, clearly distinguishing the structural boundaries formed by the obstacle region $\mathcal{A}_{\text{obs}}$ (light blue cells) from the available navigable free space $\mathcal{A}_{\text{free}}$ (white cells). Panel (b) illustrates the algorithm's execution adapted for Coverage Path Planning (CPP). The green square denotes the initial seed cell p_{seed}, while the red square indicates the final state of the sweep. The blue line traces the systematic, exhaustive expansion of the Flood Fill algorithm as it recursively visits every reachable cell within the connected component. This visual tracking confirms that the algorithm successfully maps the entire continuous region of $\mathcal{A}_{\text{free}}$ without intersecting $\mathcal{A}_{\text{obs}}$, providing a topological validation of the workspace before global planning operations.

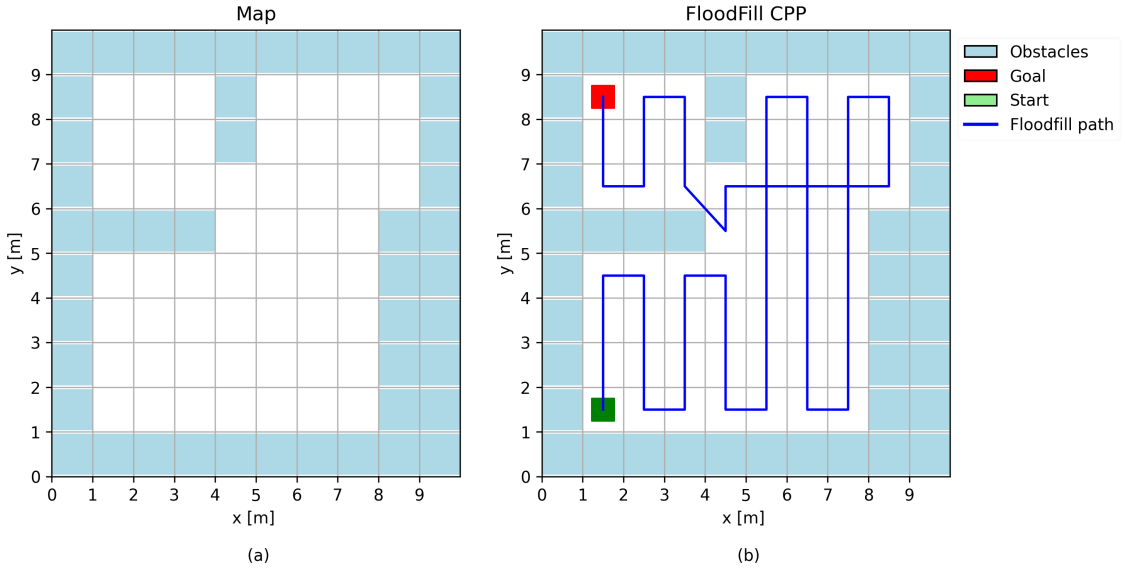


Figure 20 – Flood Fill expansion implemented over a 2D occupancy grid $\mathcal{G}$, demonstrating the identification of the reachable free space $\mathcal{A}_{\text{free}}$ from a given seed cell p_{seed}.

Algorithm 8 formalizes the iterative execution of the region expansion. The procedure initiates by defining an empty set $\mathcal{R}$ to store the topologically connected free cells, alongside a frontier queue Q_{front} initialized together with the starting coordinate p_{seed}.

During the main expansion loop, the algorithm continuously extracts a cell p_{curr} from the frontier queue. To ensure computational efficiency and prevent redundant eval-

uations, cells already present in $\mathcal{R}$ are immediately bypassed. Unvisited cells are then formally incorporated into the reachable region $\mathcal{R}$.

Subsequently, the algorithm inspects all adjacent neighbors p_{adj}. A neighbor is appended to the expansion queue Q_{front} if and only if it satisfies two conditions: it strictly belongs to the navigable free space ($p_{\text{adj}} \in \mathcal{A}_{\text{free}}$) and it has not been previously queued or visited ($p_{\text{adj}} \notin \mathcal{R}$). This systematic propagation exhausts all reachable configurations, ultimately yielding the maximal connected subset of $\mathcal{A}_{\text{free}}$ that is accessible from the initial seed, fully bounded by $\mathcal{A}_{\text{obs}}$.

Algorithm 8 – Flood Fill Region Expansion Algorithm

```
Input: Occupancy grid 𝒢, Seed cell p_seed
Output: Set of reachable free cells ℛ
ℛ ← ∅ ;                                          // Reachable region
Q_front ← Queue containing only p_seed;
while Q_front ≠ ∅ do
    p_curr ← Q_front.pop();
    if p_curr ∈ ℛ then
        continue;
    end
    Add p_curr to ℛ;
    foreach valid neighbor p_adj ∈ neighbors(p_curr) do
        if p_adj ∈ A_free and p_adj ∉ ℛ then
            Q_front.push(p_adj);
        end
    end
end
return ℛ;
```

Flood Fill guarantees complete exploration of the connected free-space component rooted at the initial seed. This makes it a valuable tool for structure verification, coverage consistency checks, and detecting isolated pockets of free cells that may require additional global replanning.

3.10.3 Wavefront

The Wavefront algorithm is a grid-based approach that propagates a distance wave from a target point p_{goal} throughout the navigable free space $\mathcal{A}_{\text{free}}$, applying a specific cost

function to each cell. This standard cost function, which can be a simple distance transform as described in Jarvis et al. (Jarvis; Byrne; Ajay, 1988), has the inherent drawback of producing trajectories with numerous sharp curves. This poses significant challenges for kinematic execution and odometry estimation during real robot operation. To address this issue, a modified cost function incorporating a penalty variable γ is introduced.

Algorithm 9 formalizes the step-by-step mechanics of this wave propagation and subsequent trajectory extraction. The procedure strategically initiates the search in reverse, placing the target destination p_{goal} into the primary expansion queue Q_{open}. Concurrently, the cost map $C(p)$ is initialized to zero across the free space, and an empty dictionary, $Parent$, is instantiated to track the optimal routing history of each cell.

During the expansive phase, the algorithm iteratively extracts a cell p_{curr} from the queue and formally incorporates it into the evaluated set $\mathcal{V}$. For every unvisited, collision-free neighbor $p_{\text{adj}} \in \mathcal{A}_{\text{free}}$, the algorithm computes its transition cost. In the standard approach, this is a simple uniform increment; however, when applying the modified path transform, this step dynamically incorporates the obstacle penalty $J(p, \gamma)$. The neighbor is then queued, and its $Parent$ pointer is directed back to p_{curr}, establishing a continuous linked list pointing towards the origin of the wave.

Once the artificial potential field C fully saturates the navigable workspace, the algorithm transitions to the path reconstruction phase. Starting from the robot's actual initial position p_{start}, it performs a discrete gradient descent by continuously following the established $Parent$ pointers cell-by-cell until p_{goal} is reached.

Figure 21 provides a detailed visual representation of the standard Wavefront algorithm execution. Panel (a) illustrates the discrete occupancy grid $\mathcal{G}$, distinguishing the obstacle region $\mathcal{A}_{\text{obs}}$ (light blue cells) from the navigable free space $\mathcal{A}_{\text{free}}$ (white cells). The nodes marked with "S" and "G" correspond to the starting position p_{start} and the goal p_{goal}, respectively. The numeric values within the free cells depict the basic distance transform propagating outward from p_{goal}, assigning an increasing cost to each cell based purely on its topological distance from the target. Panel (b) reveals the resulting trajectory $\mathcal{P}_{\text{Wavefront}}$ (solid black line) generated by executing a gradient descent over this simple distance field. As visually evident, relying solely on the distance transform produces a trajectory characterized by high curvature and numerous sharp, 90-degree turns. This geometric inefficiency justifies the necessity of the modified path transform cost function detailed in Equation 12.

Algorithm 9 – Coverage Path Planning Wavefront Algorithm

Input: Occupancy grid $\mathcal{G}$, Start cell p_{start}, Goal cell p_{goal}
Output: Set of visited cells $\mathcal{V}$, Coverage distance map C, Optimal Path $\mathcal{P}_{\text{Wavefront}}$

$Q_{\text{open}} \leftarrow$ Queue containing only p_{goal}
$\mathcal{V} \leftarrow \emptyset$
$Parent \leftarrow$ Empty dictionary
$C(p) \leftarrow 0, \quad \forall p \in \mathcal{A}_{\text{free}}$
while $Q_{\text{open}} \neq \emptyset$ do
    $p_{\text{curr}} \leftarrow$ Extract cell from Q_{open}
    Add p_{curr} to $\mathcal{V}$
    for each valid neighbor $p_{\text{adj}} \in \text{neighbors}(p_{\text{curr}}) \cap \mathcal{A}_{\text{free}}$ do
        if $p_{\text{adj}} \notin \mathcal{V}$ and $p_{\text{adj}} \notin Q_{\text{open}}$ then
            $Q_{\text{open}}.\text{enqueue}(p_{\text{adj}})$
            $Parent[p_{\text{adj}}] \leftarrow p_{\text{curr}}$
            $C(p_{\text{adj}}) \leftarrow C(p_{\text{curr}}) + \text{cost}(p_{\text{curr}}, p_{\text{adj}})$; `/* Propagate wave cost */`

`/* Path reconstruction from start to goal */`
$\mathcal{P}_{\text{Wavefront}} \leftarrow \emptyset$
$p_{\text{step}} \leftarrow p_{\text{start}}$
while $p_{\text{step}} \neq p_{\text{goal}}$ do
    $\mathcal{P}_{\text{Wavefront}}.\text{append}(p_{\text{step}})$
    $p_{\text{step}} \leftarrow Parent[p_{\text{step}}]$
$\mathcal{P}_{\text{Wavefront}}.\text{append}(p_{\text{goal}})$
return $\mathcal{V}$, C, $\mathcal{P}_{\text{Wavefront}}$

Zelinsky (Zelinsky et al., 1993) applied a path transform approach, wherein instead of propagating solely the distance from the goal, the algorithm propagates a specialized cost function. This formulation takes advantage of a weighted sum between the distance and a discomfort factor associated with moving near obstacles. The path transform cost function J for each coordinate $p \in \mathcal{A}_{\text{free}}$ is described in Equation 12, where the discomfort factor can be adjusted to influence the trajectory's adventurous or conservative nature.

$$J(p, \gamma) = D(p) + \gamma \sum_{n=1}^{N} O_n(p), \tag{12}$$

where:

$J(p, \gamma)$ is the total path transform cost function at cell p;

$D(p)$ is the Manhattan distance from cell p to p_{goal};

N is the total number of mapped obstacles in $\mathcal{A}_{\text{obs}}$;

γ is the adjustable discomfort factor;

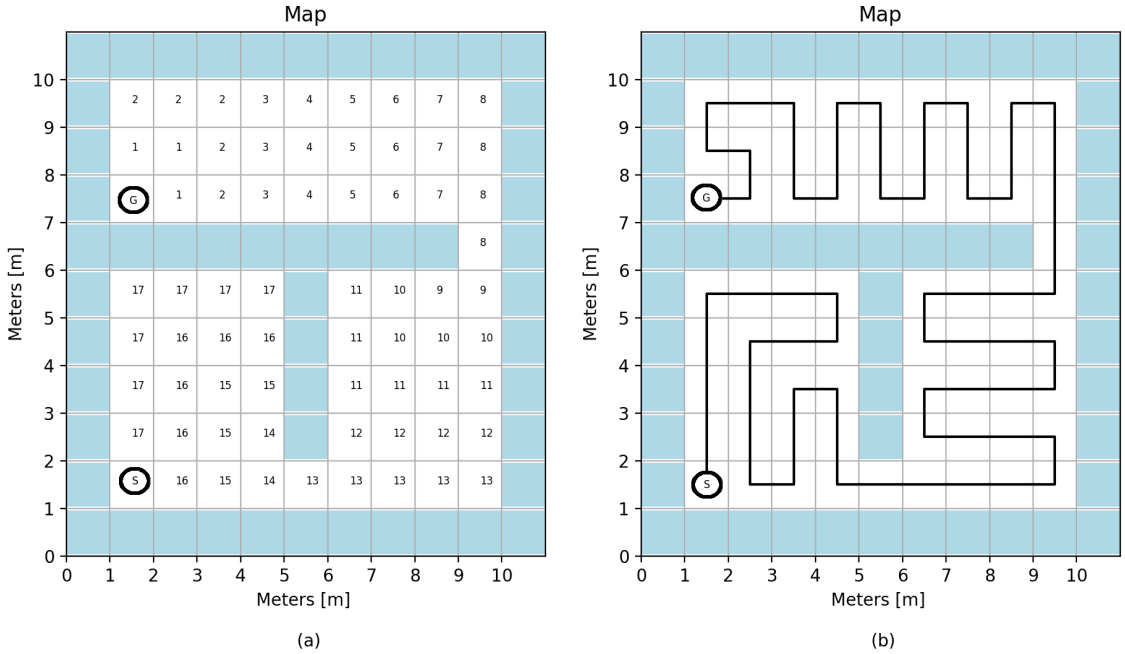


Figure 21 – Standard Wavefront algorithm execution over an occupancy grid $\mathcal{G}$. (a) Basic distance transform wave propagating from the goal p_{goal} (G) towards the start p_{start} (S) through the free space $\mathcal{A}_{\text{free}}$. (b) The resulting trajectory $\mathcal{P}_{\text{Wavefront}}$, highlighting the high curvature and numerous sharp turns generated by the unmodified cost function.

$O_n(p)$ is the obstacle penalty distance mapping the proximity of cell p to the n-th obstacle.

Figure 22 illustrates the practical impact of the modified path transform cost function on the exact same occupancy grid $\mathcal{G}$ previously analyzed. Panel (a) maintains the structural representation of the obstacle region $\mathcal{A}_{\text{obs}}$ (light blue) and free space $\mathcal{A}_{\text{free}}$ (white), along with the start p_{start} and goal p_{goal} markers. However, the numeric values within the cells now reflect the complete cost function $J(p, \gamma)$ formulated in Equation 12. By incorporating the obstacle penalty term, the propagation wave creates an artificial potential field that effectively pushes the cost gradient away from the structural boundaries. Panel (b) depicts the resulting trajectory $\mathcal{P}_{\text{Wavefront}}$. In contrast to the unmodified approach, the gradient descent over this reshaped cost landscape yields a significantly smoother path. The trajectory exhibits fewer sharp corners and naturally favors extended straight segments. This refined geometric routing is kinematically superior, facilitating more stable vehicle navigation, reducing control actuation effort, and minimizing odometry drift during real-world execution.

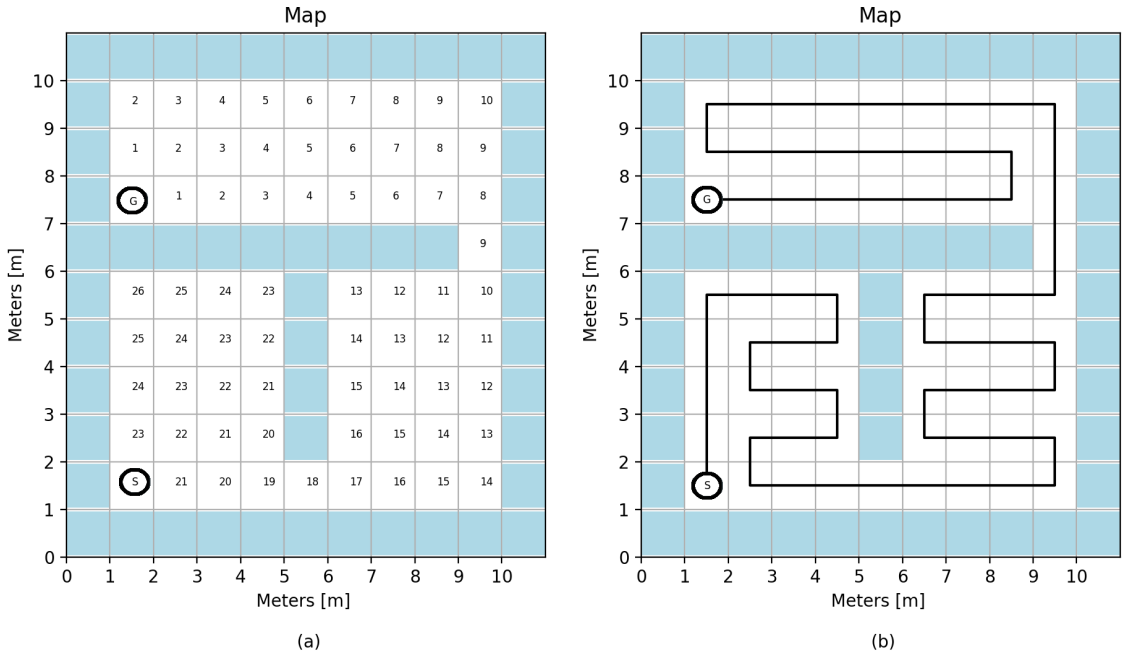


Figure 22 – Wavefront algorithm modified with the path transform cost function (Castro, G. G. R. de et al., 2024). (a) Path transform wave propagating from p_{goal}, incorporating obstacle penalties. (b) The resulting smoothed trajectory $\mathcal{P}_{\text{Wavefront}}$.

4- Results and Discussions

The results section is built upon the extensive empirical data and analysis derived from the author's prior studies (Castro, G. G. de et al., 2024, 2023). The empirical findings and analytical insights from these publications underpin the results discussed in this section, highlighting the efficacy and innovation of the hybrid planning approaches introduced.

4.1- Hardware and Environment Setup

The simulations were performed on a notebook equipped with 8GB of RAM and a 2.7 GHz Intel Core-i5-5200 processor. The operating system employed is a 64-bit Ubuntu 18.04 Bionic Beaver running the LXDE desktop environment. All algorithms were implemented using Python 3.10. The complete source code, ROS packages, and simulation dependencies are open-source and freely available at GitHub.

The aerial platform selected for these experiments is the IRIS quadrotor, depicted in Figure 23. This specific UAV model was chosen due to its native integration with the PX4 flight stack and its accurate dynamic modeling within the Gazebo software. Gazebo provides a complete physics engine that not only contributes to high-fidelity visual rendering but also allows the configuration of a dynamic interface, accurately simulating real-world physics such as gravity, wind disturbances, and inertial constraints.

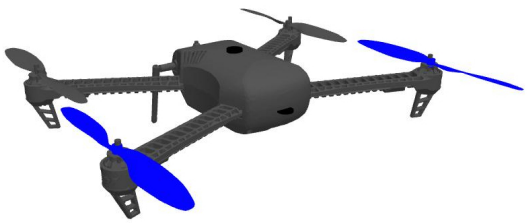


Figure 23 – IRIS UAV quadrotor model utilized in the Gazebo software simulation.

To evaluate the robustness of the proposed path planning architecture, two distinct simulation environments were designed in Gazebo. Figure 24 presents the first scenario: an olive-growing agricultural environment. This setting challenges the agent with an unstructured layout, featuring uneven terrain and scattered trees that act as cubic obstacles.

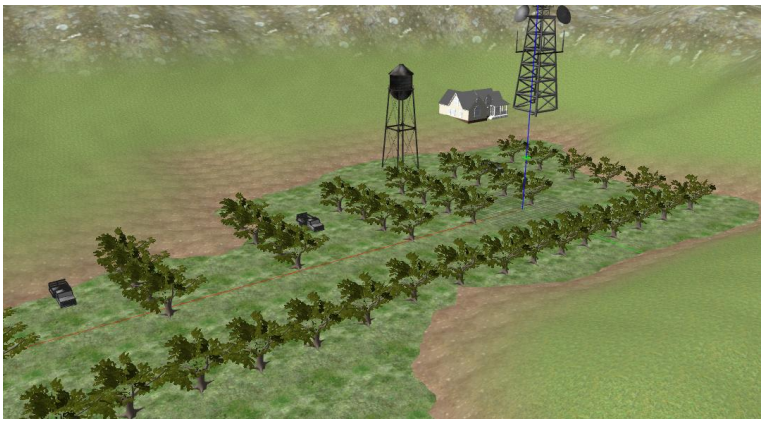


Figure 24 – Olive-growing simulation environment. The first test scenario featuring an unstructured layout with natural, sparsely distributed obstacles.

In contrast, Figure 25 displays the second scenario: a highly structured electrical substation. This industrial environment introduces complex geometric constraints, narrow operational corridors, and dense metallic structures. It serves as a rigorous evaluation environment, demanding high-precision maneuvers and strict collision-avoidance capabilities from both the global offline planner and the local DRL agent.

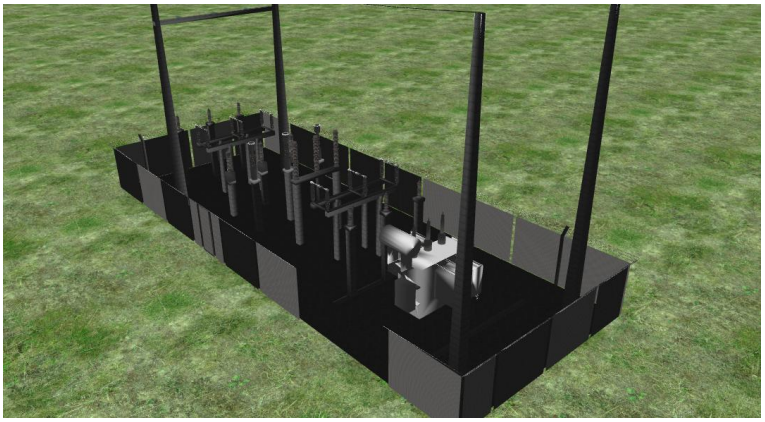


Figure 25 – Electrical Substation simulation environment. The second test scenario featuring a highly structured industrial topology with strict geometric constraints.

The UAV simulated in this work is equipped with a virtual PixHawk PX4 Flight Controller Unit (FCU) (Flight Controller Unit). The Micro Air Vehicle Link (MAVLink) communication protocol is utilized for telemetry streaming and offboard control routing.

ROS (Robot Operating System) Kinetic was used as the middleware to develop the DRL controller, the trajectory calculation modules, and the camera node packages in Python. The MAVLink stream was bridged to the ROS network through the MAVROS package.

Figure 26 provides a comprehensive schematic of the implemented software architecture, illustrating the closed-loop data flow between the simulation engine and the control algorithms. At the core, Gazebo 9.0 serves as the high-fidelity physics simulator, generating realistic sensor data and handling dynamic interactions. It communicates directly with the PX4 flight control stack, reproducing the UAV's hardware behavior. The crucial bridge between the low-level flight controller and high-level computation is established via MAVROS, which translates bidirectional MAVLink messages into standard ROS topics. Finally, the proposed DQN path planning algorithm operates as a distinct ROS node, processing incoming state observations from the middleware and publishing real-time velocity commands back through the chain to autonomously control the UAV.

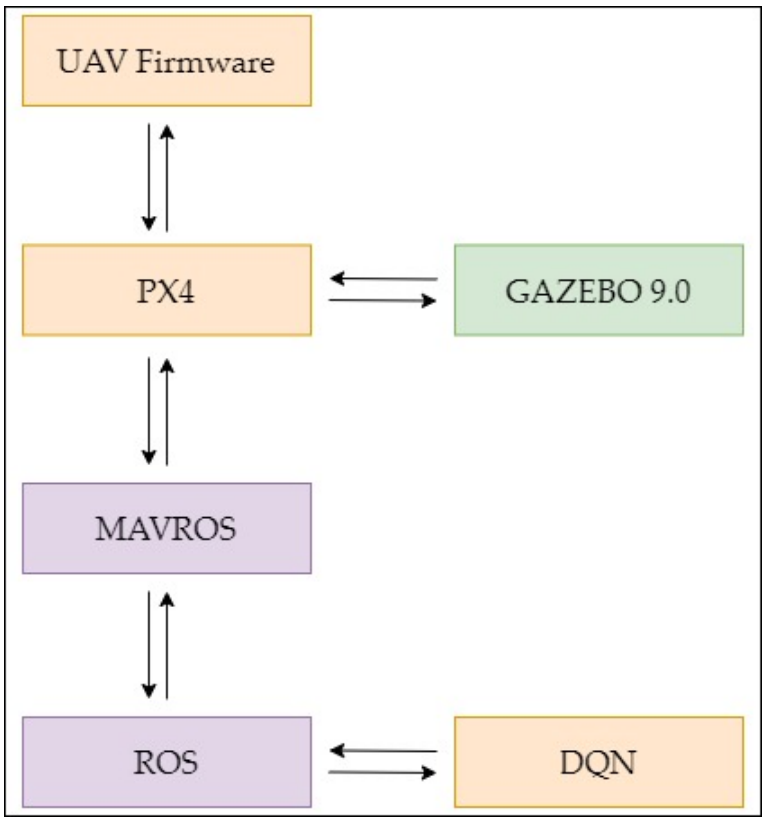


Figure 26 – Software framework overview. Gazebo simulates the physical environment and interacts with the UAV's FCU PX4. The DQN algorithm controls the agent in a closed-loop via ROS and MAVROS bridges (Castro, G. G. d. et al., 2023).

The results section is built upon the extensive empirical data and analysis derived from the author's studies (Castro, G. G. de et al., 2024) and (Castro, G. G. de et al.,

2023). The empirical findings and analytical insights from these publications underpin the results discussed in this section, highlighting the efficacy and innovation of the approaches introduced.

4.2- Integration

The interconnection and synchronization among the heterogeneous robotic systems (UAVs and UGV) were achieved using a robust ROS-based middleware framework. Figure 27 illustrates the overarching communication hierarchy, which is structurally partitioned into three primary interconnected modules.

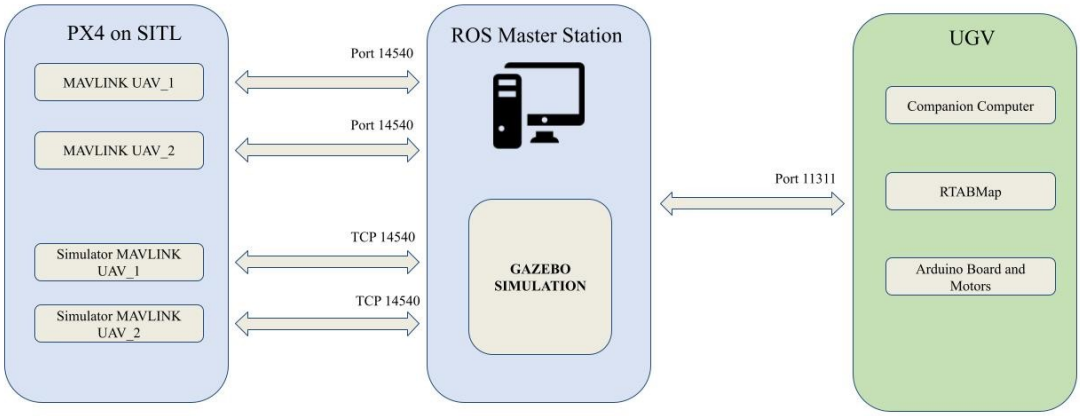


Figure 27 – Communication hierarchy of the robotic systems, detailing the network bridges between the SITL autopilots, the central ROS Master, and the UGV companion computer (Castro, G. G. de et al., 2024).

On the left, the PX4 on Software-in-the-loop (SITL) block represents the simulated flight controllers for the two UAVs. Within this Software-In-The-Loop environment, the PX4 autopilot firmware computes the low-level flight dynamics. The MAVLink protocol is heavily utilized here to establish a bidirectional communication bridge (via TCP/UDP ports, such as Port 14540) between the virtual autopilots and the high-level ground station. In the center, the ROS Master Station acts as the central coordinator. It hosts the Gazebo simulation engine and manages the parameter server and topic registry for the entire network. Finally, on the right, the UGV block represents the ground-based robot. It features a companion computer equipped with an onboard processing unit that handles

heavy computational tasks, such as real-time Simultaneous Localization and Mapping (SLAM) using the Real-Time Appearance-Based Mapping (RTAB-Map) package. This companion computer connects to the ROS Master (typically via Port 11311), ensuring that the ground agent remains synchronized with the global simulation state.

The ROS Master manages the registration and discovery of nodes within the system, integrating the communication among the subsystems inside each agent based on a publish-subscribe model. This service-oriented architecture allows individual software modules, known as Nodes, to exchange data streams through structures known as topics.

Figure 28 isolates the specific ROS node structure responsible for integrating the Deep Reinforcement Learning control loop. Three fundamental nodes are required to manage this operation: the /RRT_Mission_planner (which generates the global route), the /world_builder (which interfaces with the Gazebo environment), and the /Agent node (which hosts the DQN algorithm). As depicted by the directional arrows in the graph, the /Agent node operates in a closed continuous loop: it subscribes to the drone's spatial state via /mavros/local_position/pose and environmental data via /agent/lidar. After computing the optimal action, it immediately publishes continuous setpoints to /mavros/setpoint_accel/accel. This guarantees that the UAV actively avoids obstacles in real-time, communicating through messages published at a frequency of 1 kHz.

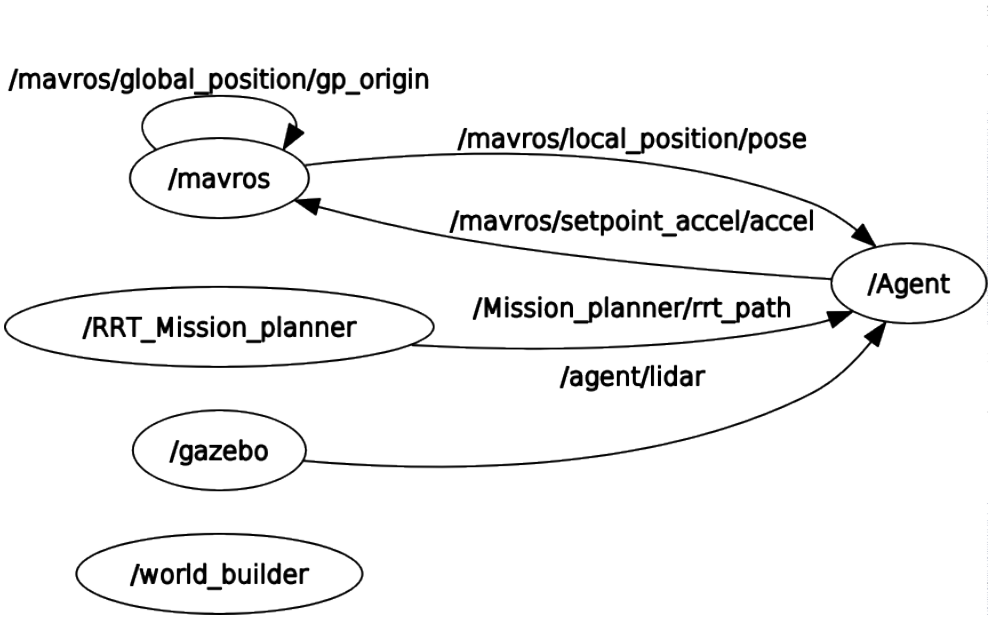


Figure 28 – ROS node structure isolating the Deep Reinforcement Learning closed-loop data flow (Castro, G. G. d. et al., 2023).

Scaling up to the full heterogeneous fleet, the complete active `rqt_graph` detailing the complex web of nodes and topics shared among all simulated robots has been divided to ensure optimal legibility. Figure 29 highlights the multi-UAV coordination. Both UAVs operate under their respective namespaces (/uav1 and /uav2), communicating with their simulated flight controllers via dedicated /mavros nodes. Crucially, the graph demonstrates that both drones subscribe to shared mission topics, such as /pt_interesse (points of interest) and /missao. This shared topology is a vital feature of the system: it guarantees that if one UAV is forced to halt or alter its current trajectory, the other agents dynamically update their awareness of the inspected coordinates, preventing coverage overlap and ensuring mission completion.

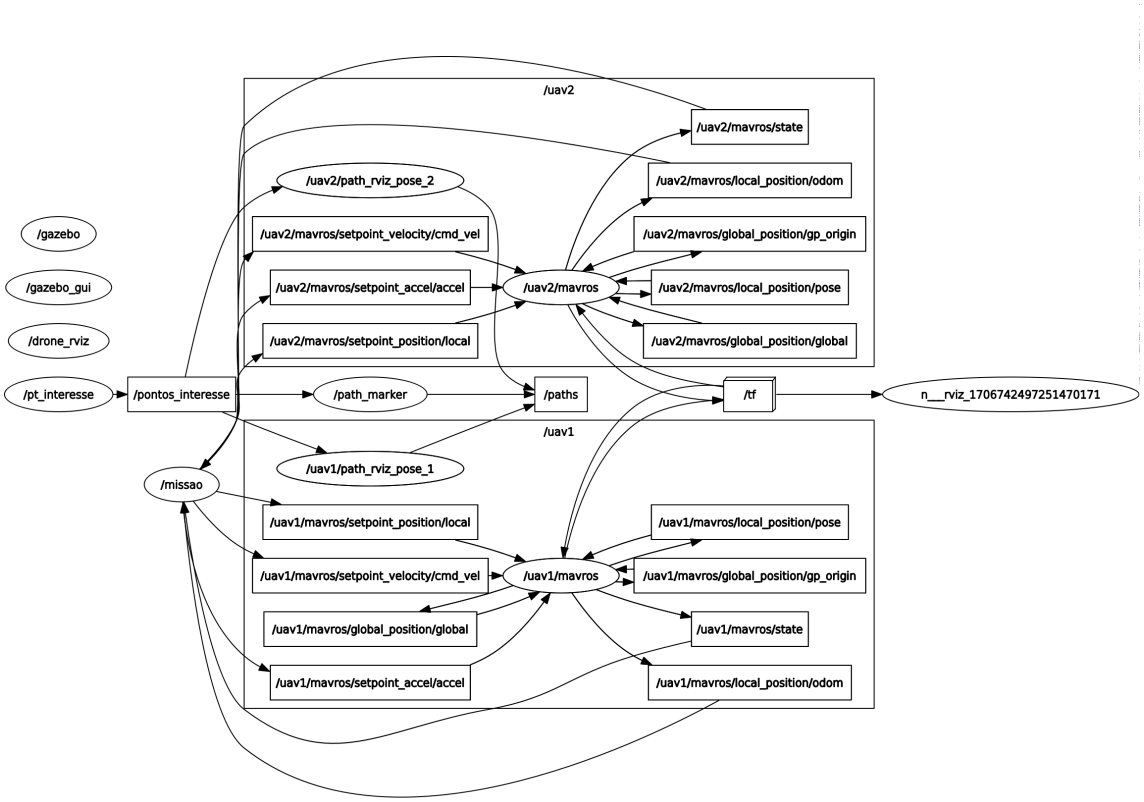


Figure 29 – Active ROS nodes and topics detailing the multi-UAV coordination (UAV1 and UAV2) and shared mission parameters.

Subsequently, Figure 30 details the UGV's internal node structure. The ground vehicle processes its spatial navigation through the /camera and /rtabmap nodes. As previously mentioned, the RTAB-Map ROS package performs simultaneous mapping and localization by fusing stereo odometry with wheel odometry using an Extended Kalman

Filter (EKF).

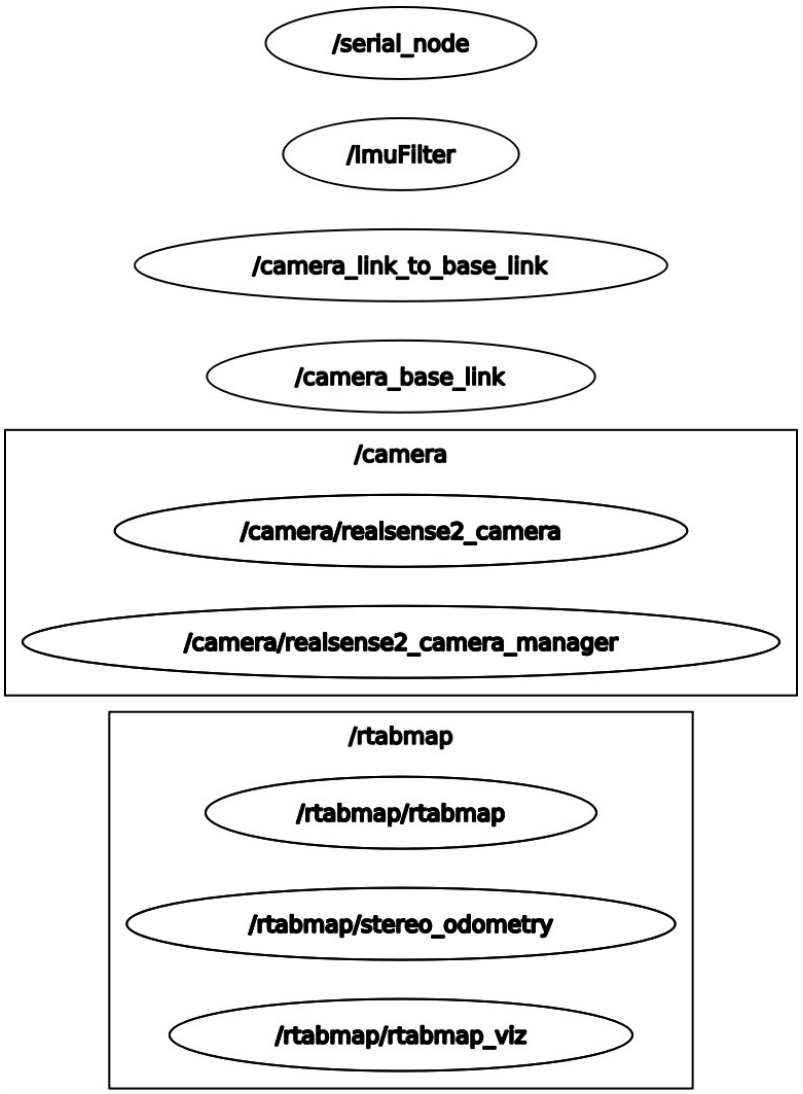


Figure 30 – Active ROS nodes and topics detailing the UGV internal structure, highlighting the mapping and serial control bridges.

4.3.1 Agriculture Scenario: Local Obstacle Avoidance

The first set of simulation results evaluates the Deep Reinforcement Learning training process. For a bounding workspace $\mathcal{W} \subset \mathbb{R}^3$ defined by the limits $x, y, z \in [0, 10]$ meters, and containing 15 dynamic obstacles, different Deep Q-Network (DQN) architectures were evaluated to optimize the agent's learning convergence, including the baseline structure proposed by de Castro et al. (Castro, G. G. de et al., 2022).

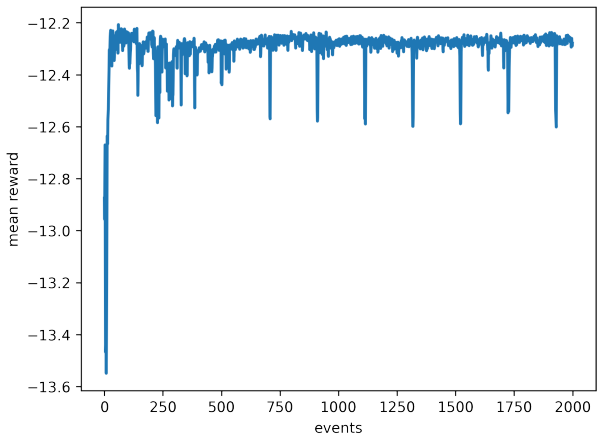
Table 6 details the structural hyperparameters of the four neural network architectures that achieved the best preliminary performance. To establish a controlled baseline for comparison, the total number of training events (2000), the learning rate (0.001), and the discount factor ($\gamma_{\mathrm{RL}} = 0.99$) were kept strictly fixed across all models. Empirical testing with different values for these specific parameters yielded significantly poorer convergence. Therefore, they were locked to allow fine-tuning of the network depth and step sizes.

Table 6 – DQN's models and hyperparameters set during the training phase.

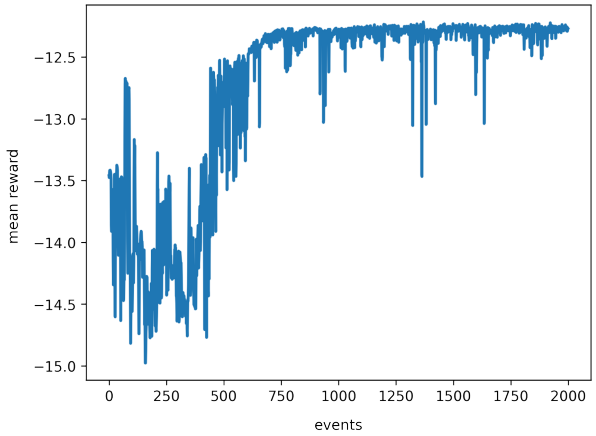
Events = 2000, lr = 0.001, $\gamma_{\mathrm{RL}} = 0.99$			
	Hidden Layers	Neurons	Steps
Model 1	1	30	1000
Model 2	1	100	500
Model 3	3	[100,60,30]	500
Model 4	3	[81,212,29]	500

By critically analyzing the learning curves in Figure 31, the performance disparities between the architectures become evident. The x-axis represents the training events over time, while the y-axis maps the mean accumulated reward. Model 1 (Figure 31a) exhibits rapid initial convergence but suffers from periodic, deep negative reward spikes, indicating structural instability during its exploitation phase. Model 2 (Figure 31b) struggles with a slow learning rate, taking over 750 events to escape the initial penalty plateau. Model 4 (Figure 31d) demonstrates complete learning failure; the agent overfits or becomes mathematically unstable, oscillating wildly without ever achieving a stable reward state. Conversely, Model 3 (Figure 31c) delivers the most robust performance, achieving rapid

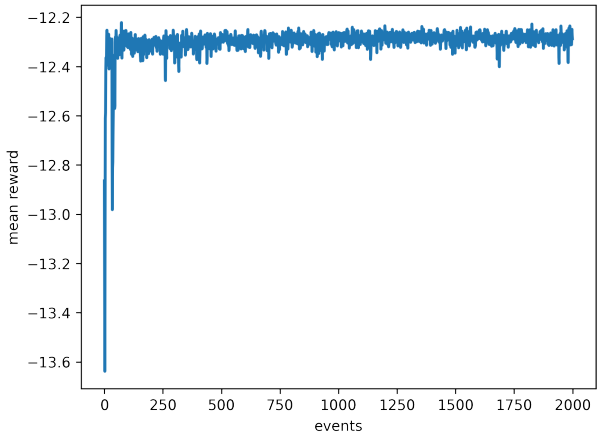
convergence and maintaining a highly stable, flat reward plateau with minimal downward variance.



(a) Model 1: 1 hidden layer, 30 neurons, step size of 1000.

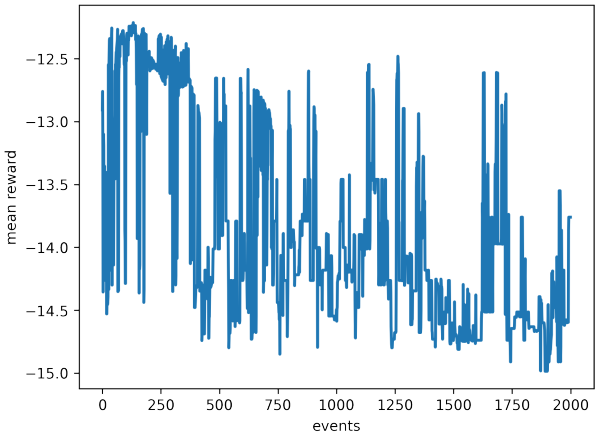


(b) Model 2: 1 hidden layer, 100 neurons, step size of 500.



(c) Model 3: 3 hidden layers (100, 60, 30 neurons), step size of 500.

To further check the exploration-exploitation trade-off of the viable candidates, Figure 32 tracks specific milestones during the training execution. The vertical green



(d) Model 4: 3 hidden layers (81, 212, 29 neurons), step size of 500.

Figure 31 – Mean reward returns for the evaluated DQN models across 2000 training events (Castro, G. G. d. et al., 2023).

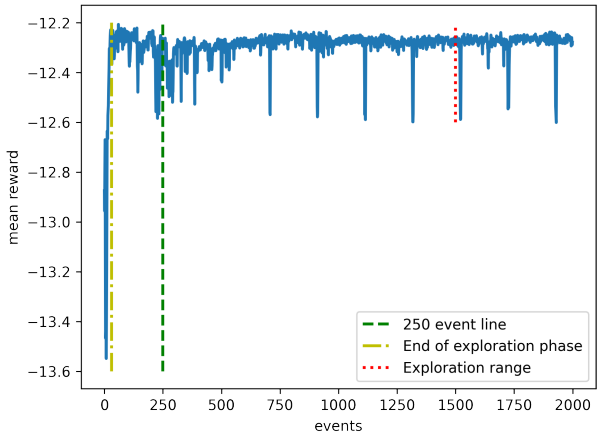
dashed line marks the 250-event benchmark (one-eighth of the total training), the yellow dash-dotted line indicates the definitive end of the exploration phase, and the red dotted line highlights the exploration variance during the exploitation phase.

As detailed in Table 7, these visual milestones translate into concrete performance metrics. While Model 1 finishes its exploration phase slightly earlier (20 events) than Model 3 (40 events) and has a marginally better numerical mean reward (-12.28 against -12.30), Model 3's policy is vastly superior in practice. As seen by the red variance line in Figure 32c, Model 3 maintains a tight, predictable behavior during exploitation, completely avoiding the negative peaks seen in Model 1. Due to this superior policy stability, Model 3 was selected as the optimal local collision-avoidance agent for the subsequent architectural integration.

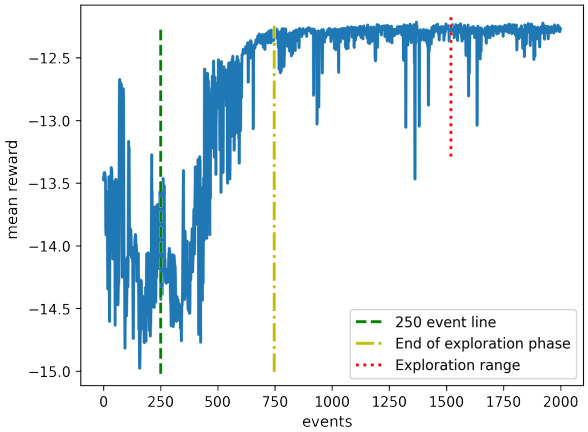
Table 7 – Performance and convergence comparison of Models 1, 2, and 3.

	Total Mean Reward	Exploration Phase End (Event)
Model 1	-12.28	20
Model 2	-12.32	750
Model 3	-12.30	40

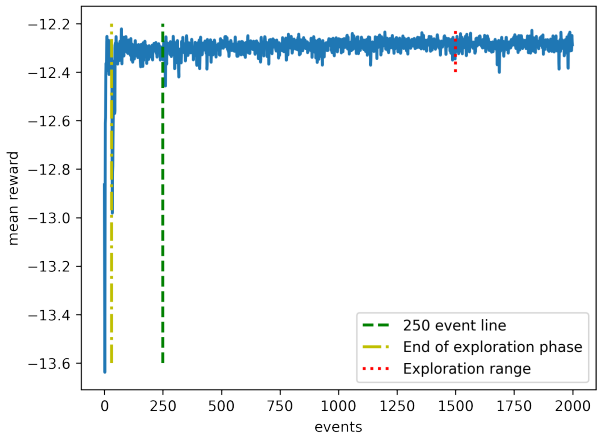
A total of twenty experiments were conducted in the unstructured olive-grove Gazebo world. The physical properties of the dynamic objects and the spatial coordinates of the insect traps were randomized in each scenario. Figure 33 provides a top-down perspective of the simulation environment, mapping the positions of the four target insect traps (labeled A, B, C, and D) and visually demonstrating the initial global paths gener-



(a) Model 1.



(b) Model 2.



(c) Model 3.

Figure 32 – Milestone comparison between the three best models. The green line marks 1/8th of training, the yellow line indicates the end of the strict exploration phase, and the red line shows exploration variance during the exploitation phase (Castro, G. G. d. et al., 2023).

ated by the RRT algorithm to link them. As observed, the RRT planner constructs a global sequence of linear, dotted segments connecting the targets. Crucially, these segments often cut across the agricultural rows without accounting for local dynamic interferences.

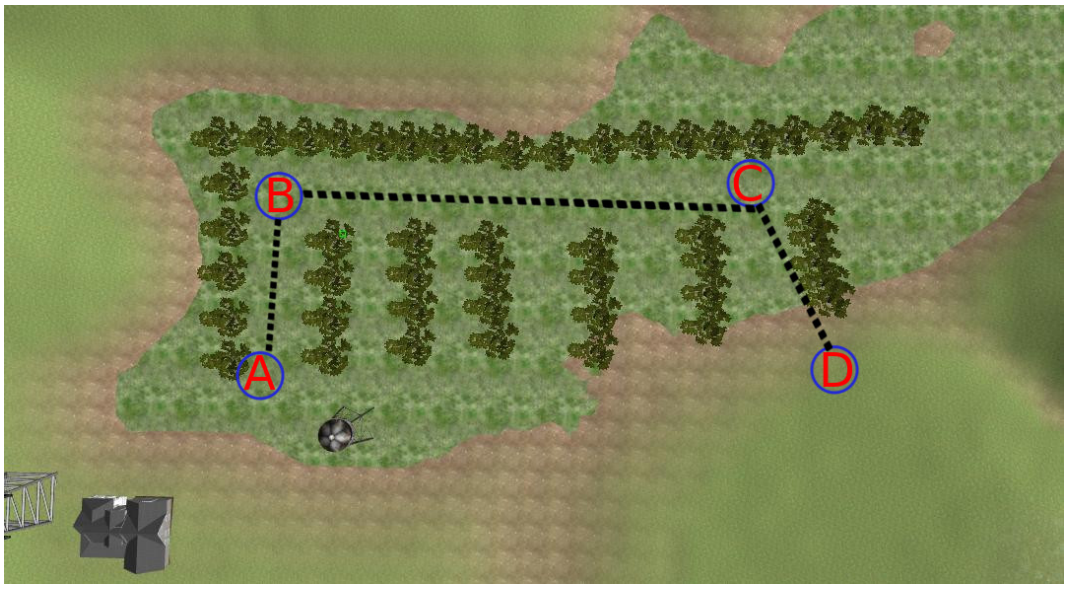


Figure 33 – Top-down perspective of the agricultural Gazebo environment. The image details the spatial distribution of the insect traps (A, B, C, D) and the dotted piecewise linear paths generated offline by the global RRT algorithm.

It is crucial to emphasize that the baseline RRT algorithm operates offline and exclusively possesses information regarding static obstacles; thus, its generated trajectory cannot guarantee collision avoidance against dynamic elements. To solve this, the angular coefficients between the RRT waypoints are continuously published via ROS topics to the local DQN agent, which then assumes real-time control to deviate from unmapped or moving obstacles.

Across all evaluated scenarios, the proposed hybrid methodology successfully completed the inspection missions. To statistically validate the real-time tracking accuracy, Figure 34 presents a boxplot analysis comparing the UAV's actual heading angle (ψ_{UAV}) with the reference RRT path angle (θ_{RRT}). The data is segmented into the three primary legs of the mission: Path AB, Path BC, and Path CD.

The statistical distribution confirms the system's robustness. The medians (indicated by the central lines within the boxes) for paths AB and BC are consistently stable around $-1°$, while path CD centers near $+1.3°$. This proves that the UAV successfully maintains the global macroscopic heading directed by the RRT. Concurrently, the interquartile ranges (the boxes) and the maximum deviations (the whiskers, reaching up to

$\pm 5°$) physically represent the precise moments when the DQN agent proactively overrides the RRT orientation to execute localized, evasive maneuvers against impending dynamic threats, before seamlessly returning to the nominal route.

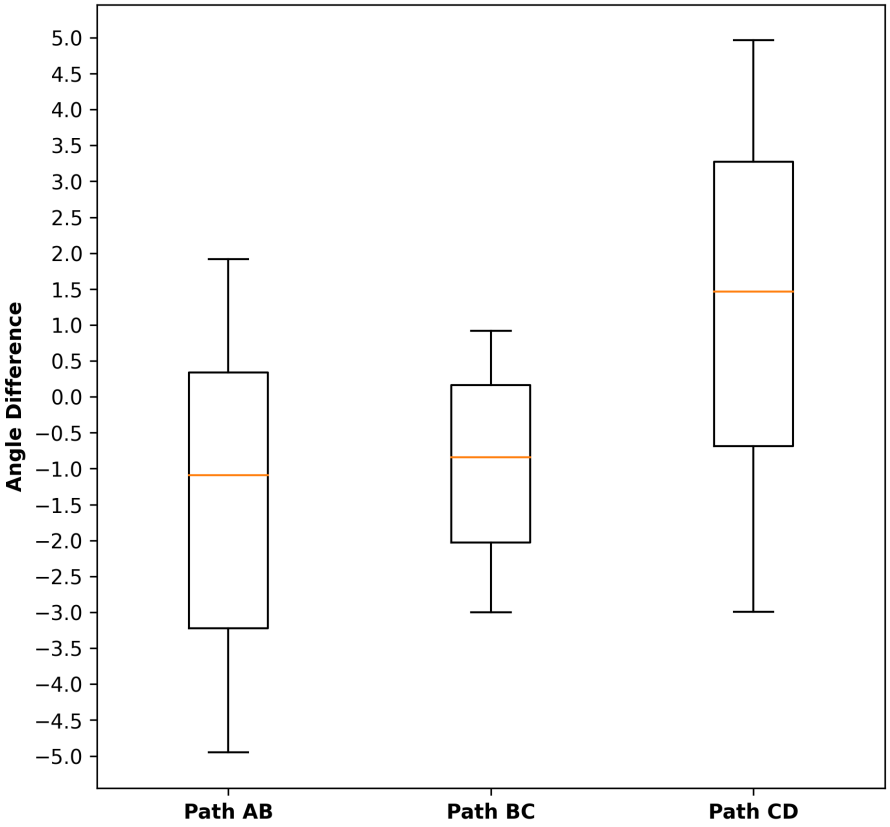


Figure 34 – Boxplot analysis of the angular tracking deviation. The y-axis represents the error between the global RRT path angle (θ_{RRT}) and the actual UAV heading (ψ_{UAV}), segmented by mission legs (Paths AB, BC, and CD).

4.3.2 Electrical Substation: 3D Mapping and Representation

The mapping framework is a foundational requirement for the autonomous inspection architecture. It enables the heterogeneous robotic fleet to spatially understand the unstructured environment, ensuring accurate localization, data collection, and robust obstacle avoidance. Since all agents operate in the same workspace, they continuously interact with the ROS mapping topics, reading from and updating the spatial data via their onboard sensors.

The initial phase of the environment reconstruction involved the UGV navigating through the substation. The ground vehicle processed its stereo camera readings using the RTAB-Map package (Labbé; Michaud, 2019) to perform simultaneous localization and mapping. Following the physical exploration, the collected data was consolidated into a 2D occupancy grid, shown in Figure 35. As directly observed in the figure, this 2D projection effectively extracts the ground-level topological layout, capturing the operational boundaries and the dense physical footprints of the electrical transformers and support pillars. This planar map is sufficient to ensure safe surface navigation for the UGV.

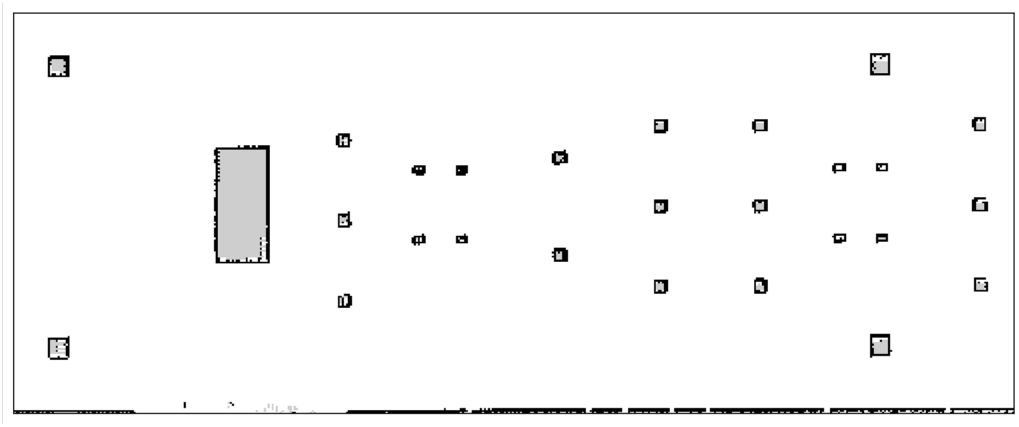


Figure 35 – A 2D occupancy grid representation of the substation environment generated by the UGV, highlighting the operational boundaries and structural footprints (Castro, G. G. de et al., 2024).

However, while the 2D grid supports planar movement, the UAV requires a comprehensive volumetric understanding of the airspace to calculate collision-free flight paths. To achieve this, a UAV was deployed to conduct sweeps at specific operational altitudes. Utilizing its built-in depth camera, it transmitted spatial point-cloud data to the OctoMap framework (Hornung et al., 2013). This probabilistic mapping package processes the incoming images to estimate the spatial occupancy of individual cubic voxels in the 3D space.

The outcome of this volumetric reconstruction is depicted in Figure 36, where a 3D modeling software was employed to refine the voxel grids for better visual interpretation. Unlike the flat 2D map, this 3D representation explicitly captures the verticality of the substation environment. It details the exact height of the tall structural pillars, the central transformer block, and suspended obstacles. Extracting this complex 3D geometry is a mandatory step to provide the necessary spatial constraints for the subsequent UAV Coverage Path Planning (CPP), preventing catastrophic collisions with elevated metallic

structures.

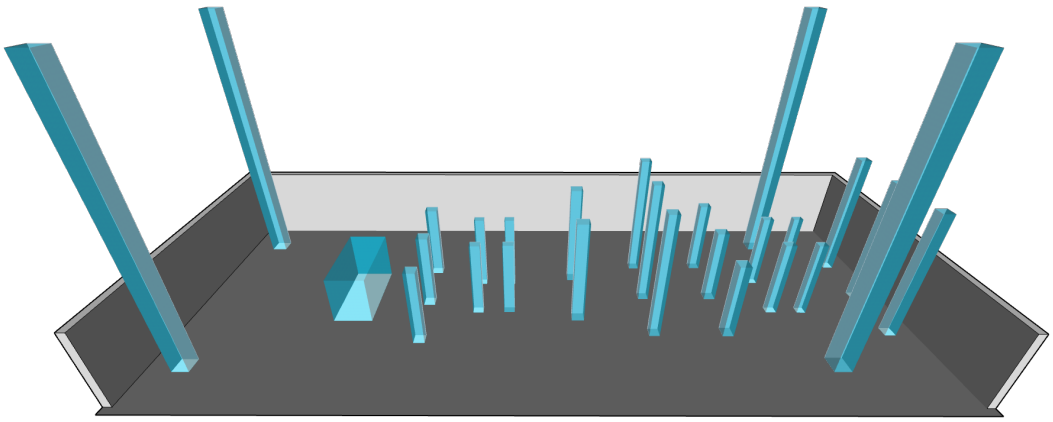


Figure 36 – A volumetric 3D representation of the substation environment generated via the OctoMap framework, detailing the vertical constraints and elevated structures (Castro, G. G. de et al., 2024).

4.3.3 Coverage Path Planning (CPP) Evaluation

Tuning the Wavefront algorithm fundamentally involves selecting an optimal obstacle penalty factor (γ_{CPP}), previously defined in Equation 12. This parameter directly shapes the artificial potential field propagated across the grid and, consequently, dictates the geometric quality of the resulting coverage trajectories. As is standard in CPP benchmarking, two primary metrics were evaluated: the number of path overlaps (redundant visitation) and the number of sharp directional maneuvers (kinematic cost). All simulation episodes began with identical initial poses, goal coordinates, and free-space configurations.

Figure 37 illustrates the profound impact of different γ_{CPP} values on the generated paths. A critical visual analysis of the benchmark reveals a massive spike in path variation at $\gamma_{\text{CPP}} = 1$. However, the system rapidly stabilizes, achieving its optimal and most predictable performance for higher penalty factors ($\gamma_{\text{CPP}} \geq 10$). In this optimized range, the algorithm significantly minimizes unnecessary kinematic movements and overlaps when compared to the baseline approach relying purely on the Distance Transform (i.e., $\gamma_{\text{CPP}} =$

0).

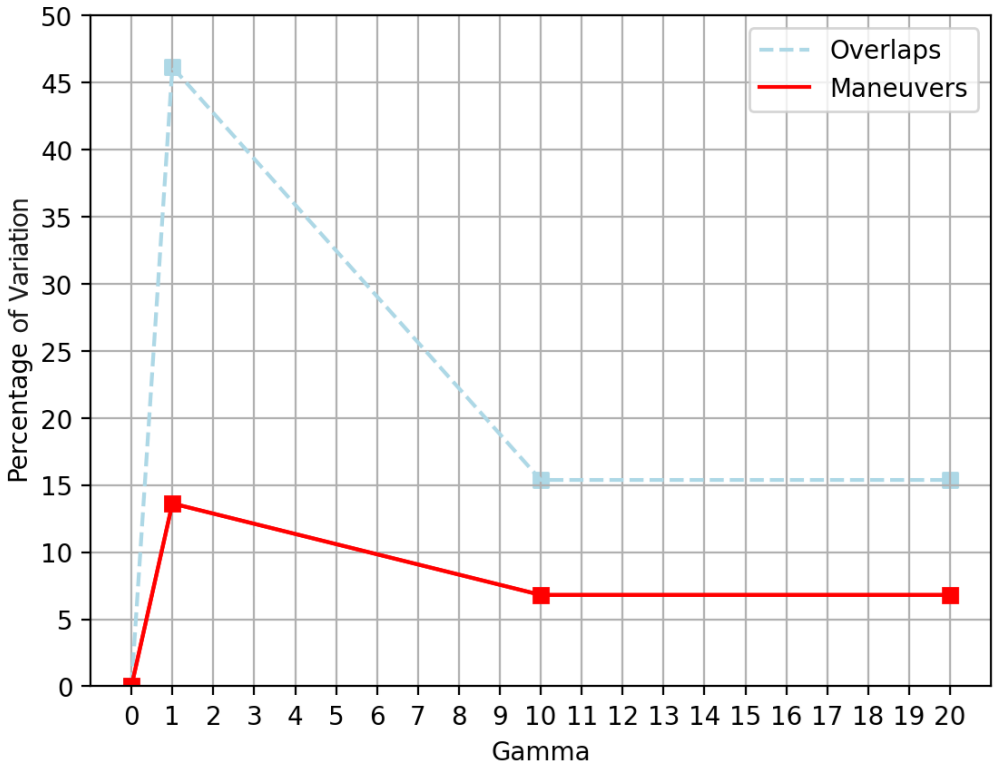
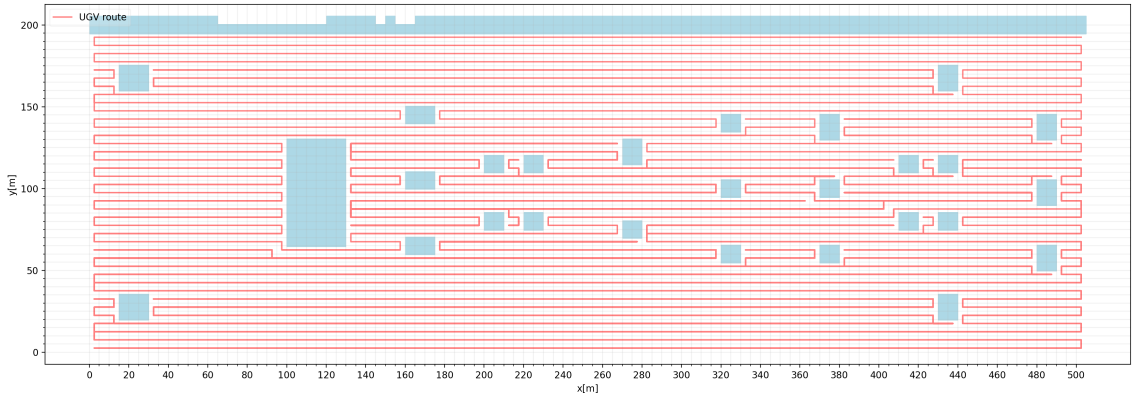


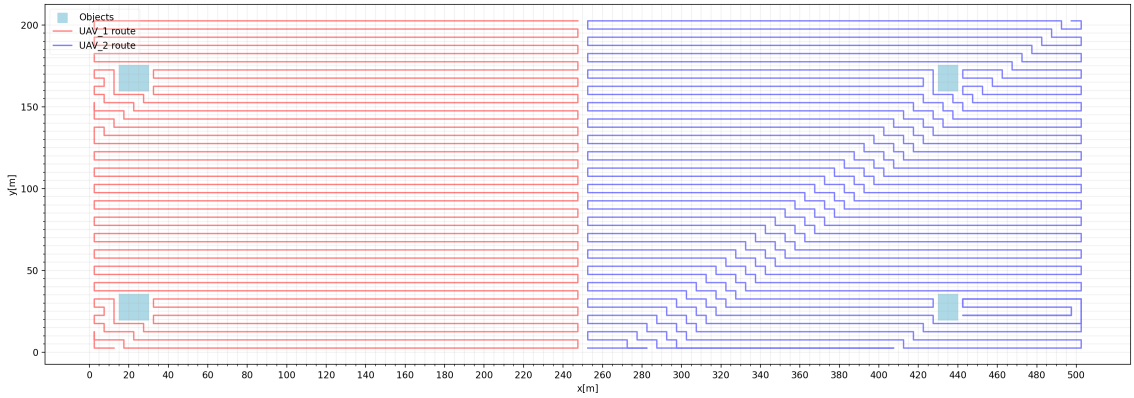
Figure 37 – Wavefront tuning benchmark. The dashed lines represent the frequency of route overlaps, while the red lines quantify the required sharp maneuvers across different penalty factors (Castro, G. G. de et al., 2024).

Following the optimization of the penalty factor, the tuned Wavefront method was deployed to compute the exhaustive coverage routes for both the UGV and the UAVs. Figure 38 presents the resulting global trajectories. Panel (a) illustrates the UGV executing a continuous horizontal sweeping pattern, effectively avoiding the mapped structural footprints. More impressively, Panel (b) demonstrates the system's capacity for spatial partitioning: the workspace is dynamically divided, with UAV 1 (red trajectory) and UAV 2 (blue trajectory) executing parallel coverage in distinct sectors, naturally preventing mid-air interference and accelerating the mission.

Beyond the standalone implementation of Wavefront, this thesis incorporates the hybrid coverage strategy extensively evaluated in (Castro, G. G. R. de et al., 2025), where Wavefront dictates the global coverage structure while a Bidirectional A* (BA*) search acts as a fallback mechanism to overcome local minima traps. In this hierarchical architecture, Wavefront determines the global visitation order, ensuring exhaustive area sweeping, whereas BA* resolves routing transitions out of dead-end cells, strictly minimizing travel



(a)



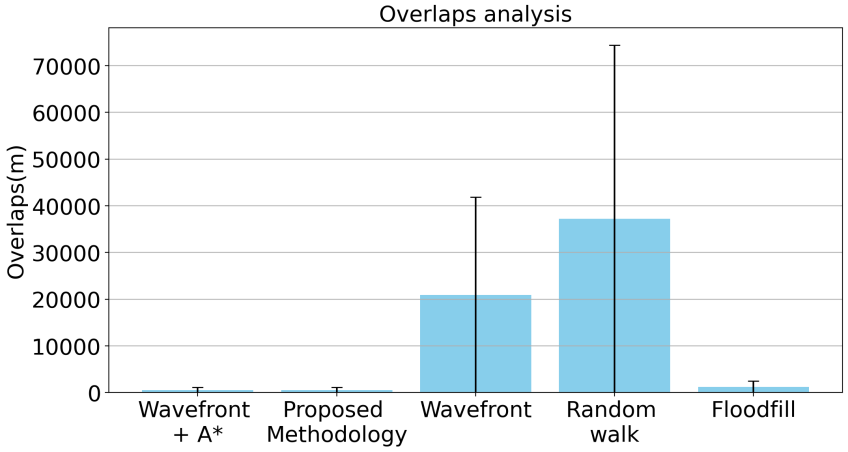
(b)

Figure 38 – Systematic map coverage generated by the tuned Wavefront CPP. (a) UGV continuous path. (b) Multi-UAV parallel paths showcasing spatial partitioning (Castro, G. G. de et al., 2024).

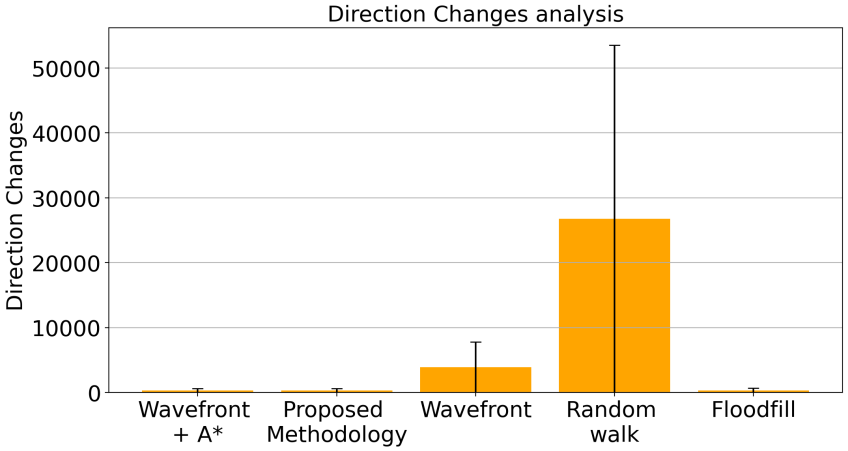
distance and unnecessary turns.

To robustly validate this approach, a series of tests were performed on a photovoltaic farm environment prior to deployment in the substation. Figure 39 visualizes the statistical performance comparison. The sheer scale of the y-axes in panels (a) and (b) highlights the inefficiency of native algorithms: pure Random Walk and classical Wavefront generate tens of thousands of redundant overlaps and direction changes. In contrast, the proposed Wavefront+BA* method (labeled as 'Proposed Methodology') flattens these metrics to near zero. While panel (c) shows a nominal increase in computation time for the hybrid approach ($\approx$ 7.5 s) compared to the raw Wavefront ($\approx$ 3 s), this trade-off is highly justified by the massive reduction in kinematic cost and battery expenditure.

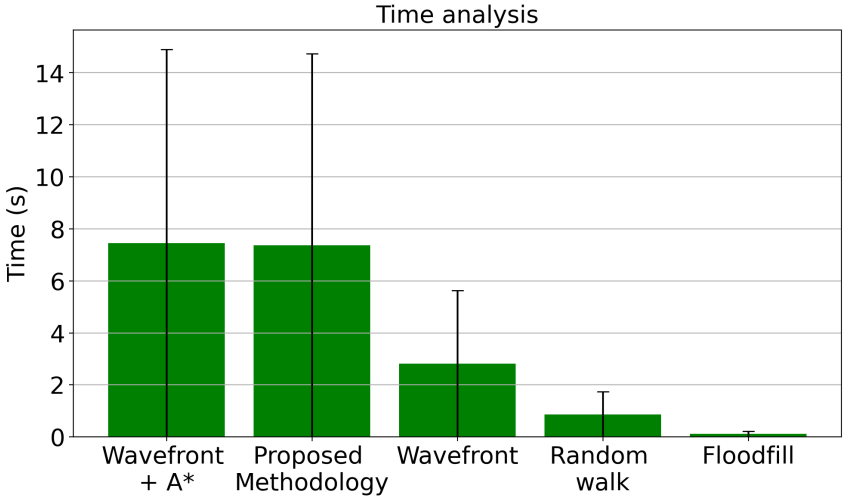
A precise numerical comparison of these improvements is tabulated in Table 8. When compared to the baseline Wavefront+A*, the proposed BA* bidirectional approach performs similarly in path overlaps (-0.13%) and direction changes (-0.33%), but achieves



(a)



(b)



(c)

Figure 39 – Mean performance analysis benchmarking Wavefront+BA*: (a) Overlaps, (b) Direction changes, (c) Time consumption (Castro, G. G. R. de et al., 2025).

a significantly faster execution time (8.74% improvement). Crucially, against the traditional Floodfill expansion algorithm, the hybrid approach reduces redundant overlaps by an impressive 126.28% and improves trajectory smoothness by 6.32%. These results categorically prove the effectiveness of the proposed hybrid approach in generating shorter, smoother, and more coverage-efficient trajectories.

Table 8 – Percentage improvement of the proposed Wavefront+BA* method against baseline algorithms (Castro, G. G. R. de et al., 2025).

Method	Overlaps (%)	Direction Changes (%)	Time (%)
Wavefront + A*	−0.13	−0.33	8.74
Wavefront	6039.85	1875.00	−32.40
Random Walk	6528.33	8382.51	−85.44
Floodfill	126.28	6.32	−98.49

4.3.4 System Resilience and Contingency Routines

During the execution of complex inspection missions, the pre-computed CPP routes are rarely static. In realistic operational environments, trajectories must adapt dynamically to hardware limitations, environmental surprises, and signal degradation. To mitigate these challenges and ensure mission survivability, the proposed framework implements three specific automated contingency routines: battery scarcity, obstacle avoidance, and communication loss.

Battery Scarcity and Hot Swap

In this extended simulation scenario, UAV 1 launches from the ground while UAV 2 remains docked onto the UGV. Both the flying agent and the ground vehicle execute their respective coverage routes. However, once UAV 1 hits a critical battery threshold of 20%, the hot-swap contingency routine is automatically invoked.

As visually depicted in the 3D plot of Figure 40, the green trajectory represents UAV 1 executing the initial coverage. Upon reaching the battery limit, the mission dynam-

ically pauses: the UGV stops moving, and UAV 1 initiates a calculated diagonal descent to land on the UGV platform. Simultaneously, UAV 2 takes off to reach the inspection altitude, effectively performing a mid-air task handover. Following the successful hot swap, UAV 2 (blue trajectory) and the UGV seamlessly resume the sweeping patterns until the global mission is completed.

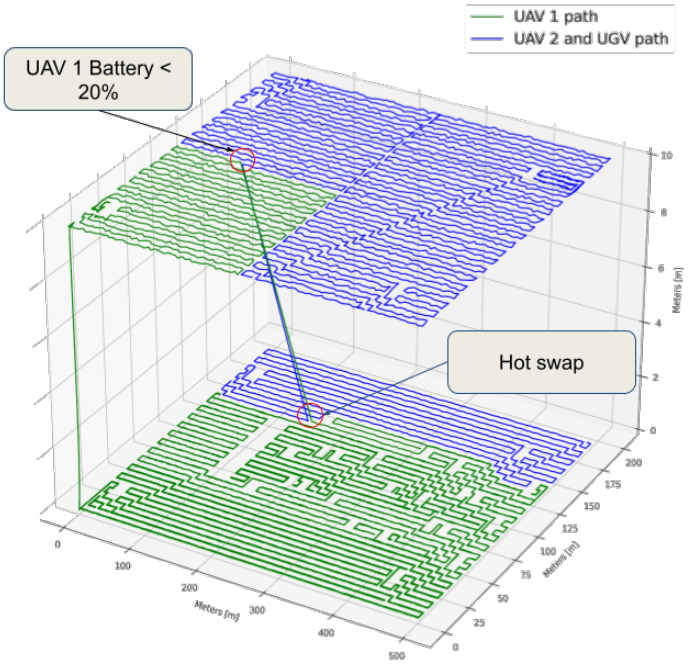


Figure 40 – 3D trajectory plot of the battery scarcity routine. The graph captures the exact spatial handover where UAV 1 (green) descends for a hot swap, allowing UAV 2 (blue) to resume the coverage mission (Castro, G. G. de et al., 2024).

Avoiding Unmapped Objects

To test the system's local reactive capabilities, five unmapped dynamic objects—varying in size from 1 to 10 grid occupancies—were randomly spawned during a standard mission execution. Figure 41 illustrates the system's robust response to these unexpected threats.

The red markers in the simulation plot indicate the exact locations of the unmapped

obstacles. When an agent's onboard sensors detect these immediate collision risks, the global planner temporarily yields control, and the local Neural Network (DQN) routine is activated. The short black trajectory segments visible in the graph highlight the precise moments the DQN successfully computes and executes localized evasive detours, steering the UAV away from the threat before safely returning control to the global RRT route.

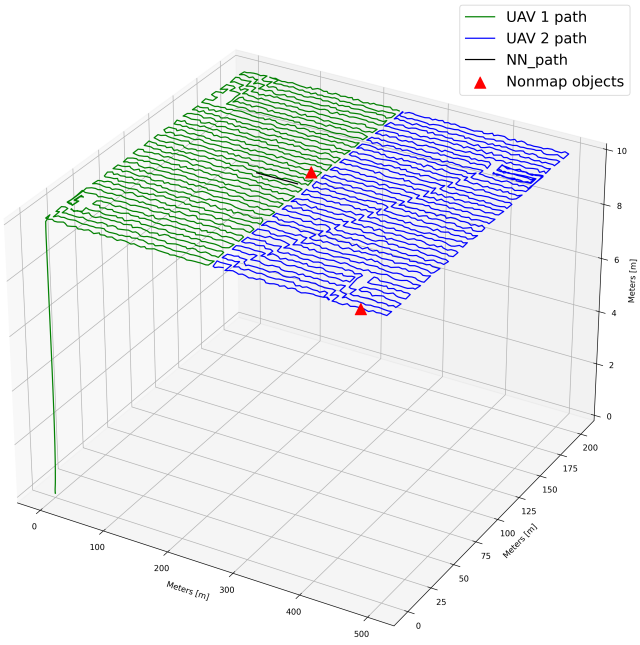


Figure 41 – Trajectory plot of the unmapped object avoidance routine. The black trajectory segments represent the DQN actively overriding the nominal path to detour around unmapped obstacles (red markers) (Castro, G. G. de et al., 2024).

Communication Loss

Because localization and map information are continuously shared among the agents via the ROS Master, maintaining network integrity is paramount. To ensure operational safety, a strict communication loss routine was implemented. It triggers the moment an agent stops receiving heartbeat signals from the central coordinator.

Figure 42 captures the exact spatial consequence of a simulated signal failure. The green trajectory of UAV 1 abruptly terminates its coverage pattern at the point of failure and executes a direct, sweeping descent arc back to its designated Home Position (Return

To Home (RTH) procedure). During this loss routine, the UGV and its nested UAV immediately halt their movements, safely holding their positions to prevent blind collisions until the system communication is fully recovered or manual intervention occurs.

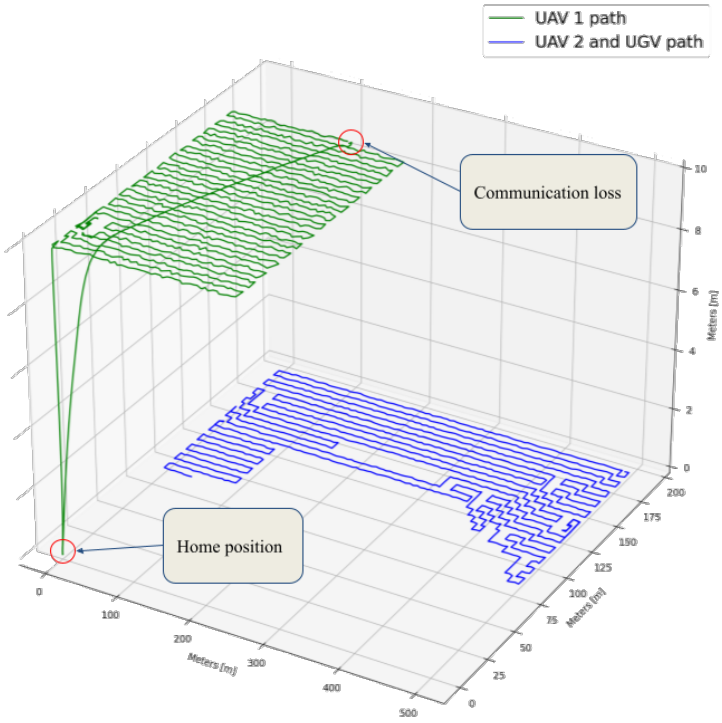


Figure 42 – Simulation of a total communication loss. The green trajectory displays the flying UAV instantly aborting its mission and executing a direct Return to Home (RTH) descent arc (Castro, G. G. de et al., 2024).

Contingency Time Analysis and Reliability

While this framework does not strictly aim to minimize absolute flight time—prioritizing safety and coverage completeness instead—evaluating the temporal cost of these contingencies is crucial for practical deployment. Table 9 compiles the mean execution metrics across multiple simulated scenarios. Notably, a full mission involving a hot swap takes an average of 28 minutes, while navigating a map with unmapped obstacles adds a marginal delay, totaling 29 minutes. Most importantly, the framework achieved an overwhelming success rate (96% to 100%) across all chaotic scenarios, proving that the

proposed contingency routines are highly reliable under stress.

Table 9 – Compilation of performance metrics and success rates during simulated contingency scenarios.

Scenario	Mean Time Cost (minutes)	Mean Path Deviation (meters)	Success Rate
Battery Scarcity / Hot Swap	28	2	98%
Unmapped Objects	29	10 (per two objects)	96%
Communication Loss	Varies on recovery	1	100%

4.4- Discussions

The comprehensive results extracted from these simulations validate the efficacy of the proposed hybrid architecture in orchestrating complex inspection tasks. By effectively merging a macroscopic Coverage Path Planner (Wavefront+BA*) with a microscopic reactive agent (DQN), the generated paths strike an optimal balance: they guarantee exhaustive coverage of the designated areas, minimize unnecessary kinematic traversal, and dynamically avoid both static and moving obstacles in real time.

Furthermore, the developed methodology exhibits significant scalability. The modular design of the ROS-based framework natively supports heterogeneous fleets, allowing for the seamless integration of additional UGVs, UAVs, or diverse sensor payloads depending on the operational demands of the industrial facility.

Finally, the inherent inclusion of strict contingency protocols—such as the Return To Home (RTH) procedure during signal loss and automated hot swaps for battery management—elevates this framework from a purely theoretical algorithm to a deployment-ready architectural blueprint. By treating communication drops and energy scarcity as expected variables rather than terminal failures, the system ensures maximum survivability. These features collectively demonstrate a robust, scalable, and highly adaptable solution for the next generation of autonomous industrial inspection.

5- Conclusions and Future Works

Conclusions

This thesis introduced a novel framework for cooperative coverage path planning in dynamic, partially known environments using a hybrid team of two UAVs and a UGV. The methodology leveraged the strengths of each robot type, allowing for enhanced efficiency and adaptability in complex scenarios such as electrical substations and precision agriculture environments. Key achievements include the development of robust coverage path planning algorithms that dynamically adapt paths to avoid collisions while ensuring comprehensive coverage, the integration of Deep Reinforcement Learning (DRL) for real-time obstacle avoidance, and extensive simulation validations that demonstrate the effectiveness of the proposed strategies.

To rigorously validate this strategy, a high-fidelity simulation environment was developed in Gazebo and ROS to mimic an authentic olive-grove culture, incorporating dynamic objects, accurate UAV kinematics, and stochastic environmental characteristics. Across twenty randomized scenarios, featuring varying insect trap poses and dynamic obstacle trajectories, the proposed hybrid architecture consistently accomplished the mission. The system successfully executed autonomous takeoffs, navigated to all designated waypoints, and returned to the home location without any collisions.

The comprehensive results extracted from these simulations validate the efficacy of orchestrating complex inspection tasks through a dual-layered approach. By effectively merging a macroscopic Coverage Path Planner (Wavefront+BA*) with a microscopic reactive agent (DQN), the generated paths strike an optimal balance: they guarantee exhaustive coverage of the designated areas, minimize unnecessary kinematic traversal, and dynamically avoid both static and moving obstacles in real time.

Furthermore, the inherent inclusion of strict contingency protocols, such as the Return To Home (RTH) procedure during signal loss and automated hot swaps for battery management, elevates this framework from a purely theoretical algorithm to a deployment-ready architectural blueprint. By treating communication drops and energy scarcity as

expected variables rather than terminal failures, the system ensures maximum survivability. It ensures uninterrupted mission progression and highlights the potential for deployment in chaotic environments where traditional static path planning falls short.

Future Works

While extensive validations were performed in high-fidelity Gazebo simulations, a natural and critical progression for further research is the deployment of this framework on physical robotic platforms. Transitioning from simulation to real-world environments (sim-to-real) will allow for the empirical assessment of hardware-in-the-loop constraints, authentic sensor noise, communication latency, and real-time physical disturbances on the hybrid fleet.

Furthermore, the highly modular design of the ROS-based framework ensures that the architecture remains future-proof. As the fields of robotics and artificial intelligence continuously evolve, the proposed system is structurally prepared to integrate new state-of-the-art methodologies. Because the architecture decouples global and local tasks, the macroscopic planner can be seamlessly swapped if a more efficient Coverage Path Planning (CPP) algorithm is developed in the literature. Similarly, the microscopic reactive agent can be effortlessly upgraded with more complex local path-planning algorithms or advanced Deep Reinforcement Learning models without disrupting the overarching system synchronization.

The contingency management system also offers a direct pathway for expansion. The current rule engine, which successfully handles battery scarcity, unmapped obstacles, and communication loss, was designed to be easily extended. Future iterations could incorporate new logical rules and routines to handle sensor degradation, dynamic weather changes, or cooperative payload transport.

Ultimately, this profound adaptability opens up vast possibilities for applications beyond the initial scope. Adapting the framework for larger swarms of heterogeneous robots deployed in critical scenarios, such as disaster response, search and rescue operations, and massive-scale agricultural monitoring, establishes the proposed architecture not merely as a static solution, but as a living, highly adaptable blueprint for the next

generation of autonomous industrial inspection.

Scientific production

As of the present moment, this doctoral thesis has resulted in the publication of three journal articles, with this thesis author being the primary author in each instance. The articles are as follows:

- de Castro, G. G., et al. Hybrid Wavefront and Bidirectional A* Coverage Path Planning for AGVs in Photovoltaic Farm Inspection. In 2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) (pp. 64-69). IEEE.
- de Castro, Gabriel GR, et al. "Heterogeneous Multi-Robot Collaboration for Coverage Path Planning in Partially Known Dynamic Environments." Machines 12.3 (2024): 200.
- de Castro, Gabriel GR de, et al. "Adaptive path planning for fusing rapidly exploring random trees and deep reinforcement learning in an agriculture dynamic environment UAVs." Agriculture 13.2 (2023): 354.
- de Castro, Gabriel GR, et al. "Dynamic path planning based on neural networks for aerial inspection." Journal of Control, Automation and Electrical Systems 34.1 (2023): 85-105.

Additionally, six related article have been published, in which the thesis author is listed as a co-author:

- Berger, Guido S., et al. "Cooperative heterogeneous robots for autonomous insects trap monitoring system in a precision agriculture scenario." Agriculture 13.2 (2023): 239.
- Coelho, F. O., Pinto, M. F., Biundini, I. Z., Castro, G. G., Andrade, F. A., & Marcato, A. L. (2024). Autonomous uav exploration and mapping in uncharted terrain through boundary-driven strategy. IEEE Access, 12, 92464-92483.'
- Dos Santos, T. M., Pinto, M. F., Biundini, I. Z., Castro, G. G., Honório, L. M., & Clua, E. (2024, November). Integrating Visual and Encoder Data for Enhanced Nav-

igation in Mecanum-Wheeled Robots via Extended Kalman Filter. In 2024 Brazilian Symposium on Robotics (SBR), and 2024 Workshop on Robotics in Education (WRE) (pp. 174-179). IEEE.

- Lisboa, K. L., Castro, G. G., Bastos, A. F., & Pinto, M. F. (2023, November). Low-Cost Embedded System Applied to Data Processing and Indicator Generation of Motor Vehicles. In 2023 15th IEEE International Conference on Industry Applications (INDUSCON) (pp. 1437-1443). IEEE.

- CARVALHO, LUCAS L. M. ; MARQUES, DIOGO G. ; CASTRO, GABRIEL G.R. ; TUXI, THIAGO M. ; PINTO, MILENA F. . Hybrid Algorithm Based on Bidirectional A* and Floodfill Applied to Coverage Path Planning for AGVs. In: 2025 Brazilian Symposium on Robotics (SBR) and 2025 Workshop on Robotics in Education (WRE), 2025, Vitoria. 2025 Brazilian Symposium on Robotics (SBR) and 2025 Workshop on Robotics in Education (WRE), 2025. p. 84.

- PINTO, MILENA F.; CARVALHO, LUCAS L.M. ; SOUSA, LUCAS C. ; AMORIM, JOHANN S.J.C.C ; NASCIMENTO, YURI S. ; CASTRO, GABRIEL G. R. . A ROS2-Based Robotics Course for Undergraduate Program. In: 2025 Brazilian Symposium on Robotics (SBR) and 2025 Workshop on Robotics in Education (WRE), 2025, Vitoria. 2025 Brazilian Symposium on Robotics (SBR) and 2025 Workshop on Robotics in Education (WRE), 2025. p. 308.

6- References

AGGARWAL, Shubhani; KUMAR, Neeraj. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. Computer Communications, Elsevier, v. 149, p. 270–299, 2020.

ALI, Hub et al. Feature selection-based decision model for UAV path planning on rough terrains. Expert Systems with Applications, Elsevier, p. 120713, 2023.

ALMADHOUN, Randa et al. A survey on multi-robot coverage path planning for model reconstruction and mapping. SN Applied Sciences, Springer, v. 1, p. 1–24, 2019.

ANDRADE, Fabio AA et al. Unmanned Aerial Vehicles Motion Control with Fuzzy Tuning of Cascaded-PID Gains. Machines, MDPI, v. 10, n. 1, p. 12, 2021.

BERGER, Christian; SCHERER, Jan; LOZANO, Tomás. Hybrid Aerial-Ground Cooperative Exploration for Heterogeneous Robot Teams. Robotics and Autonomous Systems, v. 166, p. 104412, 2023. DOI: 10.1016/j.robot.2023.104412.

BIUNDINI, Iago Z et al. Experimentation and simulation with autonomous coverage path planning for uavs. Journal of Intelligent & Robotic Systems, Springer, v. 105, n. 2, p. 46, 2022.

CABREIRA, T. M.; ALMEIDA, L.; BRISOLARA, L. Distributed Coverage Strategies for UAV Swarms: A Review. Sensors, v. 23, n. 2, p. 512, 2023. DOI: 10.3390/s23020512.

CABREIRA, Tiago M.; BRISOLARA, Lisane B.; FERREIRA, Paulo R. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. Drones, v. 3, n. 1, p. 4, 2019. DOI: 10.3390/drones3010004.

CAO, Y. U.; FUKUNAGA, A. S.; KAHNG, A. B. Cooperative Mobile Robotics: Antecedents and Directions. Autonomous Robots, v. 4, n. 1, p. 7–27, 1997. DOI: 10.1023/A:1008855018923.

CARSTEN, Joseph; FERGUSON, Dave; STENTZ, Anthony. 3d field d: Improved path planning and replanning in three dimensions. In: IEEE. 2006 IEEE/RSJ international conference on intelligent robots and systems. [S. l.: s. n.], 2006. p. 3381–3386.

CARVALHO, Lucas L. M. et al. Hybrid Algorithm Based on Bidirectional A* and Floodfill Applied to Coverage Path Planning for AGVs. In: 2025 Brazilian Symposium on Robotics (SBR) and 2025 Workshop on Robotics in Education (WRE). [S. l.: s. n.], 2025. p. 84–89. DOI: 10.1109/SBR/WRE66973.2025.11249510.

CASTRO, Gabriel G. R. de et al. Heterogeneous Multi-Robot Collaboration for Coverage Path Planning in Partially Known Dynamic Environments. Machines, v. 12, n. 3, 2024. DOI: 10.3390/machines12030645.

CASTRO, Gabriel G. R. de et al. Hybrid Wavefront and Bidirectional A* Coverage Path Planning for AGVs in Photovoltaic Farm Inspection. In: 2025 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). [S. l.: s. n.], 2025. p. 64–69. DOI: 10.1109/ICARSC.2025.1234567.

CASTRO, Gabriel GR de et al. Dynamic Path Planning Based on Neural Networks for Aerial Inspection. Journal of Control, Automation and Electrical Systems, Springer, p. 1–21, 2022.

CASTRO, Gabriel GR de et al. Dynamic Path Planning Based on Neural Networks for Aerial Inspection. Journal of Control, Automation and Electrical Systems, Springer, v. 34, n. 1, p. 85–105, 2023.

CASTRO, Gabriel GR de et al. Heterogeneous Multi-Robot Collaboration for Coverage Path Planning in Partially Known Dynamic Environments. Machines, MDPI, v. 12, n. 3, p. 200, 2024.

CASTRO, Gabriel GR de et al. Adaptive Path Planning for Fusing Rapidly Exploring Random Trees and Deep Reinforcement Learning in an Agriculture Dynamic Environment UAVs. Agriculture, MDPI, v. 13, n. 2, p. 354, 2023.

CHEN, Xia; ZHANG, Jing. The three-dimension path planning of UAV based on improved artificial potential field in dynamic environment. In: IEEE. 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics. [S. l.: s. n.], 2013. v. 2, p. 144–147.

CHOSET, Howie. Coverage for Robotics – A Survey of Recent Results. Annals of Mathematics and Artificial Intelligence, v. 31, n. 1, p. 113–126, 2001. DOI: 10.1023/A:1016639210559.

CUI, Zhengyang; WANG, Yong. UAV Path Planning Based on Multi-Layer Reinforcement Learning Technique. IEEE Access, IEEE, v. 9, p. 59486–59497, 2021.

DEBNATH, Sanjoy Kumar et al. Different Cell Decomposition Path Planning Methods for Unmanned Air Vehicles-A Review. In: SPRINGER. PROCEEDINGS of the 11th National Technical Seminar on Unmanned System Technology 2019. [S. l.: s. n.], 2020. p. 99–111.

DIAS, M. Bernardine et al. Market-Based Multirobot Coordination: A Survey and Analysis. Proceedings of the IEEE, v. 94, n. 7, p. 1257–1270, 2006. DOI: 10.1109/JPROC.2006.876939.

DIJKSTRA, Edsger W et al. A note on two problems in connexion with graphs. Numerische mathematik, v. 1, n. 1, p. 269–271, 1959.

GALCERAN, Enric; CARRERAS, Marc. A Survey on Coverage Path Planning for Robotics. Robotics and Autonomous Systems, v. 61, n. 12, p. 1258–1276, 2013. DOI: 10.1016/j.robot.2013.09.004.

GAO, Jun; SUN, Bin; LI, Feng. Dynamic Path Planning for Heterogeneous Robot Teams Using Deep Reinforcement Learning. Sensors, v. 23, n. 8, p. 3834, 2023. DOI: 10.3390/s23083834.

GERKEY, Brian P.; MATARIĆ, Maja J. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. The International Journal of Robotics Research, v. 23, n. 9, p. 939–954, 2004. DOI: 10.1177/0278364904045564.

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, IEEE, v. 4, n. 2, p. 100–107, 1968.

HOLLAND, John H. Genetic algorithms. Scientific american, JSTOR, v. 267, n. 1, p. 66–73, 1992.

HORN, Joseph; GEIGER, Brian; SCHMIDT, Eric. Use of neural network approximation in multiple-unmanned aerial vehicle trajectory optimization. In: AIAA guidance, navigation, and control conference. [S. l.: s. n.], 2009. p. 6103.

HORN, Joseph F et al. Neural network-based trajectory optimization for unmanned aerial vehicles. Journal of Guidance, Control, and Dynamics, v. 35, n. 2, p. 548–562, 2012.

HORNUNG, Armin et al. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. Autonomous Robots, 2013. Software available at http://octomap.github.com. DOI: 10.1007/s10514-012-9321-0.

JARVIS, RA; BYRNE, JC; AJAY, K. An Intelligent Autonomous Guided Vehicle: Localisation, Environmental Modelling and Collision-Free Path Finding. In: PROC. 19th ISIR The Int. Symp. and Exposition on Robots. [S. l.: s. n.], 1988. p. 767–792.

KIM, J.; KIM, M.; PARK, J. Multi-UAV and Multi-UGV Cooperation Strategies for Large-Scale Missions: A Review. IEEE Access, v. 10, p. 92310–92328, 2022. DOI: 10.1109/ACCESS.2022.3199330.

KOENIG, Nathan; HOWARD, Andrew. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566). [S. l.: s. n.], 2004. v. 3, p. 2149–2154.

KORSAH, George A.; STENTZ, Anthony; DIAS, M. Bernardine. A Comprehensive Taxonomy for Multi-Robot Task Allocation. The International Journal of Robotics Research, v. 32, n. 12, p. 1495–1512, 2013. DOI: 10.1177/0278364913496484.

LABBÉ, Mathieu; MICHAUD, François. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of field robotics, Wiley Online Library, v. 36, n. 2, p. 416–446, 2019.

LEI, Xiaoyun; ZHANG, Zhian; DONG, Peifang. Dynamic path planning of unknown environment based on deep reinforcement learning. Journal of Robotics, Hindawi, v. 2018, 2018.

LI, Jian et al. Optimizing Coalition Formation and Path Planning for UAV–UGV Teams in Obstacle-Rich Environments via Multi-Agent Reinforcement Learning. IEEE Robotics and Automation Letters, v. 9, n. 6, p. 5234–5241, 2024. DOI: 10.1109/LRA.2024.3492157.

LIU, J.; ZHANG, H.; MENG, Q. Graph-Theoretic Task Allocation for Heterogeneous Multi-Robot Systems. Robotics and Autonomous Systems, v. 151, p. 104016, 2022. DOI: 10.1016/j.robot.2022.104016.

LUQUE VEGA, Luis F. et al. UAV Flight Instructional Design for Industry 4.0 based on the Framework of Educational Mechatronics. In: IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society. [S. l.: s. n.], 2020. p. 2313–2318. DOI: 10.1109/IECON43393.2020.9255295.

MA, X.; CHEN, L.; ZHAO, F. Adaptive motion planning using deep reinforcement learning with energy constraints for UAVs. IEEE Robotics and Automation Letters, v. 8, n. 2, p. 1050–1057, 2023. DOI: 10.1109/LRA.2023.3234561.

MADRIDANO, Álvaro et al. Collaborative Aerial-Ground Robotic System for Search and Rescue Missions in Large Outdoor Areas. Sensors, v. 21, n. 6, p. 2104, 2021. DOI: 10.3390/s21062104.

MATARIĆ, Maja J. Behavior-Based Robotics: Main Properties and Implications. Proceedings of the IEEE, v. 94, n. 7, p. 1226–1238, 2007. DOI: 10.1109/JPROC.2006.885803.

MICHAEL, Nathan; FINK, Jonathan; KUMAR, Vijay. Cooperative Manipulation and Transportation with Aerial Robots. Autonomous Robots, v. 30, n. 1, p. 73–86, 2011. DOI: 10.1007/s10514-010-9205-0.

MONDAL, Suman et al. Energy-Constrained Multi-Robot Task Allocation and Motion Planning Using Deep Reinforcement Learning. IEEE Transactions on Automation Science and Engineering, v. 22, n. 1, p. 125–139, 2025. DOI: 10.1109/TASE.2025.3498271.

MOTLAGH, Omid et al. Automatic navigation of mobile robots in unknown environments. Neural Computing and Applications, Springer, v. 24, n. 7, p. 1569–1581, 2014.

MUNASINGHE, A.; PERERA, A.; DEO, R. A Review on Collaborative UAV–UGV Systems: Architectures, Challenges, and Applications. Drones, v. 8, n. 2, p. 81, 2024. DOI: 10.3390/drones8020081.

NARVÁEZ, Eduardo et al. Autonomous vtol-uav docking system for heterogeneous multi-robot team. IEEE Transactions on Instrumentation and Measurement, IEEE, v. 70, p. 1–18, 2020.

PARKER, L. E. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. IEEE Transactions on Robotics and Automation, v. 14, n. 2, p. 220–240, 1998. DOI: 10.1109/70.681242.

PINTO, Milena F et al. Case-based reasoning approach applied to surveillance system using an autonomous unmanned aerial vehicle. In: IEEE. 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE). [S. l.: s. n.], 2017. p. 1324–1329.

QIU, Yuchen; LIN, Shao; WANG, Cheng. Distributed Coordination Framework for UAV–UGV Collaborative Mapping in Unknown Environments. IEEE Transactions on Intelligent Vehicles, v. 9, n. 1, p. 556–567, 2024. DOI: 10.1109/TIV.2024.3332198.

QURESHI, Ahmed H et al. Motion planning networks. In: IEEE. 2019 International Conference on Robotics and Automation (ICRA). [S. l.: s. n.], 2019. p. 2118–2124.

RAMOS, Gabryel Silva; PINTO, Milena Faria; HADDAD, Diego Barreto. Advancing UAV swarm autonomy with ARCog-NET for task allocation, path planning, and formation control. Robotica, Cambridge University Press, v. 43, n. 7, p. 2565–2609, 2025.

SCHØLER, Flemming; COUR-HARBO, Anders la; BISGAARD, Morten. Generating configuration spaces and visibility graphs from a geometric workspace for uav path planning. Autonomous Robots, 2012, Citeseer, 2012.

SHAKHATREH, Hazim et al. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. Ieee Access, IEEE, v. 7, p. 48572–48634, 2019.

SHI, Yunfei et al. Adaptive informative sampling with environment partitioning for heterogeneous multi-robot systems. In: IEEE. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S. l.: s. n.], 2020. p. 11718–11723.

STEK, Tesse D. Drones over Mediterranean landscapes. The potential of small UAV's (drones) for site detection and heritage management in archaeological survey projects: A case study from Le Pianelle in the Tappino Valley, Molise (Italy). Journal of Cultural Heritage, Elsevier, v. 22, p. 1066–1071, 2016.

SUN, Y.; LI, P.; ZHANG, T. Learning-based hybrid navigation for multi-robot systems in dynamic environments. Autonomous Robots, v. 48, n. 2, p. 157–174, 2024. DOI: 10.1007/s10514-023-10123-0.

SUNG, Inkyung; CHOI, Bongjun; NIELSEN, Peter. On the training of a neural network for online path planning with offline path planning algorithms. International Journal of Information Management, Elsevier, v. 57, p. 102142, 2021.

TANG, X. et al. Distributed Multi-Robot Task Allocation with Graph-Based Communication Constraints. IEEE Robotics and Automation Letters, v. 8, n. 4, p. 2341–2348, 2023. DOI: 10.1109/LRA.2023.3234561.

TLALE, Nkgatho; VILLIERS, Mark de. Kinematics and dynamics modelling of a mecanum wheeled mobile platform. In: IEEE. 2008 15th international conference on mechatronics and machine vision in practice. [S. l.: s. n.], 2008. p. 657–662.

TONG, GUO et al. UAV navigation in high dynamic environments: A deep reinforcement learning approach. Chinese Journal of Aeronautics, Elsevier, v. 34, n. 2, p. 479–489, 2021.

TSAI, Jinn-Tsong; CHOU, Jyh-Horng; LIU, Tung-Kuan. Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. IEEE Transactions on Neural Networks, IEEE, v. 17, n. 1, p. 69–80, 2006.

WANG, Huijuan; YU, Yuan; YUAN, Quanbo. Application of Dijkstra algorithm in robot path-planning. In: IEEE. 2011 second international conference on mechanic automation and control engineering. [S. l.: s. n.], 2011. p. 1067–1069.

WANG, Lei; CHEN, Jiaqi; HUANG, Yifan. Multi-Layered Coverage Path Planning for UAV–UGV Teams in Large-Scale Environments. IEEE Transactions on Aerospace and Electronic Systems, v. 60, n. 3, p. 2742–2755, 2024. DOI: 10.1109/TAES.2024.3361154.

WEST, Matthew; ASADA, Harry. Design of ball wheel mechanisms for omnidirectional vehicles with full mobility and invariant kinematics, 1997.

WU, Keyu et al. TDPP-Net: Achieving three-dimensional path planning via a deep neural network architecture. Neurocomputing, Elsevier, v. 357, p. 151–162, 2019.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: PROCEEDINGS 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'. [S. l.: s. n.], 1997. p. 146–151. DOI: 10.1109/CIRA.1997.613851.

YAN, Fei; LIU, Yi-Sha; XIAO, Ji-Zhong. Path planning in complex 3D environments using a probabilistic roadmap method. International Journal of Automation and computing, Springer, v. 10, n. 6, p. 525–533, 2013.

YANG, Kwangjin; SUKKARIEH, Salah. Real-time continuous curvature path planning of UAVs in cluttered environments. In: IEEE. 2008 5th International Symposium on Mechatronics and Its Applications. [S. l.: s. n.], 2008. p. 1–6.

YANG, Yang; JUNTAO, Li; LINGLING, Peng. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. CAAI Transactions on Intelligence Technology, Wiley Online Library, v. 5, n. 3, p. 177–183, 2020.

YANG, Yang; JUNTAO, Li; LINGLING, Peng. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. CAAI Transactions on Intelligence Technology, v. 5, n. 3, p. 177–183, 2020. DOI: https://doi.org/10.1049/trit.2020.0024.

ZELINSKY, Alexander et al. Planning paths of complete coverage of an unstructured environment by a mobile robot. In: PROCEEDINGS of international conference on advanced robotics. [S. l.: s. n.], 1993. v. 13, p. 533–538.

ZHANG, H.; LIU, X.; WANG, Q. Hybrid Path Planning with Reinforcement Learning for Dynamic Environments. IEEE Access, v. 10, p. 34252–34264, 2022. DOI: 10.1109/ACCESS.2022.3162049.

ZHANG, Rui; ZHAO, Ming; LIN, Wei. Air–Ground Hierarchical Coverage Path Planning for Cooperative UAV–UGV Systems. Robotics and Autonomous Systems, v. 173, p. 104591, 2025. DOI: 10.1016/j.robot.2025.104591.

ZHANG, Wei et al. Coarse-to-Fine UAV Target Tracking With Deep Reinforcement Learning. IEEE Transactions on Automation Science and Engineering, v. 16, n. 4, p. 1522–1530, 2019. DOI: 10.1109/TASE.2018.2877499.

ZHAO, Yijing; ZHENG, Zheng; LIU, Yang. Survey on computational-intelligence-based UAV path planning. Knowledge-Based Systems, Elsevier, v. 158, p. 54–64, 2018.