FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mission planning of an autonomous surface vehicle for inspection of large infrastructures

Tiago Marques Claro

Master in Electrical and Computer Engineering

Supervisor: Andry Maykol Gomes Pinto

June 30, 2025

Resumo

Com a crescente implementação de sistemas fotovoltaicos em águas interiores e costeiras, a inspeção destas infraestruturas de larga escala apresenta desafios operacionais significativos, especialmente para veículos aéreos não tripulados com autonomia de voo limitada. Em contextos offshore, a distância até ao local de inspeção ultrapassa frequentemente o alcance efetivo destes veículos, comprometendo a viabilidade de soluções de inspeção exclusivamente aéreas.

Esta dissertação propõe uma arquitetura cooperativa que integra um veículo de superfície não tripulado e um veículo aéreo não tripulado para a realização de missões de inspeção coordenadas. O veículo de superfície é concebido como uma plataforma móvel de lançamento e recuperação, permitindo que o veículo aéreo opere mais próximo da zona de interesse e, assim, reduza o tempo de voo necessário. Foi desenvolvida uma metodologia de planeamento de missão espaço-temporal para gerir o comportamento cooperativo entre plataformas, assegurando a execução sincronizada das fases de descolagem, navegação, inspeção e aterragem.

A solução proposta inclui a aquisição automática de mapas de missão com base em dados geoespaciais, otimização de pontos de passagem considerando obstáculos estáticos e dinâmicos, e estratégias híbridas de planeamento de trajetória que combinam planeamento global com desvio local de obstáculos. Foi implementado um controlador temporal modular para gerir a execução da missão, assim como um mecanismo de recuperação de emergência para garantir a segurança da operação em caso de falhas. A plataforma de superfície, designada NEST, foi construída especificamente para esta aplicação, incorporando computação embarcada, interfaces de comunicação e uma plataforma de aterragem integrada num casco tipo catamarã.

A validação foi realizada através de simulação baseada em modelos físicos e ensaios de campo no Centro de Testes ATLANTIS. O sistema demonstrou comportamentos fiáveis de navegação, coordenação de missão e recuperação segura em condições experimentais, confirmando a viabilidade da arquitetura proposta para inspeções autónomas em ambientes marítimos.

Abstract

With the expansion of floating solar energy systems across inland and coastal waters, the inspection of these large-scale infrastructures presents operational challenges, especially for aerial vehicles limited by short flight endurance. In offshore scenarios, the distance to the inspection site often exceeds the vehicle's effective range, reducing the viability of fully aerial inspection solutions.

This thesis presents a cooperative mission architecture that integrates an autonomous surface vehicle and an aerial vehicle to perform coordinated inspection missions. The surface vehicle acts as a mobile launch and recovery platform, enabling the aerial vehicle to operate closer to the inspection area and thus reducing the required flight time. A spatio-temporal mission planning methodology was developed to manage coordinated behavior between platforms, ensuring synchronized execution of takeoff, navigation, inspection, and landing phases.

The proposed solution includes automatic generation of mission maps using geospatial data, waypoint optimization that considers both static and dynamic obstacles, and a hybrid navigation strategy that combines global planning with local obstacle avoidance. A modular temporal controller orchestrates mission execution, while an emergency retrieval mechanism was developed to guarantee safe vehicle recovery under failure conditions. The surface platform, named NEST, was purpose-built for this application, featuring onboard computing systems, communication interfaces, and a landing platform integrated into a catamaran-type hull.

The system was validated through physics-based simulation and field trials conducted at the ATLANTIS Test Center. The experiments demonstrated reliable performance in terms of autonomous navigation, coordinated task execution, and safe vehicle recovery, confirming the feasibility of the proposed architecture for autonomous inspection in maritime environments.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) outline 17 global objectives for building a safer, more inclusive, sustainable, and efficient future. This work contributes to the advancement of the following SDGs:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 14 Conserve and sustainably use the oceans, seas and marine resources for sustainable development

The corresponding targets for these SDGs are named as follows:

Target 8.8 Protect labour rights and promote safe and secure working environments for all workers, including migrant workers, in particular women migrants, and those in precarious employment

Target 9.4 By 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities

Target 14.2 By 2020, sustainably manage and protect marine and coastal ecosystems to avoid significant adverse impacts, including by strengthening their resilience, and take action for their restoration in order to achieve healthy and productive oceans

SGD	Target	Contribution	Performance Indicators and Metrics
8	8.8	Development of safe obstacle avoidance algorithms to minimize human error in vehicles operation	Reduce the number of incidents leading to injuries or fatalities in hazardous workplace environments
9	9.4	Enhancing autonomous vehicle systems for inspection and maintenance tasks, enabling more frequent operations and increasing overall industrial efficiency	Increase in operational efficiency and infrastructure longevity; Reduction in downtime and operational costs
14	14.2	Mitigate the impacts of vehicle operation in marine and coastal ecosystems	Reduction in operation incidents that affect marine ecosystems

Agradecimentos

Gostaria de começar por agradecer ao meu orientador, Andry Pinto, pela oportunidade de integrar a sua equipa e viver uma experiência de trabalho real num ambiente de investigação. Agradeço também ao meu coorientador, Daniel Campos, por todo o apoio prestado ao longo desta dissertação, estando sempre disponível sempre que precisei.

Um agradecimento especial a toda a equipa do MOTUS — Pedro Leite, Maria Pereira, Renato Silva, Pedro Pereira, João Dionísio, Francisco Neves, Celso Pereira, João Campanhã, Luís Branco, Tiago Sousa, Lourenço Pinho e Rafael Claro — pelo companheirismo, pela partilha de conhecimento e pela constante disponibilidade para ajudar, mesmo quando isso significava adiar os seus próprios objetivos.

Quero ainda agradecer aos meus companheiros de tese, Francisco Silva, Alexandre Vilhena e Bruno Costa, por partilharem comigo esta jornada e por tornarem o nosso grupo de bambis no MOTUS numa verdadeira equipa.

Por fim, deixo um profundo agradecimento à minha família, pelo apoio incondicional e pelas ferramentas que sempre me proporcionaram para alcançar esta meta. E também à SEDE, a casa que me acolheu durante grande parte desta fase universitária, e que foi, muitas vezes, o meu refúgio e ponto de encontro.

Tiago Claro

Official Acknowledgments

This work is partly funded by the European Union's Horizon 2022 research and innovation programme under grant agreement No 101119744 (TALOS).

Tiago Claro

"Sometimes my genius... it's almost frightening."

Jeremy Clarkson

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

2D	Two Dimensional
3D	Three Dimensional
ACO	Ant Colony Optimization
AMPSO	Adaptive Mutual learning Particle Swarm Optimization
ANN	Artificial Neural Network
APF	Artificial Potential Field
ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
AVOA	Adaptive Velocity Obstacle Avoidance
DA-APF	Deterministic Annealing Artificial Potential Fields
DW4DOT	Dynamic Window for Dynamic Obstacles Tree
DWA	Dynamic Window Approach
EDP	Energias de Portugal
EKF	Extended Kalman Filter
ENU	East-North-Up
GPS	Global Positioning System
IR	Infrared
LiDAR	Light Detection and Ranging
MBES	Multibeam Echosounder
MRTA	Multi-Robot Task Allocation
NED	North-East-Down
NDT	Non-Destructive Testing
NEST	Non-stationary Emersed Structure for Takeoff-landing
O&M	Operation & Maintenance
PSO	Particle Swarm Optimization
PV	Photovoltaic
QGC	QGroundControl
ROV	Remotely Operated Vehicle
RRT	Rapidly-exploring Random Tree
SCC	Shore Control Center
SITL	Software In The Loop
TRP	Traveling Repairman Problem
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
USV	Unmanned Surface Vehicle
UTM	Universal Transverse Mercator
VO	Velocity Obstacles

Chapter 1

Introduction

Due to the climate change, finding new solutions for sustainable energy is a critical need. Portugal's current average temperature has risen approximately 1ºC compared to pre-industrial times, leading to significant impacts on human populations and ecosystems [1]. Nonetheless, Portugal has emerged as a front-runner in Europe with regard to renewable energy, accounting for 63% of total energy consumption in 2023. Photovoltaic (PV) panels, particularly floating PV systems [2], are increasingly explored in Portugal, where abundant sunlight supports their expanding adoption. This technology, deployed on large bodies of water, converts solar energy into electric energy while offering environmental benefits. In dry and hot climates, solar installations on lakes and reservoirs provide the added benefit of reducing water surface radiation, which helps to minimize evaporation [3]. Water-based PV systems improve efficiency by approximately 7% compared to ground-mounted PV panels, as the cooling effect of the water mitigates overheating of the panels [4]. The solar plant in the Alqueva Dam, shown in figure 1.1, is an example of an already functioning photovoltaic farm. This solar plant developed by EDP is able to produce 7.5GWh annually over its twelve thousand PV panels and it's the largest photovoltaic farm installed in a reservoir in Europe[1].

Periodic maintenance is essential to ensure that the panels operate at maximum efficiency and continue producing optimal power output [5]. Maintenance activities typically involve thermal inspections with the aim of searching for accumulated leads, snow, soiling or bird droppings that cause the shading of the PV cells, also find modules in open and closed cycle, broken front glass, loss of connection with the junction box, as well as cells with temperature variation as described in the normative developed by International Electrotechnical Commission (2017) [6]. Monitoring the solar panels to ensure their optimal performance and prevent shading can lead to an increase of 8.7% of overall power production, while avoiding 81% of voltage sag and lowering distortions to 2% [7].

Inspection tasks are currently conducted mainly through human workers that face several challenges, namely,low safety in hard-to-reach areas that require navigating unstable platforms or

[1] https://www.edp.com/pt/europa/portugal/projetos/paineis-solares-flutuantes (Accessed in 24-06-2025)

[2] https://www.euronews.com/next/2023/01/20/could-floating-solar-panels-be-a-solution-to-both-the-climate-crisis-and-soaring-energy-pr (Accessed in 24-06-2025)

Figure 1.1: Solar Power Plant in Alqueva Dam [2].

bodies of water. This can lead to drowning, falls and exposure to harsh weather conditions. Additionally, manual inspections are time-consuming and repetitive, requiring workers to physically traverse the entire array of panels. Furthermore, the demand for skilled workers and high specialized equipment and safety will increase the costs of maintaining a floating solar plant [8].

Autonomous systems, such as Unmanned Aerial Vehicles (UAV) and Unmanned Surface Vehicles (USV), can enhance the inspection of PV panels by introducing robustness and reliability to the acquired data. Technological developments (sensors, technical cameras and control systems) in the aerial industry also provide opportunities for monitoring large areas using UAVs equipped with a payload of sensors and autonomously acquire the data from the PV farm [9, 10, 11].

1.1 Motivation

As PV farms continue to expand, challenges related to the limited autonomy of UAVs become more prominent. This issue is especially critical in floating PV farms, where landing sites are scarce. Since the start of UAV operations are from land-based locations, increases in travel distance, reduces the UAV's available inspection time and overall efficiency [12].

Combining multiple vehicles for inspection tasks can significantly enhance operational coverage, enabling the inspection of larger areas and increasing overall efficiency. For instance, employing a combination of a USV and a UAV [13], as illustrated in figure 1.2, allows the USV to act as a mobile landing pad for the UAV. This setup reduces the UAV's energy consumption during transit from the shore to the PV farm. Once at the site, the UAV can carry out aerial inspections, while the USV conducts surface-level inspections or provides support for the UAV's operations, such as emergency landing and communication relay. After completing the tasks, the USV recovers

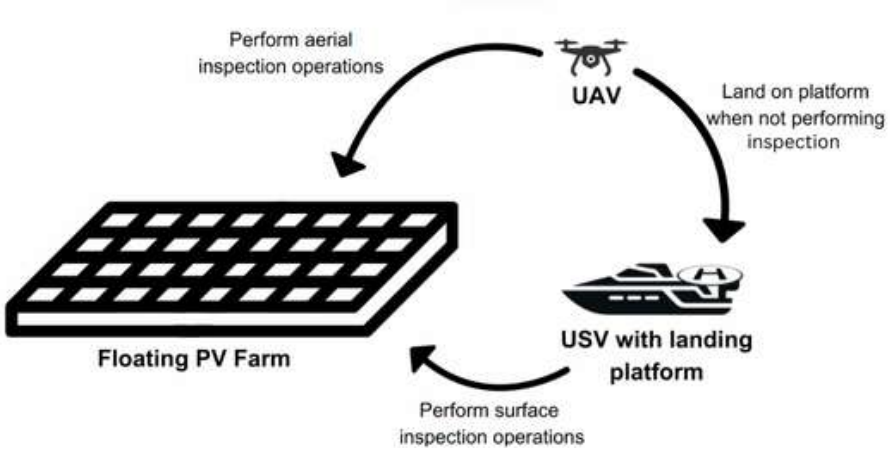


Figure 1.2: Operational scheme of the UAV and USV in inspecting the floating PV farm.

the UAV and transports it back to shore. However, this multi-vehicle approach introduces significant challenges, such as the need to synchronize tasks across both platforms while optimizing constraints related to time, energy efficiency, and spatial positioning.

This dissertation introduces a spatio-temporal mission planning methodology, where the USV serves as a mobile platform for an UAV that conducts offshore inspections. This approach aims to optimize inspection operations by using the USV to transport the UAV to specific locations, enabling autonomous task execution and retrieval considering all the constraints imposed by this system.

1.2 Objectives

Optimizing endurance and minimizing time consumption are crucial for the operation and maintenance (O&M) of photovoltaic power plants. This work focuses on mission planning for a USV-UAV setup that autonomously inspects a floating PV farm, aiming to optimize inspection efficiency by minimizing travel time. Conducted within the scope of the European TALOS project, this work aims to be demonstrated using autonomous vehicles at the Alqueva Dam, showcasing its real-world applicability. The following list specifies the objectives towards this dissertation:

- Enhance USV capabilities to support autonomous aerial-surface operations by integrating a UAV-compatible landing pad and increased autonomy;
- Enable safe autonomous operations for USVs, through the integrations of mission planning, navigation and collision avoidance techniques;
- Development of a spatial and temporal scheduling algorithm to optimize USV-UAV cooperation, ensuring efficient task allocation and synchronization in dynamic operational environments;
- Validation in simulation and near-real scenarios of the USV-UAV cooperation task.

1.3 Document Structure

This document is structured as follows:

Chapter 2 provides a detailed literature review on physical interactions between UAVs and USVs, highlighting progress and addressing unresolved challenges in this domain. It also examines path planning algorithms, focusing on their use in dynamic and complex environments. Finally, the chapter explores decision-making processes under temporal constraints.

Chapter 3 presents the methodology adopted in this work. It begins with a definition of the problem statement, followed by a description of the USV used for experimental validation. The chapter then details the system architecture and the mission planning methodology, including mission definition, map acquisition, navigation strategies, and the implementation of failsafe mechanisms.

Chapter 4 outlines the simulation experiments and their corresponding results. Additionally, it includes real-world tests conducted in operational environments to validate the proposed methodology.

Chapter 5 presents the conclusion of this work and presents future work improvements.

Chapter 2

Literature Review

Floating photovoltaic farms are emerging as a promising renewable energy source, particularly in regions with limited land and high sunlight exposure. However, operating and maintaining these installations efficiently on water surfaces introduces several challenges. Conventional inspection methods for PV panels typically require human operators, who face the risks and difficulties of the water environment, such as drowning, exposure to harsh weather and equipment damage. This literature review explores the potential of USV-UAV cooperation to overcome these challenges, enabling multi-domain inspections. In this system, the USV serves as a carrier for the UAV, improving its autonomy while providing safer navigation in a challenging environment.

This chapter begins by providing a general overview of the challenges associated with the operation and maintenance of floating photovoltaic (PV) farms, emphasizing the potential of autonomous systems to address these issues. Then it explores the concept of mission planning, highlighting its critical role in enabling autonomous systems to perform inspection tasks effectively. This section is divided into subsections that detail the global path planning techniques, which rely on pre-existing knowledge of the environment, the local path planning methods, designed for real-time obstacle avoidance in dynamic settings, and the hybrid path planning approaches, which combine global and local methods to balance long-term optimization with adaptability. Finally, the chapter explores decision-making processes, focusing on Multi-Robot Task Allocation (MRTA) frameworks and strategies for coordinating autonomous vehicles, ensuring efficient task execution and collaboration in complex environments.

2.1 General Overview

The operation and maintenance (O&M) of PV parks present several challenges due to their demanding installation environments. Employing a large fleet of maintenance technicians is costly[1], especially for offshore installations, which need specialized training and equipment. Inspections of

[1] https://commercialsolarguy.com/free-government-research-spreadsheet-model-of-solar-power-operations-and-maintenance-costs/ (Accessed in 07-12-2024)

PV systems are currently conducted using a combination of manual and automated methods. Onshore facilities often rely on technicians performing physical inspections or using handheld tools such as thermal and visual cameras to identify issues like overheating or shading [14]. In some cases, drones equipped with these cameras and sensors are deployed to cover large areas more efficiently. However, offshore and floating PV systems introduce additional challenges. Weather conditions like high waves and strong winds can limit access and complicate the transfer of personnel and equipment between vessels and PV arrays, further increasing operational complexity and costs. Specialized vessels and remote monitoring systems are increasingly employed in such environments to address these limitations, but the costs remain significantly higher compared to onshore facilities. Taking into account these constraints, an autonomous system can replace human workers and conduct more reliable inspections when equipped with visual and thermal cameras.

An emerging approach in autonomous systems involves the cooperation of multiple agents to enable operations across multiple domains. For instance, the use of UAVs enhances inspections by enabling close examination of PV panels from above. However, these vehicles have limited autonomy, which makes the cooperation between UAVs and USVs with landing platforms highly beneficial. The USV, operating in a more suitable domain, offers greater autonomy and can optimize the inspection process by extending the operational range of the UAV [15].

Physical cooperation between multiple vehicles presents a challenging coordination problem; however, it has been extensively studied by the scientific community. This coordination can be categorized into static and dynamic approaches [16]. Static coordination involves setting up a predefined framework of rules or protocols prior to task execution. While this approach is effective for managing complex tasks, it may struggle with real-time control, as it lacks flexibility to adapt quickly to dynamic situations or unforeseen challenges. Conversely, dynamic coordination relies on real-time analysis of available information during the task to determine subsequent actions. A marsupial relationship between vehicles, in which each vehicle plays a distinct role within a cooperative team, enhances the strengths of each vehicle type, enabling them to collaboratively perform complex tasks across diverse environments that would be challenging to navigate independently. In this relationship, the vehicles assume one of two roles: the container, typically a larger vehicle that provides primary support and transportation; and the passenger, a smaller vehicle carried by the container to specific areas where it performs designated tasks. The container vehicles can be classified within a large set of roles [17]:

- Carrier Role: The core function is physically transport the passenger vehicle, enabling it to conserve energy and avoid challenging terrains they are not equipped to handle;
- Coach Role: Once deployed, the container vehicle can move to different vantage points, using its sensors to monitor and guide the passenger, improving the effectiveness in the field;
- Manager Role: The container vehicle assigns and coordinates specific tasks for each passenger, functioning as a central command unit for task allocation;

- Messenger Role: In environments where communication is though, the container serves as a relay, connecting passengers with each other or with external controllers;
- Processor Role: When passengers lack computing power, the container can take on data-heavy tasks. This allows passengers to send data to the container, which processes it and returns the results, maximizing efficiency;
- Supporter Role: The container additional logistical support, such as recharging the passenger batteries, carrying supplies, and clean the path from obstacles.

This classification serves diverse applications by aligning each vehicle's role with specific operational requirements. Pinto *et al.* (2014) [18] developed a marsupial system for monitoring the river environment, using a USV as the container vehicle and an UAV as the passenger vehicles, allowing a better overview of the river environment and also providing a safe landing platform in the USV. This method optimized the energetic autonomy for long-lasting operations and protected the aerial vehicle in harsh weather conditions.

To handle the take-off and landing behavior Kalaitzakis *et al.* (2021) [19] proposed the state-machine present in the figure 2.1. The *Ride* state, represents the initial condition, is when the UAV is attached to the USV and awaiting a task request. Upon receiving a request, the UAV transitions to the *Take-Off* state, marking the start of its mission. After reaching a designated altitude, the UAV enters the *Follow* state, where it maintains a safe distance from the USV as both vehicles move. If the UAV loses track of the USV, the state changes to either *Hover* or *Return*,

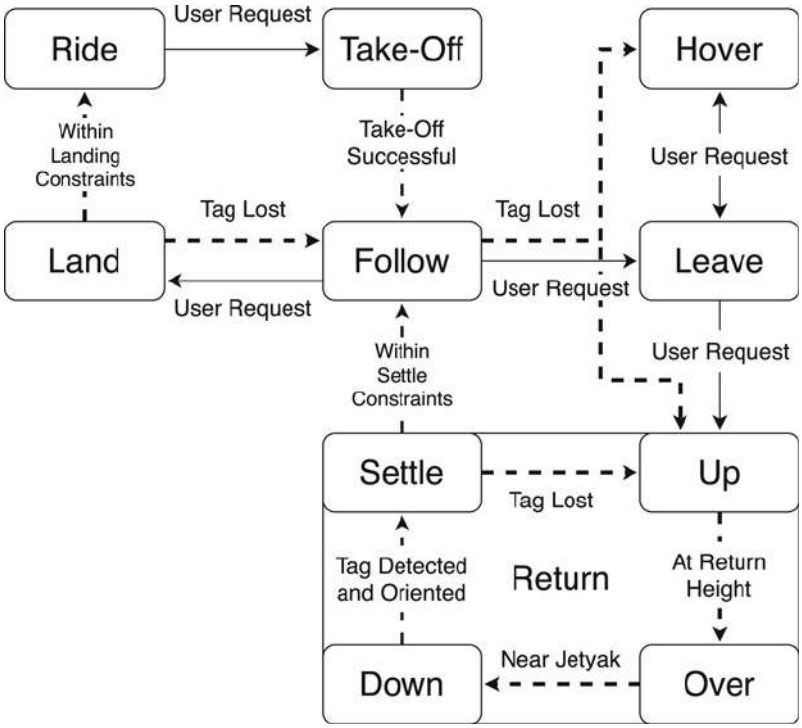


Figure 2.1: State machine diagram of the behaviors [19]. Dashed lines signal automatic transitions with conditions. Solid lines show where user input is required for transitions.

depending on the desired configuration. The *Leave* state enables the UAV to operate at a remote location, useful for tasks like mapping or monitoring. After completing such tasks, the UAV enters the *Hover* state, holding position and altitude as a standby mode. The *Return* state may then be triggered to bring the UAV closer to the USV. Finally, the *Land* state facilitates the UAV's docking on the USV; this state is accessible only from the *Follow* state to ensure proximity between the UAV and USV before landing.

2.2 Mission Planning

The demand for autonomous systems capable of completing costly, expensive, and dynamic missions has risen over recent years. The missions are defined as the extent of objectives and tasks the vehicle must achieve for the mission success [20]. To achieve the mission success two main levels need to be accomplished, namely, the path planning, responsible for the motion of each vehicle regarding the individual constraints, and the decision making domain, responsible for the coordination of multiple agents.

Path planning is a non-deterministic polynomial-time (NP-hard) problem that involves finding a continuous path from an initial to a final goal configuration. There are numerous path planning algorithms, each distinguished by the extent of environmental knowledge available [21]. Global path planning aims to find the optimal path when the environment is largely static and fully known, making it most effective [22]. In contrast, local path planning is used in unknown and dynamic environments, where obstacles may move which requires real-time obstacle avoidance algorithms [23]. These algorithms operate while the robot is in motion, relying on data from local sensors to navigate and adapt to the surroundings in real-time.

The decision making process in autonomous systems is often framed as a scheduling problem, which focuses on optimizing the mission by efficiently allocating tasks and required skills to the vehicles within a cooperative framework. This process ensures that resources are used effectively, and the mission is completed within the defined constraints. The cooperative domain is inherently influenced by bidirectional constraints that each vehicle imposes on the other, given that they operate across distinct domains with differing capabilities and requirements. In a system where one vehicle may rely on the other for transportation or task execution, introduces dependencies between their actions. The task allocation system must, therefore, be robust enough to handle dynamic changes in both environmental conditions and operational constraints. This flexibility allows the system to adapt to unexpected situations, such as changes in available time, task priority, while still optimizing the overall mission performance. By considering these factors, the decision-making process ensures that the vehicles work together effectively, enhancing mission success and efficiency.

2.2.1 Global Path Planning

Global path planning relies on pre-existing maps of the robot's surroundings, giving it complete information about the environment before its operation. To develop a global path plan, the prin-

ciples illustrated in figure 2.2 must be followed. The figure breaks down into three fundamental steps [24]:

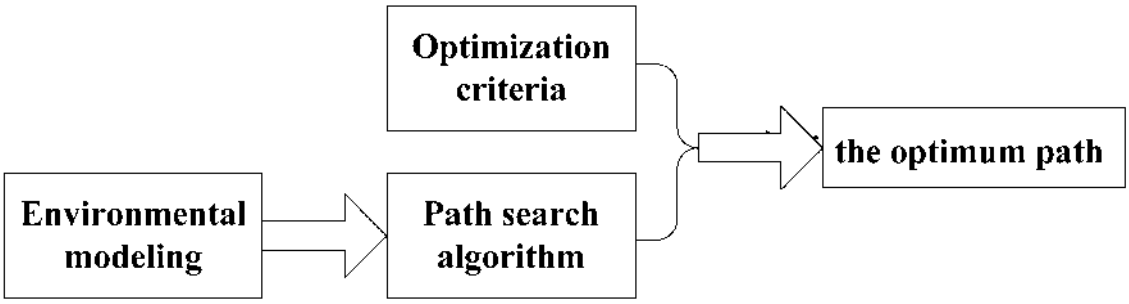


Figure 2.2: The principle of autonomous system global planning [24].

- Environmental modeling – This is the first step that involves creating a representation of the environment in which the robot will operate. As Zhang *et al.* (2018) [24] described, there are different approaches to this modeling:
  - Framework Space Approach involves modeling the environment by fitting a graph to the space, where obstacles are defined as regions that the robot cannot occupy [25]. This approach is most effective for systems with high degrees of freedom.
  - Free Space Approach constructs the navigable free space between obstacles within the robot's environment, focusing on free convex regions. This method is better suited for simpler environments where free space is easy to define.
  - Cell Decomposition Approach partitions the environment into basic, connected regions known as cells. These cells can take rectangular or polygonal shapes and must be distinct, non-overlapping, and adjacent. Cells containing obstacles are labeled as occupied, while those without are considered free of obstacles [26].
  - Topology Method simplifies the path planning process by converting a complex, high-dimensional problem (involving numerous positions or orientations) into a lower-dimensional problem that emphasizes connectivity within the environment. This approach reduces the effort needed to find a path, as it focuses on determining whether different regions of the environment are connected, rather than navigating through the entire space [24].
  - Probabilistic Roadmap Method involves randomly selecting points within the environment and connecting them to create an undirected graph, known as a roadmap. This roadmap represents the connectivity of the free space, turning the path planning problem into a graph search problem, which can often simplify the process [22].
- Optimization Criteria – To determine the most optimal path, several factors must be taken into account. The following list outlines three of the most commonly used optimization criteria:

  - Path Length objective is to minimize the path length as much as possible. To compute the length of a given path, the lengths of each segment that make up the path are summed. The length of a segment is determined by the Euclidean distance between its endpoints [27];
  - Smoothness is used to avoid sharp turns in the robot's path [28];
  - Safety Degree is the sum of deviation degrees between any segment in a path and its nearby obstacle [29].

- Path Search Algorithm – This component is responsible for determining the best path based on the environment and the selected optimization criteria. These algorithms can follow two different approaches: a heuristic approach or an artificial intelligence approach.

Dijkstra (1959) [30] presented the Dijkstra algorithm, a heuristic technique used to address the shortest path routing problem in 2D autonomous system path planning. It seeks to identify the shortest path between two nodes in a directed graph (digraph). In figure 2.3, a graph with five nodes is shown. To find the shortest path from node *A* to any other node, the Dijkstra algorithm is employed. The algorithm evaluates all possible paths, and the shortest path to each node is saved. Table 2.1 presents the sum of the shortest paths from node *A* to every other node.

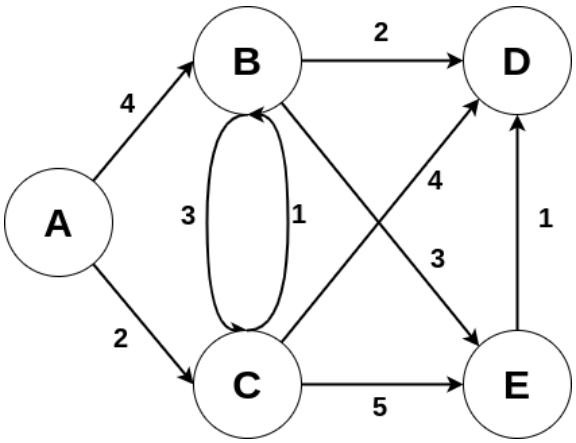


Figure 2.3: Weighted graph.

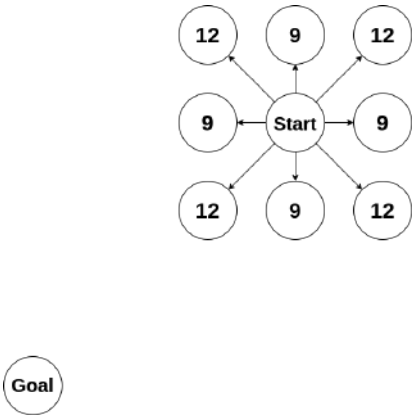
Table 2.1: Weight of the shortest path from node *A* to every other node.

A	B	C	D	E
0	3	2	5	6

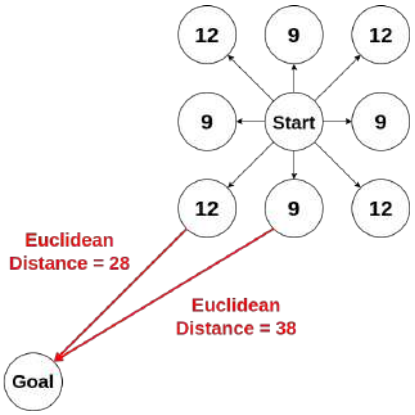
The algorithm operates on the principle of finding the shortest Euclidean distance between nodes. However, when applied to a curved surface, the accuracy of the algorithm decreases with increasing curvature. This is because Euclidean distance between two nodes differs from the surface distance between two points, leading to potential errors in path planning on curved surfaces [31]. An example of the implementation of this algorithm in robot path planning was

presented by Wang *et al.* (2011) [32], demonstrating how this method solves a maze robot path planning problem. The results were positive; however, this method requires full computation of the robot's workspace, which can lead to high computational demands in large environments with many nodes.

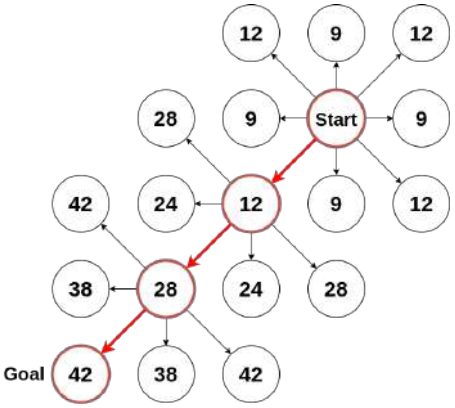
Another well-known algorithm for finding the shortest path is the A* algorithm, developed by Hart *et al.* (1968) [33]. The primary goal of this algorithm is to make decisions as informed as possible. This means that expanding a node that will never be part of an optimal path is wasted effort, while ignoring nodes that may lie on the optimal path is not acceptable. The A* algorithm is based on the principles of Dijkstra's algorithm but avoids the need to calculate the costs to all nodes in the graph. Instead, it selects the next node in the path by exploring the node with the least cost to the final destination [34]. In figure 2.4a, the first step of this algorithm is illustrated. Initially, the costs for each node around the start node are calculated. Then, as shown in figure 2.4b, the Euclidean distance from each node to the goal node is computed to identify the node closest to the goal. Once this node is found, only that node is explored. Upon reaching the goal, as depicted in figure 2.4c, the algorithm succeeds without the need to search the entire environment.



(a) First step.



(b) Iterative step.



(c) Final step.

Figure 2.4: A* Algorithm Procedure.

Although the A* algorithm is simple and faster than the Dijkstra algorithm, it has some limitations that may make it unsuitable for certain cases. A* requires the selection of specific parameters to function as intended, and these parameters greatly influence its performance [35]. Additionally, in large graphs, the algorithm can become inefficient, leading to slower processing times. Tang *et al.* (2021) [36] proposed the Geometric A* Algorithm, which generates an optimal path using the A* algorithm, followed by a screening function to correct path irregularities, and finally smooths the entire path using an interpolation algorithm. This approach demonstrated a high success rate, effectively reducing the number of turns made by the autonomous system, thereby shortening, smoothing the path and reducing the environment search area. However, this proposal did not account for the robot's size, which plays a crucial role in collision avoidance, and it was only applied to static environments. However, in the context of a autonomous system path planner, the environment must be sensed and captured by the robot. In this regard, Stentz (1994) [37] proposed the D* algorithm, which is similar to the A* algorithm but is designed for dynamic environments. The primary objective of this path planner is to avoid all obstacles in the environment while minimizing the cost function.

Lavalle (1998) [38] introduced a random approach to path search called Rapidly-exploring Random Tree (RRT), which has the advantage of being directly applicable to nonholonomic and kinodynamic planning[2]. The RRT algorithm involves selecting random points around the starting point and connecting them to form a tree-like structure. In figure 2.5, x_{start} denotes the starting point. The point x_{rand} represents a randomly selected location within the environment. Next, x_{new} is placed along the line connecting x_{rand} and $x_{nearest}$, considering the maximum predefined step length, and it is subsequently connected to the $x_{nearest}$ point.

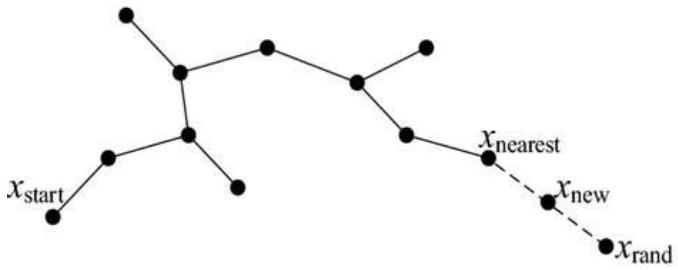


Figure 2.5: Rapidly-exploring Random Tree Algorithm [39].

Then more random points are selected in the unexplored area and connected to the closest point from the tree, progressively expanding it. Once the tree reaches the final goal, it traces back to the starting point to determine the path. The performance of this algorithm improves with an increasing number of points. Although this algorithm provides a good path plan, it is not asymptotically optimal and suffers from slow convergence and weak target tendency, particularly in maps with many narrow passages. To address these issues, Erke *et al.* (2020) [35] developed

[2]Nonholonomic and kinodynamic planning refer to path planning systems that account for movement constraints and consider the system's dynamics to ensure efficient trajectories.

RRT*, which introduces the re-selection of parent nodes and rerouting mechanisms to optimize the path. These operations result in shorter paths. However, this process is also slow, prompting researchers to propose the idea of dynamic step length, which limits sampling to create narrower trees toward the target point [40]. To tackle the problem of narrow passages, often encountered in complex environments, researchers have developed an improved algorithm capable of detecting these narrow passages, significantly enhancing the efficiency of the basic RRT algorithm in such environments [41, 42].

The genetic algorithm, introduced by Holland (1992) [43], is inspired by natural selection. Artificial entities explore new regions of the search space using genetic operators such as crossover, which resembles the reproduction phase and combines different properties of the entities. This is followed by the mutation phase, which introduces random changes to explore additional paths. The process concludes with the selection operation, where the least fit individuals are removed, leaving only the most effective path-exploring individuals for the crossover process. This cycle is repeated until the goal point is reached [44].

Bio-inspired algorithms significantly contribute to path planning, with Ant Colony Optimization (ACO), developed by Dorigo (1992) [45], serving as example. This algorithm is inspired by the behavior of ants, which uses pheromones on the ground for communication, thereby marking the paths that other ants in the colony should follow [46]. ACO has been applied to a broad spectrum of problems, particularly those classified as *NP*-hard. These problems are characterized by their optimal solutions having exponential time complexity in the worst case. However, the effectiveness of this approach heavily relies on the appropriate selection of weights, as incorrect values can lead to local optima traps [47, 48].

2.2.2 Local Path Planning

Local path planning is an approach in which the autonomous system gathers information about its surroundings to determine the path in real time while avoiding obstacles.

The Artificial Potential Field (APF) algorithm, proposed by Khatib (1986) [49], models the autonomous system as a particle within a potential field. In this framework, the goal exerts attractive forces on the robot, while obstacles generate repulsive forces as shown in the figure 2.6. However, this approach often converges to local minima, which can cause the algorithm to become trapped in a deadlock. Additionally, the algorithm exhibits poor performance when the target point is situated near obstacles, making detection more challenging.

To address the issue of local minima, Duan *et al.* (2022) [51] proposed an improved APF that incorporates a secondary virtual target attraction potential field. The results demonstrated its effectiveness in resolving the local minima problem. Another approach was introduced by Wu *et al.* (2023) [52], where the deterministic annealing APF (DA-APF) integrates a temperature parameter to enhance the repulsive potential of obstacles. This method simulates annealing and

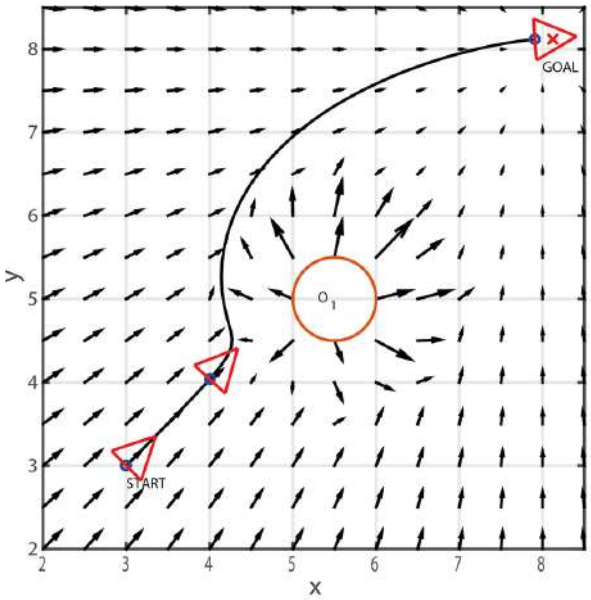


Figure 2.6: Traditional Artificial Potential Field path planning [50].

tempering processes, allowing the robot to escape local minima by employing a gradient descent potential-guided strategy that directs the robot toward the target point.

The Dynamic Window Approach (DWA), originally proposed by Fox *et al.* (1997) [53], samples multiple velocity values and simulates the trajectory based on the robot's dynamics, accounting for its limited velocities and accelerations. This algorithm is primarily designed for static environments. To address its limitations with moving obstacles, Molinos *et al.* (2019) [54] developed an enhanced version of DWA for dynamic environments. The Dynamic Window for Dynamic Obstacles Tree (DW4DOT) algorithm builds upon DWA by improving the safety of the robot's motion and enhancing obstacle avoidance maneuvers. It extends the planning horizon beyond the robot's local window, predicting possible paths further in time using a tree-building method, thereby reducing the number of paths evaluated. While this algorithm has demonstrated promising results, it is computationally intensive, which may limit its suitability for certain real-world applications.

The Timed-Elastic-Band (TEB) algorithm [55] represents the robot's path as a sequence of points, continuously adjusting it to maintain smoothness, avoid obstacles, and respect the robot's motion constraints. This is achieved by "stretching" or "compressing" the elastic band that models the vehicle's path. Due to its ability to update the path online, this algorithm is well-suited for dynamically changing environments. However, the TEB algorithm tends to get stuck in local minima. To address this, Wu *et al.* (2021) [56] proposed an enhanced version of TEB, which not only focuses on avoiding local minima and improving path smoothness but also aims to minimize the path length. This enhancement resulted in a 5% reduction in planning time in comparison with the original TEB algorithm.

Fiorini and Shiller (1998) [57] introduced the Velocity Obstacles (VO) algorithm, which offers a straightforward geometric framework for potential avoidance maneuvers and unifies the handling

of both moving and stationary obstacles. This algorithm predicts future obstacle trajectories and identifies the robot's velocities that may result in collisions. Based on these predictions, the autonomous system selects a velocity and direction that steer clear of collision-prone areas. More recently, Campos *et al.* (2019) [58] proposed an algorithm specifically designed for USV obstacle avoidance, which estimates collision-free velocities. This approach incorporates a protective zone model that accounts for kinematic constraints and evaluates danger using a multi-criteria metric. The algorithm represents danger zones as cones, adjusting them according to obstacle velocities. Figure 2.7 depicts the protective zone of the USV (center) for both static (left) and dynamic (right) obstacles. The AVOA algorithm showed a travel time improvement of at least 58% compared to the TEB algorithm.

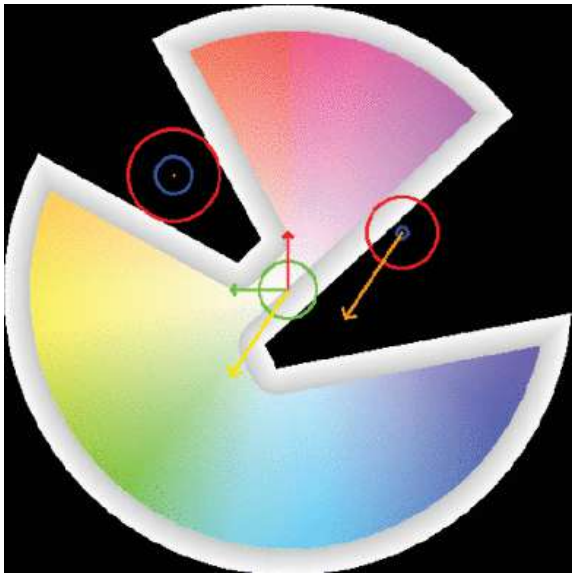


Figure 2.7: AVOA protective zone representation [58].

2.2.3 Hybrid Path Planning

Hybrid path planning effectively balances the trade-offs between long-term optimality and real-time adaptability. By integrating global and local methods, it ensures that the vehicle navigates efficiently over long distances while dynamically avoiding unforeseen obstacles. The global planner ensures an optimal route for reaching a target, while the local planner reacts to dynamic elements such as moving vessels or changing currents. This combination is particularly effective for USVs operating in unpredictable and changing environments.

A variety of hybrid algorithms have been proposed to address the shortcomings of standalone planners. For instance, the combination of Ant Colony Optimization (ACO) with Artificial Potential Fields (APF) allows for efficient exploration of unknown environments while ensuring smooth collision avoidance. Additionally, various algorithmic combinations can yield favorable results. For example, Chen *et al.* (2019) [59] proposed an algorithm that integrates the A* algorithm with

the Dynamic Window Approach (DWA), demonstrating improvements in travel time optimization and effective dynamic obstacle avoidance.

One of the primary challenges in hybrid path planning is maintaining computational efficiency while balancing long-term and short-term navigation goals. As the complexity of the environment increases, so does the demand for real-time processing, which can be limited by hardware constraints and sensor inaccuracies. Addressing these challenges requires the development of more efficient algorithms that can dynamically adjust to the complexity of the environment without sacrificing safety or performance.

2.2.4 Decision Making

To develop a fully operational mission planner for a marsupial vehicle, a robust decision-making process is essential. This process coordinates system constraints and schedules tasks to ensure mission efficiency. Critical to decision-making is assessing task feasibility to prevent time and resource wastage on unfeasible operations. A variety of constraints must be integrated to ensure mission effectiveness, such as time constraints, which dictate that certain tasks be completed within specific time frames; precedence constraints, which require tasks to be performed in a particular sequence; and cooperation constraints, which involve tasks needing collaboration among multiple robots.

In mission planning for multi-robot systems, Multi-Robot Task Allocation (MRTA) is a common approach and can be viewed as a combinatorial optimization problem structured within graph-based frameworks [60]. According to Gerkey and Matarić (2004) [61], MRTA can be categorized as follows:

- Single-task robots/Multi-task robots: Robots perform only one task at a time/Robots are capable of executing multiple tasks concurrently;
- Single-robot tasks/Multi-robot tasks: Tasks that require a single robot/Tasks that need multiple robots for completion;
- Instantaneous assignment/Time-extended assignment: Tasks are assigned to robots either instantaneously without future planning/Tasks that are assigned as a sequence of planned tasks.

This classification provides a structured method for characterizing MRTA problems based on these combinations. Furthermore, additional MRTA taxonomies expand Time-Extended assignments (TA) into two sub-problems, with temporal and ordering constraints [62]. Temporal constraints are expressed as Time Windows (TW), representing the time interval during which a task can start or end, with bounds for early and late start/finish times. Synchronization and Precedence constraints (SP) impose a required order in which tasks must be completed.

MRTA approaches can also be classified as either centralized or decentralized:

- Centralized methods: Mission coordination is managed by a single leader. This includes a weakly centralized approach where the leader role is dynamic and can change based on conditions such as environment or robot battery levels, and a strongly centralized approach, where a single leader remains fixed throughout the mission.
- Decentralized methods: This includes hierarchical and distributed approaches. In the hierarchical model, multiple leaders exist, with robots organized into groups. In a distributed model, robots operate independently without a designated leader.

To classify MRTA algorithms, Gerkey and Matarić (2004) [61] identifies three types of task assignments:

- Offline assignment: Tasks are allocated prior to mission commencement, making this approach ideal for stable environments but unsuitable for dynamic environments.
- Iterated assignment: Tasks are allocated during mission execution, allowing reallocation of tasks even if previously assigned tasks remain incomplete.
- Online assignment: Tasks are allocated continuously during the mission without overriding previously assigned tasks.

Given the dynamic nature of autonomous systems, mission planners often need to address unpredictable factors that may affect performance and potentially disrupt planned constraints. In such cases, the mission planner must adapt to unforeseen situations. For instance, if a task takes longer than expected, subsequent tasks may need to be rescheduled to remain feasible within the set time constraints.

Bischoff *et al.* (2020) [63] proposed a model that decomposes the mission into start and end points, tasks, and precedence, which is represented as a graph in figure 2.8. In this graph, pentagons represent the robots' starting points, circles indicate tasks, and triangles mark the end points. Colored arrows show the task sequence performed by each robot, while the black arrow denotes the precedence of task t_1 over t_3. This graph-based approach evaluates mission feasibility by ensuring that all assigned tasks can be completed within a finite time, maintaining a logical and executable sequence for the robots.

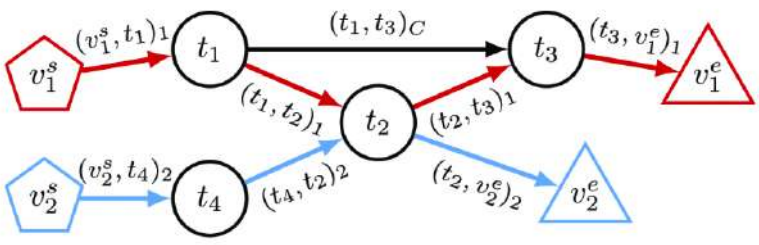


Figure 2.8: Mission plan in a graph representation [63].

Moreover, having the MRTA fully characterized all constraints should be integrated into a cost function aimed at minimizing overall cost while maximizing task completion within the shortest time frame. This task allocation problem resembles the Traveling Repairman Problem (TRP) [64], a variant of the Traveling Salesman Problem (TSP). Unlike the TSP, which focuses on minimizing total travel distance, the TRP aims to minimize average waiting time.

To optimize the mission by minimizing travel time and maximizing the rewards collected at each waypoint, Pinto *et al.* (2020) [65] introduced a cost function, represented in equation 2.1. This function combines a time cost component J_{time} and a reward cost component J_{reward}, as detailed in equations 2.2.

$$\min J = J_{time} - J_{reward}, \tag{2.1}$$

$$J_{time} = \alpha_T \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{m} T_{ij} X_{ijk}, \qquad J_{reward} = \beta_C \sum_{i=1}^{n} p_i Y_i, \tag{2.2}$$

where, α_T and β_C are weighting factors that balance the importance of minimizing travel time and maximizing rewards, respectively. The term T_{ij} represents the travel time between waypoints i and j, while X_{ijk} is a binary decision variable that indicates whether vehicle k travels between these waypoints. Additionally, p_i is the reward associated with visiting waypoint i, and Y_i is a binary variable indicating whether a specific waypoint is visited.

The algorithm seeks to determine the X decision variable that define optimal routes for the vehicles while considering to the following constraints:

- Waypoints must be visited at most once:

$$\sum_{i=0}^{n} \sum_{k=1}^{m} X_{ijk} \leq 1, \quad \forall j \in \{1, \ldots, n\}; \tag{2.3}$$

- Vehicles must follow continuous paths:

$$\sum_{i=0}^{n} X_{ipk} - \sum_{i=0}^{n} X_{pjk} = 0, \quad \forall k \in \{1, \ldots, m\}, p \in \{1, \ldots, n\}; \tag{2.4}$$

- Total travel time must not exceed the maximum allowable time (L_k):

$$\sum_{i=0}^{n} \sum_{j=0}^{n} T_{ij} X_{ijk} \leq L_k, \quad \forall k \in \{1, \ldots, m\}. \tag{2.5}$$

- Decision variables must be binary:

$$X_{ijk}, Y_i \in \{0, 1\}, \quad \forall i, j \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\}. \tag{2.6}$$

Therefore, effective decision-making within a multi-robot system is crucial for optimizing mission planning, ensuring that tasks are allocated efficiently while respecting the various constraints, such as time, precedence, and cooperation. The integration of MRTA frameworks, particularly in dynamic and unpredictable environments, allows for a more flexible and adaptable approach to task execution. By using cost functions that balance time minimization and reward maximization, mission planners can ensure optimal performance while maintaining feasibility.

2.3 Critical Review

The development of an autonomous mission planner for a marsupial system to inspect PV farms offers promising advancements since this system leverages the complementary capabilities of UAVs and USVs. The UAVs can conduct detailed inspections from above, while USVs provide crucial support, such as transportation and emergency interventions. The interaction between these vehicles is well-documented in scientific literature, with successful studies demonstrating effectiveness of UAV-USV collaboration. However, existing models often assume ideal operational conditions, overlooking real-world environmental challenges like wind and waves.

Mission success in these systems depends heavily on effective path planning algorithms. Current global path planners in the literature have demonstrated high effectiveness in addressing large-scale navigation problems in complex environments. However, these algorithms typically assume static environments, which is often an unrealistic representation of real-world conditions. In contrast, local path planning algorithms are better suited for dynamic environments with moving obstacles. Nevertheless, they often struggle in highly cluttered spaces, particularly without effective global guidance to direct navigation. Therefore, a hybrid approach that combines the strengths of both global and local path planning techniques could offer optimal navigation strategies for dynamic and complex operational scenarios.

Mission planning heavily relies on the decision-making process, which involves allocating tasks based on robot and task categorizations. These categorizations serve as system constraints, defining the problem and guiding task allocation effectively. The TRP represents a significant evolution of the TSP by prioritizing temporal constraints over distance, making it particularly relevant in scenarios where time-sensitive operations are crucial. Despite extensive research on this topic, many studies simplify the cost functions used, often failing to capture the full complexity of real-world conditions.

This dissertation aims to address the previous mentioned limitations by proposing an advanced autonomous UAV-USV system capable of overcoming environmental constraints and tested in realistic scenarios. Additionally, to mitigate the limitations of both local and global path planning, the research will explore hybrid systems that combine the strengths of both approaches. By incorporating these real-world uncertainties into task allocation models, this research ensures a more resilient and adaptable mission planning process. The use of TRP models with dynamic constraints

allows the system to prioritize urgent tasks, offering a solution better suited for time-sensitive operations in complex environments. This methodologies aim to develop a robust autonomous system capable of handling complex missions that require UAV-USV collaboration.

Chapter 3

Mission Planning for USV-UAV cooperation

Currently, the inspection of floating photovoltaic plants is typically performed manually by identifying thermal hot-spots and physical obstructions that may affect solar panel performance [66]. Existing autonomous solutions generally involve the use of an UAV, which takes off from shore, flies to the PV plant, conducts a visual and thermal survey, and then returns to land for data analysis.

This chapter aims to explore a cooperative approach for inspection of floating PV plants where a USV provides transportation and communication infrastructure to the UAV that performs the inspection of multiple PV panels. To enable the cooperative behavior of this approach, the work is proposing a mission architecture that manages task allocation, waypoint planning, and inter-vehicle communication, ensuring effective operations and responsiveness to unexpected events.

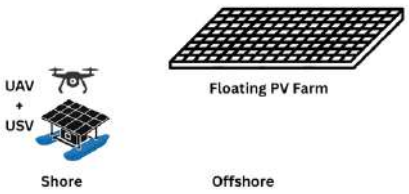
This chapter is organized as follows: section 3.1 provides an overview of the problem and the proposed approach to address it; section 3.2 introduces the NEST USV, detailing its hardware configuration and the software developed to enable autonomous navigation; section 3.3 outlines the complete mission execution pipeline, starting with user-defined pre-planning, followed by autonomous mission execution, and concluding with the implemented fail-safe procedures.

3.1 Introduction

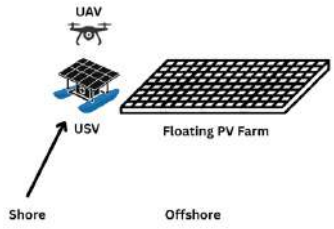
The efficiency of this approach is heavily constrained by the distance between the PV plant and the shore. Since traditional quadcopters have a limited endurance, a significant portion of this capacity is consumed merely traveling to the inspection site, thereby reducing the time available for actual inspection.

To overcome the efficiency limitations of conventional UAV-based inspection methods, this work proposes a cooperative solution where a USV serves as a mobile launch and recovery platform. By transporting the UAV to the inspection site and retrieving it afterward, the USV significantly reduces the UAV's flight time and extends its effective operation range. The proposed

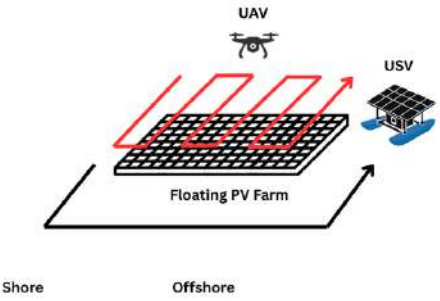
mission planning algorithm autonomously manages the entire cooperative process, coordinating both vehicles through optimized waypoint routing that accounts for static and dynamic obstacles, while ensuring synchronized operation and safe UAV recovery under all conditions. As part of this planning process, the algorithm includes an automatic map generation step that leverages geospatial data to produce an occupancy grid of the mission area, providing a reliable basis for path planning. The mission planner was developed following the Multilayer Universal Software Architecture (MUSA) framework [67], enabling adaptability to a broad range of inspection scenarios involving large-scale aquatic infrastructures. Its modularity and geo-referenced mapping capabilities allow it to operate effectively across diverse environments and locations. A schematic representation of the wanted implementation is represented in figure 3.1.



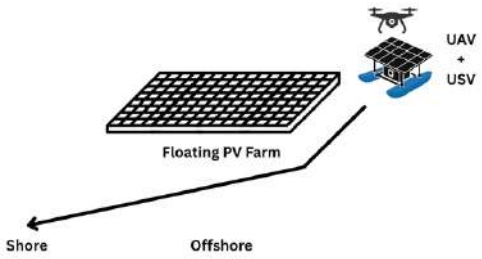
(a) Stage 1 – Mission Start: The USV and UAV are docked and initialized on shore.



(b) Stage 2 – Navigation and Take-Off: The USV navigates autonomously to the predefined UAV take-off location.



(c) Stage 3 – Aerial Survey: The UAV performs the inspection while the USV proceeds to the landing waypoint.



(d) Stage 4 – Landing and Return: The UAV lands on the USV, and both vehicles return to shore in a coupled configuration.

Figure 3.1: Schematic overview of the multi-stage autonomous inspection mission.

3.2 NEST - Non-stationary Emersed Structure for Takeoff-landing

This chapter introduces the USV developed to support the mission objectives, named NEST. It was designed for autonomous navigation and features an integrated landing pad, allowing it to serve as a mobile recovery platform for UAVs. Table 3.1 depicts a summary of the major NEST characteristics.

Table 3.1: Technical specifications of the NEST.

Characteristics	**Specifications**
Dimensions	2.0 m × 2.0 m × 1.3 m
Total weight	∼100 kg
Available payload	∼200 kg
Drive mode	Vectorial drive
Max speed	1.5 m/s
On-board sensors	Navigation: GPS, IMU, Compass. Perception: 3D LiDAR, visual and thermal cameras
Power supply	$LiFePO_4$ 25.6 V @ 200 Ah
Power consumption	∼3300 W
Operation time	∼2 h
Douglas sea scale	3 (up to 1.25 m)

This integrated design introduces a novel catamaran-style USV concept, developed to support UAV operations. The dual-hull configuration provides a stable platform that enables the UAV to safely land during and after inspection missions in maritime environments. An illustration of the NEST USV is presented in figure 3.2.

Figure 3.2: NEST USV, featuring a catamaran design and a 2×2 meter UAV landing platform for autonomous inspection operations.

3.2.1 Hardware Architecture

The hardware architecture of the NEST USV follows a modular design approach, in which the power and electronic subsystems are separated. The power module is equipped with a 24V battery

and includes essential operational and safety components such as a power switch, a fuse for over-current protection, and a dedicated charging port. Figure 3.3 shows the schematic and physical implementation of the power module, housed in a Peli Air case with an IP67 rating.

(a) NEST USV's schematic power module. (b) NEST USV's real power module.

Figure 3.3: Electrical schematic of the NEST USV's power module. Red lines indicate positive polarity connections, while blue lines represent negative polarity connections.

The electronics module is responsible for managing all navigation-related functions of the NEST USV. This module controls the four thrusters, and it integrates both an onboard computer and a CubeOrange flight controller[1]. Figure 3.4 shows the electrical schematic of the electronics module.

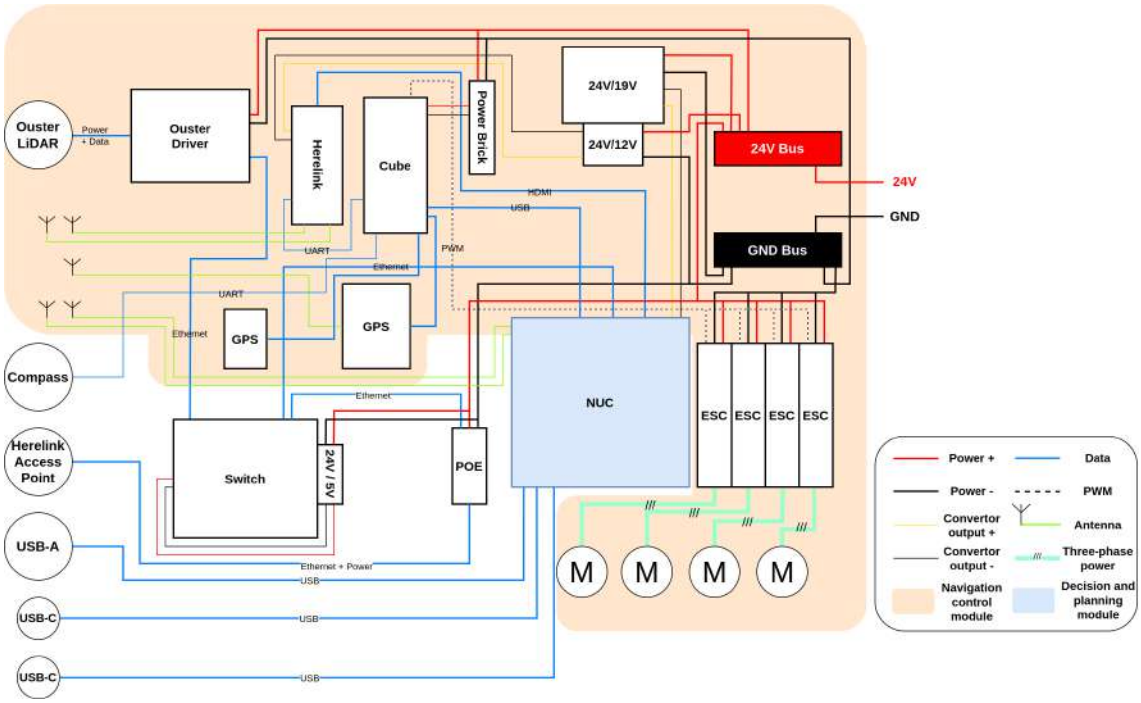
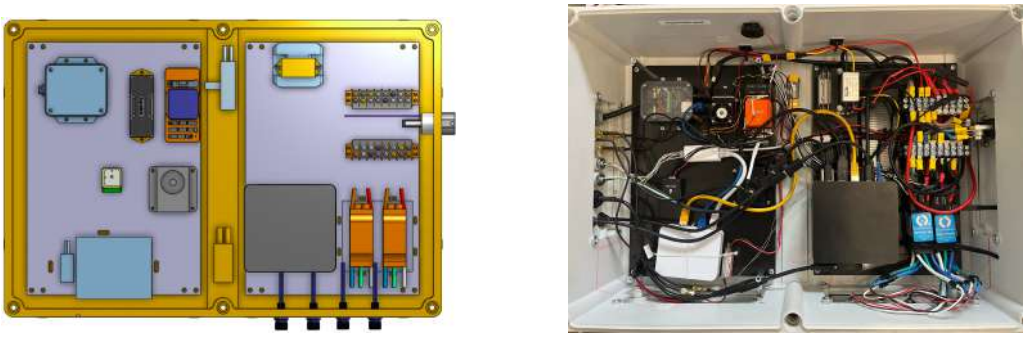


Figure 3.4: Electrical schematic of the NEST USV's electronics module.

[1] https://ardupilot.org/copter/docs/common-thecubeorange-overview.html (Accessed in 24-06-2025)

The onboard computer handles navigation tasks and communicates with the CubeOrange flight controller via MAVROS using the MAVLink protocol. The CubeOrange converts control commands into PWM signals sent to the ESCs, which then drive the thrusters using three-phase current. It also performs sensor fusion to provide accurate position and orientation data required by the navigation system. In addition, it includes the necessary sensors to provide accurate orientation and positioning data, such as GPS, IMUs and compasses. A LiDAR sensor is also incorporated, along with its corresponding driver, to capture environmental data and support obstacle avoidance during autonomous navigation. Figure 3.5 presents the layout of the electronics module components, including the CAD design used for prototyping and the final implementation. The module is housed in a Fibox enclosure with an IP67 rating.



(a) CAD model. (b) Real implementation.

Figure 3.5: NEST USV's electronics module.

The firmware running on the CubeOrange is a modified version of ArduSub[2]. Although originally developed for underwater vehicles, the source code was adapted to support a vectored thruster configuration enabling holonomic navigation. This adaptation was necessary because existing surface vehicle firmware lacks support for holonomic position hold. Holonomic control is essential for enabling position hold, which is a critical capability during UAV landing and takeoff procedures.

Communication between the onboard computer and the flight controller is established via the MAVLink protocol, used both directly and through MAVROS, which acts as a bridge between ROS2 and MAVLink. Figure 3.6 depicts the functionalities layout of the vehicles' hardware.

The system also interfaces with QGroundControl, which provides a high-level mission planning and monitoring interface with user-friendly controls.

3.2.2 Software Architecture

The software architecture integrates multiple modules to enable autonomous navigation. USV localization, achieved through sensor fusion of GPS and inertial data, is support to all modules throughout the system. The costmap captures static obstacles and navigable areas. Based on this, the global planner generates an optimal route, then a velocity controller computes the reference

[2]https://www.ardusub.com/ (Accessed on 24-06-2025)

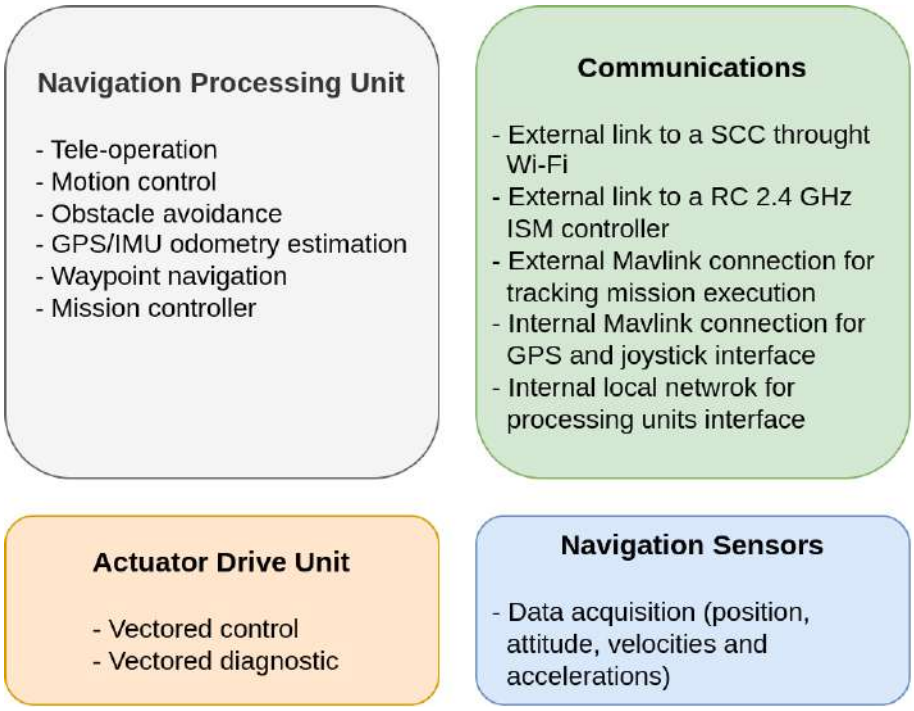


Figure 3.6: NEST functionalities layout. The grey, green, orange and blue boxes are the processing units, communications, actuator drive units, and sensors interfaces, correspondingly.

velocities to follow that route, which are refined by a local planner to avoid dynamic obstacles in real time. Finally, a thrust controller converts these adjusted velocity commands into low-level actuator inputs, ensuring smooth and responsive navigation. This navigation pipeline is represented in figure 3.7.

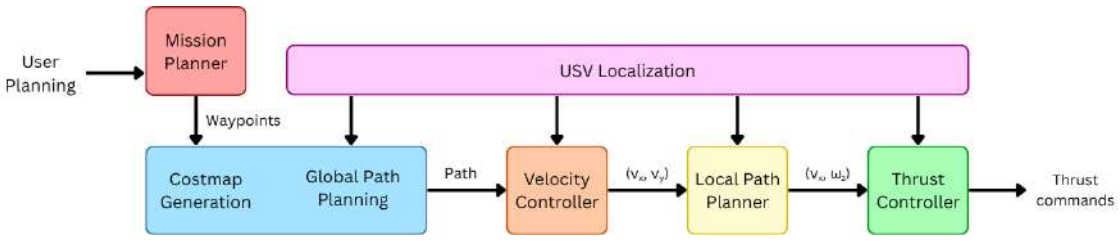


Figure 3.7: Navigation pipeline.

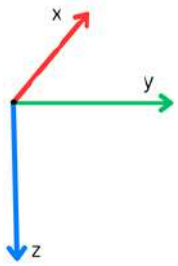
The process begins when the Mission Planner requests a path to a designated waypoint. The `nav2` framework responds by generating a global path, which is then passed to the velocity controller. This controller computes a velocity vector based on the USV's current position and the planned trajectory. The local planner then adjusts this velocity to account for surrounding obstacles, producing linear and angular velocity commands. These are passed to the thrust controller module, which translates them into low-level commands for the USV's thrusters.

To enable autonomous navigation, the `nav2` framework requires a continuous estimate of the USV's position within the map frame as the global planner rely on the current pose of the vehicle to compute trajectories, and ensure convergence to the goal. The following section outlines

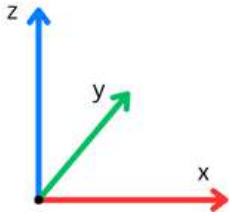
the localization system implemented to fulfill this requirement, detailing the coordinate frame transformations employed to maintain consistent pose estimation throughout the mission.

USV Localization

Autonomous navigation in outdoor environments poses significant challenges related to coordinate frame transformations and multi-sensor state estimation. GPS-based systems, along with inertial sensors such as IMUs and compasses, require accurate and consistent frame alignment to enable reliable waypoint-based navigation. Given the integration of ArduSub with ROS2-based navigation tools, it is required to align their coordinate frame conventions to ensure system compatibility. ArduSub operates using the North-East-Down (NED) frame, which is standard in underwater applications, whereas ROS2 navigation tools assume an East-North-Up (ENU) frame. Proper transformation between these frames is essential for accurate localization. Both coordinate systems are illustrated in figure 3.8.



(a) North-East-Down coordinate system.



(b) East-North-Up coordinate system.

Figure 3.8: Local tangent plane coordinates.

The transformation between these coordinate systems involves both axis permutations and sign changes. The general transformation from NED to ENU coordinates can be represented using a rotation matrix $\mathbf{T}_{NED \to ENU}$:

$$\mathbf{T}_{NED \to ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{3.1}$$

For a position vector $\mathbf{p}_{NED} = [x_N, y_E, z_D]^T$ in the NED frame, the corresponding ENU position is:

$$\mathbf{p}_{ENU} = \mathbf{T}_{NED \to ENU} \cdot \mathbf{p}_{NED} = \begin{bmatrix} y_E \\ x_N \\ -z_D \end{bmatrix}. \tag{3.2}$$

For GPS data, the conversion only requires a sign change to the altitude since the longitude and latitude remain the same:

$$\text{GPS:} \begin{cases} \text{latitude}_{ENU} = \text{latitude}_{NED} \\ \text{longitude}_{ENU} = \text{longitude}_{NED} \\ \text{altitude}_{ENU} = -\text{altitude}_{NED} \end{cases} . \tag{3.3}$$

The conversion of IMU data from North-East-Down (NED) to East-North-Up (ENU) coordinate frames requires transformation of both orientation quaternions and linear motion vectors to enable integration with ROS2 navigation systems.

Given a quaternion $q_{NED} = [q_w, q_x, q_y, q_z]$ in the NED frame, the conversion follows a systematic process beginning with the extraction of Euler angles from the NED quaternion using the standard quaternion-to-Euler conversion formulas:

$$\phi_{NED} = \arctan 2\left(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)\right), \tag{3.4}$$

$$\theta_{NED} = \arcsin\left(2(q_w q_y - q_z q_x)\right), \tag{3.5}$$

$$\psi_{NED} = \arctan 2\left(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)\right). \tag{3.6}$$

The coordinate transformation mapping then reassigns these angles according to the frame conversion where $\phi_{ENU} = \theta_{NED}$ such that NED pitch becomes ENU roll, $\theta_{ENU} = \phi_{NED}$ such that NED roll becomes ENU pitch, and $\psi_{ENU} = \psi_{NED} + \frac{\pi}{2}$ representing the yaw offset adjustment.

The ENU quaternion is subsequently reconstructed using the standard quaternion construction formula:

$$q_{ENU} = \begin{bmatrix} \cos\frac{\phi_{ENU}}{2}\cos\frac{\theta_{ENU}}{2}\cos\frac{\psi_{ENU}}{2} + \sin\frac{\phi_{ENU}}{2}\sin\frac{\theta_{ENU}}{2}\sin\frac{\psi_{ENU}}{2} \\ \sin\frac{\phi_{ENU}}{2}\cos\frac{\theta_{ENU}}{2}\cos\frac{\psi_{ENU}}{2} - \cos\frac{\phi_{ENU}}{2}\sin\frac{\theta_{ENU}}{2}\sin\frac{\psi_{ENU}}{2} \\ \cos\frac{\phi_{ENU}}{2}\sin\frac{\theta_{ENU}}{2}\cos\frac{\psi_{ENU}}{2} + \sin\frac{\phi_{ENU}}{2}\cos\frac{\theta_{ENU}}{2}\sin\frac{\psi_{ENU}}{2} \\ \cos\frac{\phi_{ENU}}{2}\cos\frac{\theta_{ENU}}{2}\sin\frac{\psi_{ENU}}{2} - \sin\frac{\phi_{ENU}}{2}\sin\frac{\theta_{ENU}}{2}\cos\frac{\psi_{ENU}}{2} \end{bmatrix}. \tag{3.7}$$

The complete IMU sensor data transformation can therefore be summarized as the simultaneous conversion:

$$q_{ENU} = \text{quaternion}(\theta_{NED}, \phi_{NED}, \psi_{NED} + \frac{\pi}{2}), \tag{3.8}$$

$$\vec{a}_{ENU} = \mathbf{T}_{NED \to ENU} \cdot \vec{a}_{NED}, \tag{3.9}$$

$$\vec{\omega}_{ENU} = \mathbf{T}_{NED \to ENU} \cdot \vec{\omega}_{NED}, \tag{3.10}$$

where $\vec{a}$ represents linear acceleration and $\vec{\omega}$ represents angular velocity vectors. This coordinate frame transformation ensures that multi-sensor data from ArduSub can be integrated with ROS2 navigation packages, enabling consistent and accurate state estimation for autonomous vehicle navigation.

While coordinate frame alignment ensures compatibility between software components, achieving robust localization in the dynamic maritime environment requires sensor fusion techniques. The integration of GPS, IMU, and compass data through an EKF provides the accurate state estimation necessary for precise navigation and control.

To address the challenges of multi-sensor state estimation, an EKF framework was implemented to enable robust sensor fusion and continuous state estimation. The EKF is well-suited for outdoor robotic platforms due to its ability to handle nonlinear motion dynamics and integrate heterogeneous sensor data. It maintains a state vector $\mathbf{x}$ that includes the vehicle's pose and velocity components:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}, \ddot{x}, \ddot{y}, \ddot{z}]^T, \tag{3.11}$$

where, (x, y, z) denote the position in 3D space, (ϕ, θ, ψ) correspond to roll, pitch, and yaw angles, and the dot notation represents time derivatives (linear velocity and acceleration and angular velocity).

The prediction step of the EKF follows a nonlinear state transition model:

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1|k-1}, \mathbf{u}_k) + \mathbf{w}_k, \tag{3.12}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k, \tag{3.13}$$

where $\mathbf{F}_k$ is the Jacobian of the transition function, $\mathbf{P}$ is the state covariance matrix, and $\mathbf{Q}_k$ is the process noise covariance and $f(\cdot)$ is defined by the 3.14 represents the prediction model based on a 3D kinematic model derived from the Newtonian mechanics.

$$f = \begin{bmatrix} P_k \\ \omega_k \\ \dot{P}_k \\ \dot{\omega}_k \\ \ddot{P}_k \end{bmatrix} = \begin{bmatrix} P_{k-1} + R_{ZYX}\left(\dot{P}_{k-1}\Delta t + \frac{1}{2}\ddot{P}_{k-1}\Delta t^2\right) \\ \omega_{k-1} + \Omega\,\dot{\omega}_{k-1}\Delta t \\ \dot{P}_{k-1} + \ddot{P}_{k-1}\Delta t \\ \dot{\omega}_{k-1} \\ \ddot{P}_{k-1} \end{bmatrix}, \tag{3.14}$$

where $P = (x, y, z)$, $\omega = (\phi, \theta, \psi)$, and the dots denote the corresponding velocity and acceleration, R_{ZYX} represents the ZYX Tait-Bryan angles rotation matrix and Ω is defined by:

$$\Omega = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\cos^{-1}\theta & \sin\phi\cos^{-1}\theta \end{bmatrix}. \tag{3.15}$$

During the update phase, sensor measurements are incorporated using:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1}, \tag{3.16}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - h(\mathbf{x}_{k|k-1})), \tag{3.17}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}, \tag{3.18}$$

where, $\mathbf{H}_k$ is the measurement Jacobian, $\mathbf{R}_k$ is the measurement noise covariance, and $h(\cdot)$ represents the nonlinear measurement model. Each sensor is assumed to measure specific state variables. Consequently, the observation matrix $\mathbf{H}$ is typically a submatrix of the identity matrix, selecting the observed states. However, since the sensors operate asynchronously and do not measure all components of the state vector simultaneously, partial updates are performed. This is done by reducing $\mathbf{H}$ to an $m \times 15$ matrix, where m is the number of measured variables. In this reduced matrix, only the elements corresponding to the measured states are set to 1, while the rest are zero [68].

To support the EKF used in localization, a structured hierarchy of coordinate frames is employed to manage spatial relationships across different reference systems. This setup allows consistent transformation of sensor data into a common reference frame, enabling accurate state estimation through sensor fusion. At the top of the hierarchy, a global reference frame is defined to represent mission-level positioning. Geodetic data—such as latitude, longitude, and altitude obtained from GPS—is converted into a local Cartesian frame using the Universal Transverse Mercator (UTM) projection, as defined by Snyder (1982) [69]. This projection simplifies computations by linearizing Earth's curved surface into a planar coordinate system. Within the local Cartesian frame, a fixed origin is defined based on the initial GPS fix, and subsequent positions are expressed relative to this origin. In parallel, local odometry—computed from IMUs—provides short-term, drift-prone pose updates with high frequency. These are fused with the global data to provide a stable and accurate estimate of the vehicle's position and orientation. The resulting frame structure integrates both absolute and relative references, allowing transformations between the inertial (global) frame, the local odometric frame, and the vehicle's body-fixed frame. This layered approach supports robust sensor fusion, compensating for the drift of local sensors with the stability of global measurements.

Costmap Generation

This architecture is built around the ROS2 `nav2` framework, a modular and extensible system for autonomous navigation. Among its core capabilities are map representation, global path planning, and real-time control. The occupancy grid generated in the mapping phase is first converted into a costmap, which defines the navigable space for the planner. The costmap construction serves as the representation for path planning, constructed through the fusion of multiple information layers. The costmap $C(x,y)$ at discrete coordinates is computed as:

$$C(x,y) = C_{\text{static}}(x,y) + C_{\text{inflation}}(x,y), \tag{3.19}$$

where $C_{\text{static}}(x,y)$ represents the base layer derived from the occupancy grid, and $C_{\text{inflation}}(x,y)$ accounts for safety margins around obstacles. The inflation layer applies a spatial decay function to expand the perceived size of obstacles, thereby creating a buffer zone around them. It is computed as:

$$C_{\text{inflation}}(x,y) = \frac{254 \cdot e^{-\alpha \cdot d(x,y)}}{r_{\text{inflation}}}, \tag{3.20}$$

where $d(x,y)$ is the Euclidean distance from the cell to the nearest obstacle, α is the cost scaling factor, and $r_{\text{inflation}}$ denotes the inflation radius beyond which obstacle influence diminishes.

This layered costmap representation allows for efficient and safe path planning by encoding both static obstacles and dynamic safety constraints. An example of the resulting costmap is shown in figure 3.9.

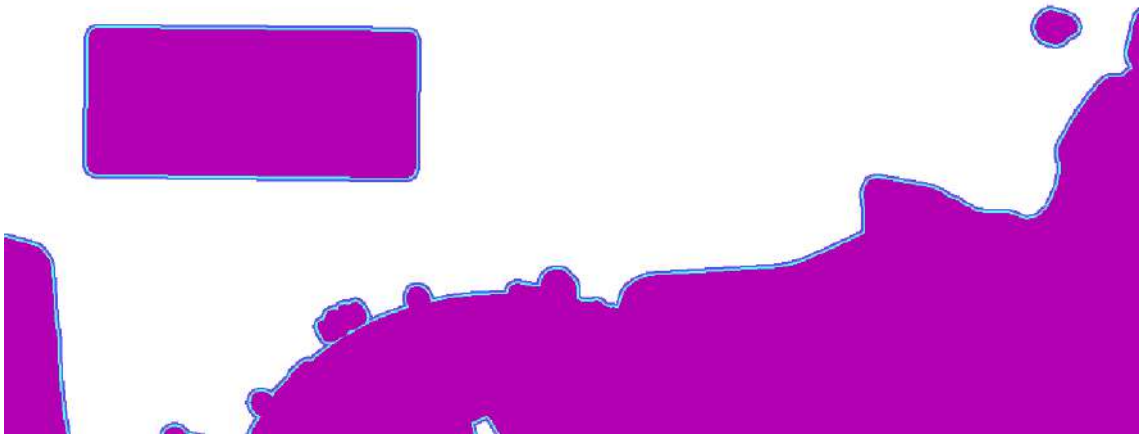


Figure 3.9: Costmap generated by `nav2` based on an occupancy grid. Pink regions represent areas of maximum traversal cost (i.e., close proximity to obstacles), while white regions indicate free space.

Global Path Planner

To enable globally optimal path planning, this work employs navigation functions—a class of artificial potential field methods that ensure convergence to the goal while avoiding local minima. A navigation function defines a scalar field Φ over the configuration space, where the gradient $\nabla\Phi$ guides the vehicle toward the target while steering away from obstacles. A valid navigation function satisfies the following conditions:

- The potential Φ is zero at the goal and approaches infinity near obstacles;
- There are no local minima except at the goal;

- The function has a unique global minimum at the goal position.

These properties ensure that gradient descent on Φ always leads to the target without becoming trapped in suboptimal regions. In practice, navigation functions are implemented using wave-front propagation on a discretized costmap [70]. The algorithm initializes the goal cell with $\Phi = 0$ and assigns infinite cost to obstacle cells. The potential value of each free cell is then iteratively updated based on the minimum accumulated cost from neighboring cells:

$$\Phi_{i,j}^{(n+1)} = \min_{(p,q)\in N(i,j)} \left\{ \Phi_{p,q}^{(n)} + c(i,j \to p,q) \right\}, \tag{3.21}$$

where $N(i,j)$ denotes the set of neighboring cells and $c(i,j \to p,q)$ represents the cost of transitioning between adjacent cells. This process spreads potential values outward from the goal to all reachable positions in the map. Once the potential field has been computed, the optimal path is extracted by following the steepest descent direction:

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \nabla \Phi(\mathbf{q}_k), \tag{3.22}$$

where α is a step size parameter and the gradient is approximated using discrete differences between grid cells.

Due to grid discretization, the resulting path may exhibit angular discontinuities. To address this, a smoothing stage refines the trajectory by optimizing waypoint positions. The smoothing minimizes the following cost function:

$$J = \sum_{i=1}^{n} w_1 \|\mathbf{p}_i - \mathbf{p}_i^{\text{orig}}\|^2 + w_2 \|\mathbf{p}_{i+1} - 2\mathbf{p}_i + \mathbf{p}_{i-1}\|^2, \tag{3.23}$$

where, $\mathbf{p}_i$ are the current waypoints, $\mathbf{p}_i^{\text{orig}}$ are the original waypoints obtained from the grid-based path, and w_1, w_2 are weighting factors. The first term enforces fidelity to the initial path, while the second penalizes curvature by minimizing the discrete second derivative. Gradient descent is used to iteratively minimize J, producing a smoother path that facilitates more stable tracking and enhances overall mission performance as it can be seen in figure 3.10.

Velocity Control

The velocity controller functions as an intermediate layer between the global path planner provided by the `nav2` framework and the vehicle's low-level actuation system. Its primary role is to convert discrete waypoint sequences into continuous velocity commands that respect the vehicle's kinematic and dynamic constraints.

To ensure smooth and accurate trajectory tracking, the controller employs a *Pure Pursuit* algorithm augmented with velocity smoothing and acceleration limiting. This enhances stability during high-frequency control updates and mitigates abrupt motion changes. A key component of the controller is its adaptive lookahead mechanism, which selects a target point on the planned path based on a fixed lookahead distance L_d. As illustrated in figure 3.11, the desired velocity vector

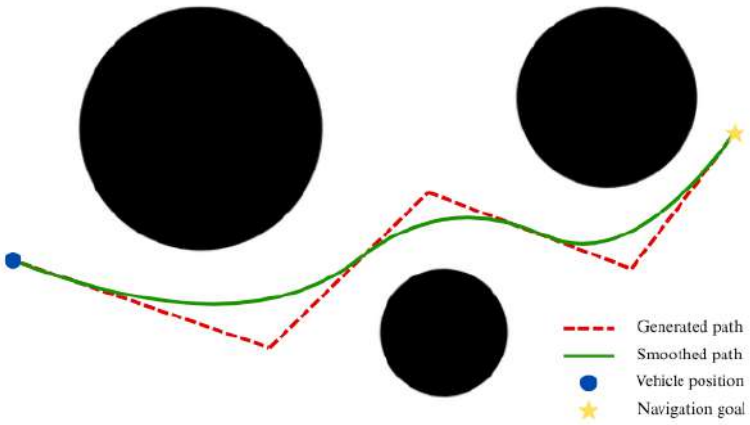


Figure 3.10: Smoothing representation of generated path.

is computed as the direction pointing from the current USV position to the intersection between the planned path and a circle of radius L_d. The algorithm 1 outlines the method used to determine the target point along the path. The algorithm begins by identifying the waypoint closest to the USV's current position. From this point, it traverses the path sequentially, accumulating segment distances until the total distance reaches the predefined lookahead distance L_d. Once this threshold is met, linear interpolation is used between the two enclosing waypoints to determine the precise lookahead target point.

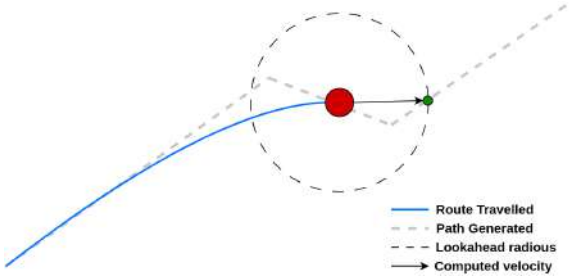


Figure 3.11: Lookahead algorithm.

With the lookahead point identified, the algorithm computes the desired velocity vector in the `map` coordinate frame. The direction vector $\mathbf{d}_{map}$ is defined as:

$$\mathbf{d}_{map} = \mathbf{t} - \mathbf{r} = (dx_{map}, dy_{map}), \tag{3.24}$$

where $\mathbf{t}$ represents the target (lookahead) point, and $\mathbf{r}$ is the current position of the USV. This vector is then transformed into the `base_link` frame using the transformation matrix $\mathbf{T}^{map}_{base\_link}$:

$$\mathbf{d}_{base\_link} = \mathbf{T}^{map}_{base\_link}\mathbf{d}_{map} = (dx_{base\_link}, dy_{base\_link}). \tag{3.25}$$

Algorithm 1 Lookahead Point Calculation

Require: Path $\mathbf{P} = \{p_1, p_2, ..., p_n\}$, Robot position $\mathbf{r} = (x_r, y_r)$, Lookahead distance L_d
Ensure: Lookahead point $\mathbf{t} = (x_t, y_t)$

```
Initialize d_min ← ∞, i_closest ← 1
for i = 1 to n do
    d_i ← ‖p_i − r‖_2
    if d_i < d_min then
        d_min ← d_i
        i_closest ← i
    end if
end for
Initialize d_acc ← 0, found ← false
for j = i_closest to n − 1 do
    p_1 ← p_j, p_2 ← p_{j+1}
    d_segment ← ‖p_2 − p_1‖_2
    if d_acc + d_segment ≥ L_d then
        d_overshoot ← L_d − d_acc
        α ← d_overshoot / d_segment
        t ← p_1 + α(p_2 − p_1)
        found ← true
        break
    end if
    d_acc ← d_acc + d_segment
end for
if found = false then
    t ← p_n {Use final waypoint}
end if
return t
```

To normalize the direction and produce a unit vector for scaling purposes, the vector is normalized as:

$$\hat{\mathbf{d}} = \frac{\mathbf{d}_{base\_link}}{\|\mathbf{d}_{base\_link}\|_2} = (\hat{d}_x, \hat{d}_y). \tag{3.26}$$

The target velocity magnitude begins at the maximum velocity v_{max} and is scaled down as the vehicle approaches a stopping condition. The deceleration factor k_{decel} is defined as:

$$k_{decel} = \sqrt{\min\left(\frac{L_d}{d_{stop}}, \frac{d_{goal}}{d_{stop}}\right)}; \tag{3.27}$$

where the stopping distance is given by:

$$d_{stop} = \frac{v_{current}^2}{2a_{max}} \cdot s_{margin}; \tag{3.28}$$

with s_{margin} providing an additional safety buffer.

The target velocity in the `base_link` frame is then calculated as:

$$\mathbf{v}_{target} = v_{max} \cdot k_{decel} \cdot \hat{\mathbf{d}}. \tag{3.29}$$

To ensure smooth control and prevent sudden accelerations, a two-stage filtering process is applied. Stage 1 applies exponential smoothing via a low-pass filter:

$$v_{filtered} = (1 - \alpha_{smooth}) \cdot v_{smooth} + \alpha_{smooth} \cdot v_{target}, \tag{3.30}$$

with $v_{smooth} = 0$ at initialization and subsequently set to the previous $v_{filtered}$ value at each iteration, and α_{smooth} as the velocity smoother factor. Stage 2 enforces acceleration limits by clamping the filtered velocity within bounds defined by a_{max} over a control period Δt:

$$v_{smooth} = \text{clamp}(v_{filtered}, v_{smooth} \pm a_{max} \cdot \Delta t). \tag{3.31}$$

Finally, the resulting velocity is saturated to ensure that it does not exceed v_{max}.

Local Path Planner

For the local path planner, the Adaptive Velocity Obstacle Avoidance (AVOA) algorithm [58] was integrated. This algorithm is a variation of the traditional Velocity Obstacles method, adapted specifically for USV navigation. AVOA provides collision-free path planning by determining safe velocity commands that maintain the desired trajectory toward a target destination.

AVOA operates by generating protective zones in the velocity space around detected obstacles, as illustrated in figure 3.12. These zones represent velocity combinations that would result in a collision with an obstacle. The algorithm uses linear and angular velocity commands $[v_x, w_z]$, which correspond to the control inputs for differential drive systems.

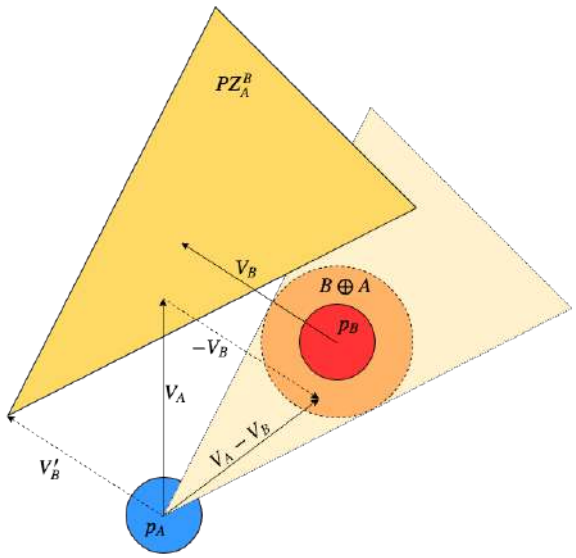


Figure 3.12: Representation of the protective zone PZ_A^B of an obstacle B relative to agent A [58].

Unlike traditional approaches, AVOA introduces a time-window constraint for collision risk assessment. It only considers obstacles that pose a threat within a predefined temporal threshold, thus avoiding unnecessary course corrections for distant or irrelevant objects. For instance, if a potential collision would occur in 30 seconds but the threshold is set to 10 seconds, the algorithm ignores the obstacle for the current planning cycle. This leads to more efficient and smoother trajectory generation. The algorithm proceeds through three main stages: first, it represents obstacles in velocity space; second, it computes the admissible velocity set that avoids collisions while respecting the USV's kinematic constraints; and third, it selects the optimal velocity command from the valid set based on predefined criteria.

This algorithm operates efficiently in real time, requiring no external communication or coordination. Moreover its predictive behavior enables preemptive maneuvering, resulting in smoother paths and improved efficiency. AVOA supports simultaneous handling of multiple static and dynamic obstacles. All generated commands are constrained to the physical capabilities of the USV's propulsion system, ensuring feasibility at the hardware level.

Thrust Control

Translating desired velocity commands into thrust forces is a key step that connects motion planning with actuator control in autonomous navigation. The implemented control system employs a decoupled PID (Proportional-Integral-Derivative) architecture that independently regulates linear and angular motion components, providing robust velocity tracking while taking into consideration the specific dynamic characteristics of surface vehicles.

The control architecture operates by subscribing to desired velocity commands generated by the obstacle avoidance system and comparing these setpoints with the current vehicle velocities obtained from odometry feedback. The velocity error for each degree of freedom is computed as:

$$e_i(t) = v_{d,i}(t) - v_{c,i}(t), \tag{3.32}$$

where $e_i(t)$ represents the velocity error for the i-th degree of freedom, $v_{d,i}(t)$ is the desired velocity, and $v_{c,i}(t)$ is the current measured velocity from the odometry system.

The core control law implements a discrete-time PID controller for each controlled axis. For linear motion control in the x and y directions, the control output is computed as:

$$u_{\text{linear}}(k) = K_{p,\text{linear}} e(k) + K_{i,\text{linear}} \sum_{j=0}^{k} e(j)\Delta t + K_{d,\text{linear}} \frac{e(k) - e(k-1)}{\Delta t}. \tag{3.33}$$

Similarly, for angular motion control about the yaw axis, the control law becomes:

$$u_{\text{angular}}(k) = K_{p,\text{angular}} e_\omega(k) + K_{i,\text{angular}} \sum_{j=0}^{k} e_\omega(j)\Delta t + K_{d,\text{angular}} \frac{e_\omega(k) - e_\omega(k-1)}{\Delta t}, \tag{3.34}$$

where K_p, K_i, and K_d represent the proportional, integral, and derivative gains respectively, Δt is the control sampling period, and $e_\omega(k)$ denotes the angular velocity error.

The system incorporates several control features to enhance performance and robustness. Gain scheduling is implemented to provide gentler control action during low-speed operations, where the proportional gains are reduced by a factor of 2-3 when the current velocity magnitude falls below specified thresholds (0.2 m/s for linear motion and 0.1 rad/s for angular motion). This approach prevents excessive control effort during precision maneuvering and station-keeping operations. Anti-windup protection is implemented to prevent integral saturation, particularly important for marine vehicles where external disturbances such as currents and waves can cause sustained velocity errors. The integral term is clamped to maximum values ($I_{\text{max,linear}}$ and $I_{\text{max,angular}}$) and is reset to zero when the vehicle is commanded to stop and the current velocity is near zero:

$$I(k) = \begin{cases} 0 & \text{if } |v_d| < 0.01 \text{ and } |v_c| < 0.1 \\ \text{clamp}(I(k-1) + e(k)\Delta t, -I_{\max}, I_{\max}) & \text{otherwise} \end{cases}. \tag{3.35}$$

Input filtering is applied to the desired velocity commands to smooth rapid changes that could cause actuator saturation or unstable motion. The filtered desired velocity is computed using a first-order low-pass filter:

$$v_{d,\text{filtered}}(k) = (1-\alpha)v_{d,\text{filtered}}(k-1) + \alpha v_{d,\text{raw}}(k), \tag{3.36}$$

where α is the filter coefficient (typically 0.3), providing a balance between responsiveness and smoothness. The control outputs are directly mapped to thrust force commands for the vehicle's propulsion system. For a typical surface vehicle configuration, the linear control outputs u_x and u_y are assigned to the horizontal thrust forces F_x and F_y respectively, while the angular control output u_{ω_z} is mapped to the yaw moment M_z:

$$[F_x, F_y, F_z, M_x, M_y, M_z] = [u_x, u_y, 0, 0, 0, u_{\omega_z}]. \tag{3.37}$$

The vertical force and roll/pitch moments are set to zero, since the vehicle operates only in the horizontal plane. The control system operates at a fixed sampling rate of 10 Hz, providing sufficient bandwidth for typical surface vehicle dynamics while maintaining computational efficiency. The decoupled control architecture allows for independent tuning of linear and angular motion parameters, enabling optimization for different operational scenarios such as transit navigation, precise maneuvering, or station-keeping operations. This velocity-to-thrust control system provides the essential interface between the high-level autonomous navigation algorithms and the low-level actuator hardware, ensuring that planned trajectories are accurately executed while maintaining stable and responsive vehicle behavior in the presence of environmental disturbances and modeling uncertainties.

The navigation pipeline provides the capability to reach designated waypoints autonomously, but mission success depends on executing appropriate behaviors upon arrival at each location. The action system defines the specific tasks performed at waypoints, ranging from simple transit operations to complex UAV deployment and recovery procedures.

3.3 Cooperative Mission Planning Framework

The integration of UAVs and USVs for autonomous survey missions presents architectural challenges that require balancing centralized and decentralized control strategies. Traditional robotic systems typically adopt either a fully centralized architecture—where a single master unit coordinates all subsystems—or a fully decentralized one, where each agent operates autonomously with inter-agent communication for coordination. However, in the context of marsupial robotic systems for inspection, a hybrid approach is more appropriate.

In the proposed architecture, the Ground Station serves primarily as a supervisory interface, responsible for mission configuration, monitoring, and logging. Once the mission is initiated, time-critical tasks—such as UAV takeoff, landing, and emergency handling—are executed autonomously through a local coordination mechanism between the USV and the UAV. This system follows a host-guest communication model, in which the USV assumes the role of the host by issuing commands-enabling flight mode, triggering takeoff or landing-while the UAV acknowledges and validates the successful completion of each operation before mission progression continues. This structured exchange reduces reliance on real-time intervention from the Ground Station, enhancing autonomy and robustness in dynamic environments. This tight coupling ensures the UAV can focus on inspection tasks while offloading navigation and communication responsibilities to the USV. A schematic representation of the full system architecture is presented in figure 3.13, illustrating the interconnection between mission layers and cooperative components.

Mission configuration begins at the Ground Station, where the user defines survey parameters using QGC, which also provides real-time mission monitoring and visualization. The Mission Planner module retrieves the map of the operational area based on the USV's current GPS coordinates. It then computes the UAV's takeoff and landing waypoints according to the survey region defined in QGC. These computed waypoints are presented to the operator for validation. Upon user confirmation, the Mission Planner transmits the navigation goals to the USV, which autonomously calculates the optimal route while considering both static and dynamic obstacles. Once the USV reaches the designated takeoff waypoint, it issues a command for the UAV to initiate the survey. During the UAV's flight, the USV navigates toward the landing waypoint. The mission concludes with the UAV landing on the USV, after it autonomously return to the home position.

3.3.1 Pre-planning

To define the mission, the operator uses QGC's Mission Planner interface. Through this tool, a survey area is selected by drawing a polygonal region over the map, within which the UAV

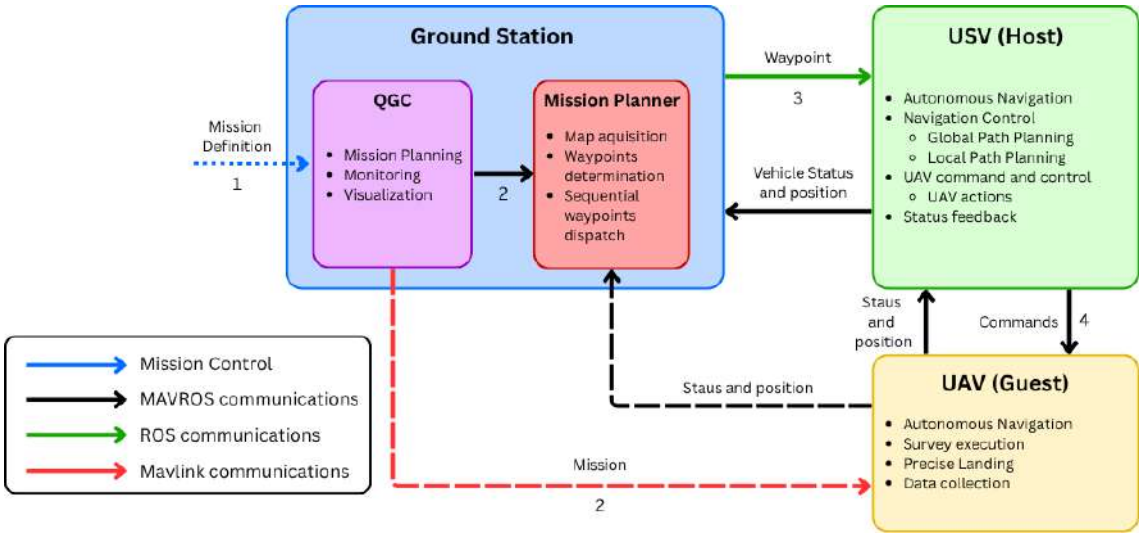


Figure 3.13: Mission Planner Architecture. Solid lines represent autonomous robot-robot communication; dotted lines represent human-robot interactions; and dashed lines indicate shared communication (human-robot and robot-robot) used for both mission coordination and human monitoring. The numbers below the communication links represent the sequential flow of information. Unnumbered communications indicate continuous data exchange.

will perform its inspection flight. In addition to the area, the user configures several key mission parameters, including the survey altitude, line spacing, angle of the survey transects, and turnaround distance. Once the mission parameters are defined, QGC automatically generates a set of waypoints in GPS coordinates representing the UAV's full flight path. These waypoints are exported and parsed by the Mission Planner module to identify the most appropriate takeoff and landing locations, ensuring alignment with the USV's navigation plan and operational constraints. A screenshot of the QGC Mission Planner is represented in the figure 3.14, where the user defines a mission in the Alqueva Dam floating PV plant.

Upon arrival at a waypoint, the USV executes a predefined action associated with that location. By default, the mission planner specifies only two primary waypoints corresponding to the takeoff and landing procedures. However, the system is designed to support user-defined waypoints with customizable actions, allowing for flexible mission configuration and extension based on operational requirements.

- None: This action instructs the USV to pass through a waypoint without stopping. It is primarily used to influence path shaping, particularly to avoid undesirable regions.
- Wait: Commands the USV to hold position at the waypoint for a user-specified duration.
- Position Hold: Maintains the USV's position until manual intervention signals mission continuation.

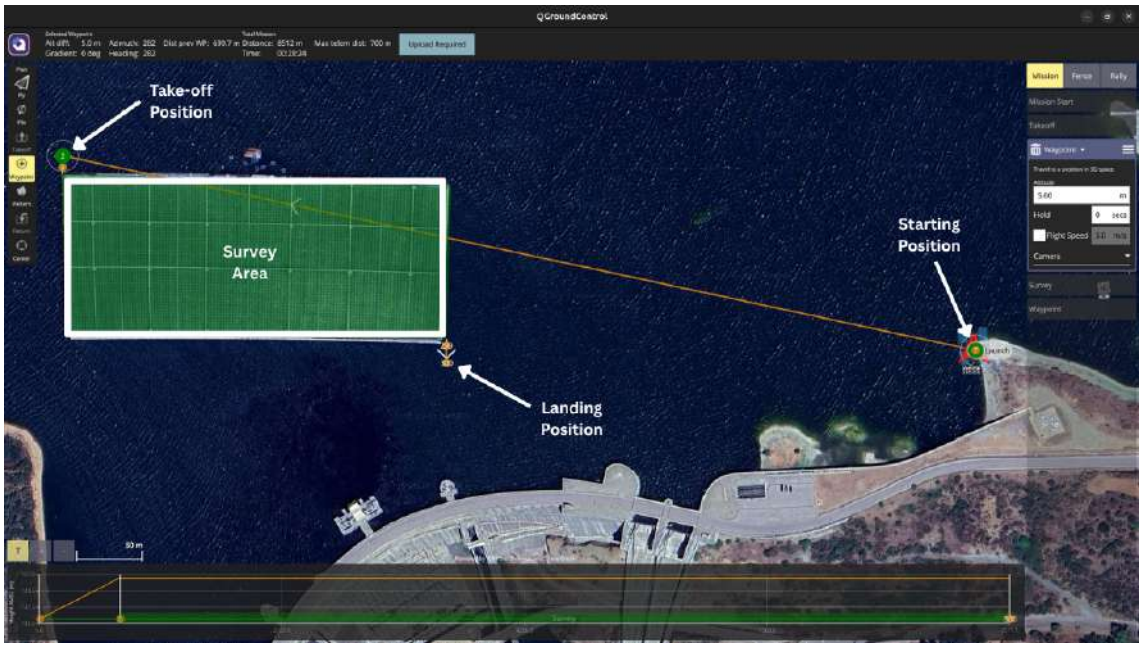


Figure 3.14: QGroundControl UAV Mission Planner.

- Takeoff: Triggers deployment of an onboard UAV. The USV first holds position and then issues a takeoff command via MAVROS. It proceeds only upon successful UAV launch confirmation.
- Landing: Prepares the USV for UAV recovery using a visual marker-based precision landing approach [71]. Once landing is confirmed, the USV navigates back to the mission origin.

Map acquisition

Creating the occupancy grid of the operational environment is crucial to ensure safe and efficient navigation. This environmental modeling structure was selected as the primary map representation due to its effectiveness in autonomous path planning since the USV operates in only two dimensions, and no altitude dimension is required. This format provides a balance between computational efficiency and path planning compatibility. To generate the occupancy grid, a custom package was developed. This module automatically constructs the required map based on the USV's current GPS position and the user-defined mission area. The internal workflow of this package is illustrated in figure 3.15, where blue module represents the input data, green modules denote the processing components, and the red module corresponds to the final output.

To construct the occupancy grid, the package first segments the user-defined mission area into map tiles, which are automatically downloaded from *OpenStreetMap*. These tiles are then stitched into a single high-resolution image, centered on the USV's current GPS position. This ensures accurate transformation from global GPS coordinates to local map coordinates.

The resulting image undergoes a series of pre-processing steps. A blurring filter is applied to reduce noise such as watermarks or minor artifacts, followed by a color threshold operation

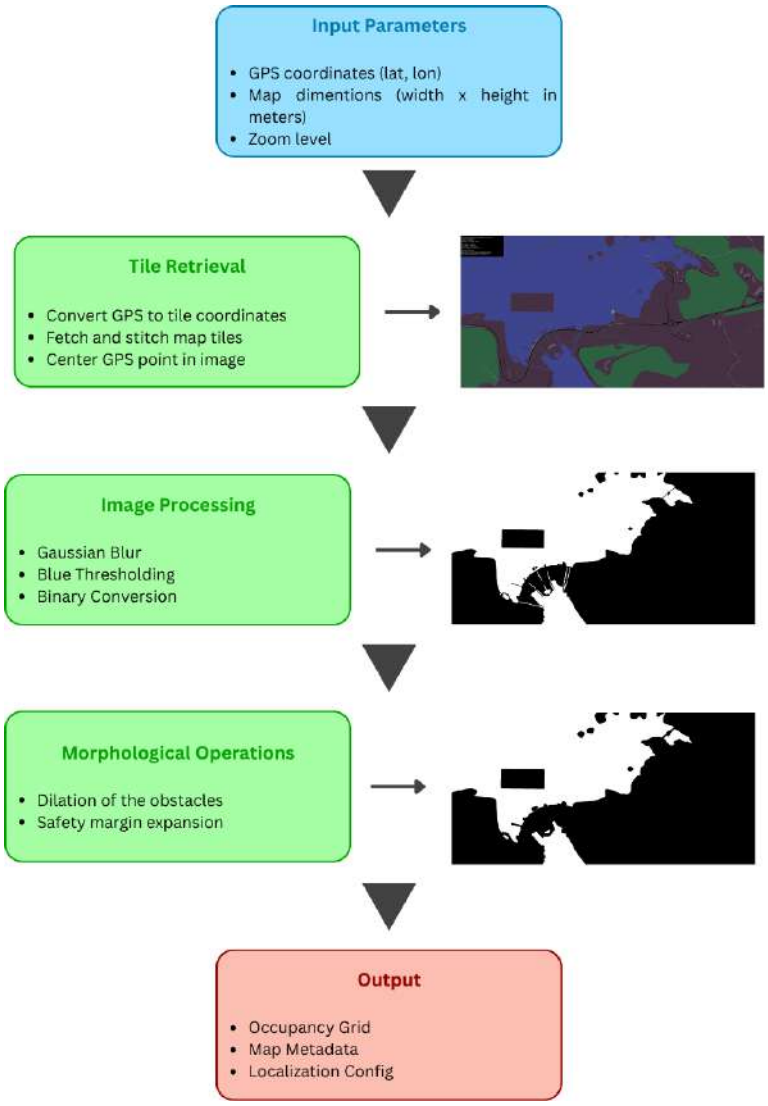


Figure 3.15: Structure of the map acquisition module used for generating the mission area occupancy grid from geospatial data.

that isolates water regions—typically rendered in blue—producing a clean binary map. A dilation operation is then performed to expand non-navigable regions in the occupancy grid, ensuring the USV maintains a safe operational buffer from shorelines and fixed sea structures. This safety margin is especially critical in environments influenced by tides, where the water level can vary significantly over time. In coastal areas like Portugal, the tidal range can reach up to approximately 4 meters[3]. Such variations can cause previously submerged obstacles—such as rocks, submerged piers, or ramps—to become exposed or dangerously shallow, posing risks to navigation. At low tide, these obstacles may extend further into the water, effectively reducing the navigable area available to the USV. Conversely, at high tide, water may temporarily cover structures close to the shore, creating a misleading perception of safe passage. To mitigate these risks, the dilation

[3]https://birdecology.wixsite.com/tidalwings/tagus-estuary-portugal (Accessed on 09/06/2025)

buffer is calculated to account for the worst-case scenario, where the water level is at its lowest and obstacles protrude the most into the navigation zone. The dilation value was conservatively determined using a slope estimate of 40°, a representative approximation of the inclination of common man-made coastal structures such as seawalls and boat ramps[4]. Under this assumption, a horizontal clearance of 5 meters provides sufficient separation across tidal conditions, as calculated by $\frac{4}{\tan(40°)} \approx 5$. This is illustrated in figure 3.16, which shows how tidal variation affects proximity to sloped coastal surfaces.

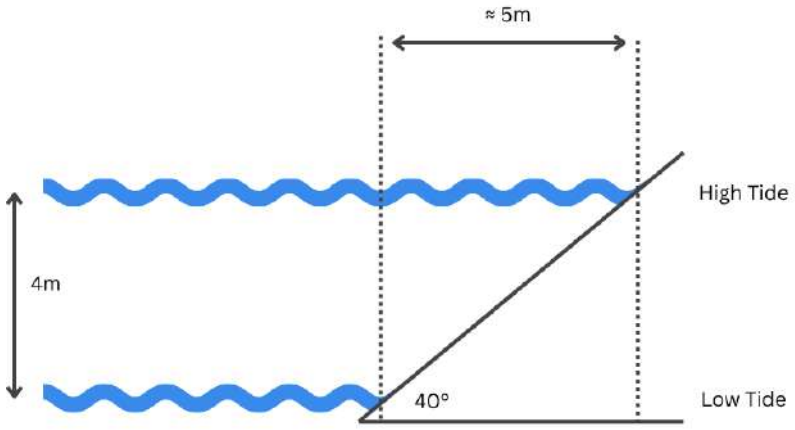


Figure 3.16: Representation of the tidal state in the map acquisition.

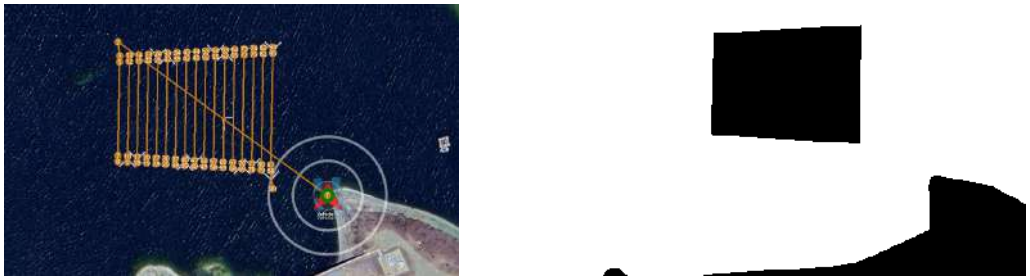
The resulting occupancy grid serves as the basis for both global and local planning modules, enabling reactive and goal-driven behaviors while maintaining real-time performance. Alongside the occupancy grid, this methodology also generates a corresponding configuration file containing essential map metadata, such as resolution, origin, and coordinate frame information, which is required for robot localization during mission execution.

Large structures are typically already present in the online map data. However, when the mission involves inspecting an unknown or newly introduced structure, the corresponding survey area should be treated as a navigational obstacle. This allows the global path planner to more effectively optimize the route by excluding the survey region from the traversable space. To support this functionality, the occupancy grid generation was extended to process the UAV survey waypoints, generate a bounding polygon around the surveyed area, and integrate it directly into the occupancy grid. An example of this functionality is shown in figure 3.17.

3.3.2 Spatial-Temporal Mission Execution

The generated occupancy grid and mission waypoints provide the spatial configuration for autonomous operation, but coordinating the temporal sequence of mission events requires a structured approach to system state management. A state machine architecture ensures that complex

[4]https://www.scribd.com/document/170813605/Coastal-Engineering-Manual-CEMPartVI-Chap5p2-Fundamentals-of-Design (Accessed on 24/06/2025)



(a) Survey waypoints in QGC. (b) Obstacle stitched to the map.

Figure 3.17: Demonstration of the obstacle creation from QGC survey.

multi-vehicle operations proceed in the correct order while handling contingencies and maintaining safety throughout the mission. The mission is governed by a robust state machine architecture designed to structure and sequence each phase of the operation. Figure 3.18 illustrates the full layout of this state machine, which provides clear transitions, systematic flow, and robust mechanisms for error handling and emergency response.

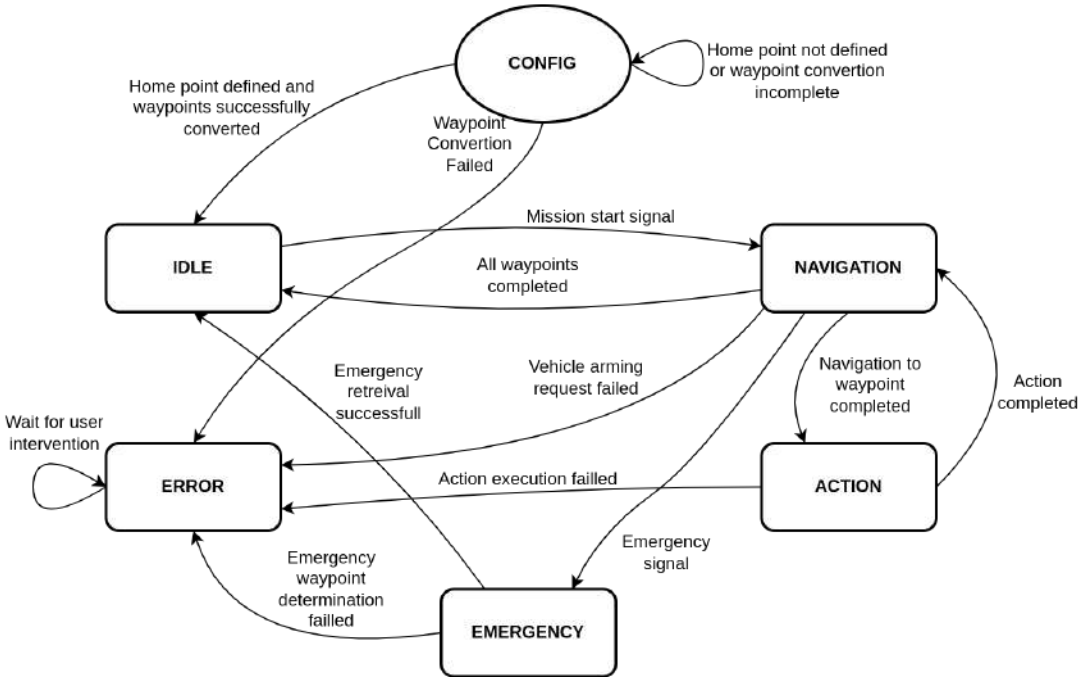


Figure 3.18: State Machine of the mission planner.

The mission begins in the CONFIG state, where the system executes all initialization routines. During this phase, the mission planner retrieves the UAV's waypoint list from QGC via the MAVROS interface, identifies takeoff and landing coordinates, and converts all relevant GPS positions to local map coordinates for navigation. If a failure occurs during coordinate conversion, the system transitions to the ERROR state, preventing further execution with invalid data.

After successful configuration, the system moves to the IDLE state. Here, it remains in

standby, fully initialized and awaiting the start signal from the operator. This state provides a safety buffer before autonomous actions begin. Upon receiving the start command, the system enters the NAVIGATION state. It initiates the arming sequence for the thrusters, and once completed, the onboard navigation manager begins guiding the USV through the mission waypoints. Throughout this state, the system actively monitors for waypoint completion and emergency conditions.

At each waypoint, the system transitions into the ACTION state, where any predefined tasks are executed. The *Action Executor* oversees these operations, providing status feedback. Once the action is complete, the mission returns to the NAVIGATION state to proceed to the next waypoint.

The EMERGENCY state serves as a critical safety feature. It can be triggered from within the NAVIGATION state by user to request an emergency retrieval. In this state, the system invokes an emergency waypoint calculation service that selects a safe rendezvous point based on current USV and UAV positions, home location, safety margins, and defined search radius. This ensures a controlled recovery of both vehicles in hazardous scenarios.

The ERROR state is activated when critical faults are detected—such as navigation failures, action timeouts, or communication loss. In this state, the system halts mission progression, logs diagnostic data, and awaits operator intervention, thereby minimizing risk and preventing undefined behavior. This state machine architecture provides an organized mission logic in a systematic way. It supports a robust error handling and emergency situations.

3.3.3 Emergency Handeling

While the primary mission architecture handles normal operational scenarios, autonomous maritime operations must also incorporate robust contingency planning for emergency conditions. Situations such as UAV battery depletion, equipment failure, or environmental hazards can compromise mission safety, requiring the system to rapidly identify safe recovery options and coordinate emergency landing procedures. In coordinated UAV-USV missions, this involves determining a safe landing zone that ensures the UAV can return safely with minimal mission disruption. The emergency landing waypoint optimization problem aims to compute an optimal meeting location $\mathbf{W} = (x_w, y_w)$ within the maritime operational area where the UAV can land safely on the USV. This solution must minimize total coordination cost while satisfying critical safety and accessibility constraints to protect both the mission and the vehicle assets during unforeseen events.

Given the operational parameters including USV position $\mathbf{V} = (x_v, y_v)$, UAV position $\mathbf{U} = (x_u, y_u)$, home base position $\mathbf{H} = (x_h, y_h)$, speed differential $\alpha = v_{uav}/v_{usv}$, occupancy grid $\mathcal{G}(x,y) \in \{0, 1, -1\}$ representing navigable water, obstacles, and unknown areas, safety radius r_{safe} and search radius R_{search}, the optimization objective is formulated as a multi-criteria problem that balances coordination efficiency, mission recovery, and safety requirements for emergency landing operations.

The emergency landing waypoint is optimized by minimizing a composite mission cost function that reduces the travel time for both vehicles, shortens the distance to the home location, and

ensures that the selected waypoint lies within a cost-free cell of the costmap, as represented in the following equation:

$$J(\mathbf{W}) = T_{coordination}(\mathbf{W}) + C_{return}(\mathbf{W}) + P_{safety}(\mathbf{W}), \tag{3.38}$$

where $J(\mathbf{W})$ is the overall cost function to be minimized; $T_{coordination}(\mathbf{W})$ represents the combined travel time cost for both vehicles; $C_{return}(\mathbf{W})$ accounts for the distance to the home location; and $P_{safety}(\mathbf{W})$ penalizes unsafe waypoint placement based on the costmap. The coordination time represents the bottleneck constraint for simultaneous arrival at the landing zone:

$$T_{coordination}(\mathbf{W}) = \max(T_{usv}(\mathbf{W}), T_{uav}(\mathbf{W})), \tag{3.39}$$

with individual travel times:

$$T_{usv}(\mathbf{W}) = \frac{\|\mathbf{W} - \mathbf{V}\|}{v_{usv}}, \tag{3.40}$$

$$T_{uav}(\mathbf{W}) = \frac{\|\mathbf{W} - \mathbf{U}\|}{\alpha \cdot v_{usv}}. \tag{3.41}$$

This formulation ensures that the landing waypoint selection minimizes the maximum travel time, accounting for the speed differential between the USV and UAV, which is critical for emergency landing coordination where timing precision is essential. To facilitate mission recovery, the algorithm incorporates a return penalty that favors waypoints closer to the home base:

$$C_{return}(\mathbf{W}) = \beta \cdot \|\mathbf{W} - \mathbf{H}\|, \tag{3.42}$$

where $\beta \in [0.05, 0.15]$ is a weighting factor that balances immediate landing coordination needs with long-term mission efficiency and recovery operations. The safety penalty ensures obstacle avoidance within the required clearance radius for safe landing operations:

$$P_{safety}(\mathbf{W}) = \begin{cases} 0 & \text{if } \forall \mathbf{p} \in \mathscr{B}(\mathbf{W}, r_{safe}) : \mathscr{G}(\mathbf{p}) = 0 \\ \infty & \text{otherwise} \end{cases}, \tag{3.43}$$

where $\mathscr{B}(\mathbf{W}, r_{safe})$ represents the safety buffer zone around the landing waypoint, ensuring adequate clearance from maritime obstacles, shallow water, or other navigation hazards that could compromise the emergency landing operation.

The complete optimization problem is formulated as:

$$\begin{aligned} \text{minimize}_{\mathbf{W}} \quad & J(\mathbf{W}), && (3.44)\\ \text{subject to} \quad & \mathbf{W} \in \mathscr{S} \subseteq \mathbb{R}^2, && (3.45)\\ & \mathscr{G}(\mathbf{W}) = 0, && (3.46)\\ & \|\mathbf{W} - \mathbf{V}\| \leq R_{search}, && (3.47)\\ & \|\mathbf{W} - \mathbf{U}\| \leq R_{search}, && (3.48) \end{aligned}$$

where $\mathscr{S}$ represents the feasible search region bounded by the maritime operational area and navigable waters.

Due to the non-convex nature of maritime occupancy grid constraints and potential optimization failures in complex marine environments, the system implements a robust four-stage hierarchical approach that guarantees solution availability for emergency landing scenarios. The primary optimization stage employs bounded L-BFGS-B optimization [72] with penalty methods for constraint handling. The initial guess is calculated using a weighted centroid approach:

$$\mathbf{W}_0 = \frac{\alpha \cdot \mathbf{V} + \mathbf{U}}{\alpha + 1}. \tag{3.49}$$

This formulation accounts for the speed differential by positioning the initial guess closer to the slower USV, improving convergence efficiency for emergency landing coordination. The search bounds are defined as:

$$\begin{aligned} x_{min} &= \min(x_v, x_u) - R_{search}, && (3.50)\\ x_{max} &= \max(x_v, x_u) + R_{search}, && (3.51)\\ y_{min} &= \min(y_v, y_u) - R_{search}, && (3.52)\\ y_{max} &= \max(y_v, y_u) + R_{search}. && (3.53) \end{aligned}$$

When gradient-based optimization fails to find a feasible landing solution, the algorithm proceeds to systematic exploration using expanding circular search regions. The search pattern employs concentric circles with radius $R \in \{50, 100, 200, 500, 1000\}$ meters, appropriate for maritime emergency landing scenarios. For each navigable water cell (x_i, y_i) within radius R, the algorithm evaluates:

$$C_i = \max\left(\frac{d_{usv,i}}{v_{usv}}, \frac{d_{uav,i}}{v_{uav}}\right) + \beta \cdot d_{home,i}, \tag{3.54}$$

where $d_{usv,i} = \|(x_i, y_i) - \mathbf{V}\|$, $d_{uav,i} = \|(x_i, y_i) - \mathbf{U}\|$, and $d_{home,i} = \|(x_i, y_i) - \mathbf{H}\|$. The optimal landing waypoint is selected as:

$$\mathbf{W}^* = \text{argmin}_{(x_i, y_i) \in \mathscr{F}_R} C_i, \tag{3.55}$$

where $\mathscr{F}_R$ represents the set of all navigable water cells within radius R suitable for emergency landing operations. There is a schematic representation of this stage in figure 3.19.

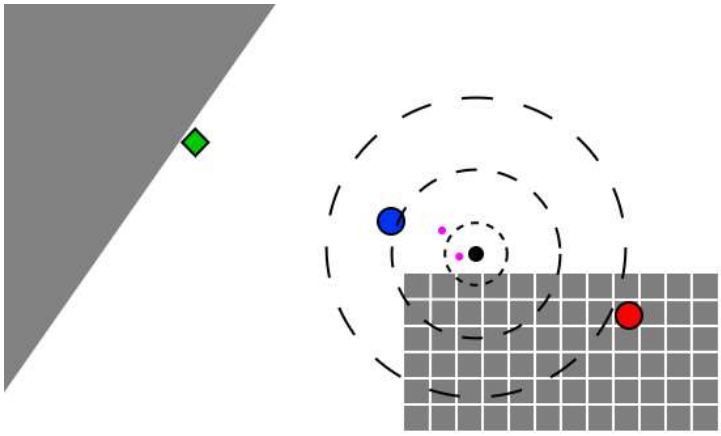


Figure 3.19: Emergency Stage 2. The green diamond indicates the home location, the blue circle represents the USV, and the red circle the UAV. The black dot denotes the weighted centroid, dashed lines illustrate the search radii, and purple dots mark the candidate emergency waypoints.

If regional search fails, the algorithm performs coarse grid sampling across the entire maritime occupancy map. The sampling strategy examines every k-th grid cell where $k = 10$ for computational efficiency. The evaluation metric uses average distance to both the USV and UAV:

$$D_{avg,i} = \frac{d_{usv,i} + d_{uav,i}}{2}. \tag{3.56}$$

The final stage provides guaranteed solution availability through hierarchical fallback mechanisms specifically designed for emergency landing scenarios: if $\mathscr{G}(\mathbf{H}) = 0$, return $\mathbf{W} = \mathbf{H}$; otherwise find closest navigable water to $\mathbf{H}$ within 2km radius; or return any accessible water cell in the map suitable for emergency landing. The proximity search employs a spiral pattern where for $r = 1$ to r_{max} step Δr and $\theta = 0$ to 2π step $\Delta\theta$, candidate landing positions are evaluated as $\mathbf{H} + r(\cos\theta, \sin\theta)$.

The multi-stage algorithm guarantees termination with a valid emergency landing solution if at least one navigable water cell exists in the occupancy grid. This follows from the hierarchical design where stages 1-2 may fail due to local optima or constraint violations, stage 3 performs exhaustive sampling with success probability $P = F/N$ where F is the number of navigable water cells and N is total cells, and stage 4 provides deterministic fallback with success probability 1 if any suitable landing area exists. Therefore, the overall success probability approaches 1 as long as the maritime map contains accessible navigable water for emergency landing operations.

Chapter 4

Experimental Results

To demonstrate the mission planning algorithm, a series of tests were conducted to validate the implemented approach by verifying each module of the pipeline. This chapter begins with section 4.1, which describes the simulation setup and validates the four main phases of the mission pipeline: the autonomous navigation architecture of the NEST, the pre-planning definition, the mission execution, and the emergency handler. Then the results obtained from the simulation tests and compares the proposed approach with a mission performed by a solo UAV. Section 4.3 describes the navigation validation of the NEST USV in a near-real scenario, and analyzes the outcomes of the real-world implementation tests.

4.1 Introduction

The simulation tests were conducted in a physics-based environment to validate the USV architecture prior to real-world deployment. The *Gazebo* simulator [73] was used, enhanced with an ocean plugin to more accurately model buoyancy and the physical interactions between the NEST and its intended maritime environment. *Gazebo* was selected due to its strong integration with ROS and the availability of extensive documentation, including tools for sensor simulation.

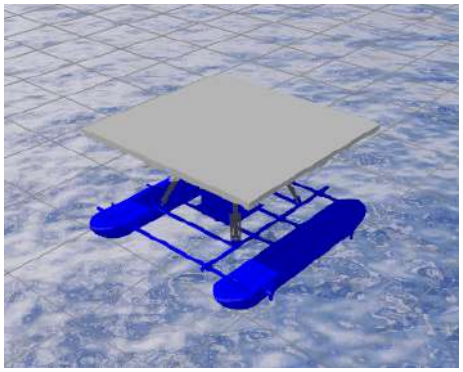


Figure 4.1: NEST USV model in the *Gazebo* simulator.

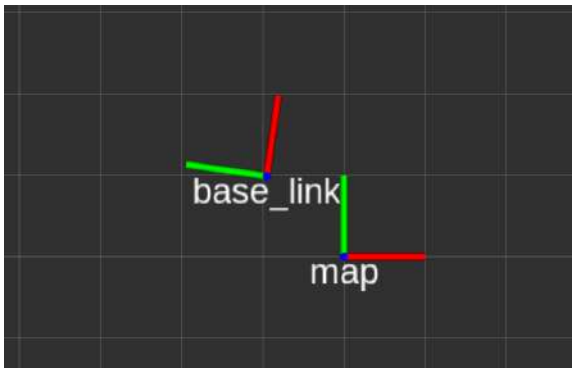


Figure 4.2: Representation of the `base_link` and `map` frames in the *RViz* tool.

Figure 4.1 shows the NEST model operating in the *Gazebo* simulator. This setup includes all core components and sensor placements, allowing realistic interactions with a simulated marine environment.

For the NEST simulation, a Software-In-The-Loop (SITL) approach was employed instead of a purely virtual simulation. This method allows the hardware-software interface to be realistically simulated, where sensor data is generated virtually and processed by the actual autopilot firmware—in this case, ArduSub. This setup enables the validation of communication protocols and system behavior without the need for physical deployment or simplified abstractions.

To support debugging and validation of localization accuracy, *RViz* was employed to visualize the coordinate frame transformations. As illustrated in figure 4.2, the `map` and `base_link` frames are offset by $(-10,\ 10)$ meters to enhance visualization and avoid frame overlap.

The simulated weather conditions were kept near-ideal, with no waves or wind, to focus the validation on system functionality and integration.

4.2 Simulation

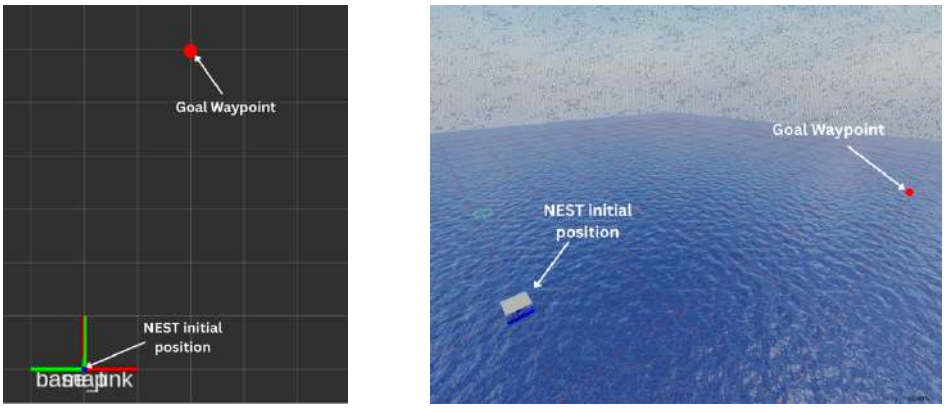
The simulation tests were designed to validate the four main contributions of this thesis. The first objective focuses on the autonomous navigation pipeline of the USV, including the evaluation of its localization accuracy during mission execution, global path generation, velocity command output, and communication with low-level control layers responsible for thruster actuation. The second objective involves validating the pre-mission planning process, which includes map acquisition and analysis, as well as the projection of the takeoff and landing waypoints into map coordinates, and the third objective evaluates the complete mission execution. Finally, the forth objective evaluates the emergency handling algorithm and its correct execution under simulated failure conditions.

Waypoint Navigation

The first test focuses on validating the NEST's ability to autonomously navigate to predefined waypoints. In this scenario, a goal waypoint was set at coordinates (20, 60) meters within the `map` frame. Both *RViz* and *Gazebo* environments use a grid resolution of 10 meters per cell to aid visualization. To simplify the evaluation, an empty costmap was used, meaning no obstacles were present in the environment. Figure 4.3 shows the initial conditions of the test, including the vehicle's starting position and the target waypoint located at the coordinates $(x, y) = (20, 60)\ m$.

Once the goal waypoint was assigned, the navigation system generated a path, as illustrated in figure 4.4, where the red line indicates the computed trajectory. Based on this path, the algorithm calculated the necessary velocity commands to enable smooth path tracking. These velocities were then translated into thrust signals, establishing low-level communication with the vehicle's thrusters.

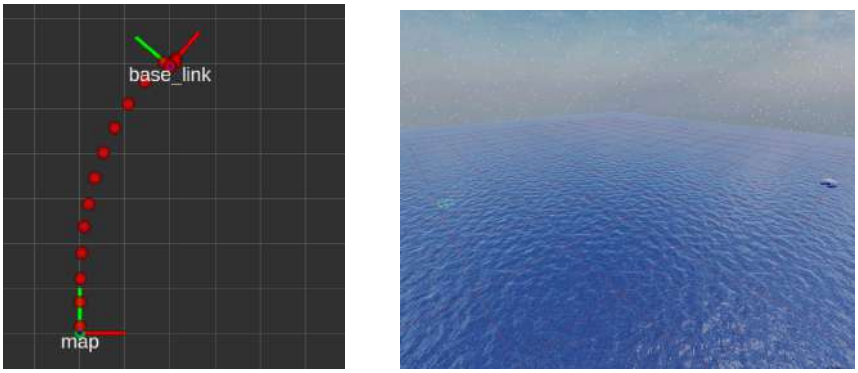
At the end of the navigation process, the final state of the navigation is illustrated in figure 4.5. The figure presents both the *RViz* and *Gazebo* environments to offer a comprehensive view of the



(a) *RViz* visualization. (b) *Gazebo* visualization.

Figure 4.3: Visualization of navigation initial conditions.

Figure 4.4: Computed path for (20, 60) meters goal coordinates



(a) *RViz* visualization. (b) *Gazebo* visualization.

Figure 4.5: Visualization of navigation final conditions.

result. In *RViz*, the trajectory is marked by a series of circles, each representing the position of the NEST at 5-second intervals. This temporal visualization allows for an intuitive understanding of the USV's motion profile throughout navigation, which exhibited a smooth and consistent trajectory to the goal waypoint. Concurrently, the *Gazebo* snapshot confirms the final position of the

vehicle in the simulated environment. Based on the visual data, the NEST traveled approximately 63.2 meters in around 60 seconds, achieving an average linear velocity of approximately 1.05 m/s, consistent with expected performance under ideal conditions.

Map Acquisition

After completing the waypoint navigation validation, the next step was to validate the pre-planning process, starting with the acquisition of the mission map. To accomplish this, the NEST vehicle was spawned in simulation using the GPS coordinates 38.198818° N, 7.491545° W, placing it in the Alqueva dam region. Figure 4.6 presents a screenshot of the QGC interface, displaying a satellite view of the environment and the initial vehicle location, denoted by the letter `L`.

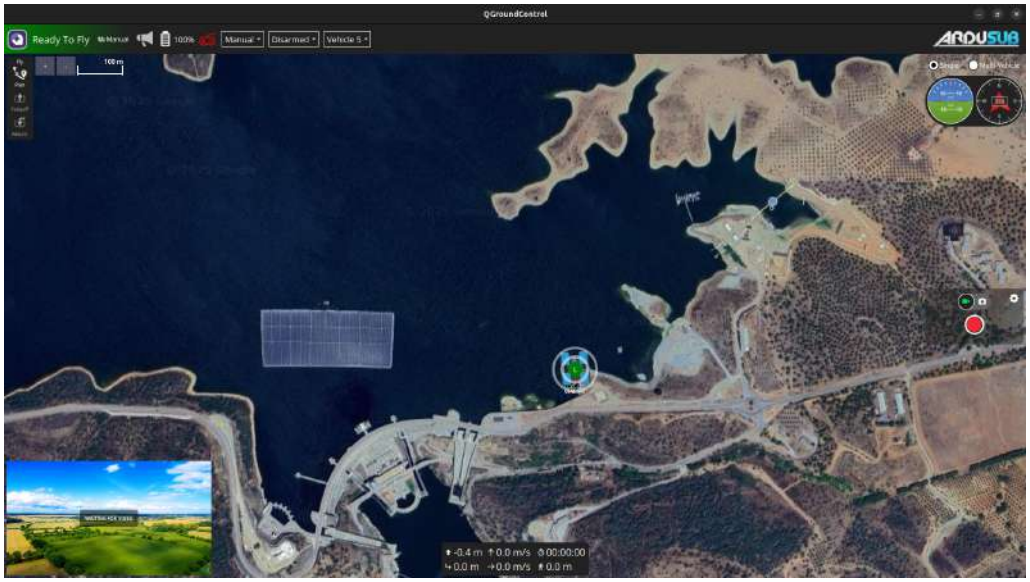


Figure 4.6: Satellite view of the Alqueva dam area in QGC, with the vehicle spawn point marked by the `L` symbol.

Since the mission objective is to survey the floating photovoltaic (PV) plant visible in the satellite imagery, the mission area was defined as 2000 meters in width and 1000 meters in height to ensure complete coverage. The map was automatically retrieved from the *OpenStreetMap* web API, and the result is shown in figure 4.7. In this image, the yellow ⊕ symbol marks the NEST's position, and the top-left corner displays the generated metadata, useful for debugging and validation.

Next, a pre-processing step applies a blur filter and converts the image to a binary format. This isolates water bodies as navigable regions, shown in figure 4.8, where land and non-navigable areas are removed.

Finally, a dilation operation is performed on the binary map to introduce a safety buffer around static obstacles and shorelines. This step ensures that the NEST maintains a minimum clearance from hazardous regions during navigation. The final processed map is shown in figure 4.9.

Figure 4.7: Raw mission map downloaded from the *OpenStreetMap* API, showing the NEST's position (yellow ⊕) and metadata for debugging.

Figure 4.8: Binary threshold of the raw map after pre-processing. Water areas are preserved as navigable zones, while land is filtered out.

Figure 4.9: Final navigable map after applying dilation for safety margins, ensuring a safe distance from shores and static obstacles.

Pre-planning

To define the intended mission, the QGC interface was used. Once the mission is configured, the mission planner extracts the take-off and landing waypoints. Figure 4.10 shows the mission setup within the QGC interface.

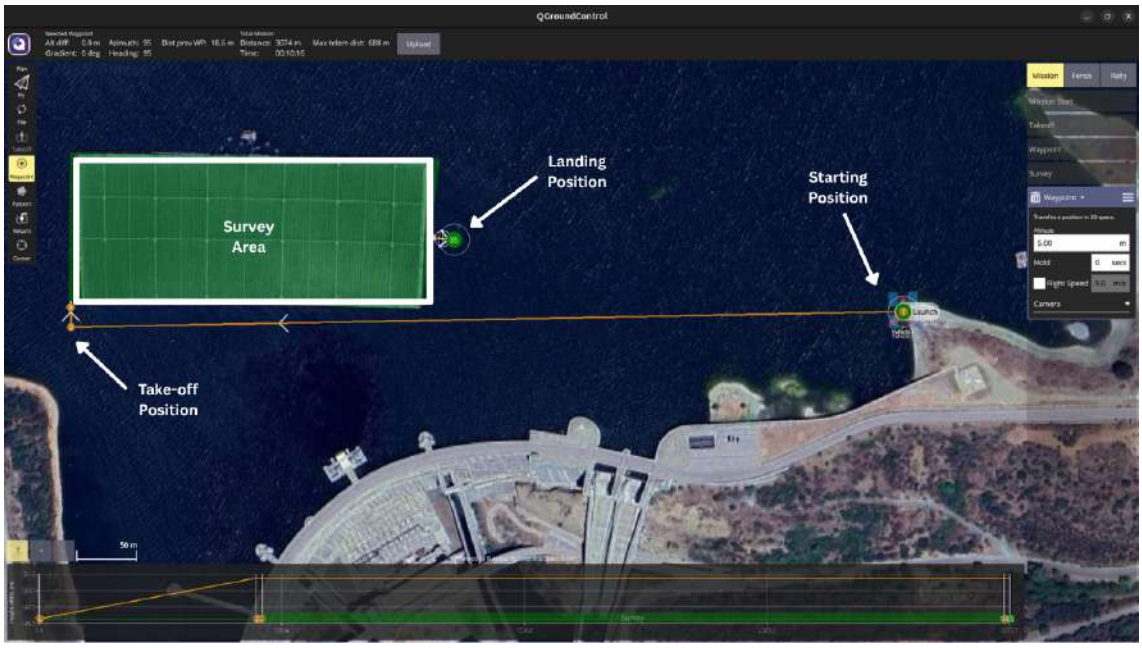


Figure 4.10: Mission setup interface in QGC.

The mission planner converts the GPS coordinates of the takeoff and landing waypoints into map frame coordinates, making them compatible with the global planning module. Table 4.1 lists the GPS coordinates and corresponding converted values. Figure 4.11 presents the result of this conversion within the *RViz* costmap environment, where the red dot indicates the take-off waypoint, and the blue dot marks the landing location.

Table 4.1: Mission waypoints and associated actions.

Name	GPS Coordinates	Map Coordinates (x, y)	Action
Take-off	38.198703° N, -7.499411° W	$(-688.17, -12.79)$	`takeoff`
Landing	38.199352° N, -7.495790° W	$(-371.42, 59.41)$	`landing`
Home	38.198818° N, -7.491544° W	$(0, 0)$	`none`

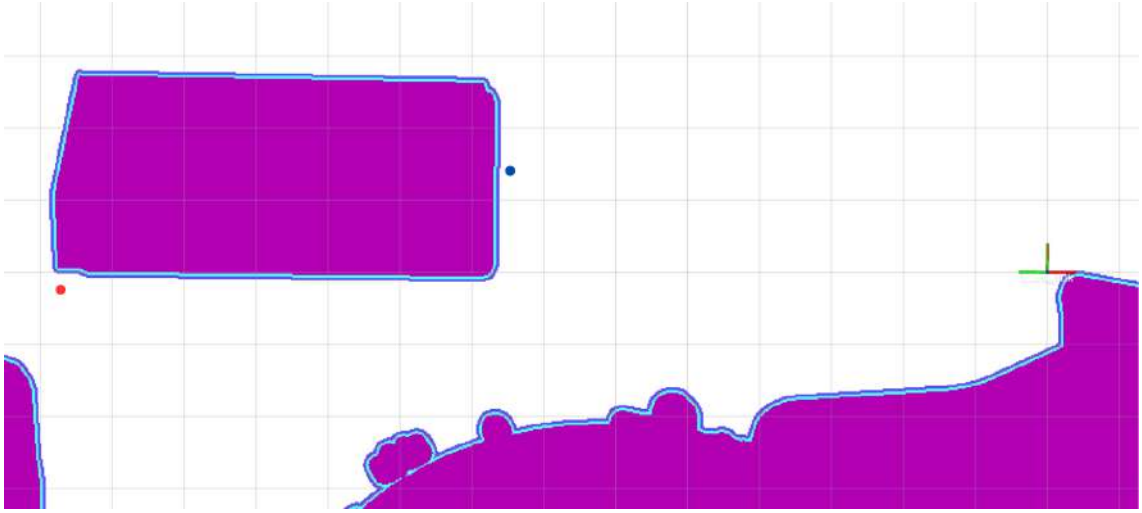


Figure 4.11: *RViz* visualization of waypoint definitions. The red dot indicates the take-off waypoint, while the blue dot marks the landing location. Grid cell size is 50 meters.

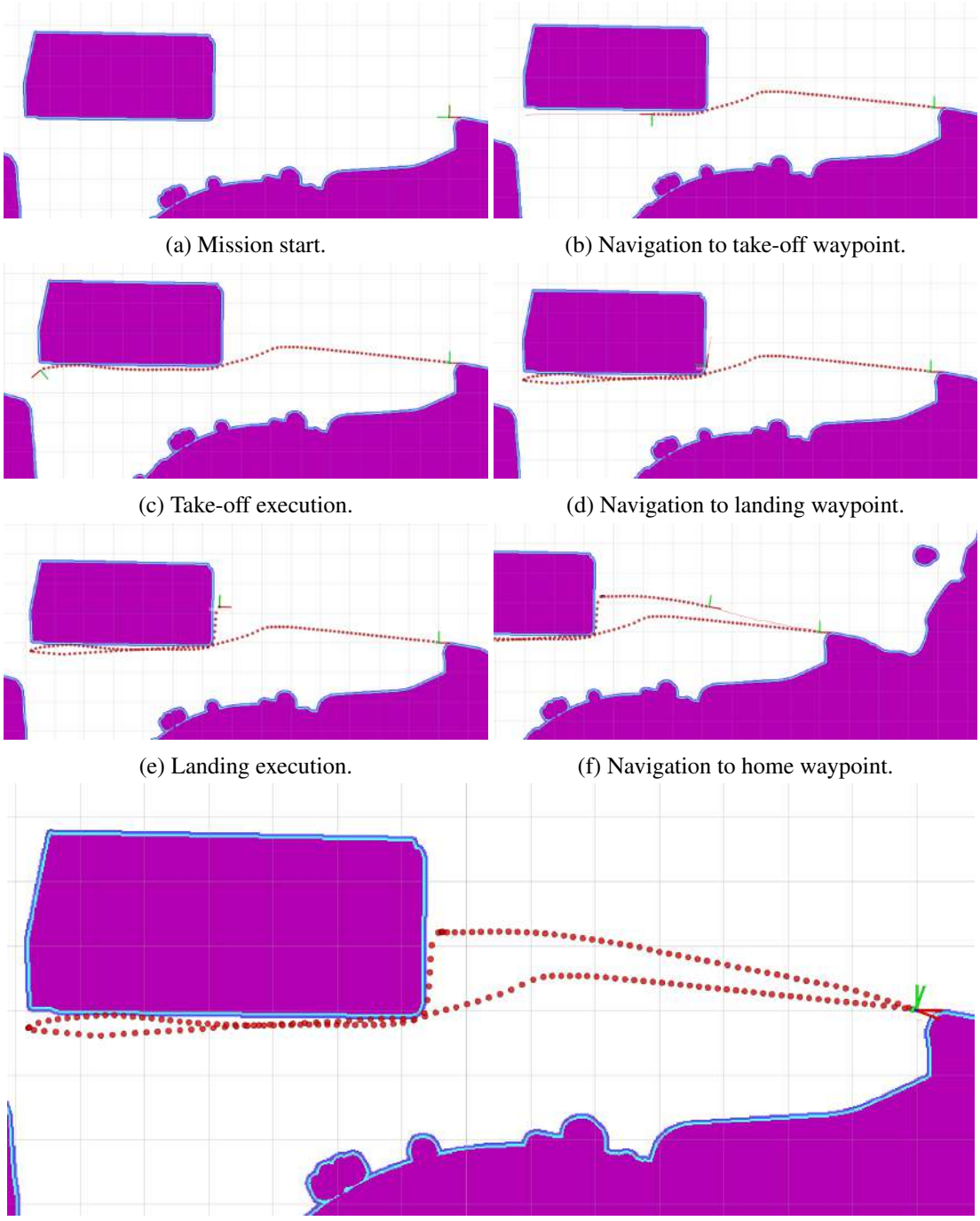
Mission Execution

Figure 4.12 illustrates the mission execution as visualized in *RViz*. During execution, NEST's navigation was monitored through the visualization of planned paths and trajectory feedback. Red dots represent the historical positions of the USV, recorded at 10-second intervals, showing its progression through the mission phases. In the first stage, the vehicle computed the global path to the take-off waypoint and successfully navigated to it, avoiding the floating PV farm along the way. Upon arrival, it correctly issued the necessary commands to deploy the UAV. Following a successful take-off, the vehicle computed the path to the landing waypoint, again avoiding the floating platform. It then waited for the UAV to complete its landing before both vehicles proceeded together to the home position in a coupled configuration.

The full mission lasted approximately 42 minutes, with the UAV conducting its survey for around 17 minutes. A post-mission overview is shown in figure 4.13, based on QGC logs. Figure 4.13a presents the NEST's navigation path, while figure 4.13b displays the trajectory of the UAV.

The mission was successfully completed, with the USV covering a total distance of 1455.2 meters and the UAV flying 2367.5 meters during the survey. The total mission duration was 42 minutes, with the UAV in flight for 16 minutes. Figure 4.14 illustrates the spatial relationship between the take-off and landing waypoints relative to the home position.

In a UAV standalone inspections, the UAV would need to fly 1038.0 meters from the home position to the inspection site and return, in addition to the 2367.5 meters required for the actual survey. This would result in a total flight distance of 3405.5 meters, with 30.5% (1038.0 m) of the total flight being non-inspection travel. Assuming an average flight speed of 2.27 m/s, this corresponds to an estimated mission duration of 25 minutes. This exceeds the average endurance of typical multi-rotor UAVs, which is approximately 20 minutes per battery cycle.



(a) Mission start.

(b) Navigation to take-off waypoint.

(c) Take-off execution.

(d) Navigation to landing waypoint.

(e) Landing execution.

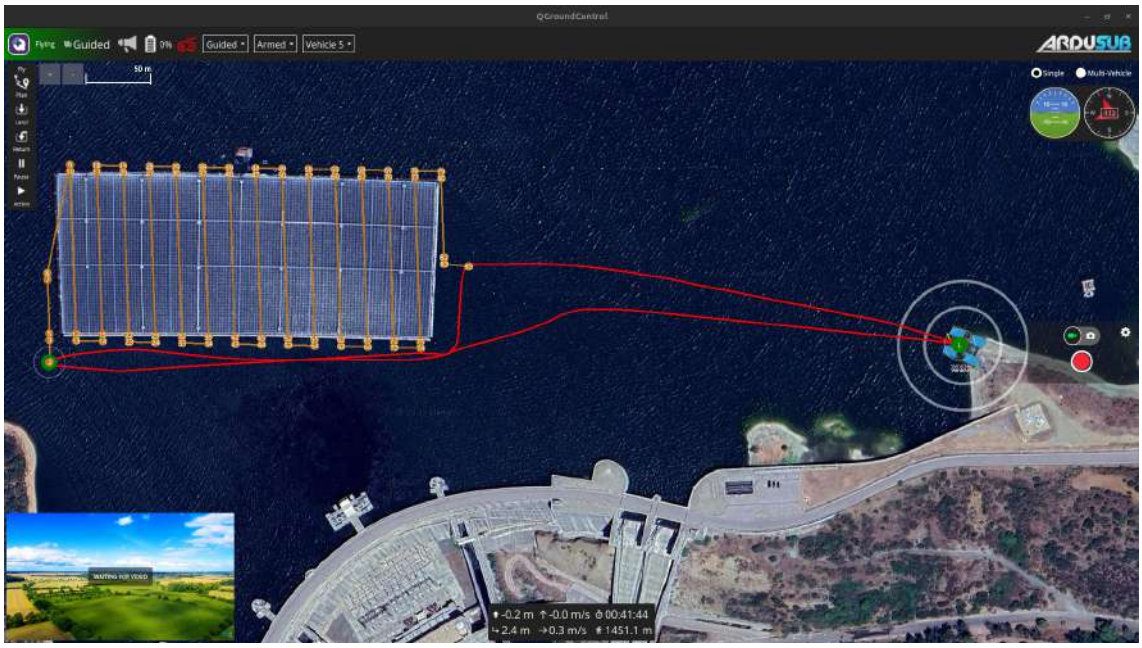
(f) Navigation to home waypoint.

(g) Mission complete.

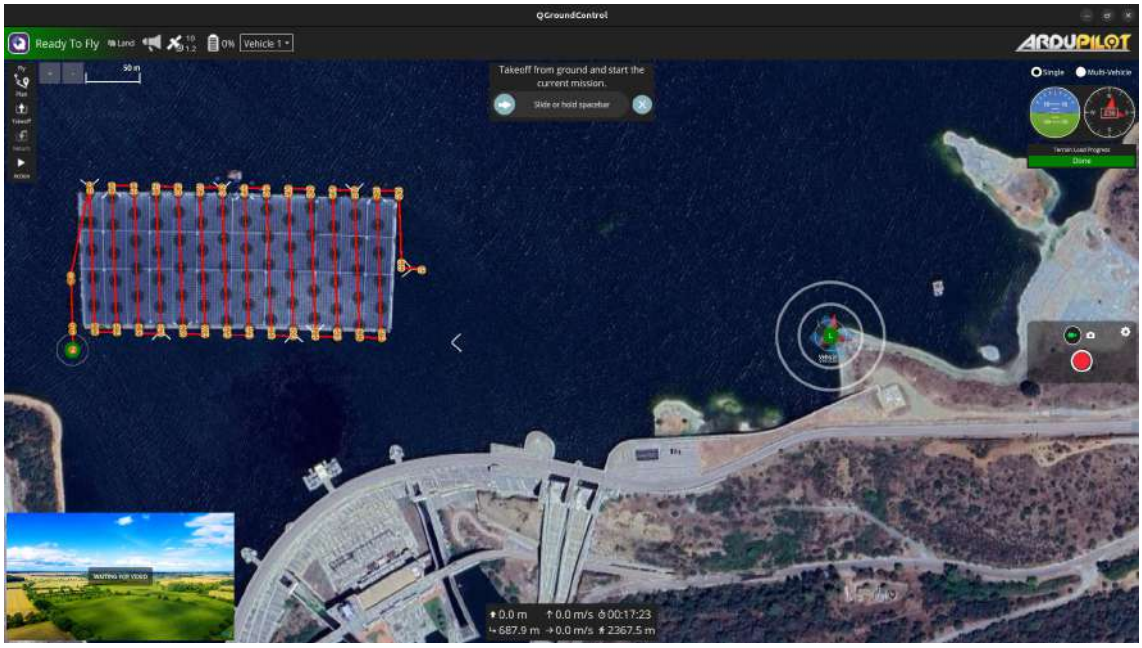
Figure 4.12: Mission progression as visualized in *RViz*, with red dots representing the NEST trajectory over time.

With the cooperative UAV–USV approach, only 50 meters of UAV flight were spent outside the inspection task[1], resulting in 97.89% of the UAV's total flight being dedicated to the inspection

[1]This distance represents the segment between the take-off waypoint and the first inspection point, and between the last inspection point and the landing waypoint.



(a) NEST navigation review in QGC.



(b) UAV navigation review in QGC.

Figure 4.13: QGC visualization of the full mission execution, showing the paths followed by both the USV and UAV.

task. This method reduced UAV flight time by 32% compared to the solo approach. However, the total mission duration increased due to the USV's lower speed: the cooperative mission took approximately 42 minutes to complete, which corresponds to a 68% increase in duration compared to the estimated 25 minutes of the UAV-only scenario. This method exhibits a linear relationship between the total mission time and the distance to the inspection site. However, regardless of the

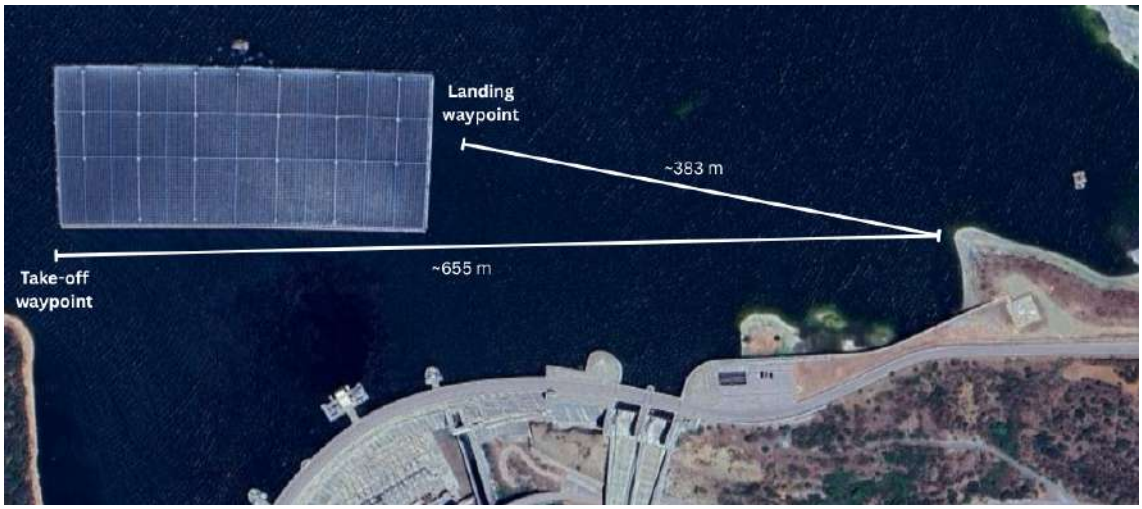


Figure 4.14: Distances from the take-off and landing waypoints to the home position.

inspection site's distance, the UAV's effective mission contribution remains constant at 97.89% of its total travel distance. In contrast, for a UAV-only approach, this efficiency would decrease linearly with increasing distance to the inspection site.

Emergency Handling

To test the emergency retrieval algorithm, a short mission was defined. Figure 4.15 shows the mission configuration interface in QGC, illustrating the intended mission setup. The mission was deliberately interrupted mid-execution to trigger and evaluate the emergency handling mechanism. As a result, the originally planned landing waypoint was not used.

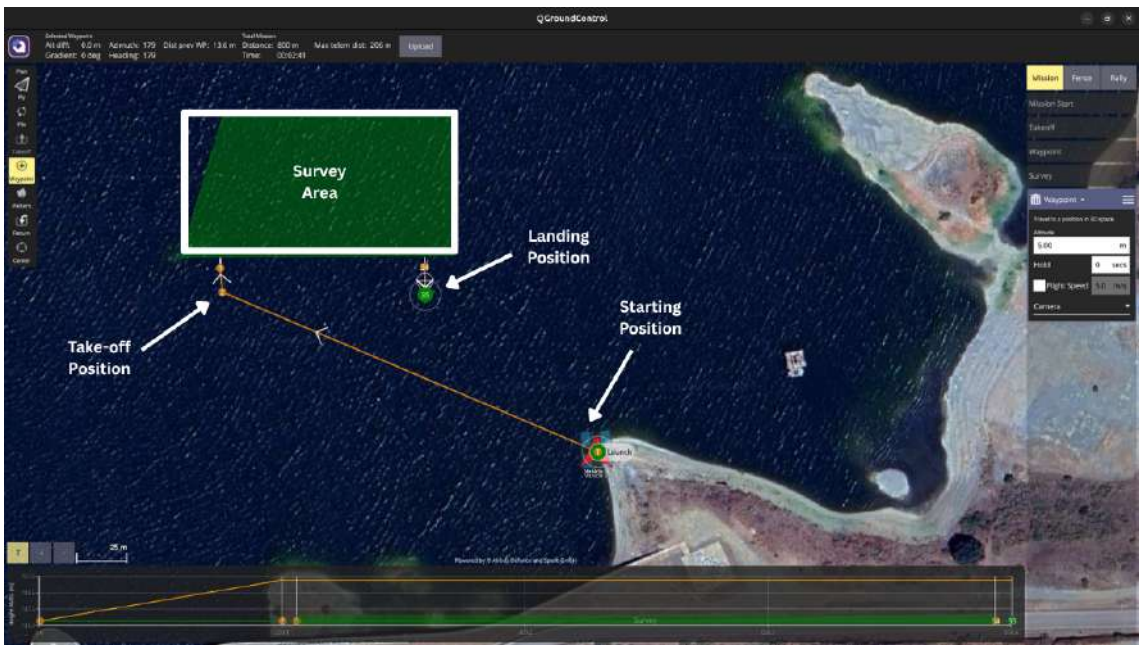


Figure 4.15: Emergency mission setup interface in QGC.

During the mission, the emergency retrieval mechanism was triggered mid-flight while the UAV was executing the survey and the NEST was navigating toward the landing waypoint. Figure 4.16 presents the sequence of events during the mission execution in *RViz*. Red dots represent the position of the NEST at 10-second intervals. Up to the take-off operation, the mission proceeded as previously described. After the UAV took off, the NEST began navigating toward the planned landing waypoint. However, an emergency retrieval was triggered mid-mission. Both the UAV and NEST immediately aborted their current operations and redirected to the computed emergency waypoint, where the UAV successfully landed. Following this, as in the previous test, the two vehicles navigated back to the home position in a coupled configuration.

Figure 4.17 shows a trajectory overview of both vehicles in QGC after mission execution.

Table 4.2 summarizes the position of each vehicle at the moment the emergency was triggered and the computed emergency waypoint used for recovery.

Table 4.2: Vehicles' position upon emergency trigger.

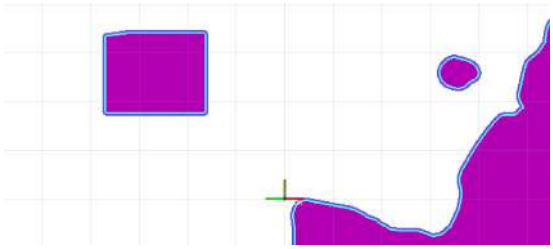
Vehicle / Waypoints	**Map Coordinates (x, y)**	**Navigation Cost**
NEST	$(-127.86,\ 84.85)$	–
UAV	$(-164.24,\ 167.34)$	–
Home	$(0,\ 0)$	–
Emergency	$(-127.65,\ 87.29)$	33.07

The emergency retrieval was completed successfully. The total mission duration was approximately 9 minutes, of which the UAV was in flight for about 3 minutes. At the time the emergency was triggered, the UAV was located 88 meters from the designated emergency waypoint, while the NEST was 2.5 meters away. This difference reflects the distinct velocity profiles of both vehicles. The parameters used for the emergency waypoint request are present in table 4.3.

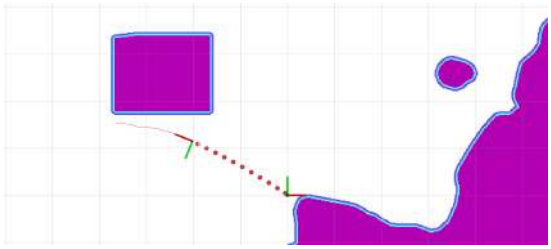
Table 4.3: Emergency waypoint request parameters.

Velocity Ratio	1:8
Safety Radius	2 m
Search Radius	1000 m

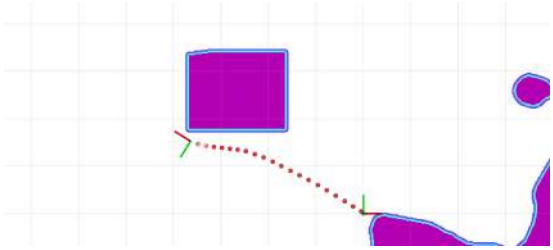
To ensure the NEST reaches the emergency waypoint before the UAV, minimizing airborne loiter time, the used ratio ensured that the emergency waypoint is much closer to the NEST than the UAV. From the moment the emergency was triggered, it took approximately 75 seconds for both vehicles to reach the emergency waypoint and an additional 160 seconds to return to the home position, totaling around 4 minutes for the full emergency retrieval.



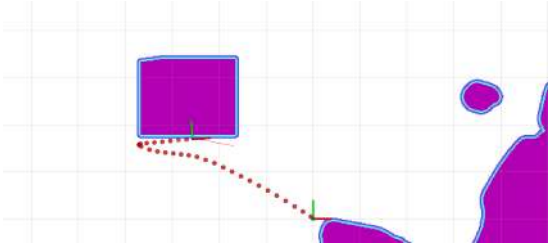
(a) Mission start.



(b) Navigation to take-off waypoint.

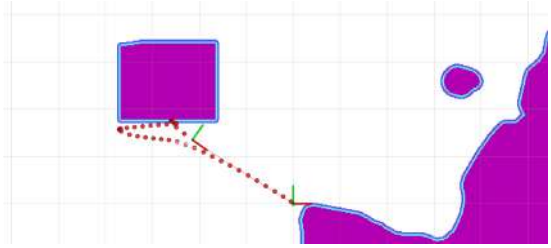


(c) Take-off execution.

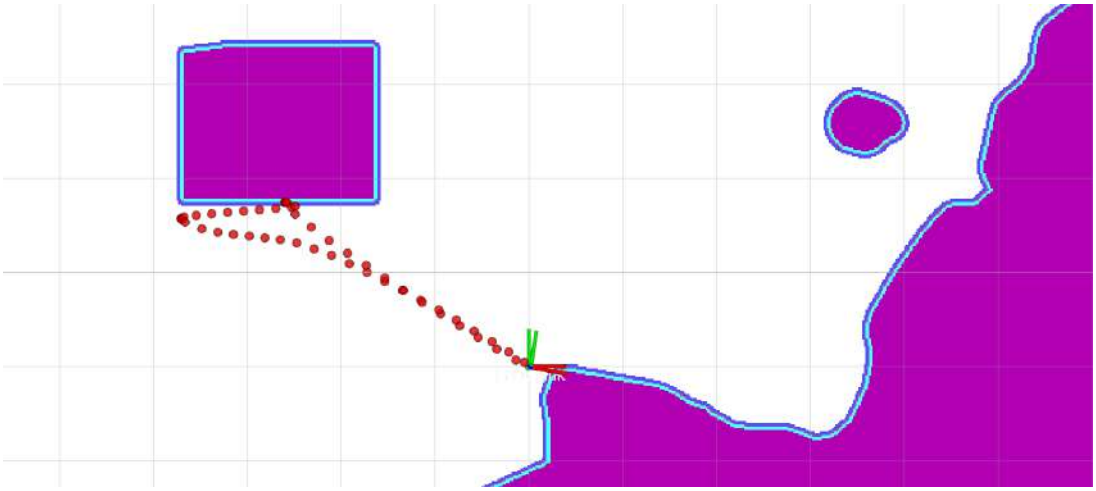


(d) Emergency triggered during flight to landing waypoint.

(e) Emergency landing execution.

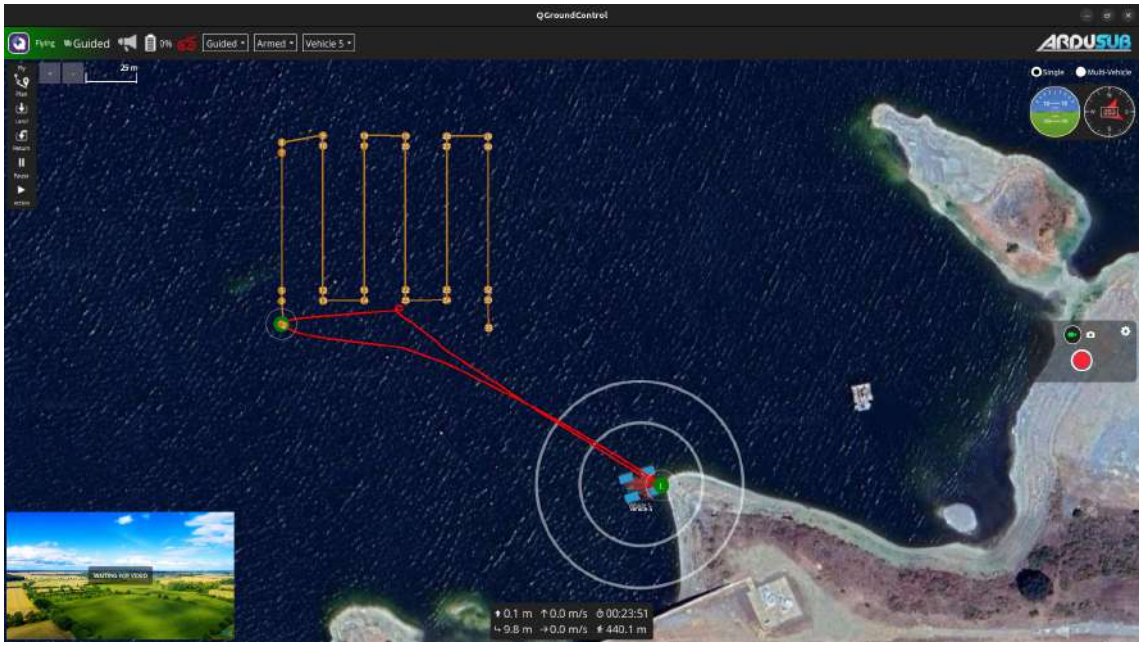


(f) Navigation to home waypoint.

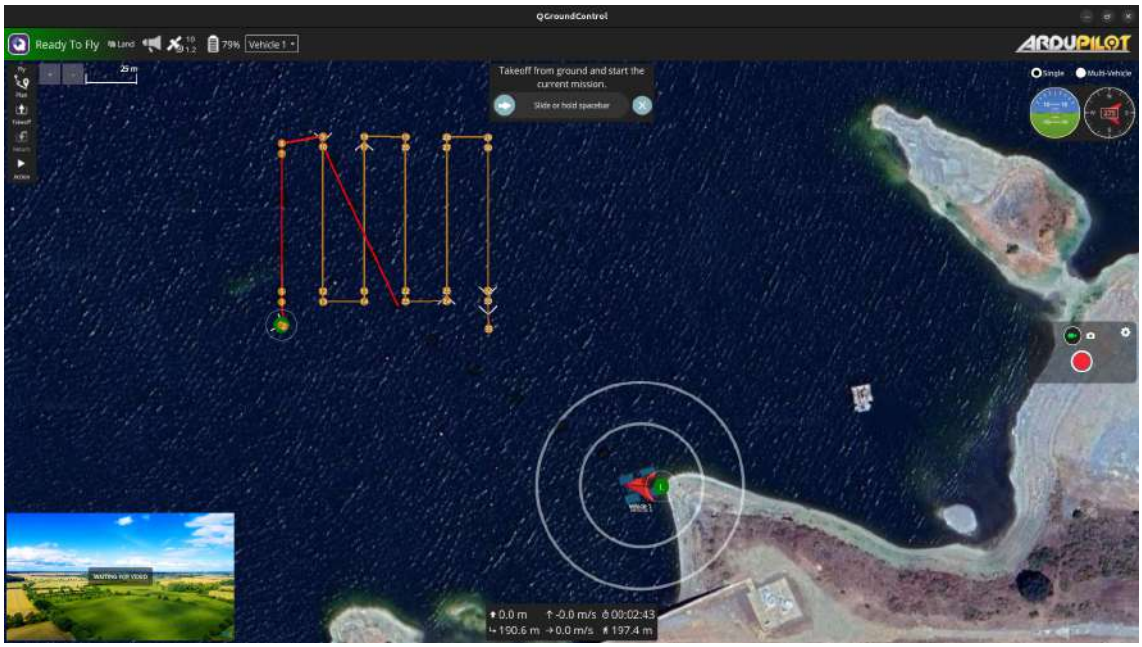


(g) Emergency retrieval complete.

Figure 4.16: Emergency handling sequence in *RViz*. Red dots represent NEST's position every 10 seconds.



(a) NEST navigation review in QGC.



(b) UAV navigation review in QGC.

Figure 4.17: QGC visualization of the emergency execution, showing the trajectories of the USV and UAV.

4.3 Near-Real Scenario

The near-real tests were conducted at the ATLANTIS Test Center in Viana do Castelo [74], comprising two main experiments to validate the proposed methodology. The first experiment involved autonomous waypoint navigation using the NEST platform, aiming to assess the mechanical integrity, kinematic configuration, and performance of the low-level control layer. Following this, a mission concept was executed with the NEST under manual control to validate the full mission planning pipeline, verify communication links with the ground station, and evaluate the effectiveness of the position hold mode required for UAV takeoff and landing operations.

Waypoint Navigation

The mission map was acquired on land at GPS coordinates 41.684714° N, 8.838482° W, as visualized in the QGC interface (Figure 4.18).

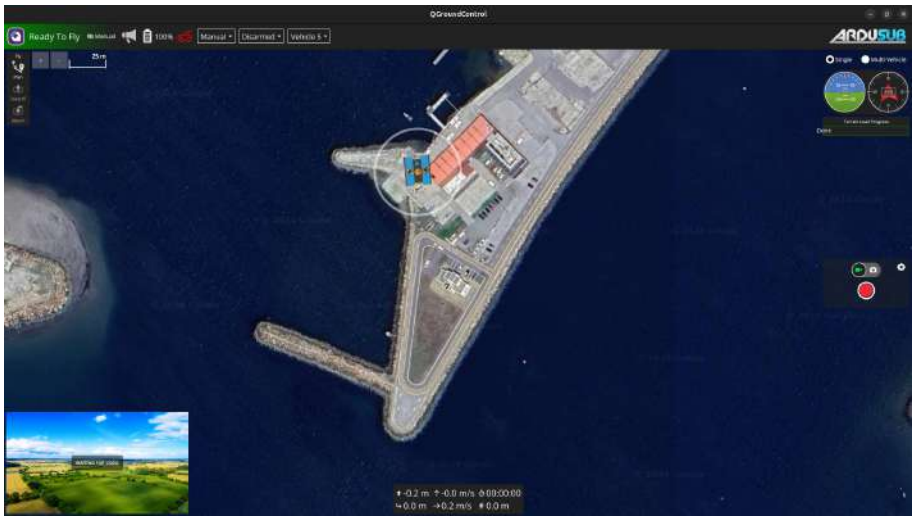
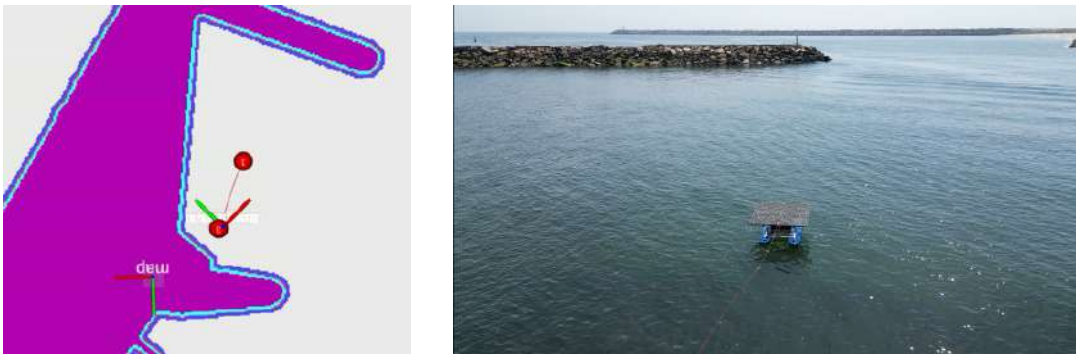


Figure 4.18: Map acquisition through QGC interface.

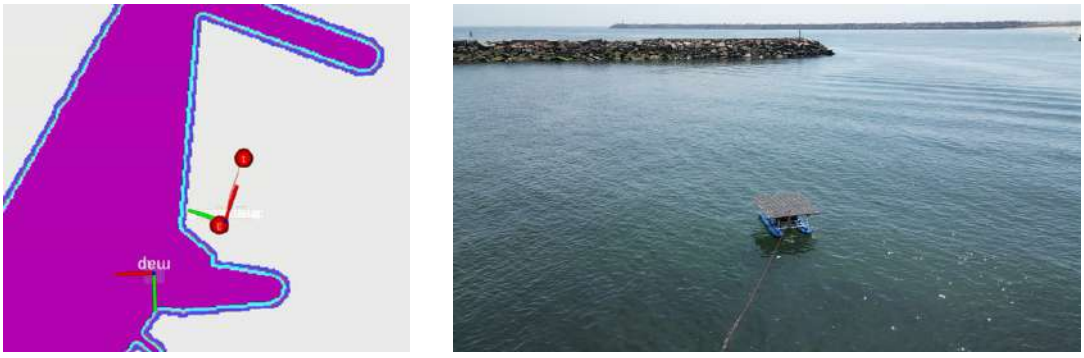
With the map defined, waypoint navigation was executed. Figure 4.19 presents a side-by-side comparison between the *RViz* visualization and real-world footage of the NEST trajectory. The waypoint navigation lasted 49 seconds and covered a distance of approximately 40 meters, corresponding to an average velocity of 0.82 m/s. The suboptimal speed can be attributed to external environmental factors such as water currents and wind.

Mission Pipeline

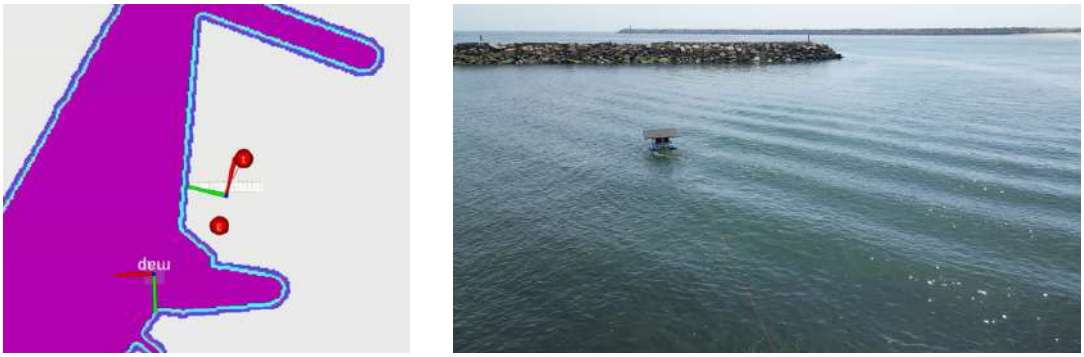
The aim of the second test was to validate the communication link between the NEST and the ground station, as well as the complete mission execution pipeline. For this purpose, the NEST was manually navigated throughout the mission. The mission plan defined in QGC is shown in figure 4.20.



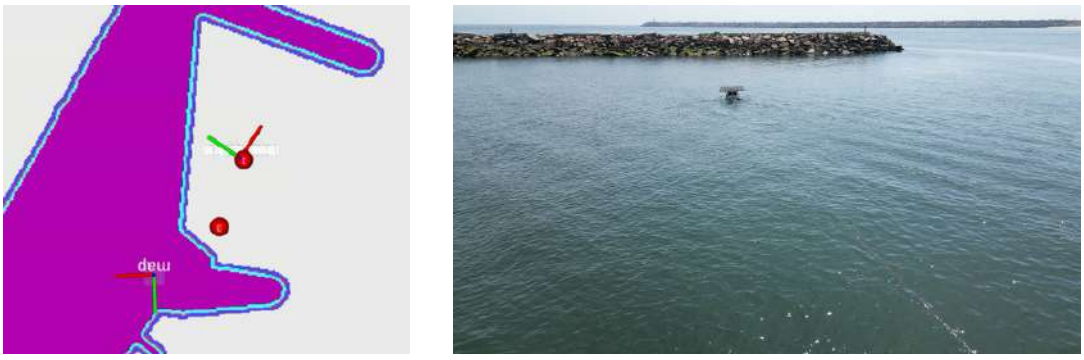
(a) Initial position: *RViz* visualization (left) and real footage (right).



(b) Path alignment: *RViz* (left) and real footage (right).



(c) Navigation to waypoint: *RViz* (left) and real footage (right).



(d) Final position: *RViz* (left) and real footage (right).

Figure 4.19: Waypoint navigation progression shown through simulation (*RViz*) and real-world footage.

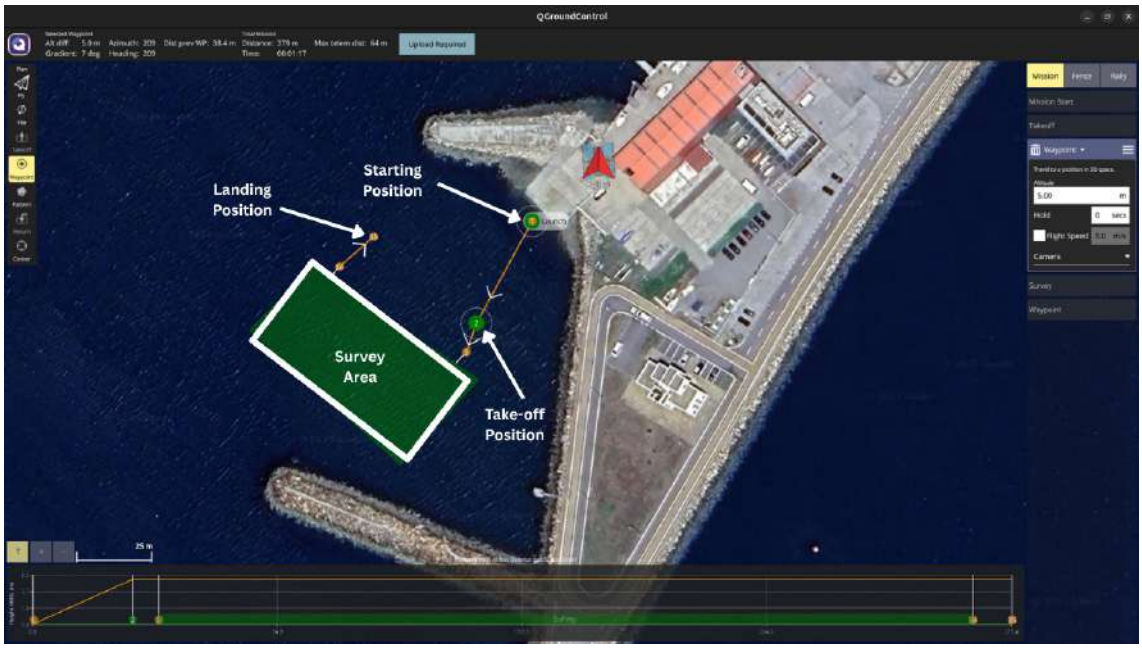
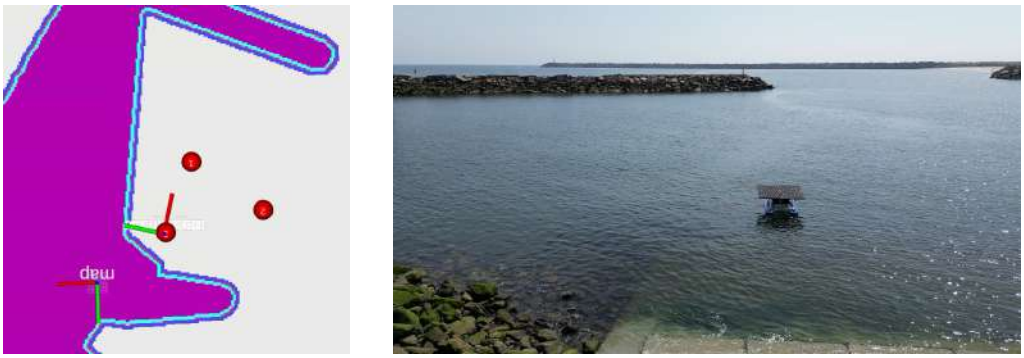
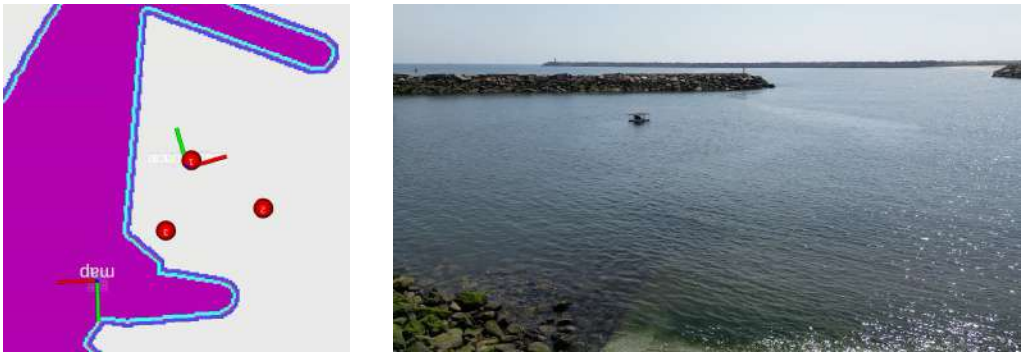


Figure 4.20: Mission setup in QGC interface.

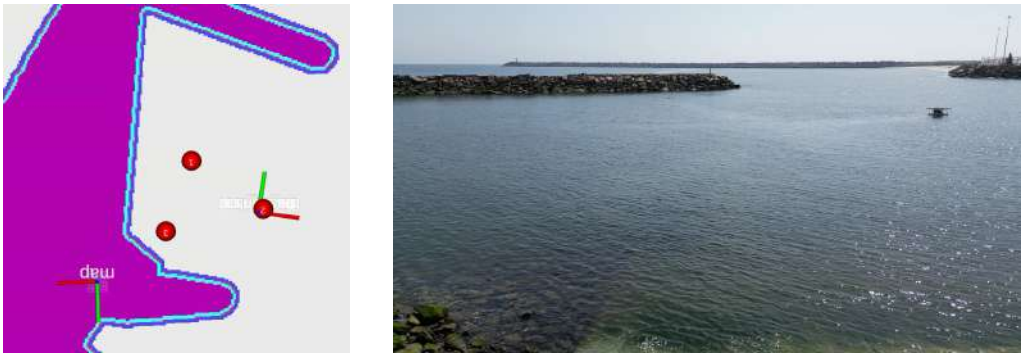
Figure 4.21 presents the progression of the mission execution. This test confirmed the proper functioning of the communication pipeline and mission control logic. At each waypoint, the NEST automatically entered position hold mode, which is a required condition for UAV take-off and landing operations. Upon stabilization at the take-off waypoint, the NEST triggered the take-off command to the UAV. Navigation to the next waypoint was only allowed after receiving a successful take-off confirmation. Similarly, upon arrival at the landing waypoint, the NEST again switched to position hold and awaited the landing confirmation signal before continuing to the home position. This verified the correct sequencing and synchronization between simulated UAV actions and the NEST's decision logic. During the mission execution, the positional data of the NEST was retrieved in order to enable a post-analysis of the position hold mode. In figure 4.22 there is a representation of the position evolution over-time of the NEST. In comparison the figure 4.23 has a evolution graphic of the latitude and longitude position of the NEST. It's possible to verify in the graph the two waypoints location where the NEST was holding position-starting around 1.3 minutes and 2.5 minutes. Through the data presented is possible to verify that the gps position variation is very low around approximately 40 centimeters, what is a promising result.



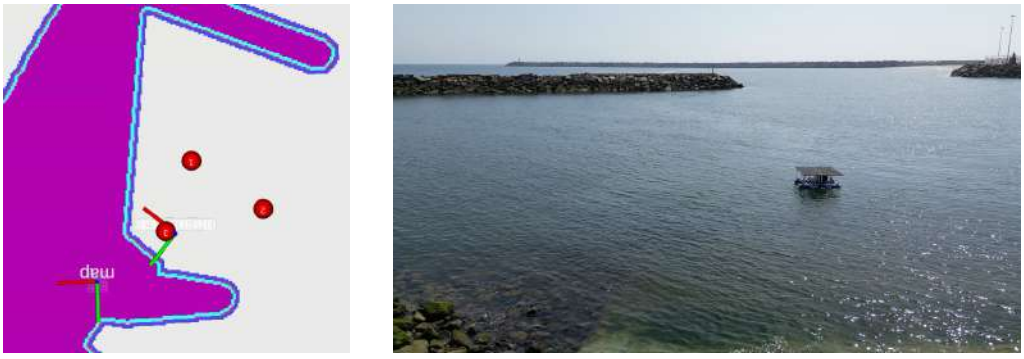
(a) Initial position: *RViz* visualization (left) and real footage (right).



(b) Take-off waypoint reached: *RViz* (left) and real footage (right).



(c) Landing waypoint reached: *RViz* (left) and real footage (right).



(d) Final home position: *RViz* (left) and real footage (right).

Figure 4.21: Mission execution overview: comparison between *RViz* simulation and real-world footage.

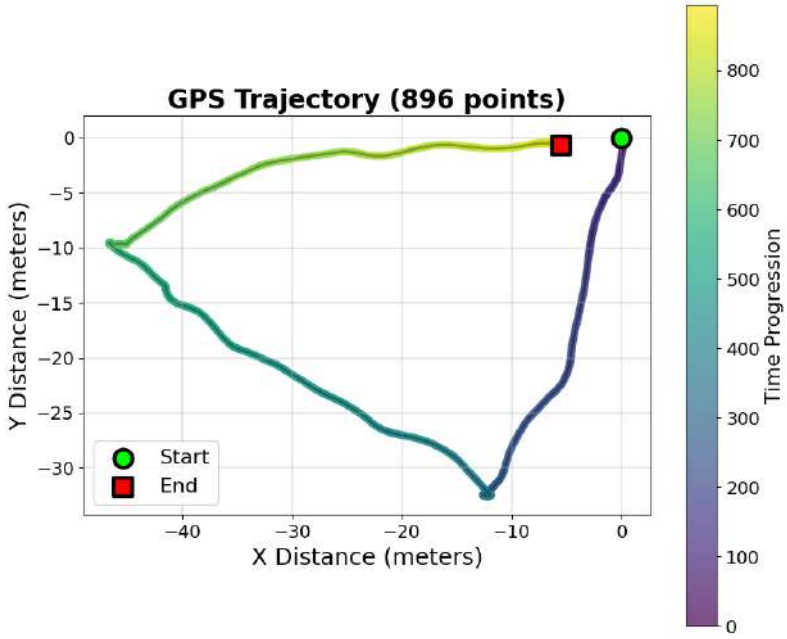


Figure 4.22: Evolution of the NEST position over time in GPS coordinates.

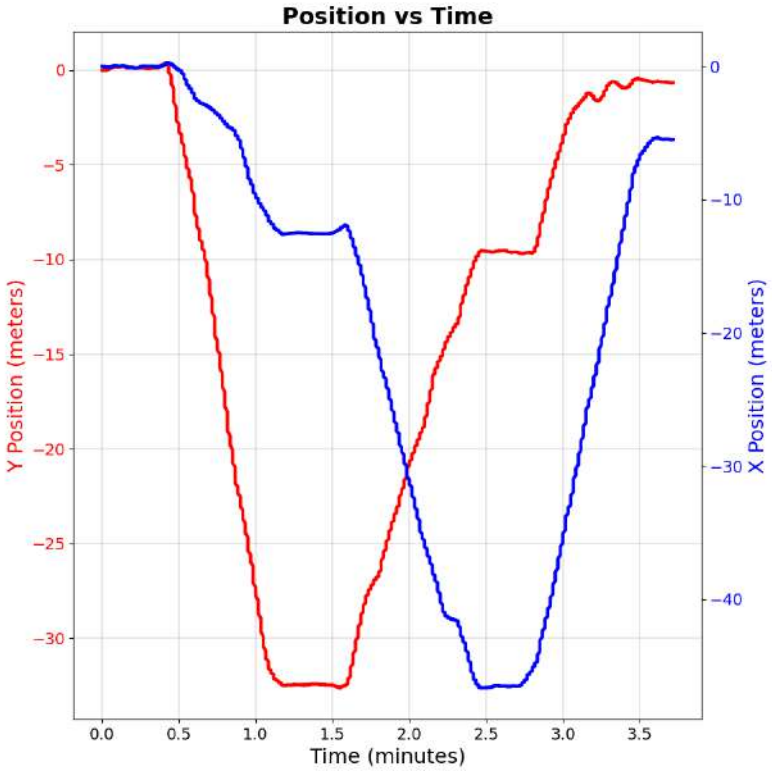


Figure 4.23: Latitude and longitude variation of the NEST over time, the red lines is latitude and the blue line is longitude.

4.4 Discussion

The proposed cooperative USV-UAV mission planning system was successfully validated through both simulated and near-real tests. Simulation results confirmed the correct operation of core components, including navigation, map acquisition, mission sequencing, and emergency handling. The cooperative strategy significantly improved efficiency by reducing UAV transit time and maximizing inspection duration, which is particularly advantageous for remote or offshore mission sites. Field testing at the ATLANTIS Test Center further demonstrated the system's viability under real-world conditions. Communication links, mission logic, and control mechanisms performed reliably, and the USV's position-hold capability provided a stable platform for UAV takeoff and landing. The full mission pipeline operated effectively with real geospatial data and environmental constraints. Overall, the results confirm that the proposed cooperative system overcomes the endurance limitations of standalone UAV operations. By leveraging the complementary roles of the USV and UAV, the architecture extends mission range and efficiency, offering a practical solution for autonomous maritime inspection tasks.

Chapter 5

Conclusion

This thesis presented a spatio-temporal mission planning methodology enabling autonomous cooperation between a USV and UAV for the inspection of floating photovoltaic (PV) plants. By addressing the limitations of UAV endurance in offshore environments, the proposed system leverages a cooperative framework where the USV acts as a mobile launch, recovery, and communication platform, effectively extending the UAV's operational range and maximizing inspection efficiency.

A mission planner was developed to autonomously coordinate both vehicles using a modular architecture that manages takeoff, navigation, inspection, and landing phases. The system integrates automatic map generation from geospatial data, waypoint optimization, and synchronized multi-vehicle task execution. Navigation is handled through a hybrid path planning strategy, combining global path generation with local obstacle avoidance, enabling robust operation in dynamic maritime environments. The cooperative approach achieved 97.89% inspection efficiency, reducing UAV flight time by 32% compared to a solo UAV mission. In a representative 42-minute mission, the UAV completed 2,367 meters of inspection flight using only 17 minutes of airborne time, remaining within battery limits that would otherwise be exceeded in non-cooperative scenarios. To ensure safety, an emergency retrieval system was developed, featuring a multi-criteria optimization algorithm that, in the tested scenario, guided the UAV to land on the USV within 75 seconds of the emergency trigger. After landing, the USV immediately navigated back to the home position, completing the recovery autonomously. The optimization balances coordination time, proximity to home, and environmental safety constraints to guarantee a safe and efficient response.

The system was validated through both simulation and real-world testing at the ATLANTIS Test Center, confirming its robustness and reliability in operational maritime conditions. The proposed framework contributes to the development of scalable and efficient autonomous inspection systems for floating PV infrastructure, with potential applications across a wide range of offshore monitoring scenarios.

5.1 Future Work

The next steps of this research include several key aspects. First, the local path planner algorithm must be validated in real-case scenarios to assess its reliability in avoiding both static and dynamic obstacles. Second, the mission planning algorithm should be tested in a real-world setting with full integration of a UAV. Third, the USV's capabilities should be expanded to include surface inspections during the UAV's survey. Additionally, future work should incorporate energy-aware planning to optimize mission execution based on the battery levels of both the UAV and USV. Finally, the integration of fault detection and recovery mechanisms is necessary to ensure system resilience in the event of hardware or communication failures.

Bibliography

[1] C Schleussner *et al.* “Climate impacts in portugal”. In: *Clim Analytics* (2020).

[2] S. Rodrigues *et al.* “Suitability analysis of solar photovoltaic farms: A Portuguese case study”. In: *International Journal of Renewable Energy Research* 7 (Jan. 2017), pp. 244–254.

[3] Luana Carolina Alves da Costa and Gardenio Diogo Pimentel da Silva. “Save water and energy: A techno-economic analysis of a floating solar photovoltaic system to power a water integration project in the Brazilian semiarid”. In: *International Journal of Energy Research* 45.12 (2021), pp. 17924–17941. DOI: https://doi.org/10.1002/er.6932.

[4] Yara Waheeb Youssef and Anna Khodzinskaya. “A review of evaporation reduction methods from water surfaces”. In: *E3S web of conferences*. Vol. 97. EDP Sciences. 2019, p. 05044. DOI: https://doi.org/10.1051/e3sconf/20199705044.

[5] Gianfranco Di Lorenzo *et al.* “Review of O&M Practices in PV Plants: Failures, Solutions, Remote Control, and Monitoring Tools”. In: *IEEE Journal of Photovoltaics* 10.4 (2020), pp. 914–926. DOI: 10.1109/JPHOTOV.2020.2994531.

[6] International Electrotechnical Commission. *IEC TS 62446-3: Photovoltaic (PV) systems – Requirements for testing, documentation and maintenance – Part 3: Photovoltaic modules and plants – Outdoor infrared thermography of photovoltaic modules and plants*. Standard. Accessed: 2024-08-05. International Electrotechnical Commission, 2017.

[7] Khaled Osmani *et al.* “A review on maintenance strategies for PV systems”. In: *Science of The Total Environment* 746 (2020), p. 141753. ISSN: 0048-9697. DOI: https://doi.org/10.1016/j.scitotenv.2020.141753. URL: https://www.sciencedirect.com/science/article/pii/S0048969720352827.

[8] Abdulrahman Khamaj Abhijit Sen Akshay Shirish Mohankar and Sougata Karmakar. “Emerging OSH Issues in Installation and Maintenance of Floating Solar Photovoltaic Projects and Their Link with Sustainable Development Goals”. In: *Risk Management and Healthcare Policy* 14 (2021), pp. 1939–1957. DOI: 10.2147/RMHP.S304732.

[9] Paolo Bellezza Quater *et al.* “Light Unmanned Aerial Vehicles (UAVs) for Cooperative Inspection of PV Plants”. In: *IEEE Journal of Photovoltaics* 4.4 (2014), pp. 1107–1113. DOI: 10.1109/JPHOTOV.2014.2323714.

[10] Dong Ho Lee and Jong Hwa Park. "Developing Inspection Methodology of Solar Energy Plants by Thermal Infrared Sensor on Board Unmanned Aerial Vehicles". In: *Energies* 12.15 (2019). ISSN: 1996-1073. DOI: 10.3390/en12152928. URL: https://www.mdpi.com/1996-1073/12/15/2928.

[11] Kuo-Chien Liao and Jau-Huai Lu. "Using UAV to Detect Solar Module Fault Conditions of a Solar Power Farm with IR and Visual Image Analysis". In: *Applied Sciences* 11.4 (2021). ISSN: 2076-3417. DOI: 10.3390/app11041835. URL: https://www.mdpi.com/2076-3417/11/4/1835.

[12] Silvia Bossi *et al.* "Floating Photovoltaic Plant Monitoring: A Review of Requirements and Feasible Technologies". In: *Sustainability* 16.19 (2024). ISSN: 2071-1050. DOI: 10.3390/su16198367. URL: https://www.mdpi.com/2071-1050/16/19/8367.

[13] Juan Liu *et al.* "UAV-USV Cooperative Task Allocation for Smart Ocean Networks". In: *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. 2021, pp. 1815–1820. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00268.

[14] H. Walker *et al.* "Model of Operation-and-Maintenance Costs for Photovoltaic Systems". In: (June 2020). DOI: 10.2172/1659995. URL: https://www.osti.gov/biblio/1659995.

[15] Mohammed Aissi *et al.* "Autonomous solar USV with an automated launch and recovery system for UAV: State of the art and Design". In: *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*. 2020, pp. 1–6. DOI: 10.1109/ICECOCS50124.2020.9314415.

[16] Zhi Yan *et al.* "A Survey and Analysis of Multi-Robot Coordination". In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399. DOI: 10.5772/57313. eprint: https://doi.org/10.5772/57313. URL: https://doi.org/10.5772/57313.

[17] Hamido Hourani *et al.* "A Marsupial Relationship in Robotics: A Survey". In: *Intelligent Robotics and Applications*. Ed. by Sabina Jeschke *et al.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 335–345. ISBN: 978-3-642-25486-4.

[18] Eduardo Pinto *et al.* "An autonomous surface-aerial marsupial robotic team for riverine environmental monitoring: Benefiting from coordinated aerial, underwater, and surface level perception". In: *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. 2014, pp. 443–450. DOI: 10.1109/ROBIO.2014.7090371.

[19] Michail Kalaitzakis *et al.* "A marsupial robotic system for surveying and inspection of freshwater ecosystems". In: *Journal of Field Robotics* 38.1 (2021), pp. 121–138. DOI: https://doi.org/10.1002/rob.21957. eprint: https://onlinelibrary.

wiley.com/doi/pdf/10.1002/rob.21957. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21957.

[20] FF Thompson. "A mission planner to improve the cost-effectiveness of autonomous marine vehicle deployments". In: (Jan. 2019). DOI: 10.25959/100.00031960. URL: https://figshare.utas.edu.au/articles/thesis/A_mission_planner_to_improve_the_cost-effectiveness_of_autonomous_marine_vehicle_deployments/23251736.

[21] Karthik Karur *et al.* "A Survey of Path Planning Algorithms for Mobile Robots". In: *Vehicles* 3.3 (2021), pp. 448–468. ISSN: 2624-8921. DOI: 10.3390/vehicles3030027. URL: https://www.mdpi.com/2624-8921/3/3/27.

[22] Chunxi Cheng *et al.* "Path planning and obstacle avoidance for AUV: A review". In: *Ocean Engineering* 235 (2021), p. 109355. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.109355. URL: https://www.sciencedirect.com/science/article/pii/S002980182100771X.

[23] Liang Zhang *et al.* "Path Planning for Autonomous Ships: A Hybrid Approach Based on Improved APF and Modified VO Methods". In: *Journal of Marine Science and Engineering* 9.7 (2021). ISSN: 2077-1312. DOI: 10.3390/jmse9070761. URL: https://www.mdpi.com/2077-1312/9/7/761.

[24] Han-ye Zhang *et al.* "Path Planning for the Mobile Robot: A Review". In: *Symmetry* 10.10 (2018). ISSN: 2073-8994. DOI: 10.3390/sym10100450. URL: https://www.mdpi.com/2073-8994/10/10/450.

[25] Chad Goerzen *et al.* "A survey of motion planning algorithms from the perspective of autonomous UAV guidance". In: *Journal of Intelligent and Robotic Systems* 57 (2010), pp. 65–100. DOI: https://doi.org/10.1007/s10846-009-9383-1.

[26] Sanjoy Kumar Debnath *et al.* "Different Cell Decomposition Path Planning Methods for Unmanned Air Vehicles-A Review". In: *Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019*. Ed. by Zainah Md Zain *et al.* Singapore: Springer Nature Singapore, 2021, pp. 99–111. ISBN: 978-981-15-5281-6. DOI: 10.1007/978-981-15-5281-6_8.

[27] Alejandro Hidalgo-Paniagua *et al.* "MOSFLA-MRPP: Multi-Objective Shuffled Frog-Leaping Algorithm applied to Mobile Robot Path Planning". In: *Engineering Applications of Artificial Intelligence* 44 (2015), pp. 123–136. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2015.05.011. URL: https://www.sciencedirect.com/science/article/pii/S0952197615001220.

[28] Prases K Mohanty and Harshal S Dewang. "A smart path planner for wheeled mobile robots using adaptive particle swarm optimization". In: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 43.2 (2021), p. 101. DOI: https://doi.org/10.1007/s40430-021-02827-7.

[29] X. Wang *et al.* "A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning". In: (2015). DOI: https://doi.org/10.15837/ijccc.2015.5.2030. URL: http://hdl.handle.net/10454/17606.

[30] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390. URL: https://doi.org/10.1007/BF01386390.

[31] Min Luo *et al.* "Surface Optimal Path Planning Using an Extended Dijkstra Algorithm". In: *IEEE Access* 8 (2020), pp. 147827–147838. DOI: 10.1109/ACCESS.2020.3015976.

[32] Huijuan Wang *et al.* "Application of Dijkstra algorithm in robot path-planning". In: *2011 Second International Conference on Mechanic Automation and Control Engineering*. 2011, pp. 1067–1069. DOI: 10.1109/MACE.2011.5987118.

[33] Peter E. Hart *et al.* "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[34] C.W. Warren. "Fast path planning using modified A* method". In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. 1993, 662–667 vol.2. DOI: 10.1109/ROBOT.1993.291883.

[35] Shang Erke *et al.* "An improved A-Star based path planning algorithm for autonomous land vehicles". In: *International Journal of Advanced Robotic Systems* 17.5 (2020), p. 1729881420962263. DOI: 10.1177/1729881420962263. URL: https://doi.org/10.1177/1729881420962263.

[36] Gang Tang *et al.* "Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment". In: *IEEE Access* 9 (2021), pp. 59196–59210. DOI: 10.1109/ACCESS.2021.3070054.

[37] A. Stentz. "Optimal and efficient path planning for partially-known environments". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. 1994, 3310–3317 vol.4. DOI: 10.1109/ROBOT.1994.351061.

[38] S. Lavalle. "Rapidly-exploring random trees : a new tool for path planning". In: *Research Report 9811* (1998). URL: https://cir.nii.ac.jp/crid/1573950399665672960.

[39] Yi Gan *et al.* "Research on robot motion planning based on RRT algorithm with nonholonomic constraints". In: *Neural Processing Letters* 53 (2021), pp. 3011–3029. DOI: https://doi.org/10.1007/s11063-021-10536-4.

[40] Wu Xiaojing *et al.* "Dynamicstep BI-RRT UAV path planning algorithm". In: *Journal of Hebei University of Science and Technology* 40.5 (2019), pp. 414–422.

[41] Liangjun Zhang and Dinesh Manocha. "An efficient retraction-based RRT planner". In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 3743–3750. DOI: 10.1109/ROBOT.2008.4543785.

[42] Wei Wang *et al.* "A learning-based multi-RRT approach for robot path planning in narrow passages". In: *Journal of Intelligent & Robotic Systems* 90 (2018), pp. 81–100. DOI: https://doi.org/10.1007/s10846-017-0641-3.

[43] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[44] Y. Volkan Pehlivanoglu and Perihan Pehlivanoglu. "An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems". In: *Applied Soft Computing* 112 (2021), p. 107796. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2021.107796. URL: https://www.sciencedirect.com/science/article/pii/S1568494621007171.

[45] M. Dorigo. "Optimization, Learning and Natural Algorithms". In: *Ph.D. Thesis, Politecnico di Milano* (1992). URL: https://cir.nii.ac.jp/crid/1573950400977139328.

[46] Marco Dorigo *et al.* "Ant colony optimization". In: *IEEE Computational Intelligence Magazine* 1.4 (2006), pp. 28–39. DOI: 10.1109/MCI.2006.329691.

[47] Dimitrios V. Lyridis. "An improved ant colony optimization algorithm for unmanned surface vehicle local path planning with multi-modality constraints". In: *Ocean Engineering* 241 (2021), p. 109890. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.109890. URL: https://www.sciencedirect.com/science/article/pii/S0029801821012385.

[48] Renato Jorge Silva *et al.* "Multi-Agent Optimization for Offshore Wind Farm Inspection using an Improved Population-based Metaheuristic". In: *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2020, pp. 53–60. DOI: 10.1109/ICARSC49921.2020.9096107.

[49] Oussama Khatib. "The Potential Field Approach And Operational Space Formulation In Robot Control". In: *Adaptive and Learning Systems: Theory and Applications*. Ed. by Kumpati S. Narendra. Boston, MA: Springer US, 1986, pp. 367–377. ISBN: 978-1-4757-1895-9. DOI: 10.1007/978-1-4757-1895-9_26. URL: https://doi.org/10.1007/978-1-4757-1895-9_26.

[50] Giuseppe Fedele *et al.* "Obstacles Avoidance Based on Switching Potential Functions". In: *Journal of Intelligent & Robotic Systems* 90 (June 2018). DOI: 10.1007/s10846-017-0687-2.

[51] Yijian Duan *et al.* "Active obstacle avoidance method of autonomous vehicle based on improved artificial potential field". In: *International Journal of Advanced Robotic Systems* 19.4 (2022), p. 17298806221115984. DOI: 10.1177/17298806221115984. eprint: https://doi.org/10.1177/17298806221115984. URL: https://doi.org/10.1177/17298806221115984.

[52] Zhengtian Wu *et al.* "Robot path planning based on artificial potential field with deterministic annealing". In: *ISA Transactions* 138 (2023), pp. 74–87. ISSN: 0019-0578. DOI: https://doi.org/10.1016/j.isatra.2023.02.018. URL: https://www.sciencedirect.com/science/article/pii/S0019057823000769.

[53] D. Fox *et al.* "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33. DOI: 10.1109/100.580977.

[54] Eduardo J. Molinos *et al.* "Dynamic window based approaches for avoiding obstacles in moving". In: *Robotics and Autonomous Systems* 118 (2019), pp. 112–130. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2019.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0921889018309746.

[55] Christoph Rösmann *et al.* "Integrated online trajectory planning and optimization in distinctive topologies". In: *Robotics and Autonomous Systems* 88 (2017), pp. 142–153. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2016.11.007. URL: https://www.sciencedirect.com/science/article/pii/S0921889016300495.

[56] Jiafeng Wu *et al.* "An Improved Timed Elastic Band (TEB) Algorithm of Autonomous Ground Vehicle (AGV) in Complex Environment". In: *Sensors* 21.24 (2021). ISSN: 1424-8220. DOI: 10.3390/s21248312. URL: https://www.mdpi.com/1424-8220/21/24/8312.

[57] Paolo Fiorini and Zvi Shiller. "Motion Planning in Dynamic Environments Using Velocity Obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772. DOI: 10.1177/027836499801700706. eprint: https://doi.org/10.1177/027836499801700706. URL: https://doi.org/10.1177/027836499801700706.

[58] Daniel Filipe Campos *et al.* "An Adaptive Velocity Obstacle Avoidance Algorithm for Autonomous Surface Vehicles". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 8089–8096. DOI: 10.1109/IROS40897.2019.8968156.

[59] Zheng Chen *et al.* "A Hybrid Path Planning Algorithm for Unmanned Surface Vehicles in Complex Environment With Dynamic Obstacles". In: *IEEE Access* 7 (2019), pp. 126439–126449. DOI: 10.1109/ACCESS.2019.2936689.

[60] Hamza Chakraa *et al.* "Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art". In: *Robotics and Autonomous Systems* 168 (2023), p. 104492. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104492. URL: https://www.sciencedirect.com/science/article/pii/S0921889023001318.

[61] Brian P. Gerkey and Maja J. Matarić. "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems". In: *The International Journal of Robotics Research* 23.9 (2004), pp. 939–954. DOI: 10.1177/0278364904045564. eprint: https://doi.

org/10.1177/0278364904045564. URL: https://doi.org/10.1177/0278364904045564.

[62] Ernesto Nunes *et al.* "A taxonomy for task allocation problems with temporal and ordering constraints". In: *Robotics and Autonomous Systems* 90 (2017). Special Issue on New Research Frontiers for Intelligent Autonomous Systems, pp. 55–70. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2016.10.008. URL: https://www.sciencedirect.com/science/article/pii/S0921889016306157.

[63] Esther Bischoff *et al.* "Multi-Robot Task Allocation and Scheduling Considering Cooperative Tasks and Precedence Constraints". In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 3949–3956. DOI: 10.1109/SMC42975.2020.9283215.

[64] Jittat Fakcharoenphol *et al.* "The k-traveling repairmen problem". In: *ACM Trans. Algorithms* 3.4 (2007), 40–es. ISSN: 1549-6325. DOI: 10.1145/1290672.1290677. URL: https://doi.org/10.1145/1290672.1290677.

[65] Vandilberto P Pinto *et al.* "Mission planning for multiple UAVs in a wind field with flight time constraints". In: *Journal of Control, Automation and Electrical Systems* 31.4 (2020), pp. 959–969. DOI: https://doi.org/10.1007/s40313-020-00609-5.

[66] Lourenço Sousa Pinho *et al.* "Anomaly Detection for PV Modules Using Multi-Modal Data Fusion in Aerial Inspections". In: *IEEE Access* 13 (2025), pp. 88762–88779. DOI: 10.1109/ACCESS.2025.3570815.

[67] Daniel F. Campos *et al.* "Nautilus: An autonomous surface vehicle with a multilayer software architecture for offshore inspection". In: *Journal of Field Robotics* 41.4 (2024), pp. 966–990. DOI: https://doi.org/10.1002/rob.22304. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.22304. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22304.

[68] Thomas Moore and Daniel Stouch. "A Generalized Extended Kalman Filter Implementation for the Robot Operating System". In: *Intelligent Autonomous Systems 13*. Ed. by Emanuele Menegatti *et al.* Cham: Springer International Publishing, 2016, pp. 335–348. ISBN: 978-3-319-08338-4.

[69] John Parr Snyder. *Map projections used by the US Geological Survey*. Tech. rep. US Government Printing Office, 1982. DOI: https://doi.org/10.3133/b1532.

[70] Adel Al-Jumaily and Cindy Leung. "Wavefront Propagation and Fuzzy Based Autonomous Navigation". In: *International Journal of Advanced Robotic Systems* 2.2 (2005), p. 10. DOI: 10.5772/5799. eprint: https://doi.org/10.5772/5799. URL: https://doi.org/10.5772/5799.

[71] Rafael Marques Claro *et al.* "ArTuga: A novel multimodal fiducial marker for aerial robotics". In: *Robotics and Autonomous Systems* 163 (2023), p. 104398. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104398. URL: https://www.sciencedirect.com/science/article/pii/S0921889023000374.

[72] Richard H. Byrd *et al.* "A Limited Memory Algorithm for Bound Constrained Optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. DOI: 10.1137/0916069. eprint: https://doi.org/10.1137/0916069. URL: https://doi.org/10.1137/0916069.

[73] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[74] Andry Maykol Pinto *et al.* "ATLANTIS Coastal Testbed: A near-real playground for the testing and validation of robotics for O&M". In: *OCEANS 2023 - Limerick*. 2023, pp. 1–5. DOI: 10.1109/OCEANSLimerick52467.2023.10244595.