NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

JOÃO AFONSO MENDONÇA JORGE

BSc in Computer Science

MULTIPLE AUTONOMOUS DRONE MISSION PLANNING

Dissertation Plan
MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

MULTIPLE AUTONOMOUS DRONE MISSION PLANNING

JOÃO AFONSO MENDONÇA JORGE

BSc in Computer Science

Adviser: João Cintra
Section Head Homeland Security and Defense in Portugal, GMV Portugal

Co-adviser: Jorge Cruz
Assistant Professor, NOVA University Lisbon

Dissertation Plan
MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2024

Abstract

The goal of this thesis is to develop an application capable of planning missions for the increasing number of drones, minimizing the need for human interaction in mission planning. Unmanned X Vehicle meaning UAV, UUV, USV or UGV (UXV)s are becoming increasingly prevalent in both military and civilian applications worldwide, offering significant advantages over manned vehicles, such as reduced maintenance requirements, enhanced safety, and freedom from human limitations. The widespread adoption and ongoing investments in UXV technologies have made them more cost-effective, allowing users to deploy not just a few but dozens of drones.

Traditionally, each drone is operated by an individual, but the growing number of drones presents challenges for efficient management, often requiring large teams. Moreover, the diverse range of UXVs—including aerial Unmanned Aerial Vehicle (UAV)s, underwater Unmanned Underwater Vehicle (UUV)s and surface Unmanned Surface Vehicle (USV)s — complicates the coordination and control of these assets on a larger scale.

This research aims to address these challenges by developing an application that focuses on optimizing the path planning of multiple drones. The system will incorporate an advanced algorithm to plan efficient and safe mission paths with minimal human input, thereby improving the scalability and autonomy of drone operations. This algorithm is designed to take a mission description, a selection of drones, their respective payloads, and the mission area, and perform two key functions. Firstly, it will choose the most suitable drones and payloads for the given mission. Secondly, it will devise an optimal plan to execute the mission efficiently, considering the unique capabilities of each selected drone.

By creating a sophisticated algorithm that combines intelligent drone selection and mission planning, this research endeavors to streamline and enhance the operational effectiveness of unmanned vehicle deployments across various scenarios.

Keywords: Unmanned Vehicles, Flight Mission Planning, Resource Allocation, Multi-UXV

Resumo

O objetivo desta tese é desenvolver uma aplicação capaz de planear missões para um número crescente de drones, minimizando a necessidade de interação humana no seu planeamento. Veículo Não Tripulado (VNT)s estão a tornar-se cada vez mais prevalentes em aplicações militares e civis em todo o mundo, oferecendo vantagens significativas sobre veículos tripulados, como manutenção reduzida, aumento da segurança e minimização do risco humano. A adoção generalizada e os investimentos contínuos em tecnologias de VNTs tornaram-nos cada vez mais acessíveis, permitindo que os usuários utilizar não apenas alguns, mas dezenas de drones.

Tradicionalmente, cada drone é operado por um indivíduo, mas o número crescente de drones apresenta desafios para a gestão eficiente de equipas, exigindo equipas cada vez maiores. Além disso, a diversidade de VNTs—incluindo Veículo Aéreo Não Tripulado (VANT)s, Veículo Subaquático Não Tripulado (VSNT)s e Veículo Superfície Aquática Não Tripulado (VSANT)s — complica ainda mais a coordenação e o controle desses ativos em larga escala.

Esta pesquisa tem como objetivo abordar estes desafios desenvolvendo uma aplicação focada na otimização do planeamento de trajetórias de múltiplos drones. O sistema incorporará um algoritmo de planeamento avançado de trajetórias de missão eficientes e seguras com mínima intervenção humana, melhorando assim a escalabilidade e a autonomia das operações de drones. Este algoritmo é projetado para receber a descrição da missão, uma seleção de drones, suas respectivas cargas e a área da missão, e realizar duas funções principais. Primeiro, deverá escolher os drones e cargas mais adequados para a missão. Em segundo lugar, elaborar um plano otimizado para executar a missão de forma eficiente, considerando as capacidades únicas de cada drone selecionado.

Ao criar um algoritmo sofisticado que combina a seleção inteligente de drones e o planeamento de missões, esta pesquisa busca otimizar e aprimorar a eficácia operacional de veículos não tripulados em diversos cenários.

Palavras-chave: Veículos Não Tripulados, Planeamento de Missão, Alocação de Recursos, Multi-VNT

Contents

List of Figures

List of Tables

Acronyms

CBRN Chemical, Biological, Radiological, and Nuclear *(pp. 10, 21)*

FOV Field of View *(pp. 30, 64)*

QGIS Quantum Geographic Information System *(p. 59)*

SAR Search and Rescue *(pp. 20, 21)*

UAV Unmanned Aerial Vehicle *(pp. i, vi, 1, 6–8, 18, 20, 21, 25, 28, 30, 34, 35, 43–45, 49, 75, 80, 89, 91–93, 96)*

UGV Unmanned Ground Vehicle *(p. 6)*

UI User Interface *(pp. 36, 42)*

USV Unmanned Surface Vehicle *(pp. i, vi, 6, 8, 9, 13, 20, 21, 28, 35, 45, 59, 75, 89, 91)*

UUV Unmanned Underwater Vehicle *(pp. i, vi, 6, 10, 11, 13, 20, 21, 26, 28, 35, 45, 49, 59, 75, 89, 91, 92)*

UXV Unmanned X Vehicle meaning UAV, UUV, USV or UGV *(pp. i, 1–8, 17, 20, 26, 32, 42)*

VANT Veículo Aéreo Não Tripulado *(p. ii)*

VNT Veículo Não Tripulado *(p. ii)*

VSANT Veículo Superfície Aquática Não Tripulado *(p. ii)*

VSNT Veículo Subaquático Não Tripulado *(p. ii)*

1 Introduction

1.1 Context and Motivation

In recent years, the landscape of Unmanned Vehicles (UXVs) has undergone a transformative evolution, exponentially expanding their utility and accessibility across various domains. These autonomous systems have become integral components of modern technological ecosystems, with capabilities extending to diverse environments such as air, ground, surface, and underwater. Equipped with an array of sophisticated payloads including high-resolution cameras, advanced sensors, and specialized tools, UXVs are tailored to undertake a multitude of missions with precision and efficiency.

The proliferation of UXVs in both military and civilian contexts is a testament to their unparalleled advantages. These vehicles operate tirelessly, devoid of the need for rest or sustenance, and their remote controllability mitigates potential risks to human operators. As a result, Unmanned Vehicles have become indispensable assets for missions ranging from surveillance, reconnaissance, and intelligence gathering to responding swiftly to emergency situations and engaging in complex warfare scenarios.

This technological progress has witnessed remarkable breakthroughs, transforming concepts like drones delivering packages and autonomously identifying and harvesting crops from mere dreams into reality. Moreover, these advancements have democratized access to the technology, bringing down costs to the point where many individuals can now purchase Unmanned Aerial Vehicles (UAVs) for recreational use.

The symbiosis between the increasing capability and availability of UXVs has ushered in a new era of technological possibilities. However, it has also presented novel challenges, particularly in the realm of mission planning and coordination. As military forces and organizations worldwide embrace the advantages of UXVs, the sheer volume of these unmanned vehicles within fleets has created a logistical conundrum. What was once manageable by small, agile teams has now transformed into a formidable challenge, requiring not only the efficient manning of these drones but also the strategic planning of missions to maximize the effectiveness of each unit.

1.2 Problem

In light of these developments, there arises a pressing need for innovative solutions that can address the complexities inherent in planning and deploying fleets of UXVs for a specified mission. This research endeavors to tackle these challenges head-on by developing an advanced algorithmic solution that not only considers the unique characteristics of each UXV but also factors in the nuances of diverse mission types. Its objective is to facilitate the selection of drones best suited for each mission and determine their optimal paths, ensuring the utmost utilization of available resources.

To accomplish this goal, it is crucial to thoroughly comprehend each mission, considering its objectives, potential constraints, and inherent complexities. An essential aspect involves identifying the criteria that define a mission plan as either effective or ineffective for each specific task. Equally important is gaining a profound understanding of each drone, encompassing their capabilities for various missions and discerning the distinctive factors that set them apart. This understanding will enable us to make informed comparisons and select the optimal resources for each mission's success.

Recognizing the complexity of these requirements, each deserving a dedicated thesis for exploration in its entirety, the strategy is to concentrate on developing an algorithm capable of managing missions and drones modeled with the minimal necessary constraints. Subsequently, complexity can incrementally introduced as the project progresses.

Given the limited time frame for this thesis, the primary focus will be on developing a proof of concept. This approach will involve constructing an architecture that lays the groundwork for adding supplementary behaviors, drones, or additional complexities such as weather conditions. By adopting this strategy, the aim is to provide a robust framework that facilitates ongoing improvements and adaptations by subsequent researchers.

1.3 Objectives

With this objective in mind, the mission is to devise a comprehensive solution capable of navigating through an array of Unmanned Vehicles, discerning the mission type, and evaluating the mission area. The aim is to intelligently select the most suitable drones tailored for the mission's specific requirements. Subsequently, the solution seeks to orchestrate precise paths for each selected drone, ensuring a synchronized performance wherein each UXV fulfills its unique role in the overall mission.

To break down this overarching goal, the mission can be delineated into two primary objectives. Firstly, to develop the capability of given a mission type and an array of drones select the most fitting drone or combination of drones for the specific mission parameters. Once the optimal drones are identified, the second objective entails the determination of the individual paths each chosen drone will traverse, ensuring an orchestrated and efficient execution of the mission at hand.

A pivotal aspect of the mission is the development of a flexible framework that facilitates the seamless addition of new drones and missions to be supported by the algorithm. This forward-thinking approach ensures the adaptability and longevity of the algorithm, making it future-proof. By incorporating such flexibility, this algorithm is designed to meet current needs while remaining robust and readily adaptable to the evolving landscape of UXV technologies and mission requirements.

It is crucial to grasp the intricacy of this issue, recognizing that the mission planning process significantly diverges depending on the mission type. This thesis will only center on missions involving area surveying/scanning, acknowledging the necessity for simplifications within the confines of a constrained timeframe.

The objective is to develop a mission planning algorithm adept at orchestrating survey/scanning missions, proficient in generating paths for the three drone environments utilized by the Portuguese Navy: air, surface, and underwater. Subsequently, the algorithm will undergo comprehensive testing within a simulator to ensure its alignment with the intended functionality.

1.4 Solution

In this solution, an effort is made to represent each Unmanned Vehicle through a set of variables. These variables serve a dual purpose: facilitating comparison between different UXVs and providing essential parameters for mission planning algorithms. The parameters encompass factors such as maximum speed, maximum height, cost of operation, endurance, etc.

The variables selected for each drone are meticulously chosen to capture nuanced differences and similarities. These variables not only facilitate comparison but also contribute to the optimization of mission planning parameters. For instance, understanding how autonomy considerations vary based on mission duration aids in tailoring the solution to the diverse operational requirements of the Navy.

For resource allocation, the approach involves representing each mission as a set of customizable parameters. Users have the flexibility to fine-tune mission characteristics, such as mission duration and mission cost with the aim to optimize mission planning and match specific mission requirements with the most suitable drone. For instance, longer-duration missions may prioritize autonomy over speed of a UXV. This flexible framework allows us to specify mission requirements and optimize drone selection accordingly.

In the pursuit of accurate performance comparison, the simulation methodology goes beyond generic metrics. Delving into the specifics of each mission scenario, considering not only the feasibility of UXV deployment but also the optimization of mission objectives. By factoring in variables like mission duration and mission cost, the comparative analysis provides a holistic understanding of each UXV's efficacy in diverse operational contexts.

Considering the mission planning aspect, the future goal is to market this product to the Portuguese Navy, focusing primarily on their mission profiles. Due to the diversity

and complexity of naval missions, the focus of this theses will be on simpler missions such as surveillance, intelligence, reconnaissance, and search and rescue—each falling under the category of area scanning.

Recognizing the complexity of naval missions, this algorithm aims to strike a balance between comprehensiveness and manageability. Although it's impractical to support every conceivable mission scenario, the modular framework allows for seamless expansion. Concentrating on foundational behaviors that constitute mission profiles will pave the way for adaptability and future mission integrations.

Looking forward, as the intricacies of UXV integration with naval operations are navigated, a commitment to modular design remains steadfast. A future is envisioned where the addition of new missions seamlessly integrates with the existing framework, minimizing development effort and maximizing adaptability. This Lego-like approach not only streamlines the implementation of chosen missions but also lays the foundation for a scalable solution that can evolve in tandem with the ever-changing landscape of naval requirements.

1.5 Contributions

While extensive research exists on mission and path generation for single UXVs or even heterogeneous UXVs of the same environment, there is a noticeable gap in the literature concerning the planning of missions that involve a combination of different UXVs operating in diverse environments. This work seeks to address this gap and propel research forward in this domain.

The primary contribution of this thesis is the development of a sophisticated system designed to select the most appropriate UXVs from a diverse lineup for a specific mission. Moreover, the system is engineered to plan the optimal path for each drone, with the overarching goal of maximizing overall mission performance. This holistic approach considers both the unique characteristics of each UXV and the varied environmental conditions in which they operate.

To validate the efficacy of the solution, testing will be conducted using a simulator. This simulator provides a platform to visualize the planned missions. Through these simulations, the aim is to rigorously evaluate the performance of the system in lifelike scenarios, ensuring its effectiveness and adaptability to diverse mission requirements.

In summary, the contributions of this work can be outlined as follows:

- Addressing the research gap in mission planning for combinations of different UXVs operating in varied environments.

- Developing a comprehensive system for selecting UXVs tailored to specific missions and optimizing their paths for enhanced overall performance.

- Demonstrating the practicality and robustness of the proposed solution through extensive testing in a realistic simulator environment.

1.6 Organization Overview

This thesis is organized into several chapters, each focusing on a specific aspect of the research and development process. The structure follows a logical progression, from reviewing existing technologies and related work to outlining the proposed solution, implementation details, and evaluation. Below is an outline of the content covered in each chapter, providing a roadmap for the reader to understand the flow of the thesis and its key contributions.

The second chapter provides a comprehensive overview of the State of the Art, detailing the technologies relevant to this thesis, including drones, their payloads, the missions they are designed to perform, and the most advanced scanning patterns currently in use.

The third chapter delves into the Related Work, offering a thorough examination of existing research and contributions in the field that directly relate to the theme of this thesis.

The fourth chapter outlines the architecture of the proposed solution to the problem at hand, including a detailed description of the algorithm's design, the application architecture, and the simulator's framework.

The fifth chapter presents the application that was developed to interface with the mission planning algorithm, highlighting its features, design choices, and its role in the overall system.

The sixth chapter describes in detail the steps taken and the various approaches explored during the implementation of the mission planning algorithm, providing a thorough explanation of the decision-making process and challenges encountered.

The seventh chapter offers a comparative analysis of all the implemented approaches from the previous chapter, discussing the strengths and weaknesses of each method and explaining the rationale behind selecting the final approach.

The eighth chapter focuses on the simulator developed to visualize the mission plans within a 3D representation of the environment, discussing its capabilities and the key insights gained from using it.

Finally, the ninth chapter summarizes the conclusions drawn from this thesis, outlines potential directions for future research, and highlights the contributions made to the broader field of UXV mission planning.

2

State of The Art

To fully comprehend this thesis, including the problem it addresses and the solutions it proposes, it is essential to establish a foundational knowledge baseline. This will ensure that readers, regardless of their area of expertise, can grasp the problem, the constraints involved, and the rationale behind the decisions made throughout the research.

The chapter begins with an overview of Unmanned Vehicles (UxVs), examining the various types that exist and their broad purposes, both globally and in the specific context of this study. This exploration will include a detailed explanation of the concept of payloads, clarifying the different types and possibilities that are integral to this thesis.

Next, the concept of a mission will be thoroughly defined, exploring the diverse range of missions and the unique challenges each presents within the scope of this research.

Finally, the focus will shift to the complexities of drone scanning patterns, providing a detailed overview of their significance and relevance to the algorithm developed in this thesis.

2.1 Technologies

Unmanned Vehicles (UXVs), as the name implies, are vehicles designed to operate without the need for a human pilot on board. They can be remotely controlled or, in some cases, operate autonomously. Since their inception, global military forces have exhibited significant interest in their development, drawn to the inherent advantages of enhanced safety by eliminating the risk to human life and the ability to operate continuously, 24/7, aligning seamlessly with the evolving landscape of technological advancements in the military sector.

Observant readers may have noted the 'X' in UXV, signifying the environment in which these vehicles operate. UXVs can be categorized into four main types: Aerial UAVs, Surface USVs, Underwater UUVs, and Ground Unmanned Ground Vehicle (UGV)s. This discussion will primarily focus on the first three, as the the target buyer for this mission planning algorithm is the Portuguese Navy, with a specific emphasis on maritime missions. Therefore, Aerial UAVs, Surface USVs, and Underwater UUVs will be explored in more

detail below.

To better understand and characterize these Unmanned Vehicles, a categorization based on weight is employed. Light UXVs weigh less than 150 kilograms, Medium UXVs fall within the range of 150kg to 600kg, and Heavy UXVs exceed 600 kg. This classification system allows for a nuanced approach to modeling and assessing UXVs, catering to the diverse requirements and capabilities within each weight category.

2.1.1 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAVs), a category likely familiar to the average reader, rotary-wing UAVs are often found in recreational settings, with family members or friends owning them for leisurely flights. The Navy, too, utilizes these compact rotary-wing drones, including quadcopters, hexacopters and octocopters. Despite their affordability and compact size, these UAVs typically weigh no more than 50 kg and offer autonomy ranging from 60 minutes to 15 hours, as indicated by Mario Marques [1]. An illustration of a commonly recognized UAV, the UX-Spyro Quadcopter developed by the Portuguese company UAVision[2], is presented in Fig. 2.1.

Figure 2.1: Illustration of the UAV UX-Spyro Quadcopter developed by the Portuguese company UAVision.

In the realm of UAVs, exists another category known as fixed-wing UAVs. Distinguished by their plane-like design featuring wings, these UAVs differ significantly from their rotary counterparts. Fixed-wing UAVs are particularly well-suited for extended flights, boasting larger autonomy, enhanced load-carrying capacity, and greater overall efficiency. Fig. 2.2 provides an example of a fixed-wing UAV, which, in this instance, incorporates rotary wings as well [3] . This hybrid design offers the flexibility of vertical takeoff and hovering while delivering significantly improved efficiency for long-distance flights compared to the quadcopter showcased in the previous figure.

The payload of an Unmanned Aerial Vehicle, pertains to the equipment designed for transportation, encompassing a versatile range of items that can be accommodated or attached to the drone. Typically, these payloads consist of sensors, cameras, or cargo.

UAVs designated for various missions often feature specialized payloads. These commonly include a high-resolution camera capable of live-streaming recordings to a base

Figure 2.2: VTOne fixed wing UAV developed by the portuguese company Beyond Vision

station. Additionally, a Multispectral Camera is integrated into the UAV's payload, designed to capture electromagnetic radiation across multiple wavelengths, spanning the electromagnetic spectrum. This distinguishes it from a high-resolution camera, which solely captures visible light, whereas the Multispectral Camera can detect a broader spectrum, including ultraviolet and infrared light. Such cameras prove invaluable for missions involving monitoring diverse environments, including agricultural, environmental, military, and defense applications.

Another prevalent payload combination comprises a High Definition, Zoom, and Thermal Camera. This trio plays a critical role in security, surveillance, and search and rescue missions. The Thermal Camera, in particular, is instrumental in scanning areas for heat sources, enabling the drone to identify and zoom in on points of interest. This comprehensive payload configuration enhances the capabilities of UAVs, making them adept at a wide array of missions across different environments.

Within the UXV family, UAVs stand out as remarkably versatile, finding applications across a spectrum of tasks. They exhibit adaptability in diverse fields, serving purposes ranging from aerial surveillance, reconnaissance, and search and rescue to precision agriculture, infrastructure inspection, scientific research, delivery services, and beyond [4].

In the context of this thesis, attention will be directed specifically toward mission types relevant to naval interests [1]. For UAVs, the primary focus encompasses reconnaissance, surveillance, search and rescue, mapping, and surveying. These mission profiles align with naval requirements, emphasizing the UAV's capability to provide critical intelligence, monitoring, and mapping functionalities in maritime contexts.

2.1.2 Unmanned Surface Vehicles

Unmanned Surface Vehicles USV, as their name suggests, operate on the water's surface. Resembling anything from a compact toy boat to a small ship, these vehicles possess the capability to assess not only the water's surface but also some depth beneath it.

Figure 2.3: HydroBoat 1200 USV developed by swedish company SatLab Geosolutions

The initial USV exemplar is a compact 10kg vessel meticulously crafted by SatLab Geosolutions [5]. This lightweight and space-efficient solution is tailored for measuring, conducting bathymetry, performing underwater surveys, and assessing water quality in sheltered waters. Moving up the scale, the U-Ranger, a substantial 1800 kg vessel boasting a high-resolution infrared camera and forward-looking sonar. This sophisticated equipment enables the U-Ranger to excel in reconnaissance and warfare operations, including mine countermeasures and Anti-Submarine Warfare [6]. Some USV even use sails as a mode of propulsion and carry solar panels to be able to perform long missions at sea.

Figure 2.4: U-Ranger Unmanned Surface Vehicle (USV) developed by Calzoni S.r.l.

Most Unmanned Surface Vehicles are equipped with both a camera for visual data capture and a payload, allowing them to perform depth measurements akin to a Single Beam Echo Sounder, a commonly utilized tool in hydrographic surveying, navigation, and marine research.

USVs tailored for environmental surveying purposes are often outfitted with an array of sensors, facilitating diverse analyses of the water conditions. Larger USVs are equipped with advanced technology, including Sonar, Lidar, and Radar systems. These sophisticated technologies contribute to enhanced navigation and mapping capabilities during surveys. With increasing vessel size, USVs typically carry more advanced payloads, optimizing their ability to fulfill the aforementioned purposes with improved precision and efficiency.

For unmanned vehicles, missions can be broadly categorized into two groups: survey-related and non-survey-related.

Survey missions encompass Hydrographic Surveying, Environmental Monitoring

and Protection, Oceanographic Research, and Chemical, Biological, Radiological, and Nuclear (CBRN) missions, where CBRN stands for Chemical, Biological, Radiological, and Nuclear. CBRN missions involve responding to and mitigating incidents related to hazardous materials [1] [4].

On the other hand, non-survey-related missions include Search and Rescue, Defense, Detection, and Security. These missions focus on activities such as emergency response, military applications, threat detection, and ensuring security in a maritime environment.

2.1.3 Unmanned Underwater Vehicles

The final Unmanned Vehicle in this discussion is the Unmanned Underwater Vehicle (UUV). UUVs operate beneath the water at varying depths, ranging from a maximum depth of 200 meters below sea level to an impressive 5 kilometers. These vehicles undertake a diverse array of missions, including surveying, mapping, monitoring, and engaging in military operations such as mine countermeasures and defense against other UUVs and vessels [4].

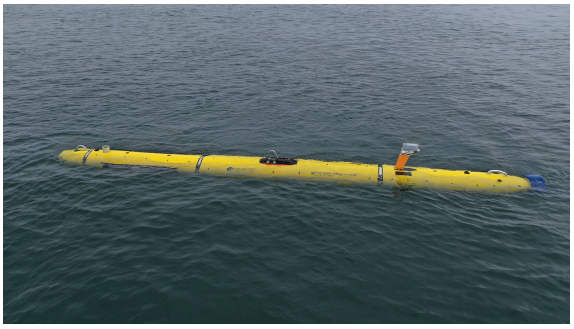


Figure 2.5: Bluefin-12 a UUV designed by General Dynamics

The primary distinguishing factors among different UUVs lie in their depth ratings and payload capacities, both of which are typically correlated with the vessel's size. For instance, in Figure 2.5, the Bluefin-12, a UUV crafted by General Dynamics, weighing 250kg with a depth rating of 200m. Contrasting this, in Figure 2.6, a similar-looking UUV known as the Bluefin-21 weighs 750kg and boasts an impressive maximum depth capability of 4.5km beneath the sea level [7].

Figure 2.6: Bluefin-21 a UUV designed by General Dynamics

Similar to Unmanned Surface Vehicles, the payloads of Unmanned Underwater Vehicles exhibit relatively consistent features across different models. The primary variation lies in the quality and detail of these payloads, a characteristic closely tied to the size of the vessel.

UUVs typically incorporate a Multi-aperture sonar, an advanced underwater imaging technology utilizing multiple acoustic transducer elements arranged in an array or aperture. This configuration enables the capture of high-resolution sonar images of the seafloor and underwater objects. In contrast to traditional single-beam sonar systems, multi-aperture sonar utilizes multiple beams simultaneously, enhancing coverage, accuracy, and imaging capabilities.

Furthermore, UUVs are equipped with a camera and a variety of sensors. These sensors cover parameters such as temperature, pressure, turbidity, and fluorometry. Such a sensor suite allows UUVs to collect comprehensive environmental data during their missions.

For UUVs designed to operate at greater depths, specialized sonar systems come into play. These systems are specifically tailored for mapping the ocean floor, providing valuable insights into underwater topography and geological features. The depth and capabilities of these specialized sonars vary based on the design and intended use of the UUV.

Again similarly to Unmanned Surface Vehicles, UUV missions can be broadly categorized into two main focuses:

Archeology, Exploration, Oceanography, and Marine Biology. UUVs contribute significantly to scientific endeavors by conducting exploration, archaeological investigations, oceanographic research, and marine biology studies. For military purposes, UUVs play a crucial role in various military applications, contributing to mine countermeasures, unexploded ordnance detection, search and recovery operations, port security, intelligence gathering, surveillance, and reconnaissance. These military-focused missions leverage the capabilities of UUVs to operate autonomously in challenging and sensitive underwater environments, enhancing naval capabilities and maritime security.

2.2 Payloads

In the context of unmanned vehicles, the payload refers to the equipment or cargo the vehicle is designed to carry and operate with, beyond its essential systems such as propulsion, navigation, and communication.

Drones can carry multiple payloads simultaneously, each serving a specific purpose or goal. The drones mentioned earlier utilize a wide variety of payloads, which we have classified into three main categories: Cameras, Sonars and Echo Sounders, and Lidar. This categorization is based on the functionality and operating principles of these payloads, making them easier to model and analyze.

2.2.1 Cameras

Starting with cameras, there are several types: standard optical cameras, thermal cameras, zoom cameras, and multispectral cameras.

Standard optical cameras, also known as RGB cameras, capture images in the visible light spectrum, similar to how the human eye perceives the world. They operate using a combination of lenses and light sensors to record light in red, green, and blue channels. These channels are combined to create full-color images that are detailed and true to human vision. These are mainly used for Surveillance and Monitoring, Navigation and Obstacle Detection and Aerial Photograghy.

Thermal cameras detect infrared radiation, or heat, emitted by objects rather than visible light. These cameras use infrared sensors to capture the heat signature of objects in their field of view, converting the thermal energy into a visual image that shows temperature differences. This allows the camera to detect objects or features even in complete darkness, through smoke, fog, or other visibility-obscuring conditions. These are used for Search and Rescue, infrastructure inspection and for military missions.

Zoom cameras have adjustable lenses that allow for variable focal lengths, enabling the camera to zoom in and out on objects without losing image clarity. They are capable of optical zoom, which adjusts the lens to magnify the image before it reaches the sensor, maintaining high resolution even at longer distances.These are used for Surveillance and Reconnaissance, Search and Rescue and wildlife monitoring.

Multispectral cameras, shown in Fig 2.7, capture data across several specific wavelengths of light, including visible and non-visible spectrums (such as near-infrared). These cameras typically use multiple sensors or filters that collect information from distinct spectral bands, allowing for detailed analysis of the composition of surfaces or materials. These are used for Disaster Response, Environmental monitoring and Agriculture.

Figure 2.7: Multispectral UXV Camera

Each of these camera systems brings unique capabilities to unmanned vehicle missions. Standard optical cameras are essential for real-time observation and navigation in visible light, while thermal cameras extend the vehicle's ability to operate in dark or obscured environments by detecting heat signatures. Zoom cameras allow for detailed observation

from a distance, and multispectral cameras provide in-depth analysis of material composition by capturing data across multiple light wavelengths. The integration of these diverse camera systems allows unmanned vehicles to perform a wide variety of tasks, from surveillance and reconnaissance to environmental monitoring and search and rescue operations, making them versatile tools for both civilian and military applications.

2.2.2 Sonars and Echo Sounders

In addition to visual systems like cameras, unmanned vehicles (particularly Unmanned Underwater Vehicles) rely on acoustic technologies to gather information about their surroundings. Two of the most commonly used acoustic systems in these vehicles are sonars and echo sounders. These technologies work by emitting sound waves and analyzing how these waves interact with objects or surfaces, providing detailed information about underwater environments that is often impossible to obtain using visual tools alone. Here, we explore how sonars and echo sounders function and their applications in unmanned vehicle missions.

Sonar systems operate by emitting sound waves into the water and then detecting the echoes that bounce back from objects or surfaces. These are very useful for Search and Rescue operations with sunken vessels as well as for Seafloor mapping. Inside the Sonar family we have three types which all scan slightly differently:

Front-facing sonar is mounted at the front of unmanned underwater vehicle or unmanned surface vehicle and is used to detect obstacles, structures, or other objects directly in the vehicle's path. It is primarily used for navigation and collision avoidance, ensuring the vehicle can safely maneuver around underwater obstacles like rocks, wreckage, or marine installations. This type of sonar typically provides real-time data, allowing the vehicle to react quickly to changes in its environment. Front-facing sonar is especially useful in complex underwater terrains or in proximity to man-made structures like ports and pipelines.

Down-facing sonar, also known as a depth sounder or echo sounder, is mounted on the underside of a UUV or USV to measure the depth of the water beneath the vehicle. By emitting sound pulses directly downward, this sonar system helps create a vertical profile of the seafloor. It is essential for applications like seabed mapping, hydrographic surveys, and depth measurement in navigation. Down-facing sonar is also used in detecting underwater hazards, such as sandbars or submerged rocks, and in determining the topography of the ocean floor, which is critical for environmental monitoring and geological research.

Side-scanning sonar emits sound waves to the sides of the unmanned vehicle, as can be seen in Fig 2.8, which create high-resolution images of the seafloor or underwater structures over a wide area. This sonar system is ideal for detailed mapping and imaging of large underwater spaces, making it popular in applications like search and recovery missions, archaeological exploration, and underwater inspections. Side-scanning sonar

can detect objects lying on or near the seafloor, such as shipwrecks, debris, or marine habitats. By scanning to the sides, it covers a broader area than downward-facing sonar, offering detailed imagery for both scientific exploration and industrial purposes like oil and gas pipeline monitoring.

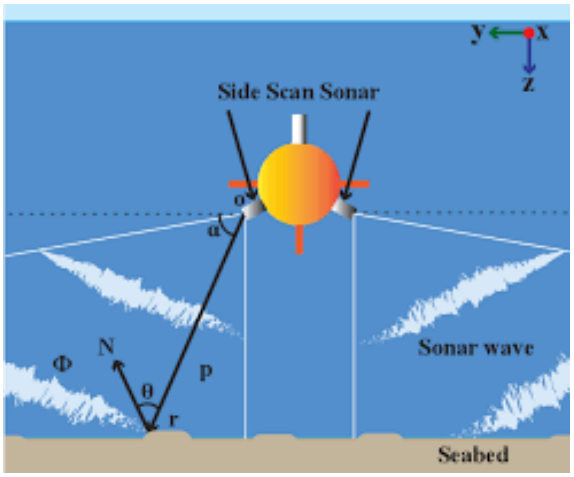


Figure 2.8: Diagram of Side Scanning Sonar

Echo sounders are specialized sonar systems primarily used for depth measurement. They emit a short burst of sound (called a ping) downward toward the seafloor and then measure the time it takes for the echo to return. This time delay is used to calculate the depth of the water beneath the unmanned vehicle. Echo sounders typically operate at higher frequencies than traditional sonar systems, providing detailed depth readings and creating vertical profiles of the seafloor. They are especially useful for underwater terrain mapping, environmental monitoring, and pipeline or cable inspections.

There are two main types of echo sounders: single beam and multibeam. Single beam echo sounders emit a single acoustic pulse directly beneath the vehicle, providing depth information along a single line. Multibeam echo sounders, on the other hand, emit multiple acoustic pulses across a wider swath, allowing for more comprehensive seafloor mapping by covering a larger area with each pass. This distinction makes multibeam systems particularly advantageous for detailed surveys and large-scale underwater mapping.

Both sonars and echo sounders play vital roles in underwater unmanned vehicle missions by using sound waves to gather detailed information in environments where visual systems may fail due to poor visibility or the lack of light. Sonar systems provide broad scanning capabilities that allow for seafloor mapping, obstacle detection, and tracking of marine life or vessels. Meanwhile, echo sounders specialize in precise depth measurement and underwater terrain mapping, which are crucial for navigation, resource management, and environmental monitoring. Together, these acoustic technologies significantly enhance the versatility and effectiveness of unmanned vehicles in underwater missions, contributing to safer navigation, resource exploration, and scientific research.

2.2.3 Radar and LIDAR

Radar and LIDAR technologies are crucial for unmanned vehicle missions, providing highly accurate data for navigation, obstacle avoidance, mapping, and environmental analysis. While Radar uses radio waves to detect objects over long distances, LIDAR (Light Detection and Ranging) employs laser pulses to create detailed 3D representations of environments. LIDAR systems come in two primary types: rotational LIDAR and non-repetitive LIDAR, each serving different roles depending on the mission.

Radar systems transmit radio waves, which reflect off objects in the environment. By measuring the time it takes for the signal to return and any frequency shifts (caused by the Doppler effect), Radar determines the distance, velocity, and size of objects. This technology is effective for detecting large objects at long distances and can operate under various environmental conditions, including fog, rain, and darkness. These are used in Marine Navigation, Collision avoidance and Long-Range Object Detection.

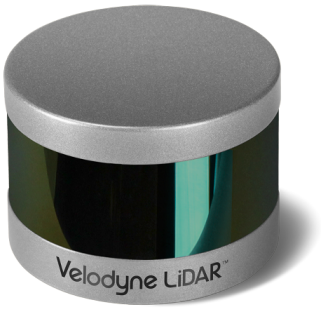
LIDAR uses laser pulses to measure distances and create high-resolution 3D maps of the environment. The system emits rapid laser pulses, which bounce off objects and return to the sensor. By calculating the time taken for the light to return, LIDAR determines the distance to the object with a high degree of accuracy. There are two key types of LIDAR systems used in unmanned vehicles: rotational LIDAR and non-repetitive LIDAR, each with distinct operating mechanisms.

Rotational LIDAR, also known as spinning LIDAR, uses a rotating sensor head to emit laser pulses in a 360-degree field of view. The rotating mechanism allows the LIDAR to continuously scan its environment, capturing detailed point clouds that represent the shape and position of objects around the vehicle. This type of LIDAR system often operates at a fixed rotational speed, providing consistent and reliable mapping over time. These are very commonly used in Autonomous driving, infrastructure Inspection and Robotic Navigation.

Non-repetitive LIDAR, does not rely on mechanical rotation. Instead, it uses electronic or optical beam steering to direct laser pulses in a random or programmable scanning pattern. Non-repetitive LIDAR can scan wider areas with fewer moving parts, leading to increased durability and reduced mechanical failure. The scanning pattern varies, providing detailed and dynamic views of the environment that traditional rotational systems may miss. These are very compact making them perfect for UXVs and frequently used for High Speed Obstacle Detection and Detailed Environmental Mapping

Both radar and LIDAR systems provide essential data for unmanned vehicle missions, each with its unique strengths. Radar excels at long-range detection and performs well in adverse weather conditions, making it essential for navigation, surveillance, and maritime applications. On the other hand, LIDAR offers unparalleled precision and high-resolution mapping, critical for autonomous navigation, environmental mapping, and obstacle detection.

Rotational LIDAR provides reliable 360-degree coverage and is commonly used in



(a) Rotation LIDAR for Unmanned Vehicles

(b) Non-repetitive LIDAR for Unmanned Vehicles

Figure 2.9: Both types of LIDAR Sensor

autonomous vehicles and infrastructure inspection, while non-repetitive LIDAR offers high-speed, dynamic scanning without mechanical components, making it ideal for fast-moving unmanned systems and environments where durability and real-time adaptability are crucial. Together, these technologies enable unmanned vehicles to operate effectively across a wide range of applications, from autonomous transportation to environmental monitoring and search-and-rescue missions.

2.3 Missions

This section explores the broad array of missions conducted by the Portuguese Navy, highlighting the various types of operations that define its dynamic and adaptive character. Through a detailed examination of these missions, we aim to uncover the distinctive operational roles and strategies employed by the Navy, illustrating how its diverse activities contribute to its overall effectiveness and adaptability.

The operational spectrum of the Portuguese Navy encompasses a diverse array of missions, that can be classified into two overarching categories: Civilian and Military. The civilian responsibilities of the Navy extend beyond traditional maritime defense, encompassing crucial roles in monitoring, scientific exploration, law enforcement, emergency services, environmental protection, and more. From monitoring pollution to responding to environmental disasters, the Navy engages in a wide range of activities that contribute to societal well-being and environmental sustainability.

On the military front, the Portuguese Navy undertakes missions vital to national security and international cooperation. Intelligence gathering, reconnaissance, mine countermeasures, anti-submarine warfare, and inspection/identification are among the key military operations aimed at ensuring the nation's defense and contributing to global maritime stability. The Navy's multifaceted role includes support for homeland defense, special operations forces, electronic warfare, maritime interdiction operations, aerial warfare, search and rescue, and combating maritime piracy.

This succinct overview offers a peek into the multifaceted missions carried out by the

Portuguese Navy, detailed extensively in Annex 1.1. Although the idea of formulating an algorithm to tackle the full spectrum of these missions is captivating, it is crucial to recognize the nuanced and specialized aspects of specific tasks undertaken by the Navy. Particularly, Anti-Submarine Warfare, a mission of utmost significance, demands a specialized naval division dedicated to meticulous planning and strategic execution.

Recognizing these inherent challenges, this thesis strategically focuses on the development of a robust framework tailored specifically for handling missions involving scanning or surveying an area. Emphasizing Search and Rescue, Surveillance, and Reconnaissance, these chosen mission types demand efficient area scanning and offer an entry point for algorithmic optimization. Honing in on these specific missions aims to establish a foundational framework that not only addresses immediate needs but also serves as a stepping stone for the seamless integration of more intricate and complex naval operations in the future.

The rationale behind this selective approach lies in the belief that mastering the fundamentals is pivotal before tackling the intricacies associated with the entire spectrum of naval missions. As the framework proves its efficacy in addressing these simpler missions, it can evolve to accommodate the diverse challenges presented by a broader range of operations. This adaptive strategy ensures a methodical and sustainable progression toward the overarching goal of creating a versatile and efficient algorithmic solution for the myriad missions undertaken by the Portuguese Navy.

2.4 Scanning Patterns

In the realm of efficient area scanning, the choice of scanning patterns plays a pivotal role, especially when considering applications such as Unmanned Aerial Vehicles equipped with cameras facing directly downward. It is important to note that these scanning patterns are not limited to visual data but can also be adapted for the use of Sonar or Lidar in the context of Unmanned Surface Vehicles and Unmanned Underwater Vehicles. This versatility underscores the need for an optimized approach to scanning large areas, necessitating the exploration of methodologies that leverage multiple UXVs simultaneously.

In the pursuit of efficient area scanning using multiple UXVs, two fundamental approaches are identified [8]. The first involves flying the drones in a coordinated swarm, wherein the drones maintain close formation to collectively cover a significant area. While this approach presents an opportunity to treat the swarm of drones as a singular entity with the capability to scan large regions, it introduces complexities associated with swarming dynamics. As this falls beyond the scope of the current thesis, the second option was the one chosen.

This approach focuses on decomposing the designated area and assigning specific portions to individual drones. This method facilitates simultaneous scanning by all drones, eliminating the need for intricate swarm dynamics. Unlike the swarm approach, the area decomposition method allows for more straightforward implementation. In the event

of a drone failure, another can seamlessly take its place, or the unscanned area can be reassigned to another drone.

However, this approach introduces its own set of challenges, particularly in managing multiple drones operating in close proximity. The potential for collisions necessitates careful coordination, and ensuring that the images captured by each drone overlap is essential. This precaution guarantees the creation of a comprehensive and cohesive image, compensating for any deviations caused by external factors such as wind.

When considering the selection of scanning patterns, a critical challenge arises when Unmanned Aerial Vehicles turn, causing the camera to deviate from a direct downward orientation and consequently leading to incomplete scans of the expected area beneath the drone. Various scanning patterns have been proposed to address this issue, each offering unique solutions [9] [10].

As depicted in Fig. 2.10, the back-and-forth or zigzag pattern entails the UAV performing alternating motions, strategically designed to minimize the frequency of turning maneuvers. Nevertheless, despite this optimization, the execution of numerous turns remains a concern, potentially resulting in non-cohesive imaging due to the gaps created by these turns.

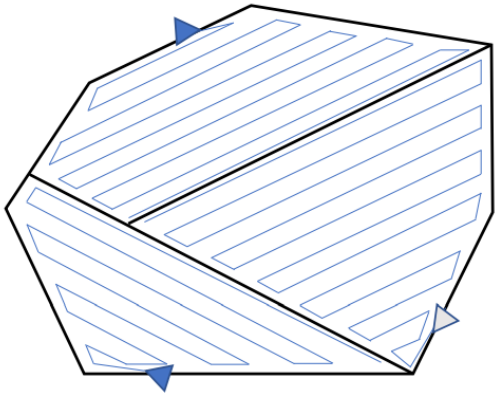


Figure 2.10: Back-and-Forth or Zigzag Pattern

The spiral pattern, showcased in Fig. 2.11, involves the UAV executing a spiral motion characterized by 90º turns. While the data collected during these turns might not be optimal, a notable advantage compared to the zigzag pattern is that these challenging scans are more easily rectifiable. This is due to the turns forming a distinct X pattern, traceable from one corner to the opposing corner, simplifying the process of redoing problematic scans.

The lawnmower pattern involves executing 180º turns outside the search area. This increases flight distance but ensures that the UAV does not turn while actively scanning. See Fig. 2.12.

Lastly, the Zamboni pattern, shown in Fig. 2.13 is similar to the lawnmower but minimizes sharp turns, making it suitable for applications where such maneuvers are not feasible.

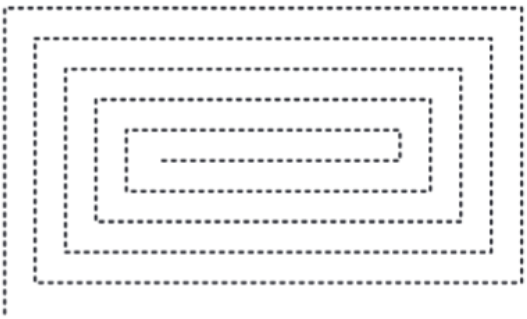


Figure 2.11: Spiral Pattern

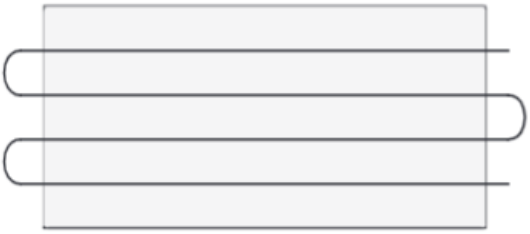


Figure 2.12: Lawnmower Pattern

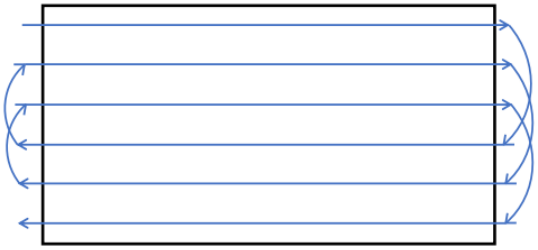


Figure 2.13: Zamboni Pattern

3

Related Work

This chapter is a comprehensive review of related work, synthesizing existing knowledge to situate the research within the broader context of unmanned vehicle missions and scanning methodologies. This review will highlight previous contributions, identify gaps, and justify the approach taken in this thesis.

The initial step in tackling the challenges posed by this thesis was to clearly define the objectives and gain a comprehensive understanding of what constitutes an unmanned vehicle mission. This process involved identifying the types of missions currently executed by the Portuguese Navy, the drones and sensors frequently employed, and the most common mission profiles encountered.

3.1 Missions, Drones and Payloads

To build a robust foundation for this research, we relied on the work of Marques [1], particularly his "Reference Model for Interoperability of Autonomous Systems." This study provides an extensive overview of unmanned vehicles (UXV), tracing their historical evolution and offering critical insights into their operational use within the Portuguese Navy. These insights were instrumental in shaping our understanding of mission requirements and were vital for developing the framework that underpins this research.

A significant real-world initiative that contributed to this study was the Robotics Exercise (REX), a collaborative project involving the Portuguese Navy. REX simulates actual missions and serves as a testing platform for technologies developed by universities, research centers, and industry. It provided valuable hands-on experience, allowing us to observe the intricate interactions between various unmanned systems—such as Unmanned Aerial Vehicles (UAV), Unmanned Underwater Vehicles (UUV), and Unmanned Surface Vehicles (USVs)—during mission execution. These exercises illuminated the distinct roles, capabilities, and limitations of each vehicle type, deepening our understanding of their operational use in the Navy's mission contexts.

For instance, REX 2014 focused on Search and Rescue Search and Rescue (SAR) operations, assessing the performance of USVs with different propulsion systems and payload

configurations [11]. In 2016, the emphasis shifted to underwater docking for UUVs and the collaborative efforts between UAVs and USVs in SAR missions, where UAVs were used to track targets from above, relaying crucial information to USVs that delivered life rafts, food, and water to survivors [12]. The most recent documented exercise, REX 2017 [13], involved a mission in the Tagus River Estuary to assess shallow water areas, offering significant insights into the application of unmanned technologies in Chemical, Biological, Radiological, and Nuclear CBRN missions, particularly in complex environments.

These combined theoretical and practical sources provided a well-rounded understanding of unmanned vehicle missions, especially within the operational framework of the Portuguese Navy. This knowledge has played a pivotal role in shaping the direction of our research and has been essential in guiding the development of the multi-drone mission planning algorithm, designed to address the dynamic and complex environments encountered in these exercises.

Further support for this research came from Ricardo Mendonça's study [14], which explored a collaborative system featuring a USV equipped with an onboard UAV capable of vertical takeoff. This system effectively managed search, tracking, and basic life support tasks for shipwreck survivors.

Moreover, Mario Marques' work with the ICARUS TEAM [15], which participated in Eurathlon 2015 [16], provided another exemplary CBRN mission. This competition, inspired by the 2011 Fukushima disaster, tasked teams with collecting environmental data, surveying the scene, and identifying critical hazards. A further exploration into chemical and radiological detection using UAVs was also documented [17], offering additional insights. Collectively, these studies have significantly enriched our understanding of cooperative robotics in real-world scenarios, providing key perspectives essential for the development of our mission planning algorithm.

The next step involved understanding the capabilities and purposes of each type of drone and identifying which drones are currently in use by various navies. The book "Autonomous Vehicles in Homeland Security" [18] provided valuable examples of drones employed by the United States Navy, offering a comprehensive overview of the specific missions assigned to different Unmanned Vehicles. Similarly, the article "Unmanned Systems in Homeland Security" [4] contributed further relevant information in this area [19].

To deepen my understanding of the technical capabilities of each UXV, I also turned to the manufacturers themselves as a key source of reliable data. In the domain of Unmanned Aerial Vehicles (UAVs), significant insights were gained from Beyond Vision's models [3] and UAVision's designs [2]. For Unmanned Surface Vehicles (USVs), I referenced notable examples from Geo Matching [5] and Maritime Robotics [20]. Finally, for Unmanned Underwater Vehicles (UUVs), models from General Dynamics [7] were critical in understanding their operational roles. With this extensive data in hand, I synthesized the information to create a detailed table, correlating each type of UXV with the specific mission profiles for which it is best suited.

3.2 Mission Planning and Algorithms

The following stage entailed a review of contemporary research and solutions for multi-UAV mission planning. One of the primary challenges was determining how to divide the search area among drones and how to model their behaviors. The paper Drone Fleet Deployment Strategy for Large-Scale Agriculture and Forestry Surveying [21] provided a foundational framework for this, offering strategies for optimizing drone fleet deployment in complex, expansive environments.

When examining the problem from a heuristic standpoint, the paper Coordinated Routing System for Fire Detection [22], which employs both drones and trucks, offered a method of modeling the variables in such a way that a single heuristic function could optimize a complex problem. Similarly, the work in Recharging Flight Planning for Logistics Drones [23] informed our approach by computing drone paths based on locations and the availability of charging stations. These two references were instrumental in shaping the modeling concepts applied in this thesis.

Additionally, the application of genetic algorithms in drone mission planning was explored through several significant studies [24], [25], [26], and [27]. These works inspired the idea of using evolutionary algorithms to create optimized paths for each drone, while also providing valuable insights into the modeling of chromosomes for genetic algorithms.

Through a careful review of these and other papers, a prevalent approach to modeling UXVs and their missions emerged. Many of these solutions relied on heuristic functions, integrating numerous parameters and variables to optimize key factors like cost and time. This method allowed for the systematic comparison of different UXV combinations by adjusting the variables within the heuristic function. Our modeling strategy follows a similar approach, with each drone and mission represented as an array of variables fed into the heuristic function to explore the optimal deployment strategy.

A turning point in this research occurred during a meeting with GMV's department in Spain, which focuses on the Future Combat Air System (FCAS). This interaction aligned closely with our theoretical approach and provided invaluable insights. The collaborative discussions with GMV colleagues significantly enhanced our understanding, especially through relevant papers shared during the meeting. One such work, Task Allocation in a Drone Coalition [28], examines algorithms based on real-world behavior within drone coalitions, exploring innovative approaches like market-based methods and animal pack behavior emulation. Another key paper, Multi-UAV Coalition Path Estimation and Task Allocation [29], introduced a cyclic process of path estimation, task allocation, and multi-UAV coordination, wherein each cycle refines the solution—an approach we intend to use in our optimization strategy.

Further exploration included surveys on algorithms used for multi-UAV tasks, with notable contributions from papers such as [30], [31], and [32], each offering comprehensive insights into the diverse range of algorithms applied to this field.

The culmination of insights from REX exercises, cooperative robotics, drone capabilities, mission modeling, and the ongoing exploration of algorithms positions this research at the forefront of innovation. By integrating both theoretical explorations and practical collaborations, this work lays a solid foundation for the unique contributions it will offer to the field of multi-drone mission planning and optimization.

4

Architecture

4.1 Solution

Our challenge is centered around the efficient selection of the most suitable Unmanned Vehicles from a diverse array of options to effectively carry out a designated mission. The system takes input data, including information about the drones and their respective payloads, mission area details, and the mission type, along with the optimization metric cost, time or cost and time. The program's output is a carefully curated selection of drones and their payloads, each accompanied by a designated path to be followed during the mission.

To address this complex task, the importance of breaking it down into manageable modules and constructing a well-defined pipeline is recognized. Consequently, we've divided the task into four main aspects as can be seen in Figure 4.1.

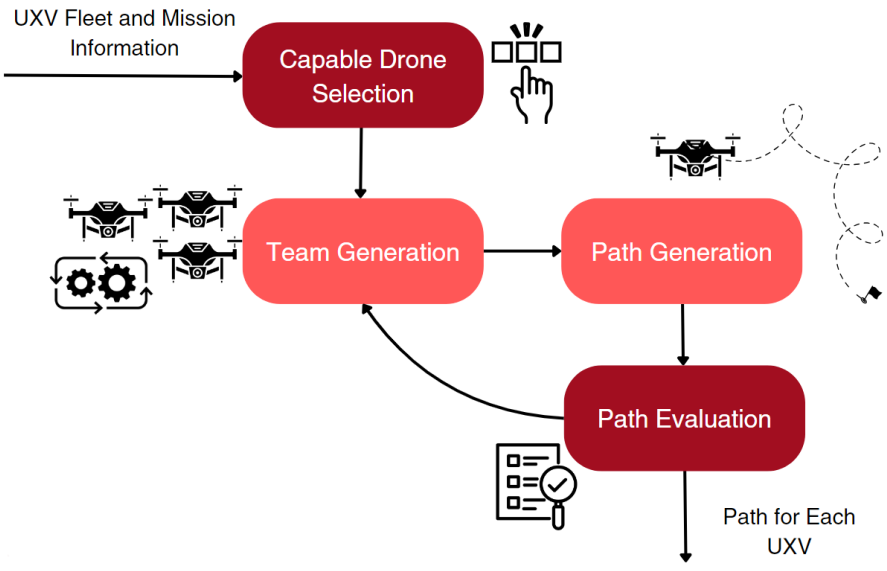


Figure 4.1: Algorithm Module Diagram

The first module, Capable Drone Selection, focuses on identifying drones and payloads that meet the mission's requirements. Next, the Team Generation module forms teams of

drones based on the selection made by the previous module. For each team, a mission path is generated in the Path Generation module, followed by an assessment in the Path Evaluation module according to user-defined guidelines. These final three steps are repeated iteratively until the optimal team for the mission is identified.

Our implementation places a significant emphasis on the independence of each module, ensuring isolated functionality. This design facilitates the seamless integration of new modules in the future, enhancing the overall utility and feature set of the application.

In the upcoming chapter, delves into the definition of a mission and meticulously explore the unique requirements associated with each mission type. This exploration aims to provide a comprehensive understanding of the myriad factors influencing the decision-making process.

4.2 Capable Drone Selection

For effective Drone Selection, it is imperative to comprehensively model all variables in this problem, with a focus on the limitations of each drone and the mission parameters, which are pivotal in decision-making.

To begin, some of the Unmanned Vehicles selected as examples feature interchangeable payloads. For instance, Unmanned Aerial Vehicles (UAVs) are limited to accommodating a single camera at a time. Consequently, a crucial decision involves selecting the most suitable camera which fits the users type preference.

In addressing this intricacy, the approach involves treating the drone and camera as an integrated component. For drones with interchangeable parts, this solution duplicates entries for each payload, effectively expanding the drone and payload combinations. This method simplifies the comparison process, making decision-making more straightforward.

Consistent with this concept, the aim to further enhance the framework by subdividing missions into smaller behaviors known as Movement Profiles. For instance, consider a mission focused on area reconnaissance. This mission can be deconstructed into three distinct phases: moving from the base to the desired scanning location, conducting the area scan, and returning to the base. These phases can then be condensed into two fundamental "Movement Profiles": moving from point A to point B and scanning the area. With this implementation, the goal is to create a versatile set of behaviors, enabling the seamless expansion of supported missions by utilizing these fundamental building blocks in varied configurations.

Each Unmanned Vehicle will be equipped with a catalog of predefined Movement Profiles that they can execute. This not only streamlines the operational aspects but also simplifies calculations. Since each Movement Profile comes with pre-calculated values for speed and flight height, it becomes effortless to optimize functions such as calculating operation time and cost.

When it comes to the UXV they will be simplified into the following variables:

- **ID** - Id indicating the model and number of the UXV.
- **Type** - Type of UXV based on its working environment.
- **Position** - Current UXV position.
- **Fixed Payloads** - Payloads which cannot be removed from the UXV.
- **Changeable Payloads** - Payloads which can be mounted on the UXV.
- **Mounted Payloads** - Changeable payloads mounted on the UXV.
- **Endurance** - Maximum time that the UXV can navigate.
- **Max Speed** - Maximum UXV speed.
- **Max Height** - Maximum height the UXV can navigate at.
- **Weather Limit** - Weather Limit in the Beaufort scale.
- **Max Turn Rate** - Maximum amount the drone can turn in a second.
- **Movement Profiles (MP)** - One or more profiles that specify at each moment the fly features of the UXV.

Each movement Profile will also have the following format:

- **MP Speed** - Speed of the UXV for the Movement Profile.
- **MP Cost Per Hour** - Cost per hour of the UXV executing the Movement profile.
- **MP Height** - Altitude of the UXV when using a route Movement profile.
- **Type** - Type of Movement (scanning or moving from A to B).

It is important to note that the height parameter for the UUVs is in fact depth of operation, the use the same parameter for the purposes of simplifies implementation.

When considering payloads, a simplified approach has been chosen by treating each UXV and its associated payload as a static object and generating all the possible combinations. Therefore the payloads will be represented by the following variables:

- **Name** - Name of the Payload.
- **Type** - Type of Payload which influences their behavior.
- **Range** - Range of the Payload sensor.
- **Maximum Depth** - The maximum depth the payload is able to work.
- **Payload Power consumption** - The consumption per hour of utilizing the payload.

- **Cost Per Hour** - Resolution of the camera.
- **Nocturnal Support** - If the payload is able to operate at night.

These variables simplify the representation of various payloads, but since each payload operates differently, it is necessary to extend the main payload class to accommodate their unique behaviors. To address this, eight specialized classes were created, each tailored to a specific type of sensor: Camera, Downward-Facing Sonar, Forward-Facing Sonar, LIDAR, Multi-Beam Echo Sounder, Radar, Side-Scanning Sonar, and Single-Beam Echo Sounder. Each class implements a unique method for calculating the area that the sensor can scan and handles distinct variables specific to that sensor type. These details will be further elaborated in the implementation chapter.

Lastly, the mission representation incorporates all the details needed from the mission:

- **Mission Type** - Specifies the nature of the intended mission.
- **Time of Day** - Crucial for discerning whether the mission will transpire during daylight or nocturnal hours, influencing payload selection.
- **Weather Information** - Recognizing the impact of weather on UXVs and ensuring tailored deployment.
- **Tridimensional Zone** - Classifies the mission environment into air, surface, and underwater categories.
- **Time Weight** - Importance of the Mission Time in the evaluation of the performance of the mission.
- **Cost Weight** - Importance of the Mission Cost in the evaluation of the performance of the mission.
- **Base Location** - Location from where the drones will depart and return.
- **Drone Quantities** - The amount of each of the different UXVs.
- **User Payload Selection** - The users selected the Type of payload for each of the drone Types.

With all necessary information mapped, the Capable Drone Selection module evaluates key parameters related to the mission's environment. For missions in three-dimensional space above the surface, drones without flight capabilities are automatically excluded. A visual representation of the operational areas for each type of Unmanned Underwater Vehicle (UUV) is shown in Figure 4.2.

Similarly, drones that do not meet weather-related criteria are also disqualified. Following this, the algorithm verifies whether each drone's payload aligns with the mission environment's requirements or the user's preferences. This meticulous filtering process

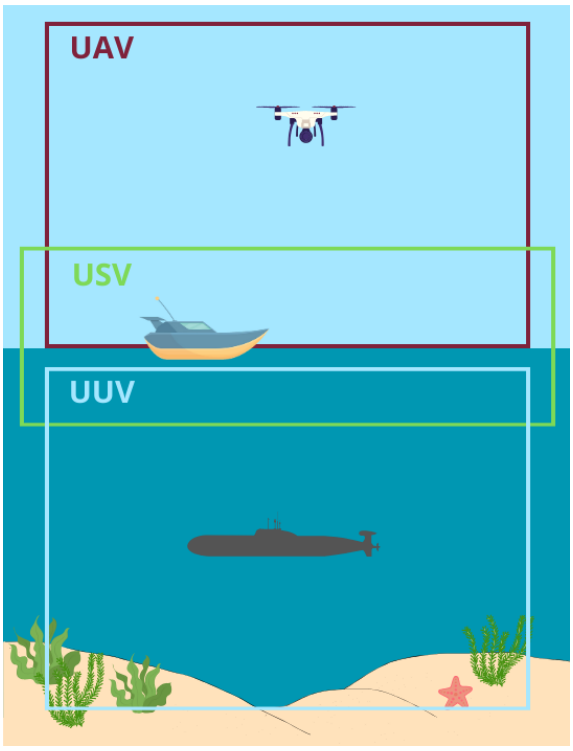


Figure 4.2: UXV working environment according to type.

significantly narrows the pool of candidates, efficiently preparing the system for the subsequent, more resource-intensive phase of allocation and mission planning.

4.3 Team Generation

The Team Generation module receives the list of drones deemed capable for the mission and is responsible for forming teams to carry out the mission tasks. This module primarily considers one critical aspect of the mission requirements: ensuring that each environment is adequately covered by the appropriate drones. Specifically, this means assigning a drone capable of scanning underwater and another capable of scanning the water's surface, if both types of scans are required by the mission.

To achieve this, the module first categorizes the drones based on their capabilities. Unmanned Aerial Vehicles (UAVs) are designated for surface scans, as they can only operate above water. Unmanned Underwater Vehicles (UUVs), on the other hand, are exclusively used for underwater scans. The challenge arises with Unmanned Surface Vehicles (USVs), which have the flexibility to scan either the water's surface or below it, depending on the attached payload.

To address this, the Team Generation module first checks the payload configuration of each USV. Based on this information, it assigns USVs to teams that require their specific scanning capability. This ensures that each team is equipped with at least one drone capable of performing the necessary scans for the mission environment. By systematically

organizing the drones in this manner, the module guarantees that all environmental scanning requirements are met by the assembled teams.

The creation of the Teams will follow different policies:

Exhaustive Approach – This approach generates all possible sub-groups from the initial set, which is also called the non-empty power set, of possible drones for the zone. This guarantees the best group of drones is found, but it is also extremely time-consuming and resource-intensive since, given N elements, the non-empty power set will have $2^N - 1$ groups, which scales exponentially as N increases. Given that this approach can be easily implemented and is the perfect baseline to compare the other implementations.

Evolutionary Approach – Following the concepts in [24], this approach starts by generating an initial population that vaguely represents the range of possible groups. This can be achieved by generating groups of size 1 to N, where N is the number of total elements in the set. Subsequently, these groups are fed into the Path Creation Module, and the group that performs the best according to the heuristic function is selected to generate the next population through mutation, recombination, or both. This establishes a feedback loop where the best groups evolve between the Mission Planning and Path Generation Modules an approach also used in [29]. After a sufficient number of iterations, the best group observed during the entire journey is selected. This approach is not exhaustive but provides improved efficiency and independently if the algorithm has finished running it will always have a viable solution ready unlike the exhaustive search.

Heuristic Approach – The Heuristic Approach in the algorithm involves leveraging domain-specific knowledge and problem-solving strategies to efficiently generate drone groups. Unlike the Exhaustive Approach, which explores all possible combinations, and the Evolutionary Approach, which iteratively refines a population, the Heuristic Approach focuses on using rules of thumb or expert knowledge to guide the generation of drone groups. For example, the Simulated Annealing algorithm, a well-known heuristic technique, explores the solution space by probabilistically accepting worse solutions to escape local optima. Similarly, the Tabu Search algorithm enhances exploration by maintaining a memory structure that prevents revisiting previously explored solutions.

The key advantage of the Heuristic Approach is its ability to rapidly generate reasonably good solutions, making it well-suited for real-time or resource-constrained scenarios. While it may not guarantee an optimal solution, it often produces satisfactory results in a fraction of the time required by exhaustive methods. The Heuristic Approach complements the other two approaches in the algorithm, providing a balance between computational efficiency and solution quality. It leverages knowledge-based strategies, such as those found in Simulated Annealing and Tabu Search, to efficiently navigate the solution space and produce effective drone groupings for diverse mission scenarios.

In exploring the variable allocation problem within the context of multi-drone missions, three distinct approaches have been presented—Exhaustive, Evolutionary, and Heuristic. These approaches, among others documented in [33], serve as foundational frameworks for addressing the complexity of allocating drones to designated zones effectively. While

these algorithms effective in their own right, they also represent a starting point for further exploration and future improvement.

4.3.1 Path Generation

The Path Generation Module is responsible for efficiently planning the flight paths of a group of drones within a specified scanning zone. To streamline this process, drones are separated by type, and their paths are calculated independently. Since drones operating in different environments, such as UAVs and USVs, will never collide with each other. This separation allows for more precise and efficient path planning, tailored to the unique operational zones of each drone type.

This module will receive the team from the Team Generation module then it will start decomposing the scanning zone, a very complex task since the zone can be any shape the user desires as long as it is convex.

Critical factors influencing this decomposition include the drone's endurance and the scanning area size each drone can cover. This last variable is dependent on the camera's Field of View (FOV) and the viable image quality at different altitudes. Notably, cameras with larger FOV can survey larger areas compared to those with smaller FOV, and a similar principle applies to altitude. Therefore, the Path Creation Module must dynamically adjust both the path and the designated zone for each drone, considering these variables.

To simplify mission planning for Unmanned Vehicles, an assumption is made such that all cameras and sensors, such as Sonar and Lidar, are oriented either directly downward or directly in front of the drone. This simplification facilitates the calculation of the scanned area by a UXV during a mission. Additionally, also assuming infinite range communication range for the drones. While limited range could introduce complexities related to the operator's position and base station locations, this consideration is deferred for potential exploration in future iterations.

To achieve the best Area Decomposition from among the drones in the team different division policies will be explored in order to find the best:

Equal Division – Serving as the baseline for path generation, this straightforward approach divides the zone into equal parts for each drone. While it overlooks the impact of crucial variables, it remains easy to implement and proves effective when drones share similar capabilities.

Endurance Based – Recognizing the endurance of UXVs as a key limitation, this approach prioritizes assigning more scanning area to drones with greater endurance. By minimizing the need for frequent recharging trips, potential time and cost savings are achievable. However, a drawback may arise if a high endurance drone with a small scanning area is selected which then drastically affects the mission completion time.

Area Scanned Per Second – Focusing on optimizing efficiency, this approach allocates more scanning area to drones capable of surveying larger regions at once. Especially

beneficial when all UXVs have comparable autonomy, this method outperforms equal division and endurance-based strategies.

Area per Euro – Striking a balance between Area Scanned and Cost, this method considers both area scanned per second and the cost. The goal is to achieve a well-rounded solution that optimizes overall efficiency, acknowledging the strengths and limitations of each drone.

By implementing and rigorously evaluating a range of diverse approaches, our objective is to identify the optimal method for dividing the area, thereby ensuring that the planned mission is executed with the highest level of efficiency and effectiveness.

Once the zone is decomposed, the path of each drone is then mapped from its starting position the the edge of the zone, following a scanning pattern and then returning to the base station.

The State of the Art review highlighted various scanning patterns, noting that the Zigzag and Spiral patterns exhibit sharp edges, which may result in suboptimal area coverage possibly producing gaps. On the contrary, the Lawnmower and Zamboni patterns, characterized by smooth lines, emerge as preferable choices. Despite the fact that these patterns may necessitate longer paths due to turns occurring outside the scanning area, they offer the advantage of seamless coverage.

Consequently, the Lawnmower pattern stands out as the primary choice for deployment. In instances where sharper turns may pose challenges, the Zamboni pattern serves as a viable alternative. This strategic selection ensures efficient area scanning while addressing potential issues related to sharp turns in the chosen patterns.

In conclusion, the Path Generation Module plays a pivotal role in efficiently planning the flight paths of a group of drones within a specified scanning zone. By dynamically dividing the zone into manageable sections based on input parameters such as drone positions, scanning area, and base station location, the module meticulously devises optimal flight paths, considering factors like drone endurance, speed and payload.

4.3.2 Path Evaluation

The Path Evaluation module receives the paths generated by the Path Generation module and evaluates them based on the user's preferred metrics. The user can adjust a slider in the application UI, as shown in Figure 4.3, to set the relative importance of mission cost and duration. The Path Evaluation module then calculates a performance value based on these metrics, enabling the algorithm to directly compare the performance of different teams and identify the most optimal one.

In order to achieve this an fitness function was implemented. A fitness function is a key concept in optimization algorithms, particularly in evolutionary algorithms like genetic algorithms. It serves as a measure of how well a given solution or individual in the population meets the desired criteria of the problem at hand. The fitness function assigns a numerical value, often called a "fitness score," to each potential solution, indicating its

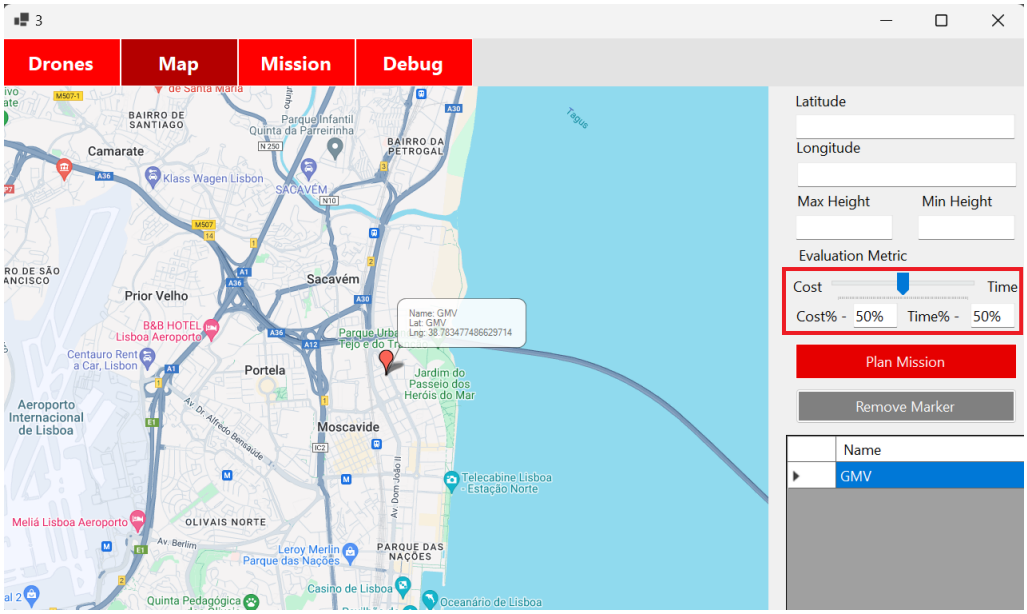


Figure 4.3: Screenshot of UI with the aforementioned slider

quality or effectiveness. The ultimate goal is to maximize (or minimize in this case) the fitness function, leading the algorithm to converge on an optimal or near-optimal solution to the problem.

In addressing the specific problem at hand, we had to merge two distinct fitness functions. One fitness function will be dedicated to estimating the time required to execute the mission, while the other will focus on estimating the cost associated with mission execution.

The output produced by the Path Generation module contains a comprehensive list detailing the drones utilized and the corresponding paths for each drone. These paths are represented by line segments, each adhering to a specific flight plan for the respective drone. The flight plan dictates the speed of the UXV, and when coupled with the line segments specifying distance traveled, it becomes straightforward to calculate the time necessary to accomplish the mission. In Table 4.1, the variables employed for the fitness functions are delineated.

Table 4.1: Variables for the fitness functions

D	The list of drones performing the mission
P	The list of Paths for each drone
Seg_p	The list of Segments in Path p
$Speed_{seg}$	The speed the UXV travels while following segment seg according to the Flight Plan in km/h
$Dist_{seg}$	The distance the UXV travels while following segment seg in meters
$Time_d$	The time drone d takes to perform its flight path in hours
$Cost_d$	Hourly Cost of operating drone d

The time required for each drone to execute its designated path in the mission is

determined by summing the division of the distances covered in each path segment, as generated by the Path Generation module, and the speed specified in the Flight Plan. Mathematically, the time T for drone d can be expressed as:

$$T_d = \sum_{seg \in P_d} Dist_{seg} \; / \; Speed_{seg} \tag{4.1}$$

Once the individual drone execution times are obtained, the overall time required for the entire mission can be calculated by identifying the maximum time taken by any drone to complete its path.

$$\text{Time to Complete Mission} = \max_{d \in D} T_d \tag{4.2}$$

Additionally, the total cost of the mission is determined by summing the costs incurred by each drone throughout the duration of their respective missions.

$$\text{Mission Cost} = \sum_{d \in D} Cost_d \times T_d \tag{4.3}$$

These two values, mission time and cost, will then be normalized to ensure consistency and prevent skewed results. The lower bound for normalization is set to 0, ensuring no negative values, while the upper bound represents the worst-case scenario using all available drones. In rare cases, the normalized values may exceed 1, but this is accounted for in the overall evaluation. The normalized time and cost are then weighted according to their importance to the user, as shown in the following equation:

$$\text{Mission Fitness} = (\text{Mission Time} \times \text{Duration Importance}) + (\text{Mission Cost} \times \text{Cost Importance}) \tag{4.4}$$

This approach allows users to adjust the evaluation based on their preferences, providing flexibility in balancing time and cost.

In summary, the equations presented allow for the comparison of the diverse paths generated by the path generation module. By utilizing these equations, the optimization of the problem becomes feasible, with the flexibility to either minimize overall mission cost or minimize the time required to successfully complete the mission. This comprehensive approach provides a robust framework for decision-making and strategic planning in the context of mission execution.

4.4 Application

The prototype created for this algorithm also included a small application to be able to interact and plan missions using the algorithm as well as exporting them into a simulator.

When considering the implementation of an application, the choice of programming language played a pivotal role, and among the myriad options available, three languages emerged as prominent contenders.

Java – stood out as a versatile and object-oriented language, developed by Sun Microsystems (now Oracle), Java's platform independence allows its programs to run on any device equipped with the Java Virtual Machine (JVM). Java boasts strengths in portability, robust multithreading support, and a wealth of libraries and frameworks. This is also the language I have most experience working with and developing application and services.

C# – C# is a powerful, modern programming language developed by Microsoft. It is widely used for building Windows applications, web applications, and games using the .NET framework. C# is known for its strong type system, automatic memory management through garbage collection, and support for object-oriented programming principles. It offers a rich set of features, including LINQ (Language-Integrated Query) for data manipulation, asynchronous programming, and strong integration with Microsoft technologies. However, C# has limitations when it comes to platform independence, as it is primarily associated with the Windows ecosystem. While efforts like .NET Core aim to address this, complete platform agnosticism is not as inherent in C# as it is in languages like Java.

Python – Takes a different approach as a high-level, interpreted language with a focus on readability and ease of use. Renowned for its simplicity, Python is a preferred choice for beginners and rapid development. It finds applications in diverse domains, including web development, data science, artificial intelligence, and automation. Python's strengths lie in its concise and readable syntax, extensive standard library, and automatic memory management through garbage collection. However, its execution speed is generally slower than that of languages.

Any of these three languages is well-suited for implementing the algorithm, but C# stands out due to its robust features, ease of UI design, and efficient implementation support, enabling the rapid development of a small application with minimal code. Additionally, GMV's preference for C# and my personal interest in learning the language further solidified the decision to proceed with C# for the project's development.

4.5 Simulator

The simulator serves a critical role in assessing the algorithm's proficiency and its real-world performance without incurring the substantial costs associated with live exercises involving actual Unmanned Vehicle. Consequently, selecting the most fitting simulator becomes a pivotal decision.

The goal of the simulator was to enable visualization of the drones within their 3D operational environment, providing an additional perspective on the paths generated by the algorithm and serving as a tool for validating those planned paths.

The chosen simulator must possess the capability to faithfully replicate the behaviors of all three types of Unmanned Vehicles—Unmanned Aerial Vehicles (UAVs), Unmanned

Underwater Vehicles (UUVs), and Unmanned Surface Vehicles (USVs).

The first idea was to use an of the shelf simulator and mold it to our purpose where three options stand out: Gazebo, AirSim, and Webots.

Gazebo – is a versatile, general-purpose robotics simulator capable of accommodating a wide range of unmanned vehicles, including UAVs, UUVs, and USVs. Notably, Gazebo stands out for its flexibility in simulating environmental conditions, encompassing dynamic factors such as wind and water conditions. This makes it a suitable choice for comprehensive UXV simulations.

AirSim – Developed by Microsoft, AirSim primarily focuses on UAVs but extends its support to ground vehicles as well. However, support for UUVs may be more limited compared to other vehicle types. AirSim is renowned for its realistic environments and sensor simulations for UAVs. While excelling in aerial simulations, its support for simulating water conditions for UUVs may be more limited compared to its capabilities for air-based simulations.

Webots – is a general-purpose robot simulator that supports a variety of robotic systems, including UAVs, UUVs, and USVs. Its versatility extends to the simulation of different environments, including underwater scenarios. Webots offers capabilities to simulate factors such as water conditions, making it suitable for a broad spectrum of UXV simulations.

Both Gazebo and Webots initially appeared to be great options, but as we delved deeper into the problem, it became clear that the available time (1 month) would not be sufficient to learn these new tools and write the necessary code to implement the Unmanned Vehicles. Additionally, managing the physics and adapting to the different environments each tool operates in would have been extremely challenging, especially given my lack of experience with either platform.

Given the challenges of using Gazebo or Webots within the limited time frame, we opted to build a custom visualization tool with integrated validation steps. To achieve this, we utilized Unity, a powerful and widely used cross-platform game engine. Unity, initially released in 2005, has become a popular choice for developing 2D, 3D, and AR/VR experiences due to its user-friendly interface, extensive asset store, and support for multiple platforms, including mobile, desktop, consoles, and web. It uses C# as its primary scripting language, allowing for efficient game logic management. Beyond gaming, Unity's versatility makes it an invaluable tool for simulations, architectural visualizations, and interactive media projects.

Not only did I have prior experience with Unity from participating in game jams, but the abundance of free educational resources available on the platform also made it the optimal choice for rapidly developing a visualization tool. While it would be ideal to extend this tool into a fully-fledged simulator with realistic physics, including the effects of wind and sea currents on drone behavior, this was beyond the scope of the time frame we had available.

5

Application

The focus of this thesis is the development of the algorithm and framework to plan the Unmanned Vehicle Missions, however this development and testing of the algorithm would be much easier with a way of quickly and easily inputting the desired parameters and selecting the mission area from a maps while also being able to visualize the results on a map.

Therefore the usefulness of a User Interface (UI) was large and one was created but it is important to note the interface is not a polished final interface but a prototype.

5.0.1 Technologies

The Application had two main requirements, the first was having a User Interface which would allow the user to interact with it and the second was to have a map similar to google maps where the user was able to place markers delimiting the mission area and visualize the drone paths once the mission was planned.

To build this User Interface the tools provided by Visual Studio IDE were crucial. Leveraging the Windows Forms .NET framework within the Visual Studio IDE, we could swiftly implement user interfaces with ease.

Windows Forms is a graphical user interface (GUI) framework for designing and building desktop applications in C#. Developed by Microsoft, Windows Forms provides developers with a rich set of tools and controls for creating interactive and visually appealing applications for the Windows operating system.

With Windows Forms, developers can quickly design user interfaces using drag-and-drop functionality and customize the behavior and appearance of controls through properties and events. The framework includes a wide range of pre-built controls such as buttons, text boxes, listboxes, and more, making it easy to create versatile applications for various purposes.

Additionally, Windows Forms integrates seamlessly with other .NET technologies, allowing developers to leverage the power of the .NET Framework for tasks such as data access, networking, and security. Overall, Windows Forms simplifies the process

of developing desktop applications in C#, enabling developers to create robust and user-friendly software solutions for Windows environments.

Next was a library to represent an interactive map where users can input mission limits, we chose one that seamlessly integrates with Google Maps, given its widespread adoption and familiarity among users. However, we also placed a premium on flexibility, selecting a library that can easily adapt to other map providers should the need arise. This approach ensures that our application remains versatile and accommodating to different mapping platforms, catering to diverse user preferences and requirements.

Given the plethora of APIs available for Google Maps, our selection criteria emphasized thorough documentation. GMAP.Net stood out for its comprehensive documentation, backed by valuable examples and a helpful video tutorial playlist.

This chosen library seamlessly integrates with the Windows Forms User Interface, facilitating easy display of maps and user interaction, including features like moving, zooming, and adding markers. Its compatibility with multiple map providers enhances versatility and allows for a broader range of map options, which aligns perfectly with the requirements of our application.

5.0.2 Design

When it came to the design of the application the goal was to have it as straightforward as possible with all the information and buttons the user might need while also trying not to crowd the screen with too much information.

The interface has three main pages:

- **Mission Details Page** - This page is dedicated to inputting the preferences of drones and payloads as well as mission time and weather.
- **Map Page** - Contains the map where the user marks the mission area as well as setting the height limits and the preferred judging metric.
- **Mission Plan Page** - Shows the final mission plan with all the drone paths in colors and the chosen drone team.

The initial page shown in Figure 5.1 features an interactive drone board situated on the left-hand side, showcasing all available drones at a glance as well as increase or decrease their quantity by using the arrows next to the name. Users can also seamlessly navigate through this board, clicking on individual drones to explore the payloads which can be mounted on that drone. As they select a drone, the adjacent Payload board on the right dynamically updates, presenting a comprehensive list of payloads compatible with the chosen drone. This intuitive setup empowers users to effortlessly browse and select drones and payloads tailored to their specific needs.

Located on the right-hand side are several selection boxes designed to streamline the mission planning process. The first box, dedicated to mission types, offers users a

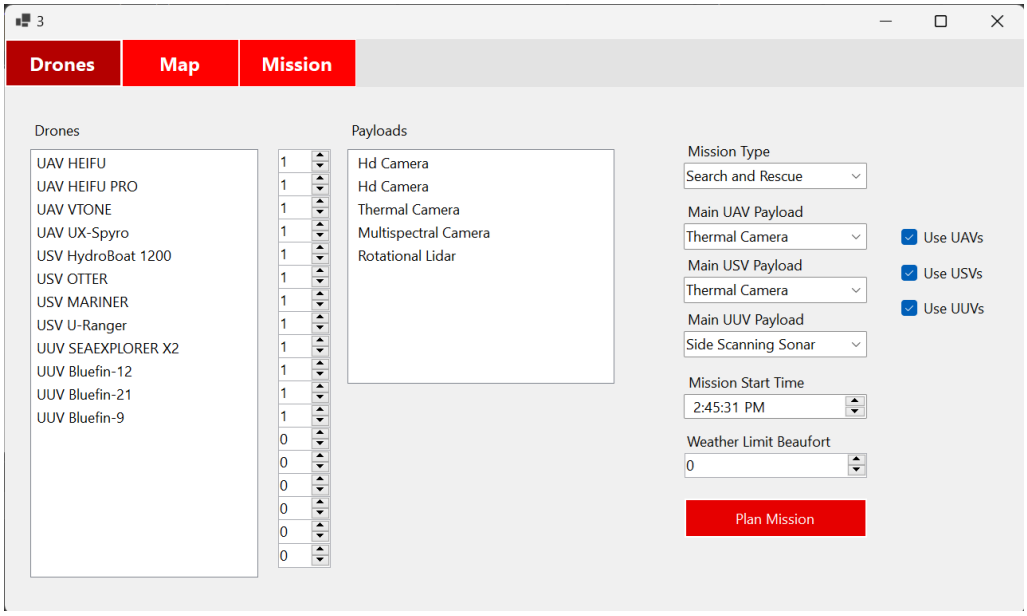


Figure 5.1: Drone Page User Interface

comprehensive list of supported missions upon selection. Once a mission type is chosen, the subsequent three boxes automatically populate with suggested main payloads tailored to each drone type, optimizing selection for the intended mission. This intuitive interface ensures efficient mission planning by providing users with relevant payload options aligned with their selected mission.

Furthermore, upon selecting a mission, the checkboxes for drone types dynamically adjust according to the mission choice, ensuring compatibility and optimal drone selection for the task at hand.

Adjacent to these selections, users can specify the mission start time and check the weather status for mission planning. Once all necessary parameters are chosen, users simply click the 'Plan Mission' button to initiate the process. Subsequently, the page seamlessly transitions to the Map Page, where users can visualize and further refine their mission parameters.

The Map Page, shown in Figure 5.2 features an interactive map interface where users can easily designate the base station and mission area by simply double-clicking and then double-clicking again to place markers delimiting the mission area. On the right-hand side, users can conveniently view the latitude and longitude coordinates of the last marker placed, displayed in the first two boxes for quick reference.

Adjacent to the latitude and longitude display, users can input the maximum and minimum height parameters required for the mission, ensuring precise altitude control. Additionally, users can use the metric slider to select the importance of each of the metrics Mission Duration or Mission Cost the user wants to prioritize.

In the bottom-right corner, a convenient button allows users to remove the currently selected marker, streamlining marker management. Below this button, a board displays

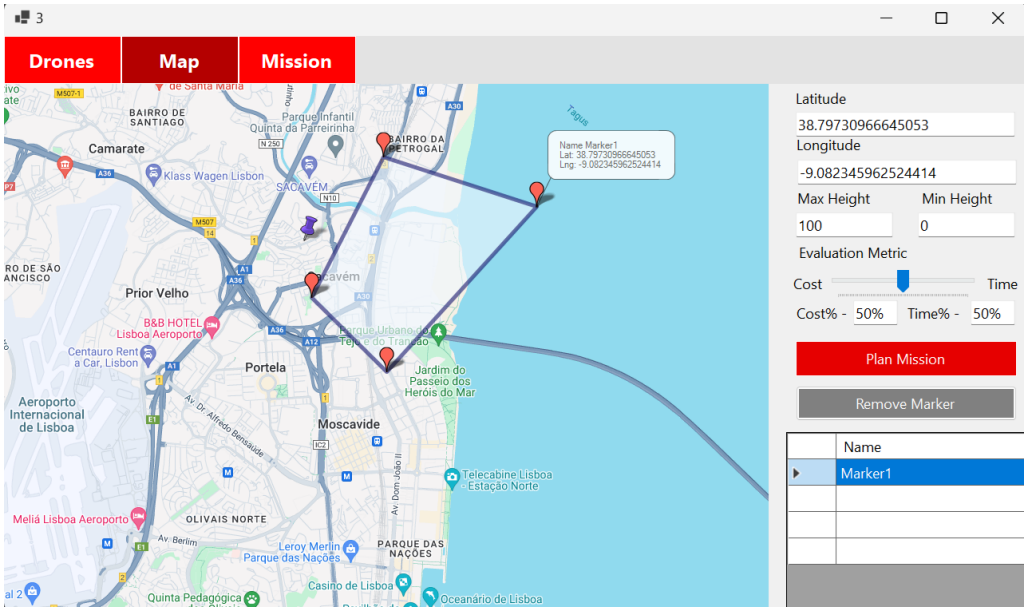


Figure 5.2: Map Page User Interface

a comprehensive list of all placed markers. Clicking on any marker within this list automatically centers the map on that location and selects the marker for deletion, providing users with efficient marker navigation and removal.

After filling all necessary input boxes and outlining the mission area on the map, users simply click the "Plan Mission" button to proceed to the final screen.

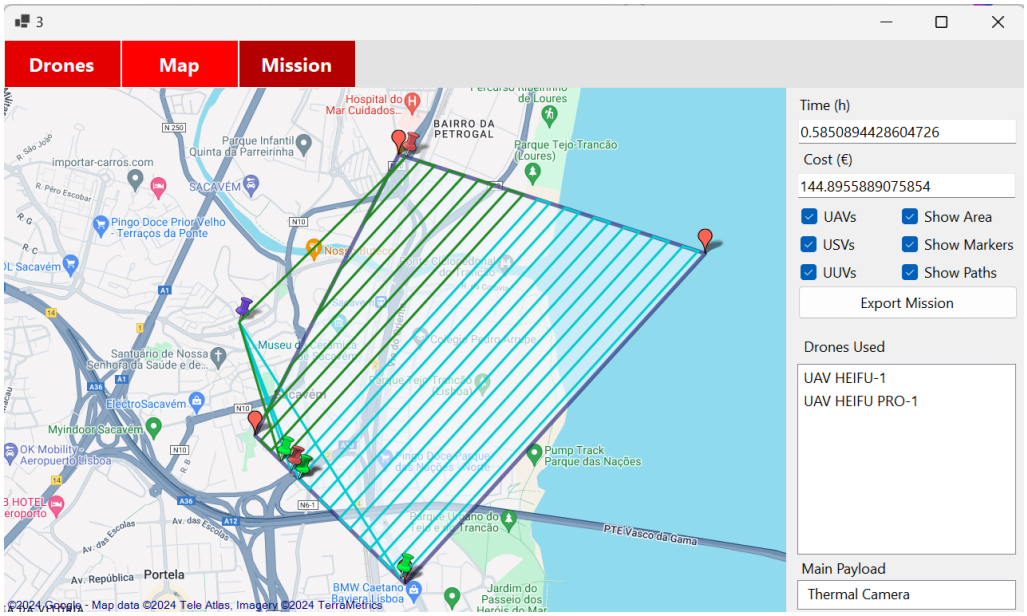


Figure 5.3: Mission Page User Interface

Upon reaching the Mission Planning Page, shown in Figure 5.3 users encounter an updated map displaying colored paths representing the flight routes of each drone participating in the mission. In addition, green and red markers signify the start and

end points of each drone's scan, providing users with a visual overview of the mission execution.

Adjacent to the map, users will find essential mission details such as the estimated mission duration in hours and the associated cost in euros. To manage the visual clutter and streamline information presentation, checkboxes are provided. These checkboxes enable users to selectively toggle the display of drone lines based on drone type, as well as control the visibility of area limits, markers, and flight paths, ensuring a tailored and uncluttered view according to user preference.

Below, a dynamic panel showcases all drones comprising the optimal team selected for executing the mission. This interactive panel empowers users to click on individual drones, revealing the main payload selected for each drone in the table below, enhancing transparency and facilitating informed decision-making.

5.1 Conclusion

Initially conceived primarily for debugging purposes, the UI featured a rudimentary display of the map and flight paths. However, as the algorithm matured and demanded more diverse inputs, the UI underwent significant transformation. This transition culminated in the creation of distinct pages, each tailored to accommodate specific sets of information and streamline user interaction.

Subsequent iterations aimed to enhance user experience by prioritizing intuitiveness, interactivity, and ease of understanding. Through these refinements, the UI evolved into a more cohesive and user-friendly interface, capable of efficiently handling complex mission planning tasks while remaining accessible to users of varying skill levels.

6

Algorithm

The algorithm is the cornerstone of this thesis, representing one of the most intricate components of the entire implementation. In this chapter, we will explore the various challenges and problems encountered during the development of this algorithm, along with the steps and compromises made to overcome these hurdles.

We will begin by outlining the technologies employed in the implementation, followed by a detailed examination of each module that constitutes the algorithm.

6.1 Technologies

As mentioned in the previous chapter the algorithm will be written in C# therefore there was a need to find the best Geometry library which would allow the algorithm to make complex operations with geometric shapes.

When considering geometries in C#, especially for tasks like polygon division and path line creation, the options for handling geographical coordinates within the .NET framework were somewhat limited. Among the available libraries, a few notable entries stood out:

- **GeoCoordinate.Net** - This is a lightweight library for basic geospatial operations like distance calculations and conversions between coordinate systems. It provides simple methods for working with latitude and longitude coordinates.

- **NetTopologySuite** - A powerful library for advanced geometric operations, including support for geographic coordinates. It offers functionalities for geometric shapes, spatial analysis, and more complex operations like buffering and intersection calculations.

- **DotSpatial** - DotSpatial is an open-source GIS library for .NET that provides a wide range of functionalities for working with geographic data, including support for handling latitude and longitude coordinates, spatial analysis, and visualization.

- **GDAL** - GDAl is an open-source tool for reading, writing, and processing various geospatial data formats. It offers a unified interface for handling raster and vector data, making it essential for geospatial applications, remote sensing, and GIS projects.

Among these libraries, NetTopologySuite stands out as more than just a coordinate manipulation tool—it's a comprehensive geometry library with robust support for geographical coordinates. While other options have their strengths, NetTopologySuite offers unmatched completeness, enabling advanced operations like polygon intersection and area calculation. Its versatility makes it an excellent choice for tasks requiring sophisticated geometric computations in geographical contexts.

Moreover, while other libraries provide some support for shape manipulation, this is not their primary focus, resulting in sparse documentation and code examples for such applications. In contrast, NetTopologySuite, being primarily a geometry library, offers extensive resources for these tasks. However, despite its thorough documentation, it can be challenging to understand and apply. In this regard, technologies like ChatGPT and GitHub Copilot were crucial in finding code snippets and suggestions for solving complex geometric problems using the library.

With the technologies and libraries selected, we can now move forward with the detailed implementation of the algorithm, beginning with the Capable Drone Selection Module.

6.2 Capable Drone Selection

This module is the first to receive mission information when the user initiates the "Plan Mission" command from the UI. Serving as the starting point of the entire algorithm, as illustrated in Figure 6.1, its primary function is to select, from the available drones, those that are capable of successfully performing the mission.

Before we must understand the information the algorithm receives as well as the decision and implications of each of the users inputs;

- **Mission Type** - The type of the mission performed(e.g., Search and Rescue, Surveillance).
- **User Preferences** - The user preferences for unmanned vehicles (UXVs), which types to use.
- **Payload Type** - The payload specifications for each UXV type, the specific type of payload which will be scanning the area for each UXV type.
- **Time of Day** - The hour the mission is to be performed.
- **Weather Condition** - The weather status in the Beaufort Scale.
- **Mission Area** - The markers delimiting the mission area.

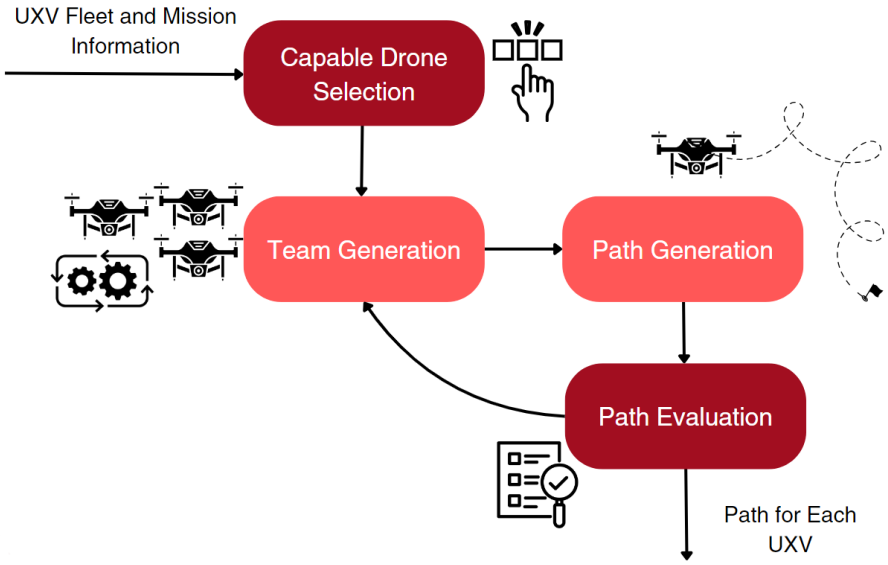


Figure 6.1: Algorithm Module Diagram

- **Base Location** - The location from where the drone are deployed and return to.
- **Height Restrictions** - The maximum and minimum height limit for the mission.
- **List of Drones Available** - The UXVs available to perform the mission and the amount of each one.

In the Drone Page, Fig 5.1, the User will select a main payload for each of the different UXV types this payload will be the one that will perform the scan. Therefore it is based on this payload's specifications that we will determine the width of the scanning path for that drone.

Initially, there was extensive debate regarding whether the user or the algorithm should select the main payload. The original idea leaned towards algorithmic selection based on mission optimization. However, it became evident that many missions, particularly surveillance with UAVs, could be effectively executed with various payloads. For instance, HD cameras, thermal cameras, and zoom cameras all suffice for surveillance missions, with user preference being the distinguishing factor. Consequently, the decision was made to allow users to choose the main payload for each drone type.

We aimed to understand user preferences while also offering payload recommendations for each mission. As a result, when a mission type is selected from a dropdown menu, the input boxes for main payloads are automatically populated with the most versatile option suitable for that mission type. This approach provides users with choice while offering valuable guidance. We achieve this by maintaining a prioritized list of payloads based on their performance in specific missions. The system selects the first applicable payload based on mission requirements and available drones, ensuring efficient mission planning.

The choice of mission timing significantly influences payload recommendations. For instance, using an HD camera for nighttime surveillance may yield suboptimal results, prompting a preference for payloads equipped with night vision capabilities. Therefore our algorithm takes these aspects into consideration.

Lastly, the mission's maximum height represents the upper vertical boundary of the mission area, while the minimum height serves as the lower limit. These values are crucial for the algorithm to determine the appropriate types of UXVs to employ for the mission, as each type comes with its own height restrictions. By defining these parameters, the algorithm can effectively select the most suitable UXVs for the mission, taking into account their respective height limitations.

With all the variables explained. This module initiates by retrieving the list of available drones and swiftly verifies if their capabilities align with the mission requirements. To optimize performance, it executes the fastest verifications first.

Initially, it filters out drones not selected in the user preferences. Subsequently, it checks if each drone's weather limitations surpass those indicated in the mission. The weather limit is quantified using the Beaufort Scale, a measure of wind and water intensity derived from observed conditions, predominantly employed in maritime contexts.

Following the assessment of weather constraints, this module proceeds to verify the vertical limits of the mission area. It matches these limits with the individual height restrictions of each drone carrying the desired payload for the mission.

Next, the module examines the mission area to determine whether the mission will be conducted over land, over water, or in a mixed environment, as this directly influences the selection of drones. To achieve this, the mission area is compared against an overlay of the Portuguese coastline. This comparison allows the module to assess whether the mission area is entirely within, outside, or intersecting the coastline, thereby identifying the specific environment in which the mission will take place.

With this information, the module filters out drones that are not suited for the mission's environment. For instance, if the mission is to be conducted over land, any drones designed exclusively for water operations are removed from the list of potential candidates.

Once the drones suitable for the mission environment are identified, the algorithm checks whether they have the payload specified by the user and whether this payload can operate effectively in the intended mission environment. For instance, if the mission involves reaching a depth of 1000 meters below sea level, the algorithm verifies that the payload is rated to withstand that depth by comparing its depth rating against the mission requirements.

As a final step, the algorithm evaluates the remaining drones to ensure they have sufficient endurance to complete the mission. This is particularly important for smaller UAVs, which may have an endurance of less than one hour. To conduct this verification, the algorithm identifies the furthest point in the Mission Area relative to the drone's starting point (Base Station) and checks whether the drone, using its Movement Profile for point-to-point travel, can complete a round trip without exceeding its endurance rating.

In scenarios where a drone supports multiple payloads matching the user's selection, the algorithm duplicates the entry, each reflecting the different payloads attached. This process guarantees that only drones equipped with the designated payload are considered for the mission, ensuring precision and effectiveness in mission execution.

At the conclusion of the module, the output is a list of drones, each equipped with the user's preferred payload for the specific type of UXV. This list may include duplicated entries of the same drone, albeit with different payloads attached. These duplicates serve the purpose of allowing the algorithm to explore all potential configurations of drones aligned with the user's preferences.

All of these complex verifications are prone to errors when being implemented, necessitating thorough testing to ensure that the implementation aligns with intended behavior. To facilitate this, tests were developed to validate each step of the process and accommodate potential future changes. These tests were implemented using xUnit, a widely used testing framework for the C# programming language.

xUnit is an open-source tool designed specifically for creating automated unit tests. It adheres to the principles of test-driven development (TDD), prioritizing simplicity, flexibility, and extensibility. By employing xUnit, the testing process becomes systematic and efficient, enabling rigorous validation of the module's functionality at every stage. This ensures robustness and reliability in the system, fostering confidence in its performance and facilitating seamless adaptation to evolving requirements.

In essence, the Capable Drone Selection module ensures precise mission planning by aligning drone capabilities with user preferences and environmental constraints, all while maintaining rigorous testing standards with xUnit. This streamlined approach not only optimizes mission effectiveness but also fosters confidence in the system's reliability and adaptability.

6.3 Team Generation

The Team Generation Module orchestrates the formation of teams from the pool of capable drones provided by the Capable Drone Selection module. Initially, the module aims to generate all possible teams by combining the drones in the list.

Teams are subject to two primary requirements: firstly, there should be no duplicate entries of the same drone within the same team, and secondly, each team must contain the minimum required number of drones to fulfill the mission.

The minimum number of drones necessary depends on the mission area's characteristics, as it can be segregated into two distinct scanning environments: scanning over water, suitable for UAVs and USVs, and scanning underwater, achievable with USVs or UUVs. Consequently, the module ensures the presence of at least one drone for each of these environments if the mission area encompasses them.

6.3.1 Exhaustive Approach

In the Exhaustive Approach, the primary objective is to generate all possible team combinations from the list of capable drones. This method ensures that every feasible configuration is considered, allowing for a comprehensive evaluation of potential drone teams.

To achieve this, a recursive function is utilized, systematically generating different team combinations while adhering to constraints, such as the maximum allowable number of drones of each type. The function operates by exploring each possible inclusion or exclusion of a drone in a team, continuing this process until all combinations are covered. As each team is formed, it is immediately checked to ensure compliance with the predefined mission requirements, such as environmental suitability, payload capacity, and endurance. This exhaustive generation and validation process guarantees that no potential team configuration is overlooked, providing a robust foundation for selecting the optimal drone team for the mission.

The Exhaustive Approach guarantees the best solution but it becomes increasingly impractical as the number of drones grows, leading to an exponential increase in the number of possible team combinations and a corresponding deterioration in the algorithm's performance.

This exponential growth in complexity necessitates an alternative solution that can efficiently navigate large search spaces without sacrificing solution quality. To address this, two advanced search algorithms were selected: a Genetic Algorithm for the Genetic Approach and a Tabu Search for the Heuristic Approach.

6.3.2 Genetic Approach

A genetic algorithm (GA) is a search heuristic inspired by the process of natural selection, used to solve optimization and search problems. It starts with a randomly generated population of potential solutions, each encoded as a string (chromosome). These individuals are evaluated using a fitness function, which ranks them based on how well they solve the problem.

The algorithm then selects the fittest individuals to reproduce through crossover (combining parts of two parents) and mutation (randomly altering some genes). This process creates a new generation of solutions, which ideally inherits the strengths of the previous generation while introducing variation. Over successive generations, the population evolves, converging towards an optimal or near-optimal solution.

For this application, the goal of the genetic algorithm is to find the best possible team to complete the desired mission. To achieve this, the genome of each individual in the population encodes a potential team of drones selected from those capable of performing the mission.

Each team is then evaluated using a predefined evaluation function that assesses its performance based on various criteria. The best-performing teams are selected to generate the next population through genetic operations such as selection, crossover, and mutation.

This process is iteratively repeated, with each new generation ideally improving upon the previous one, until a satisfactory solution is reached.

To implement the genetic algorithm, the Genetic Sharp library was chosen. This decision was based on the library's widespread use, strong reputation, and extensive support within the developer community. While implementing the algorithm from scratch was considered, it was deemed inefficient to invest time and resources in developing a custom solution when a proven and highly regarded library was readily available.

Genetic Sharp provides a robust and well-documented framework for genetic algorithms, allowing us to leverage its capabilities and focus on fine-tuning the algorithm to meet the specific needs of our mission planning application. By utilizing this established library, we ensure a reliable and efficient implementation, accelerating the development process and enhancing the overall quality of the solution.

A notable feature of GeneticSharp is its support for parallel execution, which enhances performance by allowing concurrent evaluation of the fitness function, particularly beneficial for computationally intensive tasks. The library also provides great interfaces which facilitate the implementation of genetic algorithms making it a seamless venture.

When implementing a genetic algorithm, several key concepts need to be understood:

- **Population:** A collection of individual solutions to the problem. Each individual, often represented as a string of binary digits (a chromosome), encodes a possible solution.

- **Chromosome:** A single solution within the population. It is usually represented as a string (binary, real numbers, or other forms) that encodes the parameters to be optimized.

- **Gene:** A part of the chromosome that represents a single parameter of the solution.

- **Fitness Function:** A function that evaluates and assigns a fitness score to each individual in the population based on how good a solution it is to the problem. The fitness score is used to rank individuals.

- **Selection:** The process of choosing individuals from the population to create offspring for the next generation. Individuals with higher fitness scores are more likely to be selected. Common selection methods include roulette wheel selection, tournament selection, and rank selection.

- **Crossover (Recombination):** A genetic operator that combines two parent chromosomes to produce offspring. This mimics biological reproduction and is intended to combine the good traits of parents. Common crossover methods include single-point crossover, multi-point crossover, and uniform crossover.

- **Mutation:** A genetic operator that introduces small random changes to an individual's chromosome. This helps maintain genetic diversity within the population and

allows the algorithm to explore a broader search space. Mutation rates are typically kept low.

- **Generation:** A single iteration of the genetic algorithm, where selection, crossover, and mutation are applied to create a new population of individuals.

The first step in modeling this problem using a genetic algorithm is selecting the representation of each team within the Chromosome. A Chromosome is a collection of genes; therefore, if a Chromosome represents a team of drones, each Gene will denote the number of specific drones in that team.

The drone teams have two constraints:

- They must be formed only from the list of capable drones.
- The number of drones of each type in the team must adhere to the availability constraints for the mission.

The Genetic Sharp library greatly facilitates this representation by providing several pre-made interfaces: Floating Point Chromosome, Integer Chromosome, and Binary Chromosome. The Floating Point Chromosome is not suitable for our application since it doesn't make sense to have fractional drones in a team. Similarly, the Binary Chromosome is inadequate as it can only indicate whether a drone is in the team or not, without specifying the quantity. Therefore, the Integer Chromosome is the best fit for our purposes.

Using the Integer Chromosome, each Chromosome will represent the number of each specific drone in the team. The Chromosome will have the same length as the list of capable drones, and each Gene in the Chromosome will encode the specific quantity of the drone corresponding to the same index in the capable drones list, as illustrated in Figure 6.2.

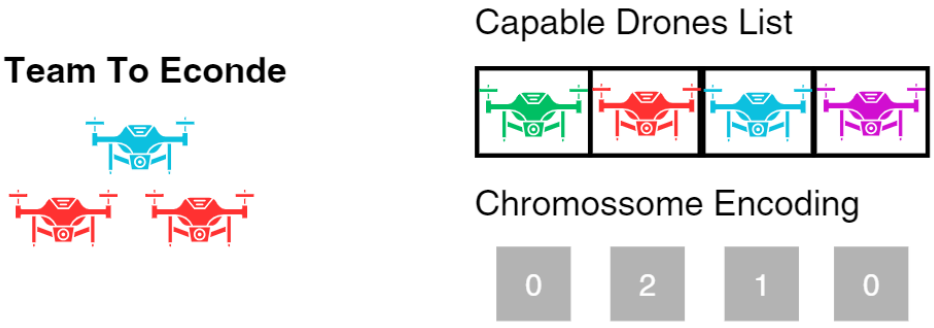


Figure 6.2: Drone Encoding in Chromosome

The next step is the Fitness Function which will be the function which measures the performance of the Team in the mission input by the User. This fitness function follows

the logic of the path evaluation function mentioned previously from the Path Evaluation module only slightly altered to fit the libraries requirements.

To integrate with the Genetic Sharp Library, every Fitness Function must adhere to the IFitness interface. This interface mandates the implementation of the evaluate method, which accepts a chromosome as input and returns its fitness value, represented as a double.

The process begins by decoding the chromosome into a list of Drones, leveraging a predefined list of capable drones. Subsequently, the fitness function assesses the team's ability to execute the mission. While individual drones may be capable of fulfilling specific roles, some missions demand a combination of different drone types. For instance, a mission might require both UAVs for aerial surface surveys and UUVs for deep underwater exploration.

During the course of recombination and mutation across successive generations, it is possible for teams to emerge that lack the collective capability to execute the mission. In such cases, these teams are assigned the lowest possible fitness value of 0. This deliberate devaluation discourages the algorithm from promoting these individuals into subsequent generations, thereby maintaining focus on evolving more capable teams.

The teams that can successfully perform the mission are then passed to the Path Evaluation module. Here, their paths are computed and the cost/time required to complete the mission is calculated. In the Genetic Sharp library, fitness values typically range from 0 to 1. To align with this scale, the computed cost/time value from the Path Evaluation module is inverted. This means that the lower the cost/time required to execute the mission, the higher the fitness score assigned to the individual.

The Genetic Sharp library streamlined the implementation of our genetic algorithm, primarily through two key classes. Our remaining tasks involved defining essential parameters and executing the algorithm. The detail of these parameters will be explained as we walk through the genetic algorithm workflow.

When running a genetic algorithm, the initial population is generated with a size determined by the *population_size* parameter. This size is crucial because it directly influences the algorithm's effectiveness. A sufficiently large population ensures diversity, allowing a broad representation of possible solutions. However, if the population is too large, it can adversely affect the algorithm's performance by increasing computational load and slowing convergence. Balancing these factors is key to optimizing the algorithm's success.

The initial population is created by randomly generating chromosomes based on the rules established in the custom chromosome class, which implements the Integer Chromosome interface. This class, called *TeamChromosome*, which contains the two necessary methods for the genetic algorithm to function: *GenerateGene* which is responsible for generating all of the genes in the Chromosome.

When generating a gene, there is only one constraint: the number of drones of a specific type. The *TeamChromosome* class receives an integer array indicating the maximum

number of available drones for each type from the capable drones list. To generate a gene, the method requires only the index of the gene in the chromosome and randomly generates a value between 0 and the maximum for that drone type. This approach ensures flexibility and adherence to constraints.

Once the initial population is generated, the selection phase begins, where certain individuals are chosen based on their fitness to generate the next population. This selection can be done using various policies, such as:

- **Elitist:** Involves directly copying a certain number of the best individuals (elite) from the current population to the next generation. This ensures that the best solutions are retained.
- **Roulette Wheel:** Assigns a probability of selection to each individual proportional to their fitness. Individuals with higher fitness have a greater chance of being selected, akin to a weighted lottery.
- **Stochastic Universal Sampling:** This method improves upon roulette wheel selection by ensuring a more even distribution of selected individuals. It uses multiple equally spaced pointers to select individuals from a cumulative probability distribution.
- **Tournament:** A subset of individuals is chosen randomly, and the best individual from this subset is selected. This process is repeated to fill the new population.
- **Truncation:** Sorts the population based on fitness and selects the top portion of individuals to form the new generation. The selected top individuals are reproduced until the population size is restored.

Each selection method in genetic algorithms has its advantages and disadvantages. Elitist selection ensures that the best solutions are retained across generations, promoting convergence, but it may reduce genetic diversity.

Roulette wheel selection provides a probabilistic approach favoring fitter individuals, which can maintain diversity, but it may suffer from bias towards highly fit individuals. Stochastic universal sampling offers a more uniform distribution, reducing the randomness inherent in roulette wheel selection, but it is more complex to implement.

Tournament selection is efficient and simple to implement, allowing control over selection pressure by adjusting the tournament size; however, it may still reduce diversity if the tournament size is too large.

Truncation selection is straightforward and can quickly improve the population's fitness, but it risks losing diversity by consistently selecting only the top individuals, potentially leading to premature convergence.

Making the choice of which policy to use is complex, as each policy impacts the algorithm differently. This complexity increases with subsequent steps in the genetic algorithm, adding further considerations to the selection process.

Next is the Crossover phase, where two or three parent solutions from the Selection phase are chosen, and segments of their genomes are exchanged to produce one or more offspring. Crossover can be performed in various ways, but a distinction must be made between Ordered Crossovers, which maintain the order of genes, and Non-Ordered Crossovers, where order does not matter. In our algorithm, the order of genes is not important; what matters are the values in each gene position.

Therefore, only Non-Ordered Crossovers were considered. Examples of these include:

- **Cut and Splice:** Two parent chromosomes are cut at randomly chosen points, and the segments are spliced together to create offspring of potentially different lengths. This allows for varying chromosome lengths in the population.
- **One-Point:** Involves selecting a random crossover point on the parent chromosomes. The segments to the right of this point are swapped between the two parents to create two offspring.
- **Three Parent:** Generates an offspring by combining genes from three parents. For each gene, the value is chosen based on a majority rule or some other decision mechanism among the three parents.
- **Two-Point:** Selects two random crossover points on the parent chromosomes. The segments between these points are swapped to produce two offspring, preserving more genetic material than one-point crossover.
- **Uniform:** In uniform crossover, each gene in the offspring is chosen randomly from one of the corresponding genes of the parents, with a uniform probability, allowing for a high degree of mixing.
- **Voting Recombination (VR):** A voting mechanism among several parents to decide the value of each gene in the offspring, aiming to combine the most favorable traits from the parent set.

Each crossover method in genetic algorithms has its unique benefits and limitations. Cut and splice crossover allows for variable chromosome lengths, providing flexibility but potentially disrupting genetic material continuity. One-point crossover is simple and effective for preserving contiguous gene sequences, yet it can be too limiting in genetic diversity. Three-parent crossover increases genetic diversity by combining genes from three sources, though it may be more computationally intensive.

Two-point crossover maintains more genetic information by swapping two segments, promoting diversity but requiring careful point selection. Uniform crossover allows extensive mixing of genes, ensuring high diversity but risking the loss of valuable gene combinations. Voting recombination (VR) amalgamates traits based on a consensus from multiple parents, promoting robustness but adding complexity to the recombination process.

After the Crossover phase, there is an option to introduce mutations in some individuals. This mutation occurs randomly, based on a defined probability parameter, and is used to increase the diversity of the population. In this implementation, the chosen mutation method was Uniform Mutation, which randomizes one of the genes of an individual when applied.

The result of the Crossover phase, after potential mutation, constitutes the next population. This population then undergoes the cycle of Selection, Recombination, Crossover, and Mutation repeatedly until a termination clause is triggered. The available termination rules were:

- **Generation number:** The algorithm stops after a pre-specified number of generations. This criterion is straightforward and ensures the algorithm does not run indefinitely.
- **Time evolving:** The algorithm stops after a certain amount of computational time has elapsed. This is useful when the runtime is limited by external constraints.
- **Fitness stagnation:** The algorithm stops if there is no significant improvement in the best fitness value over a predefined number of generations. This helps in avoiding wasted computations when the solution has converged.
- **Fitness threshold:** The algorithm stops once a solution with a fitness value exceeding a predefined threshold is found. This ensures that the algorithm stops as soon as a satisfactory solution is discovered.

Each termination criterion in genetic algorithms has its advantages and disadvantages. Using a fixed generation number is simple and predictable, making it easy to plan computational resources, but it may either stop too early before finding an optimal solution or run longer than necessary. Time evolving ensures the algorithm adheres to a time constraint, which is useful for practical applications with strict deadlines, yet it might not always yield the best possible solution within the allotted time.

Fitness stagnation helps to prevent wasted computation by halting the algorithm when further improvements are unlikely, promoting efficiency, but it may prematurely terminate the search in cases where a slight improvement can lead to significant breakthroughs. Fitness threshold termination guarantees that the algorithm stops once a satisfactory solution is found, which is highly efficient when such a solution exists within reach, but it requires prior knowledge to set an appropriate threshold and might miss better solutions if the threshold is set too low.

Among these four termination criteria, some were deemed unsuitable for our purpose. For instance, the fitness threshold and time-evolving termination criteria were excluded. Determining a suitable fitness threshold is challenging because it heavily depends on the specific mission being executed. Similarly, time-evolving criteria do not work well because missions with a large number of drones inherently require more time to find optimal solutions.

Prematurely stopping them could significantly compromise result quality. The generation number criterion was also ruled out due to its variability: some solutions converge quickly in early generations, while others may require more time, making it difficult to set an effective stopping boundary without wasting time or stopping prematurely.

Fitness Stagnation emerged as the best termination criterion for this application because the majority of solutions converge rapidly, allowing those that require more generations the opportunity to do so. While there is a risk that a solution may not converge in a reasonable time frame, our extensive trials showed this was rarely an issue.

Nevertheless, Genetic Sharp allows for multiple termination conditions, which could include using fitness stagnation alongside time-evolving criteria as a precautionary measure. This ensures that a solution is always reached within a reasonable time, even in cases where convergence takes longer.

This concludes the genetic algorithm cycle, where the individual with the best fitness from the final population is elected as the best team and used to calculate the paths users will see. While the termination rule has been selected, decisions regarding the population size, selection method, crossover method, and mutation probability still need to be made.

Due to the complexity and interdependency of these parameters—where changing one can impact the effectiveness of others in achieving optimal solutions—we opted to run all possible combinations of these parameters and the results as well as comparisons between the different approaches will be available in the next chapter.

6.3.3 Heuristic Approach

A heuristic approach to finding the best solution involves employing practical methods and strategies that prioritize speed and efficiency over guaranteed optimality. Unlike exhaustive search techniques that systematically evaluate all possible options, heuristics leverage experience, rules of thumb, or educated guesses to make problem-solving more manageable, especially in complex or large-scale scenarios. While this approach does not always yield the absolute best solution, it often produces satisfactory results within a reasonable timeframe, making it particularly valuable in real-world applications where time and computational resources are limited.

For our Heuristic Algorithm we have chosen the Tabu Search which operates on the premise of iterative improvement by maintaining a memory of previously visited solutions to navigate through the search space more efficiently. This text delves into the application, principles, and comparative analysis of Tabu Search, highlighting its effectiveness in selecting the best team for each mission.

Tabu Search is a metaheuristic optimization algorithm. It extends local search algorithms by using memory structures to escape local optima and enhance the search process. The core idea is to iteratively explore the solution space by moving from one potential solution to another that is in the neighborhood of the current solution.

To prevent cycling back to previously visited solutions, Tabu Search employs a tabu list, which is a short-term memory that stores recently visited solutions or attributes of solutions. Moves that would result in solutions on the tabu list are temporarily forbidden, or "tabu," unless they satisfy certain aspiration criteria that allow them to be overridden if they lead to a significantly better solution.

However, Tabu Search also has some weaknesses. One of the primary challenges is the proper management of the tabu list, including its size and the criteria for tabu tenure.

The tabu tenure is a crucial parameter in tabu search algorithms because it balances between exploration (allowing for temporary "tabu" solutions to be revisited after a certain period) and exploitation (guiding the search towards promising areas of the solution space). Adjusting the tabu tenure can influence the algorithm's performance in terms of convergence speed, solution quality, and ability to escape local optima.

If the tabu list is too short, the algorithm may cycle back to previously visited solutions,whereas if it is too long, it may restrict the search space excessively. Another drawback is the potential for increased computational complexity due to the maintenance of memory structures and the evaluation of aspiration criteria. This can make Tabu Search computationally intensive, especially for large and complex problems. Additionally, the performance of Tabu Search can be sensitive to the choice of parameters, requiring careful tuning and experimentation to achieve optimal results.

Tabu Search was chosen as an additional approach because it employs a heuristic based on a concise set of rules, making it quick to implement. Unlike genetic algorithms, Tabu Search is noted for its ability to avoid getting trapped in local optima, which can be a limitation for genetic algorithms. Therefore, Tabu Search is an excellent candidate to complement and evaluate the effectiveness of the Genetic Approach.

For the Tabu Search implementation, teams will be encoded similarly to how they are in a genetic algorithm. Each team will be represented by an array of integers that denotes the quantity of each type of drone present in the team. This array serves as a genotype, capturing the composition of drones within the team.

The Tabu Search algorithm begins by initializing with a random solution within the search space. In our specific application, this initial solution ensures inclusion of at least one drone type for each required search environment in the mission.

For instance, in an aerial survey mission, the solution includes at least one UAV. Similarly, in missions combining aerial and underwater surveys, the solution includes a UAV and a USV/UUV capable of underwater operations. This approach guarantees that each critical search environment is adequately represented from the outset, aligning with the mission's diverse operational needs.

During each iteration of the Tabu Search, starting from the initial solution in the search space, the algorithm proceeds through a predetermined number of iterations, a key parameter of the method. In every iteration, the algorithm explores neighbors of the current solution. A neighbor is defined as a potential solution where a single drone's quantity has been increased or decreased.

For instance, if the initial solution represents a mission with 3 available drones, like "010", its neighbors could be "000", "110", "011", or "020". Each neighbor solution is then evaluated based on its fitness using criteria similar to those in a genetic algorithm.

This evaluation involves assigning a fitness score between 0 and 1, where configurations that do not meet mission requirements receive a score of 0, while valid configurations receive scores reflecting their performance against the mission objectives.

This iterative process of generating and evaluating neighbors aims to refine the solution iteratively, seeking configurations that optimize the team's performance across various mission environments.

After evaluating all neighbors in a Tabu Search iteration, the best neighbor—typically the one with the highest fitness score—is selected as the new current solution for the subsequent iteration. This best neighbor is also added to the Tabu List, which keeps track of solutions that have been explored in the last x cycles. Here, x is a parameter defining the size of the Tabu List, crucial for enforcing short-term memory within the algorithm.

If the selected best neighbor surpasses the current best solution in terms of fitness, the best solution is updated to this new neighbor. This mechanism ensures that the Tabu Search algorithm continually seeks to improve upon the best solution found so far, effectively navigating the solution space in pursuit of optimal or near-optimal configurations of drone teams tailored to the mission's requirements.

The Tabu List plays a critical role in guiding the search by temporarily prohibiting revisiting recent solutions, promoting diversity in exploration and preventing premature convergence to suboptimal solutions.

When the Tabu Search algorithm reaches either the maximum number of iterations defined by a parameter or a point where no further neighbors can be found for the current solution, it concludes its execution. At this point, the algorithm returns the best solution found throughout its exploration process.

To optimize the Tabu Search algorithm, two key parameters require fine-tuning: the total number of iterations of the Tabu Search and the size of the Tabu List which we will delve deeper in the following chapter.

6.3.3.1 Tabu with Estimation Function

During a meeting with Professor Jorge Cruz about my thesis progress, an intriguing idea emerged. Instead of planning the entire mission for each team to measure its fitness, we could develop an estimation function to evaluate team performance based on the characteristics of all the drones.

This approach would significantly streamline the selection of the optimal team by reducing the time required to identify the best solution. The most time-consuming aspect of the application is generating paths for all possible drone teams, so this method would enhance efficiency and improve overall decision-making. Moreover, it could allow for quicker iterations and adaptability in dynamic environments.

This estimation function could replace the fitness function in both the genetic and tabu approaches. While this assumes that the estimation function accurately assesses mission complexity and each drone's strengths and weaknesses—a significant assumption—if successful, it would dramatically enhance algorithm performance. By optimizing team selection more efficiently, this approach could lead to faster and more effective solutions, ultimately improving mission outcomes.

In this application, we aim to estimate based on either the mission cost or the time required to complete it. Thus, we will develop two estimation functions: one for cost and another for time. These functions will enable more precise evaluations, allowing us to optimize both financial and temporal aspects of mission planning effectively.

Both the estimation functions will follow the same principle, with the information about the mission the algorithm will find the furthest two vertexes from the mission area and calculate the distance between them, next it will place a perpendicular line in the middle of the line formed by these two points and see where it intersects with the polygon then calculate this distance two. The idea is to simplify the shape of the mission area into a simple rectangle to make the estimations easier.

Both estimation functions will follow the same principle. The algorithm will identify the two furthest vertices in the mission area and calculate the distance between them. Then, it will draw a perpendicular line at the midpoint of this line, determining where it intersects with the polygon and calculating that distance as well. This approach simplifies the mission area into a basic rectangle, making estimations more straightforward and efficient.

In this process, the rectangle is divided equally among the drones. Each drone's necessary number of scans is determined by dividing the area width by the drone's scan width. The distance of each scan corresponds to the largest side of the rectangle, and with the known maximum speed of the drone during scanning, the scanning time is easily calculated.

Additionally, the total time includes travel from the base to the scan location and back. This comprehensive time is compared against the drone's endurance. If the endurance is less than the calculated time, the number of round trips required will be the ceiling of the division between the total time and the drone's endurance.

This ensures efficient planning and allocation of drone resources, optimizing both scanning operations and battery management. By adjusting for the endurance, operational continuity and mission success are maintained without risking drone performance or safety.

From this, we determine the time each drone takes to perform its scan. The overall mission time is defined by the maximum time taken by the slowest drone in the team. This ensures that all drones complete their tasks efficiently and within the required parameters.

The cost estimation for each drone is determined by multiplying the time taken for its scan by the operational cost per unit time. To calculate the total cost for the entire team, the individual costs of all drones are summed.

This approach allows us to efficiently estimate the cost and duration of performing the mission without the need to generate paths for each drone, a process that is significantly more time-consuming and resource-intensive. As long as the estimation function accurately captures the complexities of path generation, we can achieve comparable results in a fraction of the time. In the following chapter, these three approaches will be tested and compared.

Following the data flow of our algorithm, once a team is generated in the Team Generation Module, it is passed to the Path Generation Module, where paths for each individual drone are generated.

6.4 Path Generation

The Path Generation module is by far the most complex and intricate of this algorithm and as a consequence the one where most of the time was spent. The process of path creation can be divided in several phases:

- **Coastline Overlay** - Overlaying the coastline to distinguish and separate the water from the land.
- **Polygon Division** - The entire polygon is divided into equal parts.
- **Scanning Line Creation** - The scanning lines are drawn based on the drone and camera characteristics.
- **Scanning Line Connection** - The scanning lines are connected according to the turn rate of the drone.
- **Path Endurance Evaluation** - The path is edited to include roundtrips when the drone cannot complete it in a single trip.

Before delving into the path generation process, it's important to discuss the two distinct philosophies underlying this module, both of which have been implemented and will be explained in this chapter. The first philosophy named *Collision Avoidance Approach* emphasizes designing drone paths that inherently avoid intersecting, thereby eliminating the risk of potential collisions. This proactive approach was initially chosen during the planning phase for several reasons.

One advantage of this method is the elimination of the need for collision detection mechanisms, which simplifies the algorithm and reduces complexity. Additionally, this approach was seen as more straightforward to implement compared to alternatives that involve plotting optimal paths for each drone and then checking for intersections, a process that could impose significant computational overhead. However, this strategy does have its drawbacks. By prioritizing avoidance from the outset, it can lead to longer paths for the drones, which was considered an acceptable trade-off given the simplicity and reduced computational load.

The second philosophy named *Collision Remediation Approach* which involves generating the most optimal paths for each drone independently, with the addition of a collision detection and diversion mechanism to address potential intersections. While this approach initially seemed more complex, requiring precise prediction of drone positions and potentially intricate maneuvers, it offers the advantage of optimizing individual paths and potentially reducing the overall distance traveled.

During implementation, the initial strategy of avoiding path intersections proved more complex than anticipated. While it aimed to simplify the flight paths, it instead led to longer routes and introduced unexpected challenges in path planning. One of the key issues was the need to sequence the drones' paths based on the proximity of their scanning areas to the base station. Drones with scanning areas closer to the base station had to be the last to make their turns, ensuring that their paths would not overlap with those of other drones. This added layer of complexity is illustrated in Fig 6.3.

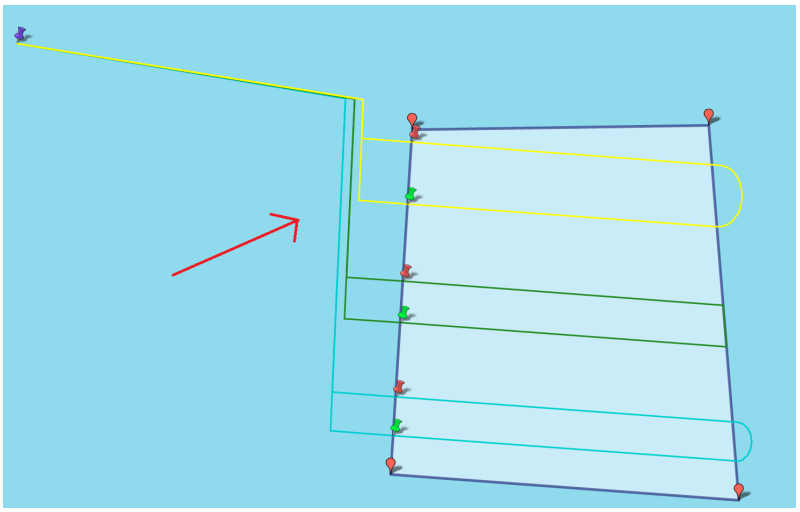


Figure 6.3: Drone Highway

Recognizing these issues, an alternative approach was explored, which focuses on generating optimal paths first and then managing potential collisions as they arise.

Both strategies have their pros and cons: the avoidance-first method simplifies the process but may lead to inefficiencies, while the collision detection approach is more complex but allows for more optimal path generation. This chapter will explore both approaches in detail, discussing their implementation, benefits, and challenges. Furthermore, the effectiveness of these approaches was tested against each other, and the results of these comparisons will be discussed in the next chapter.

Each approach begins with the same initial steps: dividing the polygon, generating scan lines, and connecting those lines to form the scanning paths. The approaches differ only in how the drones travel to and from the Base Station at the start and end of their scanning paths. Consequently, the following sections will describe the shared process, with specific differences highlighted where the methods diverge.

6.4.1 Coastline Overlay

In this step, data from Portugal's coastline is integrated into the algorithm and overlaid onto the polygon to distinguish between water and land areas. This enables the deployment of water-based drones exclusively over water, while UAVs are assigned to land regions, preventing potential mishaps such as beaching of USV or UUV.

To achieve this we firstly gathered detailed information on Portugal's coastline. Finding this data in the required format and detail was challenging, but we succeeded through the European Environment Agency, which provided a ShapeFile containing the coastline of the entire European continent. This resource was instrumental in enhancing the algorithm's ability to accurately differentiate between land and water.

A shapefile is a widely-used digital vector storage format for storing geometric location and associated attribute information of geographic features. Developed by Esri, a shapefile is composed of a collection of files that work together to represent spatial data. The primary file components include the '.shp' file, which contains the geometry of the features (such as points, lines, and polygons). Shapefiles are integral to geographic information systems (GIS) because they allow for the storage, manipulation, and analysis of spatial data, facilitating a range of applications from urban planning to environmental management.

Since the ShapeFile contained data for the entire European continent, we needed to extract the relevant portion for the Iberian Peninsula. This step was crucial because the Portuguese Navy's missions primarily occur near the Portuguese coastline. By focusing on a smaller, more relevant area, we significantly improved the loading and processing speed of the ShapeFile. To accomplish this, we utilized a program specifically designed to handle geographic information system (GIS) files, called Quantum Geographic Information System (QGIS), allowing us to efficiently trim the data to the necessary region. This targeted approach ensured that our algorithm could quickly and accurately access the required geographic details, enhancing its overall performance and reliability.

QGIS is an open-source geographic information system that facilitates the creation, visualization, analysis, and interpretation of spatial data. It supports a wide array of vector, raster, and database formats, including ShapeFiles, GeoTIFFs, and PostgreSQL/PostGIS databases. Due to its user-friendly interface and extensive plugin architecture, QGIS is suitable for diverse applications ranging from cartography and spatial analysis to environmental monitoring and urban planning. This versatile tool enables users to perform complex geospatial analyses, generate detailed maps, and integrate with other GIS tools and data sources.

For this project, QGIS was instrumental in processing the ShapeFile containing the European coastline data. Given the focus on missions near the Portuguese coastline, it was necessary to extract and simplify the relevant section pertaining to the Iberian Peninsula. Using QGIS, the ShapeFile was efficiently trimmed to this specific region, significantly enhancing the algorithm's performance by reducing the data size. Additionally, the shapes within the ShapeFile were simplified to further optimize computational efficiency.

These steps ensured that the geographic data was both precise and manageable, thereby improving the overall effectiveness of the mission planning algorithm.

The goal of utilizing the ShapeFile was to overlay it on the polygon representing the mission area and check for any overlaps. Depending on the mission area's position relative to the ShapeFile, the decomposition of the polygons will vary significantly. Previously, the polygon was divided among the drones without considering the specific environments each drone operated in.

With the addition of the ShapeFile, we now have three distinct cases that need to be handled differently:

1. **The mission area is inside the Shoreline Polygon:** In this case, since the whole mission area is inside the Shoreline Polygon it means the whole mission will be performed on the ground, therefore any aquatic drones will not be attributed any mission area to scan.

2. **The mission area intersects the Shoreline Polygon:** This is the most complex case since the mission area will have both a aquatic part and a terrestrial section the processing of this area division will be explained below

3. **The mission area is outside the Shoreline Polygon:** In this case, the mission will be performed fully over water therefore the mission will be computed as before since any of the drones can perform the mission.

When the mission area intersects the Shoreline Polygon, the algorithm segregates the water area for aquatic drones and the entire mission area for UAVs. Within this scenario, we have two options for further decomposing the area:

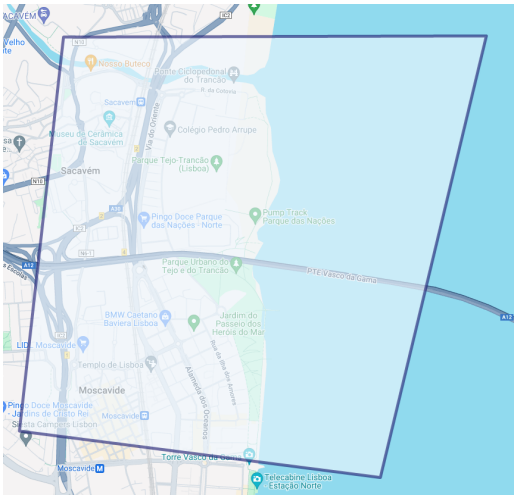
2.1 **Subsurface Scanning by USVs:** If the USVs in the team are scanning below the water surface, the algorithm divides the above-water portion of the mission area among the UAVs and allocates the underwater scanning tasks to the USVs and UUVs. This ensures that each drone type operates within its optimal environment without overlap.

2.2 **Surface Scanning by USVs:** If the USVs are scanning the water surface along with the UAVs, an additional division must be made based on the specific areas each USV will scan. In this case, the algorithm carefully partitions the water surface area among the USVs while the UAVs continue to cover the above-water regions.

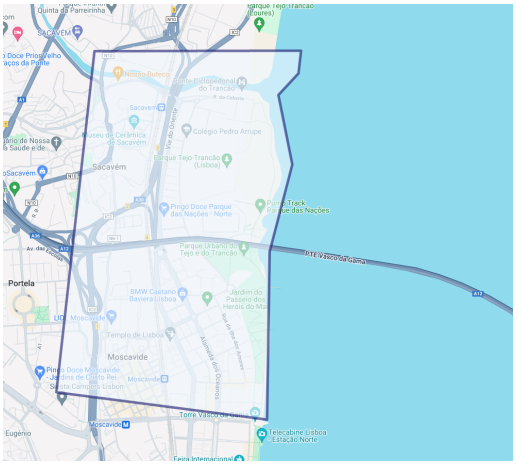
These steps ensure that both surface and subsurface scanning tasks are distributed efficiently, maximizing the capabilities of each drone type and enhancing the overall effectiveness of the mission.

Using the simplest area decomposition method as an example, when dividing the entire mission area equally among each drone in the team, we must ensure that the water area of the mission can adequately accommodate the desired scan area for the USVs.

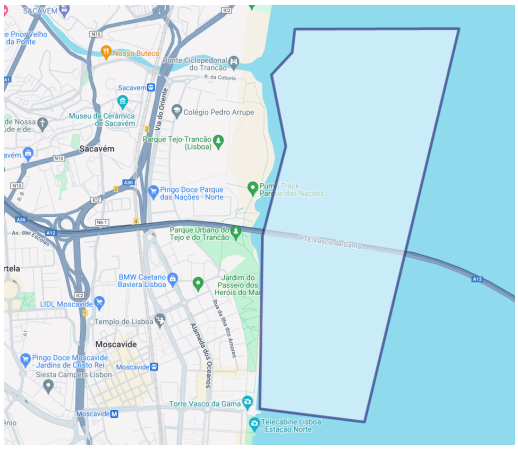
If the water area is equal to or smaller than the combined scan areas required for all USVs, we allocate the ground area among the UAVs and assign the water area to the USVs accordingly. However, if the water area exceeds the sum of all USVs' scan areas, we limit the water area to the total scan area required by the USVs. The remaining water area is then added to the ground area designated for UAVs to scan.



(a) Pre division Area



(b) Land Area after division



(c) Water Area after division

Figure 6.4: Example of the division based on the overlay of Portugal's Coastline on the Polygon

This process significantly enhances the complexity of the area decomposition module but is essential to ensure each drone operates strictly within its designated environment while striving to maintain optimal efficiency.

In Figure 6.4, we observe an outcome of this area decomposition. It is noteworthy that due to irregularities in the coastline, the paths of aquatic drones may occasionally overlap with the beach., a buffer zone of 50 meters along the coastline to mitigate such issues.

6.4.2 Polygon Division

The next step in this module involves dividing the polygon into N segments, where N corresponds to the number of drones in the team provided by the Team Generation module. To accomplish this, four different division policies were implemented to determine which one yields the best performance.

The proposed strategy emphasizes dynamically allocating more area to higher-performing drones while assigning smaller areas to drones with lower performance capabilities. This adaptive allocation ensures that each drone operates within its optimal capacity, maximizing overall efficiency and solution quality. By leveraging this strategy, the algorithm can effectively capitalize on the strengths of each drone in the team, leading to more balanced and efficient mission planning outcomes.

This method not only acknowledges the diversity in drone capabilities within teams but also aims to optimize resource utilization and minimize operational bottlenecks. It represents a proactive approach to achieving more tailored and effective solutions in complex mission planning scenarios.

The three metrics that dictate the amount of area for each drone are the following, with the fourth policy being the equal division of the polygon among the drones.

Endurance - This metric focuses on maximizing mission efficiency by selecting drones with the highest endurance. Endurance directly impacts the number of roundtrips a drone can perform without requiring recharging or refueling. By choosing drones with superior endurance, the number of roundtrips is reduced, thereby minimizing the time required to complete the mission. This metric is crucial for optimizing missions where time efficiency is paramount.

Area Scanned Per Second - This metric assesses the speed and efficiency of drones in scanning an area over time. It measures the rate at which a drone can scan a given area, emphasizing performance in time-sensitive missions where rapid completion is prioritized. Drones with higher area scanned per second are preferred when maximizing operational efficiency and throughput is critical.

Area Scanned Per Euro - Cost-effectiveness is evaluated through this metric, which quantifies the amount of area a drone can scan relative to its operating cost. It enables decision-makers to optimize missions with a focus on minimizing costs while achieving specified scanning requirements. Drones that offer higher area scanned per euro are advantageous for missions where budget efficiency is a primary consideration.

Each metric serves a distinct purpose in mission planning, catering to different optimization objectives such as time efficiency and cost effectiveness. By considering these metrics collectively, teams can strategically select drones that best align with the specific goals and constraints of each mission, ensuring optimal performance and resource utilization.

Initially, two distinct methods for dividing the polygon were explored. The first method involved a mathematical approach capable of dividing both convex and concave

polygons into equal parts. This approach began by computing the area one of the resulting subpolygons and incrementally dividing the large polygon into one subpolygon of the desired area and another with the remained of the polygon.

This was made using a dividing line which to be computed involved complex geometric calculations through a process which relied on the principle that, for a given edge pair, the minimum cut line within a trapezoid must be perpendicular to the angle-bisector of the two edges in the pair [34].

While the concept itself wasn't overly complex, translating it into code using the resources available in the NetTopologySuite library presented significant challenges. The limited functionality of the library made implementation inherently overly complex and, more importantly, excessively time-consuming. Compounding this issue was the exponential time complexity of the operation, arising from the recursive trial-and-error nature of the approach. These factors combined underscored the impracticality of pursuing this method within the constraints of the chosen library.

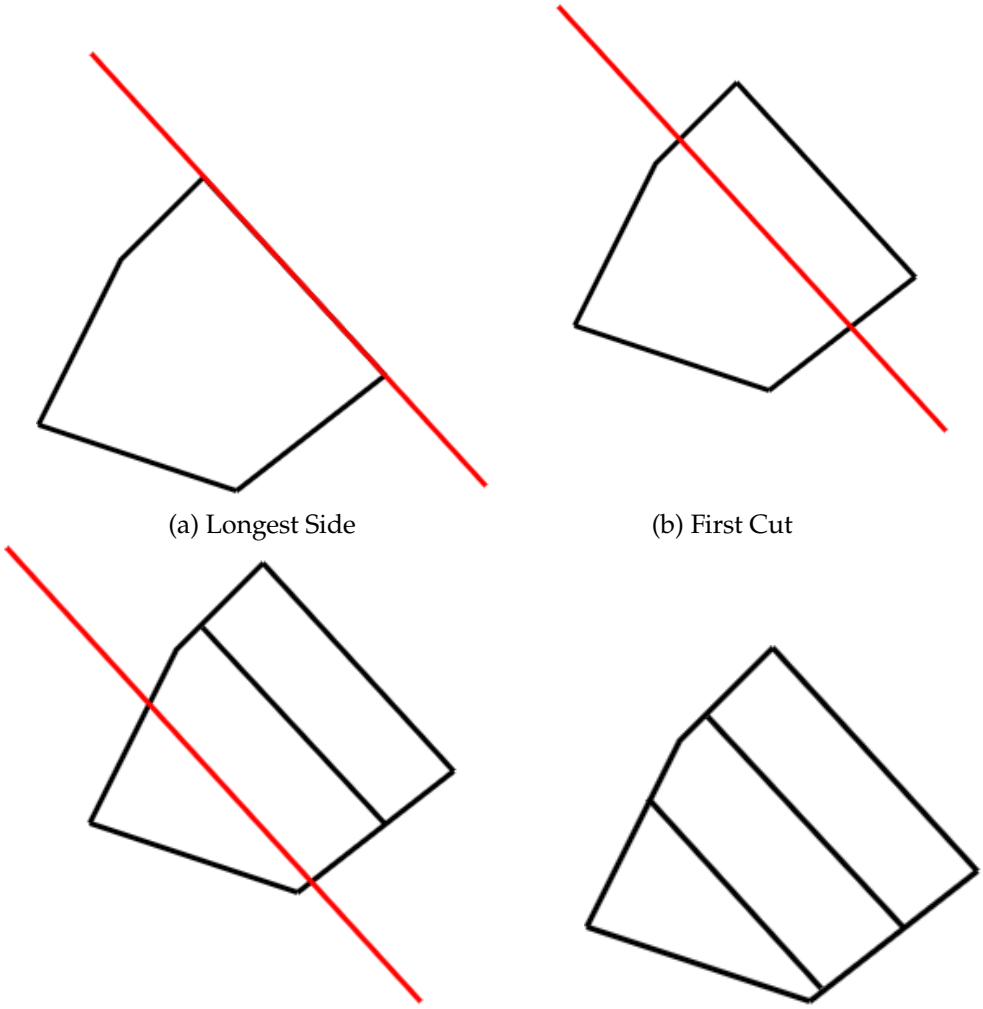
The second strategy, though simpler, offers an effective alternative. It involves identifying the longest side of the polygon and the vertex furthest from it. Subsequently, a parallel line to the longest side is incrementally moved along the perpendicular line from this side to the furthest vertex. This moving line serves as the cutting line for the division.

While this method is limited to convex polygons, its implementation is notably straightforward. Moreover, it boasts a superior time complexity due to the efficient distance calculation for displacing the cutting line. Utilizing binary search for this purpose results in a time complexity of O(log n), enhancing the overall efficiency of the division process. Despite its limitation to convex polygons, the simplicity and efficiency of this approach make it a viable option in many scenarios.

Thus, the second strategy was selected. The algorithm initiates by identifying the longest side of the polygon, followed by determining the vertex furthest from this side. Subsequently, a line parallel to the longest side is shifted towards the furthest vertex until the area enclosed between the longest side and the intersection of the cutting line with the polygon equals the percentage of the total area for each drone.This percentage is calculated by dividing the chosen metric's value for a specific drone by the sum of that same metric across all drones. For example, if endurance is the selected metric, the percentage would be determined by dividing the endurance of the individual drone by the total endurance of all the drones combined.

This approach prioritizes the longest side for division, as it yields the longest possible cuts across the polygon. Moreover, since the scanning lines will run parallel to these cuts, it results in the largest possible scanning lines. This is crucial because drones collect useful data when moving in straight lines, minimizing the number of turns necessary to scan the entire polygon. By maximizing the length of the scanning lines, we optimize the efficiency of data collection, enhancing the overall performance of the system.

At the end of this step each drone will have an assigned piece of the mission area for which it will be responsible and will effectively scan that area.



(a) Longest Side (b) First Cut

(c) Second Cut (d) Divided Polygon

Figure 6.5: The steps taken by the division algorithm

6.4.3 Scanning Line Creation

The next step involves creating scanning lines for each drone. These lines traverse the polygon parallel to its longest side, guiding the drone as it conducts its scans. The lines are spaced to ensure a slight overlap between the scans, while also maximizing the area covered by the drone with each pass.

When considering the area scanned, the width is the primary measure of interest, as it determines the maximum width of the scanning path. However, the area scanned by each payload varies depending on its type, as different payloads operate differently.

For cameras, the scanned area is computed akin to the base of a pyramid, where the height of the drone represents the height of the pyramid. The dimensions of the base of the pyramid can be calculated as follows: Assuming H represents the height of the drone, $Hfov$ denotes the horizontal component in radians of the FOV of the camera, and $Vfov$ indicates the vertical component in radians of the camera's FOV.

$$length = \text{height} \times \tan\left(\frac{\text{Vfov}}{2}\right) \times 2 \tag{6.1}$$

$$width = \text{height} \times \tan\left(\frac{\text{Hfov}}{2}\right) \times 2 \tag{6.2}$$

This calculation determines the scanning area for all types of cameras. However, for Down Facing Sonar, Forward Facing Sonar, Multi Beam Echo Sounder, and Single Beam Echo Sounder, the equations are slightly different, with the height parameter replaced by the range of the sensor. The equations are as follows:

$$length = \text{range} \times \tan\left(\frac{\text{Vfov}}{2}\right) \times 2 \tag{6.3}$$

$$width = \text{range} \times \tan\left(\frac{\text{Hfov}}{2}\right) \times 2 \tag{6.4}$$

Next, the most complex among them is the Side Scanning Sonar. Unlike the others, its range doesn't signify the depth of the scan but rather the slant range, meaning the diagonal distance as shown in Figure 6.6.

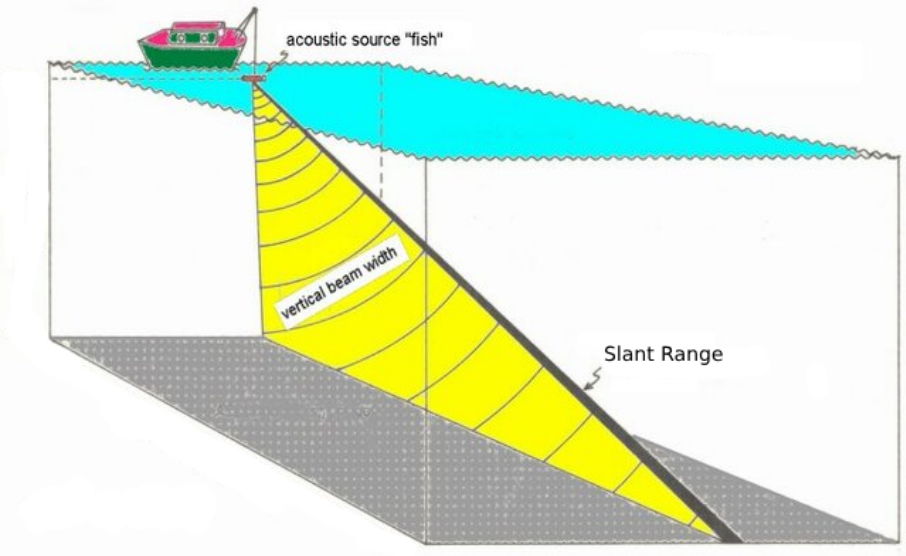


Figure 6.6: Side Scanning Diagram

Therefore, utilizing Pythagoras' theorem once more, with the hypotenuse and the Vertical Beam Width (VBW), which represents the angle in radians between the vertical line perpendicular to the ocean floor and the limit of the scanning area, we can compute the width of the scanning area using the following method:

$$width = \text{Slant Range} \times \sin(\text{VBW}) \times 2 \tag{6.5}$$

Lastly, the only ones remaining are the rotational sensors. The area scanned by these can be conceptualized as a cylinder with the drone at its center. Therefore, the width of

the area can be computed as twice the range of the sensor. This principle applies to both the Rotational Lidar and Radar.

$$width = \text{range} \times 2 \tag{6.6}$$

From this, the algorithm can calculate the width of the scanning area for any drone and payload combination. With this width, we can determine the distance between every scanning line. It's crucial to ensure that each scan overlaps slightly with the neighboring scans. This precaution ensures that even minor deviations of the drone won't leave gaps in the final scan. Additionally, overlapping scans facilitate piecing together the scanned zone to guarantee full coverage. To achieve this, we ensure a 5% overlap of the scan width on each side of the scan.

When positioning the scanning lines, the first is situated at half the ScanWidth from the starting side of the polygon. Subsequently, each line is placed a ScanWidth apart until reaching the end of the polygon. These scanning lines intersect the polygon and are cropped to fit within its boundaries, resulting in a pattern similar to the one in Figure 6.7

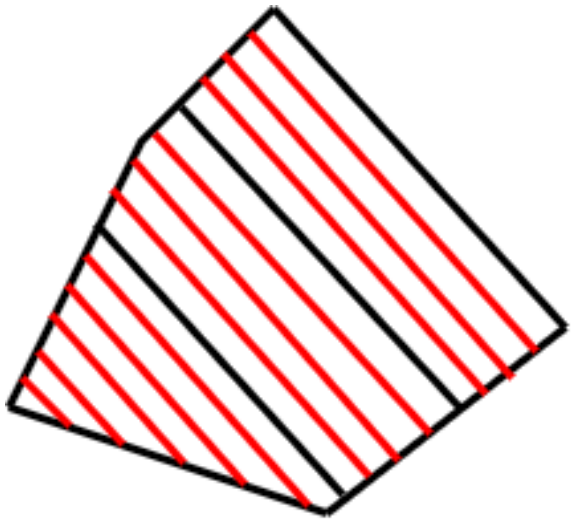


Figure 6.7: Scanning Lines Generated for the Polygon

6.4.4 Scanning Line Connection

With the scanning lines plotted, we are left with the essence of the drone paths, but it is necessary to connect them. This can be achieved in three different ways, depending on the distance the drone needs to travel to execute the 180° turn between the two parallel lines.

During the planning of this module, the intention was to dynamically adjust the curve to minimize the distance required for executing a 180° turn, taking into account the turn rate of each drone. However, this proved to be challenging as the library being utilized offered only rudimentary support for curved shapes and was computationally intensive.

As a compromise, three types of turns were established for the drones based on the diameter of their minimum turning radius between the two lines.

To calculate the minimum turn diameter, we first determine the distance required for the 180° turn as follows: From the drone, we extract the movement plan related to scanning, obtaining the velocity (Velocity) at which the drone will be traveling and its maximum turn rate in degrees per second (TurnRate). Using these parameters, we compute the minimum distance needed to execute the turn as follows:

$$minTurnDistance = \frac{180^\circ}{\text{TurnRate}} \times \text{Velocity} \tag{6.7}$$

From the minimum turning distance, we can calculate the minimum turn diameter as shown below:

$$minTurnDiameter = \frac{\text{minTurnDistance} \times 2}{\pi} \tag{6.8}$$

Now, there are three possible turns:

1. Two 90° turns, which are only performed by drones with exceptionally high turn rates.

2. A constant-angle 180° turn, resembling a half circle with a diameter equal to the distance between the lines. This maneuver is typically executed by most UXVs.

3. A teardrop/muffin top curve, where the drone executes an almost complete circle, with the radius determined by the maximum turn rate of the drone.

If the diameter of the turn is less than a quarter of the distance between the lines, the drone will not curve and will instead perform a 90° turn. This maneuver is typically executed by quadcopters, as their high turn rate enables them to sustain sharp turns at high speeds, thus facilitating computation. The turn is as follows:

When the diameter is larger than a quarter of the distance between the lines but smaller or equal to said distance, the constant-angle turn is chosen. It appears as follows in Figure 6.9

When the diameter exceeds the distance between the lines, the teardrop/muffin top curve is selected, as illustrated in Figure 6.10.

The library in use, NetTopologySuite, offers limited support for operations involving curved shapes. It provides only a single function capable of creating curved shapes, restricted to circles. This limitation becomes particularly evident in the case of teardrop curves. The original intention was to create a smooth connection between the scanning line and the circle, resembling a teardrop shape. However, due to the library's constraints, the resulting curves more closely resemble muffin tops. Achieving the desired teardrop shape would require dynamically generating curves with varying diameters, a task beyond my current understanding of geometry. Consequently, the lines connect directly to the circular shape.

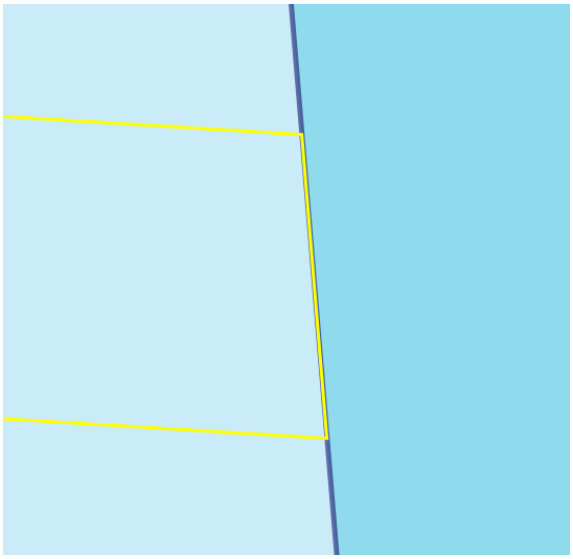


Figure 6.8: Straight Turn

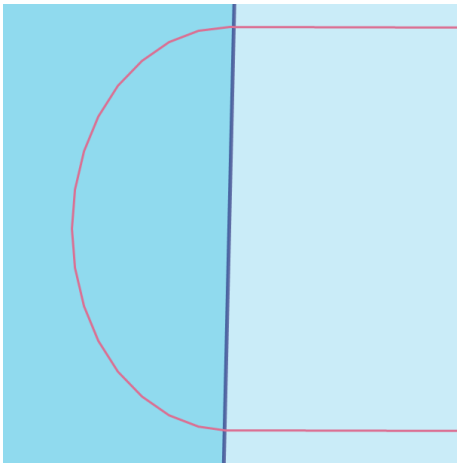


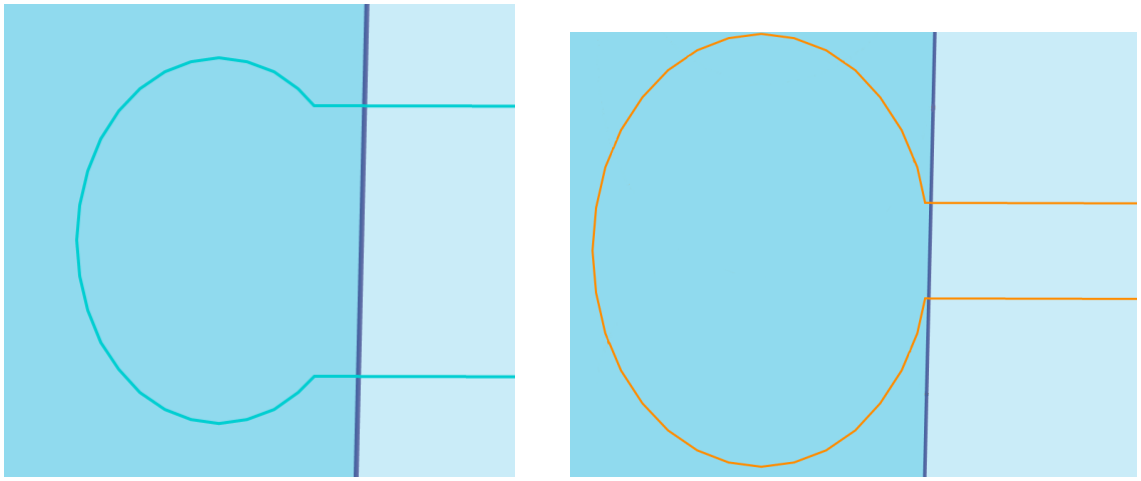
Figure 6.9: Half Circle Turn

The final result of this computation will the scanning path of the drone for the whole area it was assigned to as well and all the extra turns it will have to perform to go from one scanning line to the next. During the creation of these lines one thing that is not taken into account is the length of the path and consequently if the drone has the needed endurance to perform such a path, this and possible roundtrips the drone might need to do to recharge are handled on the next step.

However, this step differs in the way it deals with creating the paths for the roundtrips based on the two previously mentioned philosophies Collision Avoidance and Collision Remediation.

6.4.5 Collision Avoidance

Even before assessing the feasibility of the drone's designated path, a preliminary check is conducted to ensure the drone can safely traverse from the base station to the beginning



(a) High turn Rate Full Circle Turn (b) Low Turn Rate Full Circle Turn

Figure 6.10: Difference TurnRates Effect on the Full Circle Turn

of the scanning area. However, this check poses a challenge: How can we prevent drones departing from the base station from colliding en route to the scanning area?

To address this concern, we've devised the concept of a "Drone Highway." As can be seen in Figure 6.11, each drone is allocated a dedicated lane that remains independent of intersecting with other drone paths. This systematic approach ensures collision-free navigation and allows for smooth and efficient movement from the base station to the scanning area.

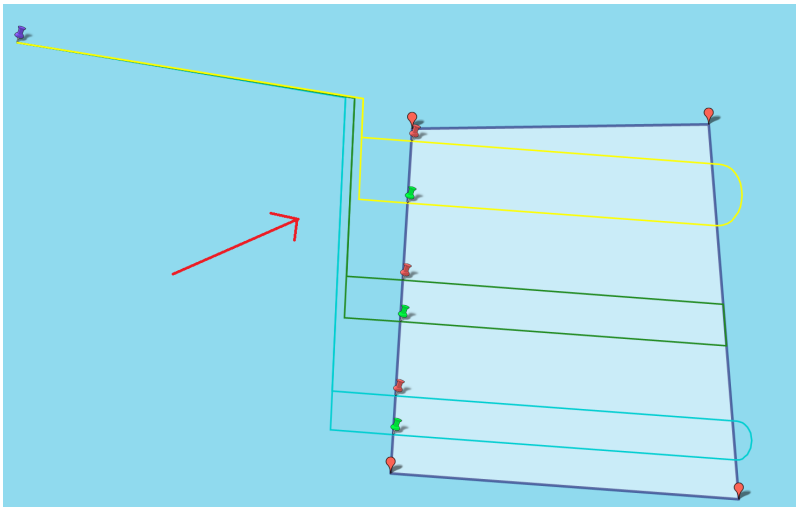


Figure 6.11: Drone Highway

This concept necessitates assigning the drone starting closest to the base station to the lane nearest the polygon, ensuring non-intersecting paths for different drones. This presented an intriguing challenge, as numerous factors influenced the drone-lane arrangement, including the polygon's orientation and the base station's positioning relative to it.

To ensure a safe return from the mission area, the initial check assesses the drone's capability to travel from the base station to the scanning area and back. This involves generating round-trip paths and calculating their respective travel times using the drone's movement plan from point A to point B. If the calculated time falls within the drone's total endurance, it passes the check. However, if it fails, the entire team is rejected.

Once the first check is cleared, the drone's scanning path undergoes a similar evaluation. If the drone can complete the entire path and return to base without requiring a roundtrip for recharging, the path remains unchanged. However, if a roundtrip is necessary, the algorithm identifies the farthest point along the path that the drone can reach while ensuring a safe return to base. This point becomes the location for the roundtrip. The roundtrip involves navigating back to the Drone Highway, returning to the base station for recharging, then resuming the scan from the midpoint.

This approach introduces minor adjustments to the original path, seamlessly incorporating the required roundtrips for the drone to effectively execute the scan.

The path derived from this process represents the definitive route provided to the drones and displayed to the user. It encompasses the journey from the base station to the scanning area, the intricate scanning path with its lines and connecting curves, as well as any essential roundtrips for recharging, culminating in the return to base. This comprehensive path offers a clear visualization of the drone's entire mission, ensuring efficient execution and effective communication with the user.

6.4.6 Collision Remediation

During the trials and testing of the algorithm, it became apparent that the strategy of having each drone follow a predefined "drone highway" to navigate to and from the mission area was not optimal. Additionally, many missions involve only one drone operating in each environment, rendering path intersections irrelevant since, for instance, an aerial drone will not scan at the same height as a surface drone.

In response to these observations, a decision was made to revise the policy governing drone path calculations. Instead of avoiding potential path intersections between drones, the new approach focuses on planning the shortest path for each drone individually and then checking for collisions during the execution of these paths.

The collision check involves verifying if the paths of any two drones operating within the same environment intersect. If an intersection is detected, the algorithm calculates the exact times at which the drones cross the intersection point and assesses whether a collision would occur. This is easily done since we know the distance the drone will travel and the speed it will travel at.

To resolve detected collisions, the path of the slower drone is adjusted. This drone is directed to perform a circular maneuver to delay its arrival at the intersection point, thereby avoiding the collision. This maneuver is the smallest full circle turn the drone can perform at that speed and this is achieved by using the previously implemented full circle

turn 6.10b. The choice of the slower drone is also strategic as its circle turn has a smaller radius, potentially reducing the chance of creating additional collision points.

There is a possibility that adding a turn could inadvertently cause another collision, potentially leading the algorithm into an infinite loop where it continually resolves collisions without finding a stable solution. However, this issue has never occurred during our testing. In practice, collisions are quite rare, as drones typically cross each other's paths only when traveling to or from the base, and this usually happens at different times.

An alternative solution could involve temporarily slowing down one of the drones to avoid a collision. However, this approach introduces the challenge of determining how slow a UAV can travel without losing altitude, which would add another layer of complexity to an already intricate function.

After modifying the path of the slower drone, the entire process is restarted. Since one path has been altered, it must be rechecked to ensure that all collisions are avoided and the updated path complies with the revised policy.

This was the final step in the Path Generation module, resulting in a comprehensive list of drone paths. Each path consists of segments and corresponding Movement Profiles. The segments are defined by a series of coordinates that outline the route the drone should follow, while the Movement Profiles specify the speed and altitude at which the drone should travel along each segment.

6.5 Path Evaluation

The objective of the Path Evaluation module is to establish a metric for comparing different missions, while also calculating the duration and cost associated with each mission. This module provides a standardized way to assess and compare mission performance, ensuring that both time and cost factors are accurately evaluated.

With this information, calculating both the duration and cost of the mission becomes straightforward. For each drone path, we determine the length of every segment, then divide it by the speed specified in the segment's Movement Profile to find the time required for the drone to travel that segment. Summing the time it takes to complete all segments gives the total mission duration for that drone.

The overall mission time is determined by the slowest drone, as all drones must finish for the mission to be considered complete. As for the cost, it is calculated based on the time required for each drone to complete the mission. The cost is simply the mission duration multiplied by the operating cost per hour of the drone and its payload.

The most complex step in the Path Evaluation module is normalizing both the cost and time, allowing us to calculate mission performance using a single value.

The importance of the Cost and the Duration of the mission is set by the user when he inputs the mission parameters as shown in Figure 4.3. Allowing users to input the percentage weight of each metric for the mission being planned. One side of the slider represents cost, while the other represents time. By adjusting the slider, users can combine

the cost and time metrics according to their selected percentages, creating a new, combined metric to guide mission planning. This functionality provides a flexible and user-centric approach to optimizing mission parameters.

The challenge with this approach lies in normalizing both metrics. The cost values are typically much larger than the possible values for the mission duration. Without normalization, if we were to multiply each metric by their respective percentages (e.g., 50/50), the cost would disproportionately influence the combined metric. Therefore, normalization is essential to ensure that both cost and time are equally represented when combined, providing a balanced evaluation for mission planning.

Normalization adjusts data values measured on different scales to a common scale, facilitating meaningful comparisons and improving algorithm performance.

To normalize a value we must know the bounds by which it is limited meaning the maximum and minimum values for the cost and time the mission takes. The normalized cost, C_{norm}, and the normalized time, T_{norm}, can be calculated as follows:

$$C_{\text{norm}} = \frac{C - C_{\text{min}}}{C_{\text{max}} - C_{\text{min}}}$$

$$T_{\text{norm}} = \frac{T - T_{\text{min}}}{T_{\text{max}} - T_{\text{min}}}$$

However, achieving this would require the algorithm to calculate the performance for each possible team combination, which contradicts our previous approach. In our earlier methods, we aimed to avoid exhaustive searches of the entire search space to ensure faster and more effective results. This comprehensive calculation would undermine the efficiency and speed that our previous strategies provided.

Therefore, we opted for a compromise. For the lower bound, we used a value guaranteed to be lower than any possible time and cost values. This is crucial because, without this precaution, the metric could produce negative values, making it impossible to rank the solutions accurately. To ensure the metric remains positive and meaningful for all potential solutions, we chose 0 as the minimum bound for both cost and time.

As for the maximum value, the algorithm calculates a mission using all capable drones, which typically results in the worst solution in terms of both time and cost. However, in the rare cases where this isn't true, the value produced by normalization might exceed 1, indicating a poor rating since the objective is always to minimize the search metric. This can slightly skew the distribution of values, making them higher for metrics where the maximum value was underestimated.

This has a minor impact on the overall metric. If the user is dissatisfied with the result, they can always adjust the slider slightly toward their preferred direction. This flexibility allows users to fine-tune the evaluation according to their priorities and preferences.

The Path Evaluation module serves as the final stage of the algorithm. Based on its assessment, the planned missions will either be sent back to the Team Generation module for further refinement or the final solution will be presented to the user.

7

Parameter Optimization and Comparative Analysis

In this chapter, we delve into the intricate processes of optimizing algorithmic performance and strategic planning for drone-based missions. We begin with **Parameter Tuning the Genetic Algorithm**, where we explore the essential adjustments needed for selection, crossover, and mutation phases to achieve optimal genetic algorithm configurations. This is followed by **Parameter Testing Tabu**, which focuses on fine-tuning the Tabu Search algorithm by adjusting key parameters such as the total number of iterations and the size of the Tabu List, rigorously testing these configurations across various missions to enhance problem-solving efficacy. We then transition to **Comparing Team Generation Approaches**, where we evaluate the performance of the Tabu Search algorithm against the Genetic and Exhaustive implementations, including an advanced version utilizing an Estimation function for refined decision-making. Next, **Path Generation Approach Comparison** examines the limitations of predefined navigation strategies and explores more effective alternatives for path optimization. Finally, in **Comparative Analysis of Area Decomposition Strategies**, we analyze four distinct metrics—Endurance, Area Scanned Per Second, Area Scanned Per Euro, and Equal Division—to determine which strategy best balances time efficiency, cost management, and resource allocation, thereby guiding the selection of the most effective approach for diverse mission requirements.

7.1 Parameter Tuning the Genetic Algorithm

In this section, we explore the crucial process of parameter tuning for the genetic algorithm, with a focus on the selection, crossover, and mutation phases. To identify the optimal configuration, various strategies and parameters are systematically tested.

During the selection phase, different strategies are evaluated to determine which individuals from the current population will contribute to the next generation. The strategies tested include EliteSelection, which ensures the survival of the fittest individuals, RouletteWheelSelection, which introduces probabilistic selection based on fitness, and

TournamentSelection, which selects the best individuals from randomly chosen subsets.

In the crossover phase, multiple methods are tested to combine genetic material from selected parents. These include OnePointCrossover and TwoPointCrossover, which involve swapping segments of parent chromosomes, ThreeParentCrossover, which combines genes from three parents, UniformCrossover, which allows for a high degree of gene mixing, and VotingRecombinationCrossover, which uses a voting mechanism to decide each gene in the offspring.

For the mutation phase, UniformMutation is employed, with various mutation probabilities (0.01, 0.05, 0.1, and 0.2) being tested to determine the optimal rate of genetic variation. Additionally, the impact of elitist reinsertion is evaluated by testing both true and false settings to assess how retaining the best individuals influences the algorithm's performance.

The goal of this parameter tuning is to identify the combination of selection, crossover, and mutation strategies that minimize the objective function. This systematic exploration will guide the choice of the most effective parameters for achieving optimal results in the genetic algorithm.

Due to the complexity and interdependency of these parameters—where changing one can impact the effectiveness of others in achieving optimal solutions—we opted to run all possible combinations of these parameters.

To evaluate the performance of each combination, a test was devised to closely mimic real use cases for this application. The following aspects were considered:

- **Mission Area Shape** - Since the application supports any convex shape as a mission area, a variety of shapes were tested to ensure comprehensive assessment.
- **Mission Area Size** - The size of the mission area affects distribution strategies. Larger areas are more complex for the algorithm to handle as they typically require more drones due to the increased scanning area.
- **Mission Area Location Relative to Shoreline** - The position of the area relative to the shoreline significantly impacts the drones' capabilities and the scanning path generation. It is crucial to test areas that are inside, outside, or intersecting the shoreline.
- **Number of Available Drones** - The number of available drones influences the range of possible team configurations the algorithm must evaluate.
- **Mission Requirement Variability** - The mission requirements dictate how the mission will be performed and which drones are suitable, making this an important variable to consider.

Following these five testing requirements, a series of tests were devised involving three different mission areas:

- The first mission area is a small triangle located inside the shoreline, where the mission is an aerial scan using only UAVs.
- The second mission area is a medium-sized rectangle intersected in the middle by the shoreline.
- The third mission area is a large pentagon fully overwater, away from the shoreline.

For the second and third mission areas, three different missions will be performed:

- A surface scan using UAVs and USVs.
- A surface and underwater scan using UAVs, USVs, and UUVs.
- An underwater scan using only USVs and UUVs.

This setup results in a total of seven individual tests. Next, these seven tests will be performed with a varying number of drones, starting at 12, which equates to one drone of each type. The number of drones will then increase in increments of six, distributed evenly among the drone types, until it reaches 48 total drones, or four drones of each type. Therefore, the seven tests are repeated another seven times, for the following drone counts: {12, 18, 24, 30, 36, 42, and 48}. This results in a total of 49 tests.

As for the combinations of the hyperparameters, there are 5 selection strategies, 6 crossover methods, 1 mutation strategy, 4 mutation probabilities ({0.01, 0.05, 0.1, 0.2}), and the option to apply elitist reinsertion or not. All these together sum up to 240 different combinations. Each combination runs the 49 tests twice, once for minimizing mission cost and once for minimizing mission time.

The results from the 49 tests are then summed to provide a metric that allows for direct comparison between the different setups for the genetic algorithm. For our application, the optimal algorithm will be the one that performs best in both minimizing time and minimizing cost, as most missions planned will use both metrics via a slider implementation. Therefore, to compare the results from the two different metrics, the sums from each were normalized and added together to create a single performance value called the evaluation metric.

In Table 7.1, we present the results for the best 10 setups, including the Evaluation metric and the normalized runtime sum for both tests. In the table, SUS stands for Stochastic Universal Sampling.

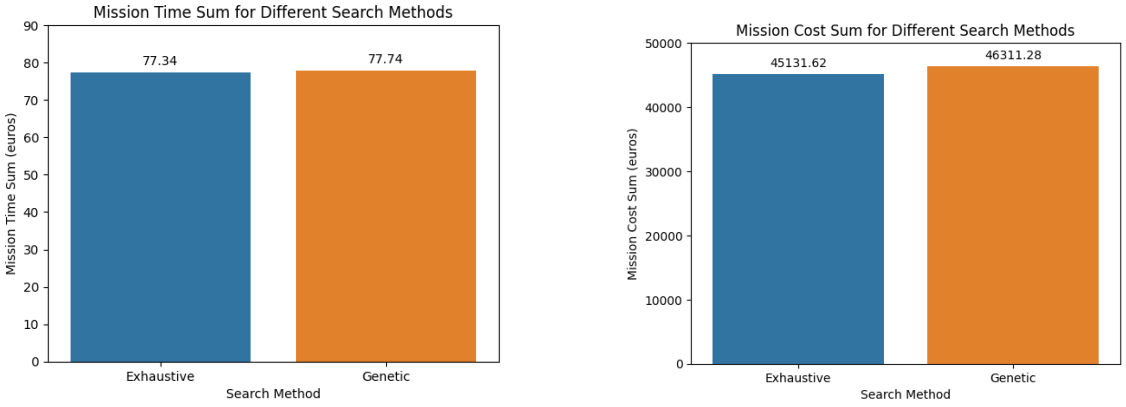
From these results, we observe that the top three setups employ the Stochastic Universal Sampling method for selection. The two best configurations are nearly identical in metric evaluation, making the deciding factor the time they take to reach a solution. Thus, the setup using Stochastic Universal Sampling with Uniform Crossover, a mutation probability of 20%, and no Elitist Reinsertion emerges as the optimal hyperparameters for the genetic algorithm tailored to this application.

Evaluation Metric	Runtime	Selection	Crossover	Mutation %	Has Elitist Reinsertion
0.003319	0.93334	SUS	TwoPoint	0.2	False
0.003374	0.89830	SUS	Uniform	0.2	False
0.004803	1.13300	SUS	TwoPoint	0.1	False
0.004813	1.12074	Roulette Wheel	Uniform	0.1	False
0.005927	1.06845	Roulette Wheel	Uniform	0.2	True
0.009845	1.15484	Tournament	TwoPoint	0.2	False
0.010305	1.22519	Roulette Wheel	TwoPoint	0.2	True
0.011689	0.93567	Tournament	Uniform	0.2	False
0.012794	0.83070	SUS	Uniform	0.2	True

Table 7.1: Top 10 Best performing hyperparemeters for the Genetic Algorithm

To validate the quality of the genetic algorithm, we compared its performance with the exhaustive approach. The testing conditions were similar to those used for selecting the genetic algorithm's hyperparameter combination. Seven different missions were tested; however, instead of incrementing by six drones per iteration, the increment was one drone per iteration. This approach aimed to better visualize the impact of the increasing number of drones on the algorithm's performance. For the exhaustive approach, the performance drop is drastic, making it impractical to test more than 18 drones within a reasonable timeframe.

The comparison results are represented in Figure 7.1.



(a) Minimizing time

(b) Minimizing cost

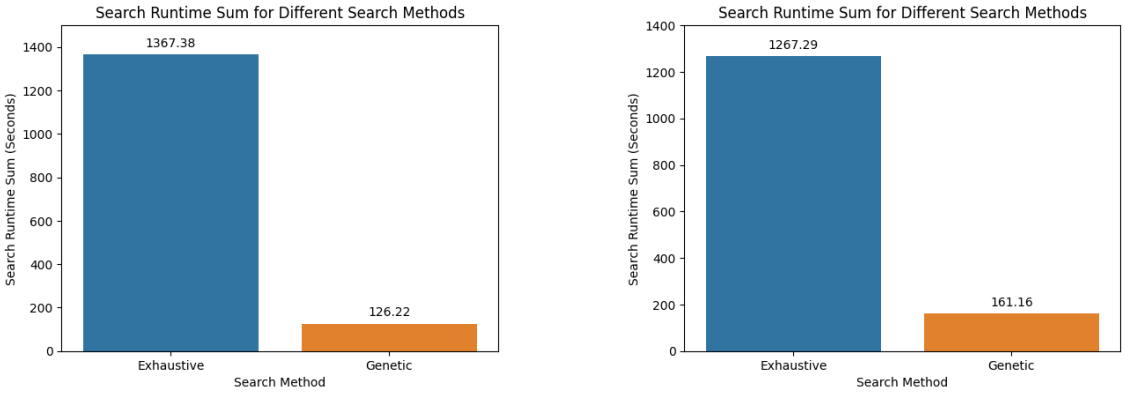
Figure 7.1: Performance comparison between Exhaustive and Genetic algorithms

In both plots, we observe that the genetic algorithm achieves results very close to those of the exhaustive approach, with an error of only 0.52% for time-based missions and 2.61% for cost-based missions. These errors are relatively small given the complexity of the search space.

However, it would be interesting to investigate how this error varies as the number of drones increases. Unfortunately, this analysis is not feasible with the exhaustive search

due to its prohibitive time requirements.

It was also crucial to compare the runtime of each approach in the conducted tests. Comparing the runtime of the genetic algorithm with that of the exhaustive approach in these tests yielded the results depicted in Figure 7.2.



(a) Minimizing time

(b) Minimizing cost

Figure 7.2: Runtime comparison between Exhaustive and Genetic algorithms

From these results, it is evident that the genetic algorithm offers a significant improvement in runtime performance. When evaluating by time, the genetic algorithm is 90.77% faster than the exhaustive approach, and when evaluating by cost, it is 87.28% faster.

Therefore, the genetic algorithm proves to be a highly efficient and effective alternative to the exhaustive search method, particularly for larger-scale problems where exhaustive search becomes impractical. Despite a minimal decrease in performance accuracy, the substantial gains in computational efficiency make the genetic algorithm a superior choice for mission planning. The slight trade-off in accuracy, with an error of only 0.52% for time-based missions and 2.61% for cost-based missions, is acceptable given the complexity and size of the search space.

Furthermore, the scalability of the genetic algorithm allows it to handle an increasing number of drones more effectively than the exhaustive approach. As mission requirements grow and the number of variables increases, the genetic algorithm's ability to quickly converge on a near-optimal solution becomes even more valuable. This makes it an indispensable tool for applications requiring efficient and reliable mission planning with large datasets.

7.2 Parameter Tuning Tabu Search

In this section, we focus on the parameter tuning process for the Tabu Search algorithm, specifically targeting two crucial parameters: the total number of iterations and the size of the Tabu List. To optimize the algorithm's performance, we adopted a systematic approach similar to that used for the genetic algorithm. Each combination of parameters was rigorously tested across seven different missions, with variations in the number of

drones used. This methodical exploration aims to identify the most effective configuration for the Tabu Search, ensuring robust and efficient problem-solving across various scenarios.

For the number of iterations, values ranging from 10 to 90 in increments of 10 were tested. For the tabu list size, the values 4, 8, 10, 12, 14 and 16 were used. All possible combinations of these parameters were tested by running the algorithm with varying numbers of drones across the seven test missions. The results of these tests were aggregated, and the top 10 best combinations for optimizing mission duration and mission cost are presented in Table 7.2.

Time Sums (sec)	Max Iterations - Tabu Size	Runtime (sec)	Cost Sums (€)
144.549065	20-12	48.136629	78927.478323
144.814936	60-04	134.516626	78927.478323
145.012002	70-08	157.817074	78927.478323
145.066667	40-08	87.981828	78927.478323
145.116227	70-04	154.589894	78927.478323
145.126314	60-08	133.524531	78927.478323
145.126874	50-08	111.412530	78927.478323
145.156567	60-10	139.546367	78927.478323
145.161367	40-12	109.116152	78927.478323
145.163487	60-12	135.804857	78927.478323

Table 7.2: Best 10 Mission Sum times with varying parameter Tabu Searches

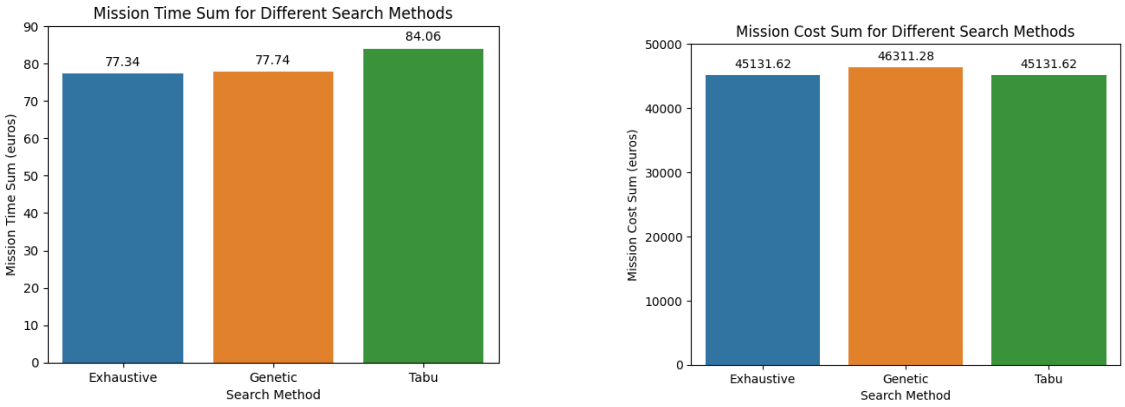
Paying attention to the scale, we observe that for the time sums are very similar between the different tabu algorithm setups, varying by less than 1 second from each other and on the last colum we can see that all of the parameters produced the same result for the mission cost sum. Therefore, to choose the best combination, we must consider the runtimes of the different setups, as the solution must not only perform well but also do so as quickly as possible.

In the runtime colum, we can clearly see the difference in runtimes for each combination. As expected, the higher the number of iterations, the longer it takes to reach a solution. This makes the combination of 20 total iterations and a tabu list size of 12 the best performing solution in terms of both performance and efficiency. Thus, we selected the 20-12 configuration as the best parameter set for this application of the Tabu Search.

7.3 Comparing Team Generation Approaches

Having fully implemented the Tabu Search algorithm and meticulously fine-tuned its parameters, we conducted a thorough comparison of its performance against our previous two implementations. The results of this comparative analysis are visually presented in Figure 7.3.

In Figure 7.3a, we observe that the performance of the Tabu Search in minimizing mission time is worse compared to the previous two implementations, exhibiting an 8.68% error relative to the exhaustive approach. However, Figure 7.3b, shows that the Tabu



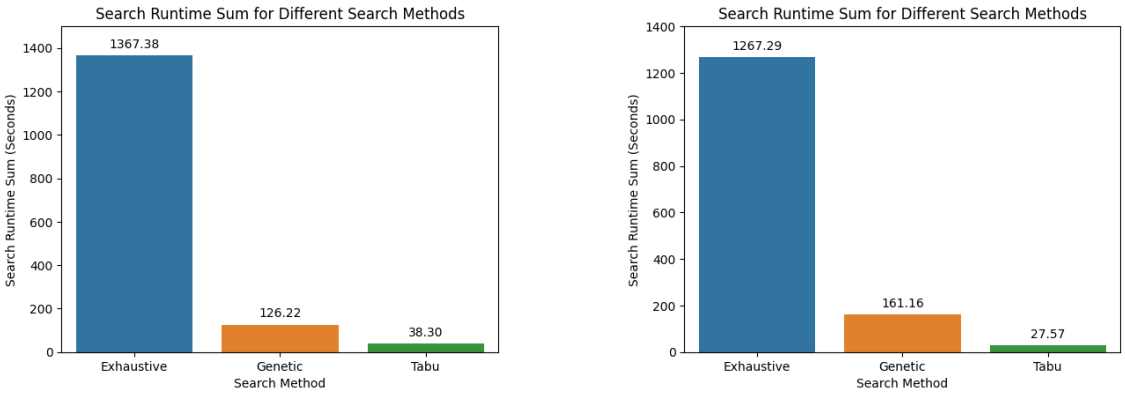
(a) Minimizing time

(b) Minimizing cost

Figure 7.3: Performance comparison between Exhaustive, Genetic and Tabu algorithms

Search matches the performance of the exhaustive approach when minimizing mission cost.

When considering runtimes, the results of each approach are displayed in Figure 7.4. It is evident that the Tabu Search is the fastest of the three, being 97.20% faster than the exhaustive approach when minimizing time and 97.82% faster when minimizing cost.



(a) Minimizing time

(b) Minimizing cost

Figure 7.4: Runtime comparison between Exhaustive, Genetic and Tabu algorithms

Since there were no clear winners between these differentiated approaches, additional testing was necessary, particularly to evaluate how these algorithms handled searches with a larger number of drones. To this end, a test was developed using the same seven mission scenarios, but the quantity of drones started at 12 and was incrementally increased by one drone at a time for each type until reaching a maximum of 50 drones.

The results depicted in Figure 7.5 clearly demonstrate that the genetic algorithm significantly outperforms the Tabu Search, showing a 23.30% improvement in adapting to the increase in drone quantities. This makes the genetic algorithm the superior choice when minimizing Time.

In this graph, we can also easily visualize the decrease in the sum of mission times

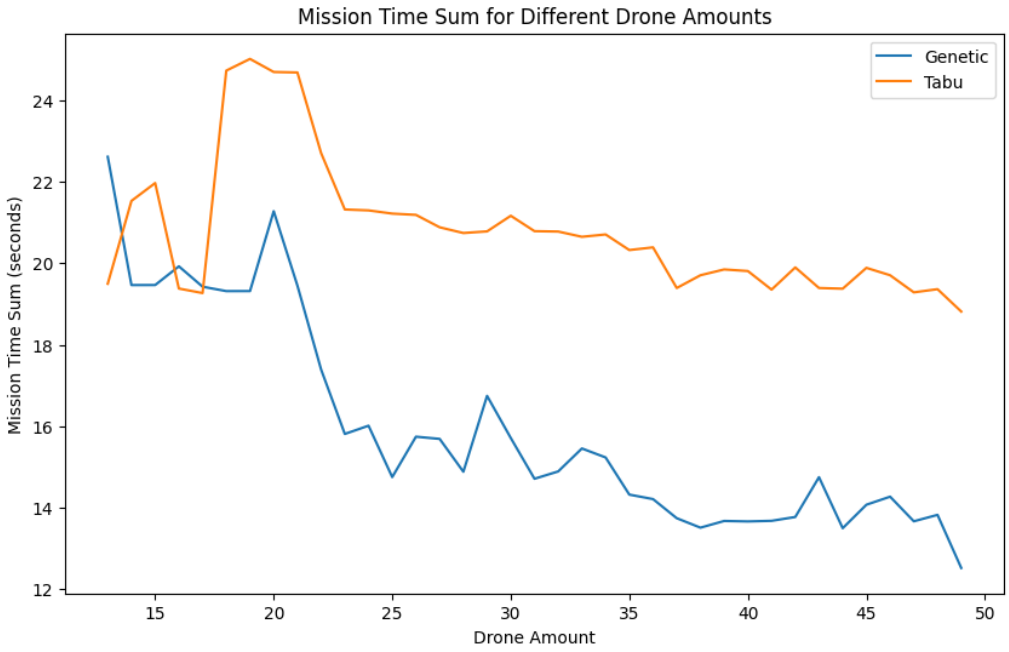


Figure 7.5: Performance Comparison Between Tabu Search and Genetic Algorithm with Varying Drone Quantities for Time Optimization

as the number of drones increases. This trend is expected, as a larger number of drones working on scanning an area results in faster completion times. A deeper analysis of the data could reveal specific impacts of different drone types on various missions. For instance, an increase in UAVs would likely reduce mission time for aerial missions but might not affect missions involving underwater scans. This nuanced understanding highlights the importance of matching drone capabilities to mission requirements for optimal performance.

Unexpectedly, as shown in Figure 7.6, the Tabu Search outperforms the genetic algorithm when optimizing for the lowest cost. In this plot, unlike the previous one, the expected result is for the cost to remain stable. Generally, the solution with the lowest cost is the one that utilizes drones in the most cost-effective manner, minimizing the number of drones used and not changing the solution as more drones are added. The Tabu Search maintains this stability, demonstrating a 19.02% improvement over the genetic algorithm.

This outcome presents a challenge, as our goal is to employ a single algorithm for both time and cost optimizations. The current results indicate that neither algorithm is universally superior; the genetic algorithm excels in minimizing mission time, while the Tabu Search is more effective in minimizing cost. To address this, further refinements and hybrid approaches may be necessary to ensure one algorithm can accommodate both objectives effectively.

Additionally, a deeper analysis into the specific factors contributing to the performance of each algorithm could provide insights for improvement. For example, understanding why the genetic algorithm adapts more effectively to increasing drone numbers while the

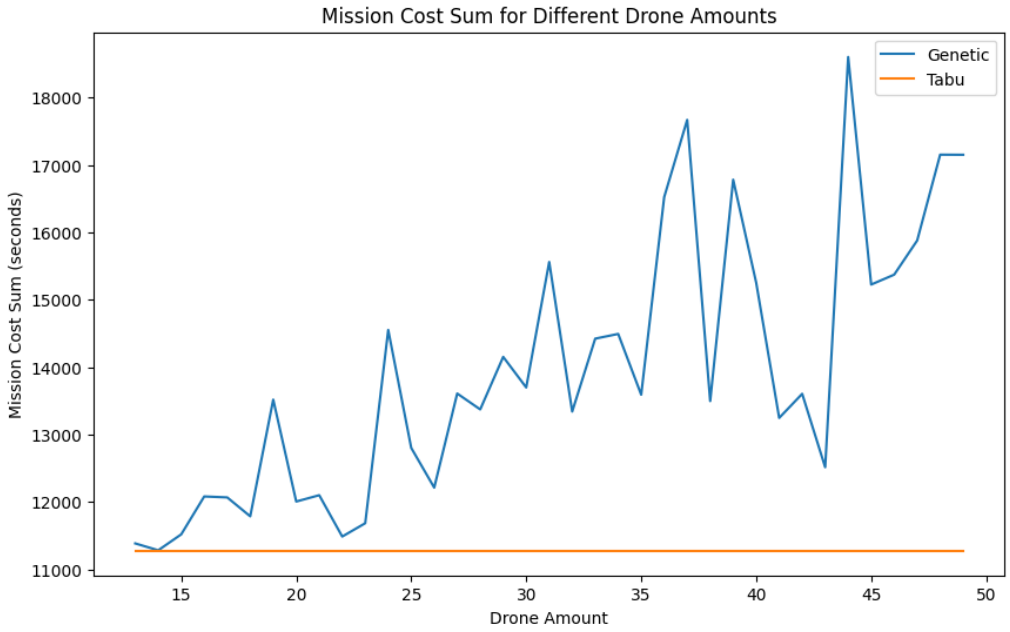


Figure 7.6: Performance Comparison Between Tabu Search and Genetic Algorithm with Varying Drone Quantities for Cost Optimization

Tabu Search maintains cost stability could inform the development of a more versatile solution. However, limited as we were by the time frame of this thesis a better analysis could not be conducted.

7.3.1 Tabu Search with Estimation Function

The third and final approach to Team Generation is the Tabu Search algorithm enhanced by the Estimation function introduced in the previous chapter. This implementation utilizes the optimized parameters identified during the fine-tuning of the standard Tabu Search, ensuring consistency and leveraging the most effective settings. By incorporating the Estimation function, this approach aims to refine the decision-making process, enhancing the algorithm's efficiency in generating well-balanced and optimized teams.

The performance of the new implementation was tested using the same conditions as in Figure 7.5. The results presented in Figure 7.7 indicate that the Tabu Search with the estimation function slightly outperforms the standard Tabu Search, reducing the performance gap to 16.98% compared to the genetic algorithm, an improvement over the previous 23.30% gap. This suggests that incorporating the estimation function enhances the efficiency of the Tabu Search, bringing it closer to the genetic algorithm's performance.

Further analysis, reveals that the Tabu Search with the estimation function matches exactly the performance of the standard Tabu Search when minimizing cost, producing the same graph as in Figure 7.6. This indicates that while the new implementation excels in time-based evaluations, it maintains cost performance parity with the traditional Tabu Search.

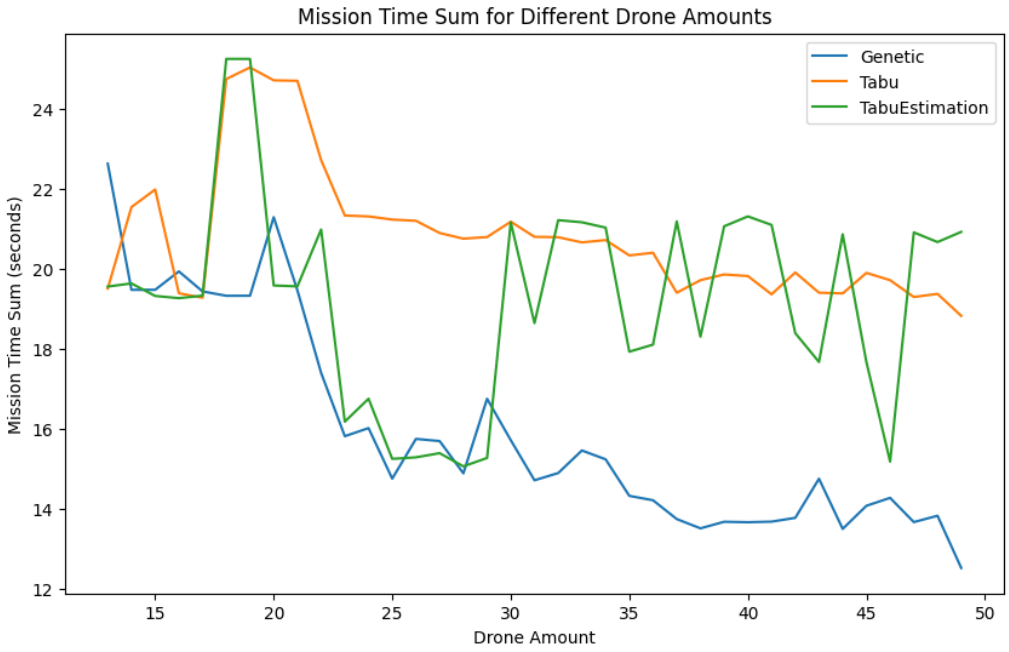
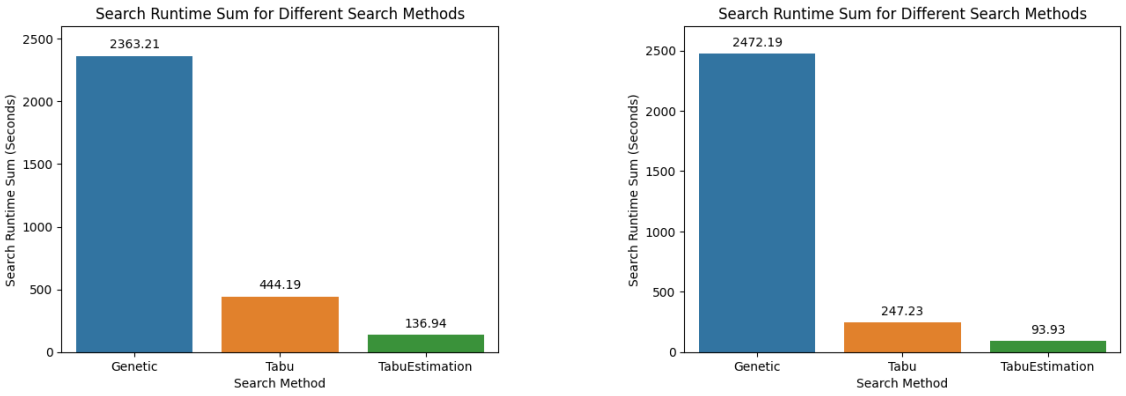


Figure 7.7: Comparison between Genetic, Tabu and Tabu using Estimation Function when minimizing time with varying drone amounts

The time-based evaluation of the Tabu Search with the estimation function is particularly promising. By avoiding the need to calculate paths for every candidate solution, the estimation function significantly improves performance, as shown in Figure 7.8. This substantial performance gain highlights the potential of the estimation function to streamline computations and enhance overall efficiency. Future work could focus on further optimizing this function to bridge the remaining performance gap and potentially develop a unified algorithm that excels in both time and cost optimizations.



(a) Minimizing time

(b) Minimizing cost

Figure 7.8: Runtime comparison between Genetic, Tabu and Tabu with Estimation algorithms

Based on these results, it is clear that a single algorithm cannot currently optimize both cost and time simultaneously. The genetic algorithm excels in minimizing mission time,

while the Tabu Search, particularly with the estimation function, proves more effective in reducing mission cost. The Tabu Search, despite improvements, still struggles with local maxima, especially as the search space expands with increasing drone quantities. This suggests that further refinements to the Tabu algorithm are needed and could possibly enhance its exploration capabilities and potentially address these limitations in future work.

Based on the current findings, the implemented approach uses Tabu Search with an estimation function for missions where cost is of equal or greater importance than time, while the Genetic Algorithm is applied in the remaining cases. This configuration works well when optimizing for either time or cost individually. However, when the objective is to minimize both simultaneously, it may not always yield the best solution.

A more optimal approach, aside from developing a single algorithm that performs well in both optimizations, would be to run both the Tabu Search and Genetic Algorithm for missions near the 50% cost and 50% time importance threshold. By conducting tests to identify the boundaries where running both algorithms adds value versus where it becomes inefficient, the process could be fine-tuned. However, due to time constraints, this implementation and the required testing were not possible within the project's timeframe, and therefore, it was not implemented. This strategy remains feasible, as the runtime of Tabu Search with estimation is negligible compared to the Genetic Algorithm, allowing both to be run without significant time loss.

In conclusion, a hybrid strategy leveraging both algorithms based on mission requirements will provide the most effective solution in the current timeframe. Future research could focus on developing methods to integrate the strengths of both algorithms into a unified approach or improving the Tabu Search to handle larger search spaces more effectively. Such advancements could offer a more comprehensive solution for optimizing both cost and time, potentially leading to better performance across a wider range of mission scenarios.

7.4 Path Generation Approach Comparison

During the trials and testing of the algorithm, it became apparent that the strategy of having each drone follow a predefined "drone highway" to navigate to and from the mission area was not optimal. Additionally, many missions involve only one drone operating in each environment, rendering path intersections irrelevant since, for instance, an aerial drone will not scan at the same height as a surface drone.

In response to these observations, a decision was made to revise the policy governing drone path calculations. Instead of avoiding potential path intersections between drones, the new approach focuses on planning the shortest path for each drone individually and then checking for collisions during the execution of these paths.

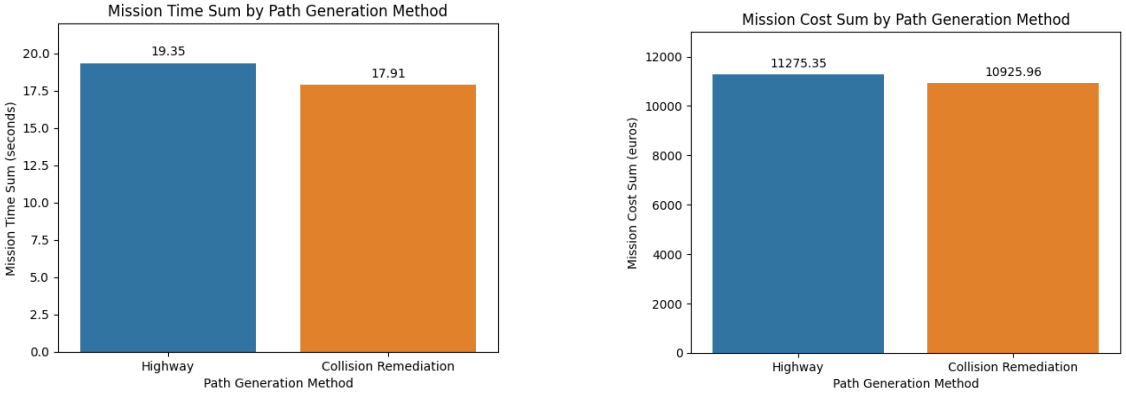
The collision check involves verifying if the paths of any two drones operating within the same environment intersect. If an intersection is detected, the algorithm calculates

the exact times at which the drones cross the intersection point and assesses whether a collision would occur.

To resolve detected collisions, the path of the slower drone is adjusted. This drone is directed to perform a circular maneuver to delay its arrival at the intersection point, thereby avoiding the collision. The choice of the slower drone is strategic: its circle turn has a smaller radius, potentially reducing the chance of creating additional collision points.

After modifying the path of the slower drone, the entire process is restarted. Since one path has been altered, it must be rechecked to ensure that all collisions are avoided and the updated path complies with the revised policy.

To evaluate the impact of these changes on algorithm performance, tests were conducted where the optimal paths for the seven different missions were generated using both the original "drone highway" policy and the new "Collision Remediation" policy. The results of these tests are illustrated in Figure 7.9.



(a) Minimizing time

(b) Minimizing cost

Figure 7.9: Runtime comparison between Genetic, Tabu and Tabu with Estimation algorithms

For both minimizing cost and time, the Collision Remediation policy proves superior to the highway path generation policy. This outcome aligns with expectations, as most missions do not involve collisions among drones. When collisions do occur, they can be effectively resolved, leading to more efficient mission paths. This efficiency translates to reduced overall mission time and, consequently, lower costs, as expenses are based on the duration of drone operations.

Moving forward, the Collision Remediation approach will be adopted for path generation. It offers a 7.44% improvement over the highway policy in time-based evaluations and a 3.10% improvement in cost-based evaluations. While these percentage improvements may seem modest, they translate to approximately a 2-hour reduction in mission time across the seven missions tested. This efficiency gain underscores the value of implementing the Collision Remediation policy to enhance overall performance.

7.5 Comparative Analysis of Area Decomposition Strategies

In this section, we conduct a comparative analysis of various area decomposition strategies, focusing on four key metrics: Endurance, Area Scanned Per Second, Area Scanned Per Euro, and Equal Division. Each metric offers a distinct approach to optimizing drone performance based on mission objectives. Endurance emphasizes mission efficiency by selecting drones with longer operational times, reducing the frequency of recharging or refueling. Area Scanned Per Second assesses the speed and efficiency of drones in covering an area, which is crucial for time-sensitive missions. Area Scanned Per Euro evaluates cost-effectiveness by measuring the area a drone can scan relative to its operational cost. Finally, Equal Division seeks to evenly distribute the scanning workload among drones to ensure a balanced operational approach. By comparing these metrics through rigorous testing, we aim to determine which strategy provides the most effective balance between time efficiency, cost management, and equitable resource allocation, guiding optimal drone selection for diverse mission requirements.

To compare the performance of these metrics we ran the 7 mission performance test with the 12 drone pool and the exhaustive algorithm to judge which performed best. Two separate tests were ran: one being judged by time and the other by cost. The following figures illustrate the results 7.10 7.11.

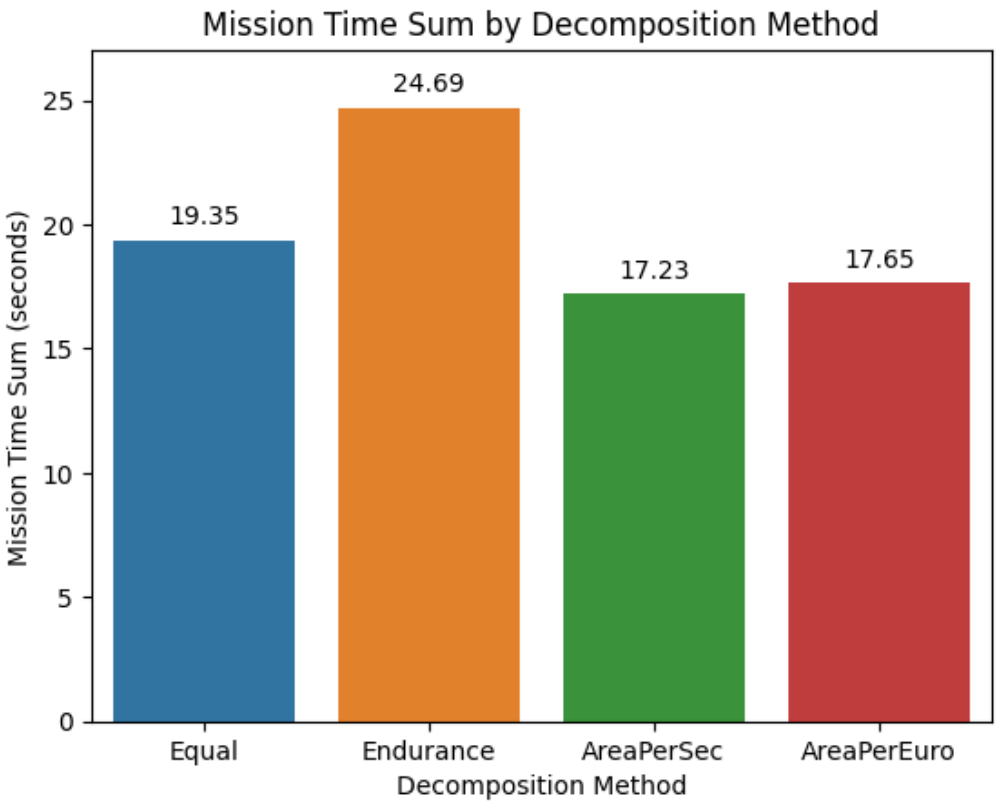


Figure 7.10: Runtime comparison of Area Decomposition approaches when evaluating a mission based on time

When minimizing time, it was anticipated that the metric of Area per Second would be the most effective, as prioritizing drones that can scan the largest area in the shortest time would likely lead to the most efficient outcome. This expectation is confirmed by Figure 7.10, which demonstrates that Area per Second is indeed the best metric for time optimization, closely followed by Area per Euro. The latter metric also performs well due to its similar rationale: drones that offer high area coverage are beneficial for time efficiency.

Figure 7.11 reveals that, for minimizing cost, the Area per Euro metric is the most effective. This outcome is expected, as the goal of minimizing cost is to prioritize drones that not only cover a large area but also have a low operational cost. By focusing on the Area per Euro, we ensure that the chosen drones offer the best balance between coverage and cost, thereby achieving the most cost-effective solution.

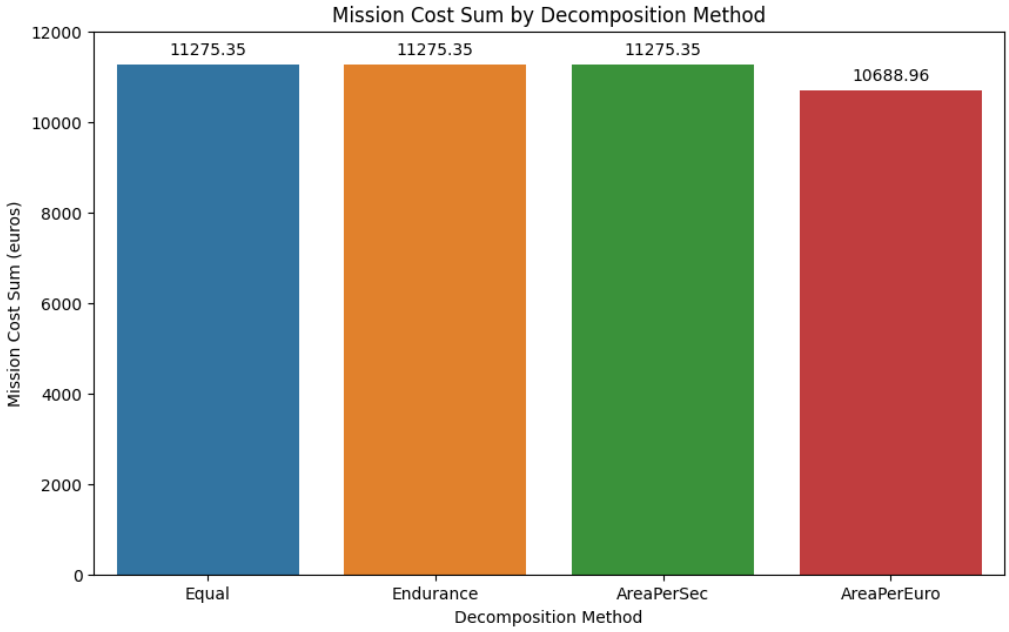


Figure 7.11: Runtime comparison of Area Decomposition approaches when evaluating a mission based on cost

Since only one of these approaches can be used for our mission, we compared the metrics to determine which produced the largest improvement. When focusing on time minimization, the Area per Second metric outperforms the Area per Euro by 2.38%. Conversely, for cost minimization, the Area per Euro metric demonstrates a 5.20% improvement over the Area per Second metric.

Given these findings, the Area per Euro metric emerges as the superior choice for decomposing the search area. It not only aligns better with cost efficiency but also balances the coverage and operational costs effectively. This selection ensures that the approach used will optimize both cost and coverage in the most efficient manner.

7.6 Conclusion

In conclusion, the comprehensive analysis of the genetic algorithm and Tabu Search has highlighted their respective strengths and optimal use cases. The genetic algorithm demonstrates exceptional performance in scenarios where minimizing execution time is critical, offering up to 90.77% faster runtime compared to exhaustive methods. Its efficiency in handling larger datasets makes it a robust choice for complex mission planning tasks. Conversely, the Tabu Search, especially when enhanced with the Estimation function, excels in cost minimization, providing an effective solution for cost-sensitive missions while maintaining efficiency.

Further improvements were achieved through the adoption of the Collision Remediation approach for path generation, which offers notable enhancements in both time and cost evaluations. This approach significantly reduces mission time and underscores the value of adaptive path strategies in optimizing performance.

Additionally, the evaluation of area decomposition strategies revealed that the Area per Euro metric outperforms the conventional even division method. This metric better aligns with cost efficiency and ensures a more balanced coverage relative to the drones' varying performance characteristics. By adopting this refined area decomposition approach, the algorithm achieves optimized cost and coverage, enhancing overall solution quality.

8

Simulator

The Simulator, as its name suggests, is designed to visualize the routes generated by the application for drones within a 3D representation of the mission area. This tool allows users to import a JSON file containing the mission plan and load it into a three-dimensional depiction of the mission's operational space. Additionally, users can play the mission in real-time, observing the movement of each drone along with the progression of time.

The Simulator also provides a means for validating the mission output. It ensures that the mission plan generated by the application is devoid of any undesirable actions, such as drone collisions or marine drones traversing over land.

8.1 Starting Point

During the implementation of the main application, an intern, António Festas, was assigned to develop a simulator for the drone paths generated by this application, as well as for another project involving drone swarms, which is being developed by my colleague, Martim Gouveira. António spent three months creating an application in Unity that successfully utilized real-world data to generate a 3D map accurately reflecting the topography of Portugal as well as the drone routes provided in a JSON file. This simulator will work as the beggining of the simulator for my application.

António's Simulator relies on GeoTIFFs as the foundation for world generation. GeoTIFFs are a specialized raster data format that embeds geographic metadata within standard TIFF files, allowing them to be accurately aligned with real-world locations. This georeferencing capability ensures that each pixel in the image corresponds to specific coordinates on the Earth's surface, making GeoTIFFs indispensable for mapping, remote sensing, and geographic information systems (GIS). They can store various types of spatial data, such as satellite imagery, elevation models, or land cover maps. Due to their open standard, GeoTIFFs are widely supported across different platforms, facilitating interoperability and enabling precise spatial analysis

The GeoTIFFs provide the data necessary to generate a scaled representation of real-world terrain within the Simulator. These files contain vast amounts of information,

making them quite large; for example, the GeoTIFFs covering the region of Portugal require 3GB of storage. Loading all of this data each time the Simulator runs would be both inefficient and time-consuming. To address this, the information is divided across multiple GeoTIFFs, each representing a different section of the world map. The naming convention typically includes the coordinates, with 'N' indicating positive latitude (north) and 'E' indicating positive longitude (east). Each GeoTIFF covers just one degree of latitude and longitude. For instance, the file ASTGTMV003_N30E000_dem represents the area from 30°N to 31°N and from 0°E to 1°E. Given that each degree of latitude/longitude corresponds to roughly 111.32 kilometers, each GeoTIFF covers an area of approximately 12,392 square kilometers.

The Simulator's first step is to identify all the necessary GeoTIFF files by looping through the drone path coordinates and correlating them with the GeoTIFFs containing the corresponding height data. Once all the required GeoTIFFs are identified, the Terrain Renderer script is executed, which loads the terrain in chunks, with each chunk containing the height data from a GeoTIFF. In a GeoTIFF, the heightmap is a 2048 by 2048 matrix, where each entry represents the height in meters for a specific point on the map. However, since the Unity terrain generation engine requires heightmaps to be normalized, these values are adjusted accordingly before being fed into the Unity Terrain Generator. After this process, the chunks are loaded, forming the complete terrain.

With the terrain generated, the next step is to plot the routes and position the drones. For this implementation, António used the mission plan provided by my colleague Martim, who was able to generate it earlier than I could. Consequently, the paths and drones were loaded in discrete time instances, rather than in a real-time representation. This approach caused the drones to jump from one position to the next with each button click, a method that would need modification to allow real-time mission visualization.

The following image illustrates the outcome of António's work, in Fig 8.1, which served as an excellent foundation for the further development of the Simulator. His efforts in gathering and processing the GeoTIFFs, enabling Unity to correctly render them, and placing the drones in the appropriate relative coordinates significantly streamlined my implementation process.

8.2 Simulator Improvements

8.2.1 Terrain

The initial simulator only supported USVs (Unmanned Surface Vehicles) and UAVs (Unmanned Aerial Vehicles) because the minimum height data in the GeoTIFFs represents the surface of the water, with a height value of zero. Since UUVs (Unmanned Underwater Vehicles) need to travel below the water's surface, we had to create depth in the bodies of water. Although the goal was to incorporate real-world data for water depth into our simulator, finding a data source that matched the quality of the terrain data obtained

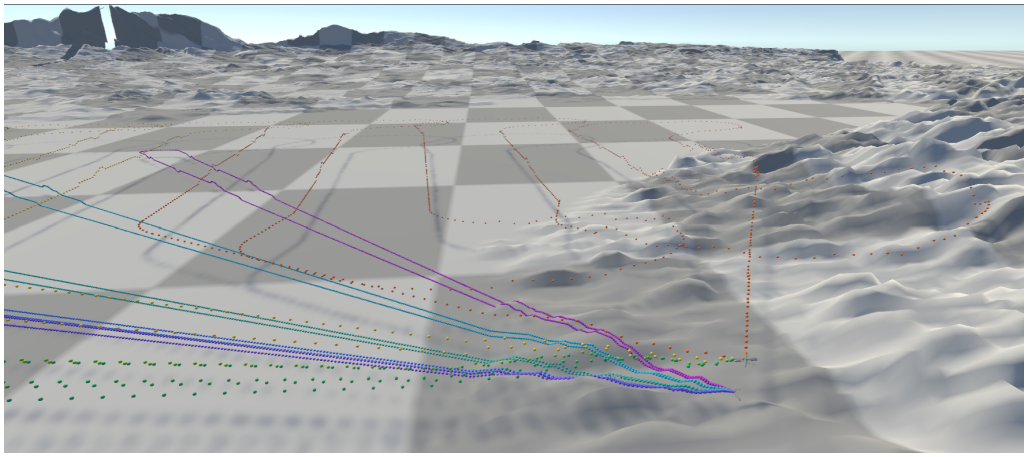


Figure 8.1: Starting Point of the Simulator

from NASA proved too time-consuming. Additionally, water depth does not significantly impact the travel paths of underwater drones, as they primarily navigate close to the sea floor. As long as the depth does not exceed the drone's operational limits, variations in water depth do not substantially affect their movement.

As a compromise, a maximum underwater depth of 30 meters was implemented, with a smooth gradient transition from the coastline to the maximum depth, simulating a natural shoreline. In the future, real-world data could potentially be incorporated into the simulator to enhance this feature.

The next focus was on visual improvements. At that stage, the simulator was entirely gray, making it difficult for users to visualize and locate the drones. The first step in enhancing the visuals was to apply color to the terrain based on elevation. This allowed the underwater areas to be depicted with sand, the land near the shore with dirt, and the peaks with a lighter shade of terrain.

To achieve this in Unity, we created a custom material for the terrain. The terrain is a 3D mesh, and the material determines its color and texture. We applied a shader to this material—a shader controls the appearance of materials by defining how light interacts with surfaces, creating effects like lighting, shadows, textures, colors, and transparency. In this case, the shader sampled the height of each pixel on the terrain and, based on the height, displayed one of five textures, ranging from a sandy texture for the beach and underwater areas to a light gray for the hilltops.

After the terrain was colored and textured, water was added to the world. This was accomplished by importing a premade object from the Unity Asset Library, which provided a body of water that could be adjusted to fit our terrain. The water object is also a terrain element that covers the entire chunk area but is placed at a height of zero, intersecting the terrain where the water should be.

The results of these two changes can be seen in the following Figure 8.2. These enhancements bring the simulator to life, making it look more and more like a finished product.

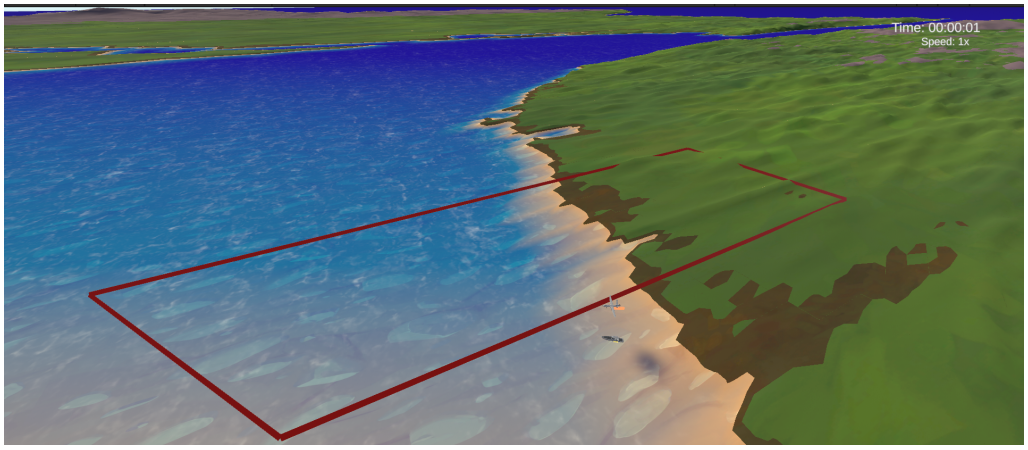


Figure 8.2: Terrain Update on the Simulator

8.2.2 Drone Paths

In the initial version, drone paths were indicated by a row of 3D circles that the drones intersected as time progressed by clicking the arrow keys. Additionally, the drones were modeled using simple shapes, such as cylinders.

To enhance the simulator, we updated the drone assets to include free 3D models from the Unity Asset Store. These new assets provided distinct models for UAVs, USVs, and UUVs, allowing users to easily differentiate between them.

The representation of the paths and the movement of the drones required a complete overhaul. First, the way the mission data was encoded was revised. Previously, each drone's position was associated with an integer value representing the point in the time series when the drone should be at that position. For a real-time representation of drone movement, each position is now paired with a timestamp indicating when the drone traverses that location.

The script responsible for running and loading both the drones and their paths was updated to use Unity's Time object, which tracks the elapsed time in the simulation. At the start of the simulation, this time is set to zero, and all drones are positioned at their starting point, labeled the base station. As time progresses, Unity interpolates the drones' current positions between the starting and next positions based on the elapsed time and the timestamps associated with each position.

The simulator also allows users to adjust the speed of the simulation and to pause it. This is achieved by applying a speed multiplier to the elapsed time, while pausing stops the time from advancing.

It is crucial to consider that each type of drone operates at a different height, which must be accounted for when loading and moving them within the simulation. USVs are the simplest, as they maintain a constant height equal to the water surface level. UAVs and UUVs, however, require adjustments based on terrain elevation. UAVs operate at a height of 100 meters above the terrain to ensure accurate scanning. Therefore, during simulation, the UAVs altitude is adjusted by sampling the terrain height and adding 100

meters, resulting in the UAV adjusting its altitude up and down in response to terrain changes. Similarly, UUVs navigate close to the underwater terrain and adjust their depth based on changes in underwater topography, especially near the coastline.

To enhance the visualization of drone paths, the display method was revised. Instead of loading the entire path at the start of the simulation, the path now appears progressively as the drone moves during the mission. The representation of the paths was also updated: rather than using 3D circles, a line similar to those seen in the movie Tron was introduced. Each drone is now accompanied by a bright, semi-transparent line that is easily identifiable but does not obstruct the view of other drones. This effect was achieved using Unity's LineRenderer component, which renders a line between specified points.

Additionally, a red delimiting line was added around the perimeter of the scanning area as can be seen in Fig 8.3. This boundary helps users visualize the limits of the scanning region and the areas covered by the drones.

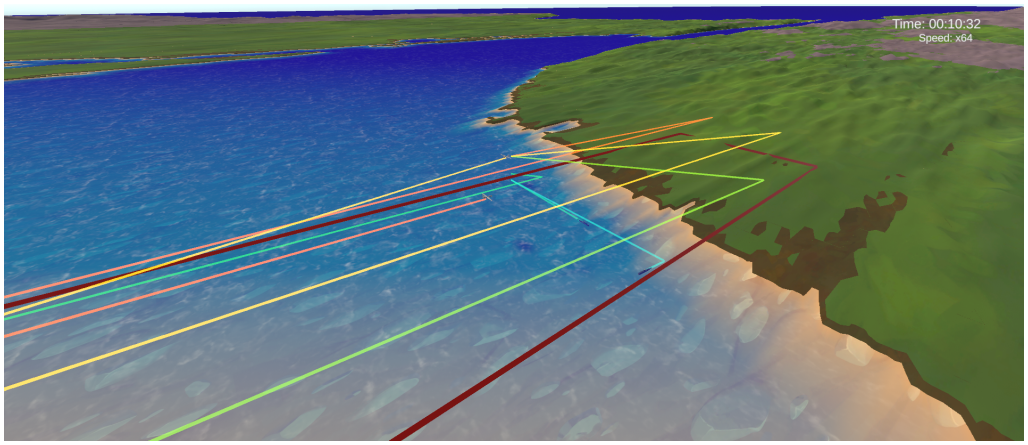


Figure 8.3: Drone Path indicated by the line following the drones

8.2.3 Verifications

The primary goal of this simulator was not only to visualize the drone paths but also to verify the quality and validity of the paths generated by the application. Two key verifications were implemented: checking for drone collisions and ensuring that water-environment drones do not traverse land.

These checks were implemented as follows:

1. **Drone Collision Check**: During each update, the simulator checks if drones are too close to each other based on their simulated positions. If a collision is detected, an explosion effect is displayed on the screen, accompanied by an error message that specifies which drones collided. The simulation is then paused, with the colliding drones highlighted to indicate the issue.

2. **Water Drone Beached Check**: For water drones, the simulator checks if they traverse onto land. If a water drone is detected to be on land, a message is displayed, and the

simulation is halted. The affected drone is highlighted to show where it has gone off course.

Example of one test cases can be seen in the following image:

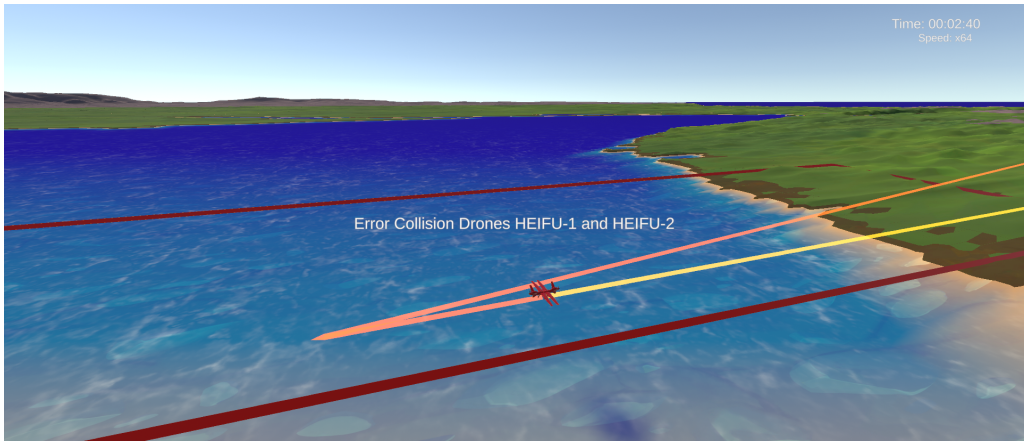


Figure 8.4: Collision Between two drones and warning message

The goal of these verifications is to address potential aspects that the main application might have overlooked. Future enhancements could include energy usage calculations to estimate the drones' consumption during missions, as well as accounting for wind interference in the drone paths. Additionally, a real-world simulation could be developed to account for changes in elevation, wind, and other variables that might cause discrepancies between the theoretical paths generated by the application and actual conditions.

8.3 Algorithm Improvement

Building this simulator and visualizing the drone paths in 3D, instead of the previous 2D view, revealed an aspect we had not previously considered. In a flat-world scenario, where UAVs travel 100 meters above the ground, the drones would not need to adjust their altitude, so we initially didn't account for it. However, when terrain elevation was introduced, it quickly became clear that we had overlooked a crucial factor: the drones constantly need to gain or lose altitude to maintain the necessary 100-meter distance from the ground.

These altitude changes are not currently reflected in the algorithm, as we have not yet integrated elevation data. This will likely be an important next step, as these frequent altitude adjustments will undoubtedly impact battery life, potentially reducing the drones' endurance during missions.

Addressing this issue is one of the primary reasons the simulator was developed. It is essential to replicate real-world conditions as closely as possible, allowing us to identify any gaps in our initial assumptions and ensure the success of our approach.

8.4 Conclusion

The Simulator significantly enhances the user experience by providing a clearer visualization of the mission and the paths of each drone. It not only improves the quality of user interaction but also serves as a secondary verification tool for the mission plans generated by the application. Unity has proven to be an excellent platform for developing this high-level prototype efficiently, thanks to its comprehensive set of tools and readily available assets.

9

Conclusions

The primary goal of this thesis was to develop an algorithm capable of planning missions involving multiple autonomous drones in various environments. To achieve this, we focused on creating a flexible framework that could support increasingly complex mission behaviors as they were added. Initially, our focus was on search-related missions, as these represent the most common type of operation performed by unmanned vehicles. The scanning behavior used in these missions is foundational to many other types of tasks, making it an ideal starting point.

After implementing the algorithm for scanning-related missions, we turned our attention to optimizing and improving its performance. Specifically, we explored different approaches to selecting the best team of drones for each mission, comparing Exhaustive Search, Genetic Algorithms, and Heuristic Methods. Through this exploration, we found that a hybrid approach worked best: Genetic Algorithms were most effective for optimizing mission time, while the Heuristic approach, using an estimation function, proved superior for cost-focused optimizations.

To further enhance mission planning, we implemented four methods of dividing the mission area among drones: Equal Division, Endurance-Based Division, Area-Per-Second Division, and Area-Per-Euro Division. Among these, the Area-Per-Euro method produced the best overall results when both time and cost were considered.

We also optimized the path-planning process. Initially, paths were designed with a focus on collision avoidance. However, we shifted to a collision remediation approach, which significantly improved the quality of the planned paths.

The algorithm was integrated into a user-friendly interface that allows users to input available drones, define mission parameters, and outline the mission area on a map. The system then generates 2D paths, which can be exported into a 3D simulator for further validation.

The simulator enables users to visualize the mission in a 3D environment, providing the ability to assess paths more effectively and to simulate mission execution. Users can pause, fast-forward, and review the mission in real time, while the system performs necessary checks for potential collisions with the ground or other drones.

9.1 Contributions

This thesis makes several notable contributions to the field of autonomous drone mission planning. The primary achievement is the development of a versatile algorithm capable of handling complex mission behaviors for multiple drones in varied environments. Initially focused on search-related missions, this algorithm lays the groundwork for broader applications by efficiently managing diverse operational contexts.

A significant contribution is the integration of multiple optimization techniques. By comparing Exhaustive Search, Genetic Algorithms, and Heuristic Methods, we identified a hybrid approach that optimizes mission time and cost effectively. This hybrid solution enhances the efficiency of mission planning.

Additionally, the thesis explores four methods for dividing mission areas—Equal Division, Endurance-Based Division, Area-Per-Second Division, and Area-Per-Euro Division—demonstrating that the Area-Per-Euro method provides the most balanced results in terms of time and cost efficiency.

Improvements were also made in path planning by shifting from collision avoidance to collision remediation, leading to higher-quality path generation. The development of a user-friendly interface, which includes a 3D simulation capability, further enhances the system's practicality by allowing for real-time path visualization and collision assessment.

Lastly, the thesis highlights future research directions, including the integration of topographic data for endurance calculations and advanced simulation features to account for environmental factors. These contributions advance autonomous drone technology and offer practical solutions for real-world mission planning.

9.2 Limitations and Future Work

The proposed solution, while effective, has several limitations, primarily due to time constraints on this project. With additional time, many of these issues could have been addressed or refined.

Firstly, the mission area selected by the user must be a convex polygon. This restriction arises from the division approach used, which fails with non-convex shapes. For practical mission planning, especially in real-world scenarios, it's essential to account for areas where certain types of drones cannot operate, such as no-fly zones for UAVs. To support non-convex polygons as mission areas, a more advanced geometric algorithm would need to be implemented.

In terms of no-fly zones, the algorithm would require modifications to how scanning paths are generated, allowing for deviations that account for restricted areas. A related issue involves coastline data, which was integrated late in the development process. In rare cases, the scanning paths conflict with the shoreline, causing water-based drones to become stranded near the beach. While this issue has been mitigated, it can still occasionally occur, which can render the mission invalid.

Another limitation that became apparent with the implementation of the simulator is the lack of accounting for vertical movement in the drone's endurance calculations. Vertical movement is primarily influenced by terrain irregularities, as drones need to maintain a stable altitude above ground level. Currently, the path-planning algorithm does not have access to topographic data for the mission area, which would be necessary for more accurate endurance estimations.

Improvements can also be made in the search for optimal solutions. While the Genetic Algorithm was selected for time-based optimization, there is potential to enhance the Tabu Search method. With further development, Tabu Search could be refined to explore a broader range of solutions, reducing the likelihood of converging on local optima. Additionally, combining both the Tabu Search and Genetic Algorithm for missions near the 50% cost and 50% time importance threshold could provide a more robust approach, leveraging the strengths of both algorithms. This enhancement, along with the dual-algorithm approach, could significantly improve both the quality and performance of the solution.

Lastly, the simulator is relatively simple, functioning mainly as a visualization tool. For future improvements, it would be crucial to virtually simulate drone flight dynamics, including environmental factors such as wind and waves, which affect both the planned paths and energy consumption. Enhancing the simulator's capability in this regard would provide more realistic mission assessments.

Despite these limitations, this project represents a meaningful step forward in the realm of autonomous drone mission planning. We believe the framework developed here lays a solid foundation for future research and development in this field.

Bibliography

[1] M. M. Marques. "Reference Model for Interoperability of Autonomous Systems". In: 2018. URL: https://api.semanticscholar.org/CorpusID:86587661 (cit. on pp. 7, 8, 10, 20, 101).

[2] *SPYRO*. pt. URL: https://www.uavision.com/spyro?lang=pt (visited on 2024-01-03) (cit. on pp. 7, 21).

[3] *HEIFU Class 3 Hexacopter*. en-US. URL: https://beyond-vision.pt/heifu-drone-hexacopter/ (visited on 2024-01-03) (cit. on pp. 7, 21).

[4] M. H. Fleming. "Unmanned Systems in Homeland Security". en. In: () (cit. on pp. 8, 10, 21).

[5] *HydroBoat 1200*. en. URL: https://geo-matching.com/products/hydroboat-1200 (visited on 2024-01-03) (cit. on pp. 9, 21).

[6] D. Bertin. *U-Ranger - An Unmanned Surface Vehicle for Surface and Underwater Missions*. 2009-04 (cit. on p. 9).

[7] *Bluefin Robotics Unmanned Underwater Vehicles - General Dynamics Mission Systems*. en. URL: https://gdmissionsystems.com/underwater-vehicles/bluefin-robotics (visited on 2024-01-03) (cit. on pp. 10, 21).

[8] M. R. Silva et al. "Performance Evaluation of Multi-UAV Network Applied to Scanning Rocket Impact Area". In: *Sensors* 19.22 (2019). ISSN: 1424-8220. DOI: 10.3390/s19224895. URL: https://www.mdpi.com/1424-8220/19/22/4895 (cit. on p. 17).

[9] J. Araújo, P. Sujit, and J. Sousa. "Multiple UAV area decomposition and coverage". In: *2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. 2013, pp. 30–37. DOI: 10.1109/CISDA.2013.6595424 (cit. on p. 18).

[10] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr. "Survey on Coverage Path Planning with Unmanned Aerial Vehicles". In: *Drones* 3.1 (2019). ISSN: 2504-446X. DOI: 10.3390/drones3010004. URL: https://www.mdpi.com/2504-446X/3/1/4 (cit. on p. 18).

[11] M. M. Marques et al. "REX 2014 - Robotic Exercises 2014 multi-robot field trials". In: *OCEANS 2015 - MTS/IEEE Washington*. 2015, pp. 1–6. DOI: 10.23919/OCEANS.2015.7404497 (cit. on p. 21).

[12] M. M. Marques et al. "REX 16 – Robotic Exercises 2016 Multi-Robot Field Trials". In: *2019 IEEE Underwater Technology (UT)*. 2019, pp. 1–5. DOI: 10.1109/UT.2019.8734390 (cit. on p. 21).

[13] M. Monteiro Marques et al. "Assessment of a Shallow Water Area in the Tagus Estuary Using Unmanned Underwater Vehicle (or AUV's), Vector-Sensors, Unmanned Surface Vehicles, and Hexacopters – REX'17". In: *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*. 2018, pp. 1–5. DOI: 10.1109/OCEANSKOBE.2018.8559177 (cit. on p. 21).

[14] R. Mendonça et al. "A cooperative multi-robot team for the surveillance of shipwreck survivors at sea". In: *OCEANS 2016 MTS/IEEE Monterey*. 2016, pp. 1–6. DOI: 10.1109/OCEANS.2016.7761074 (cit. on p. 21).

[15] G. De Cubber et al. "The EU-ICARUS project: Developing assistive robotic tools for search and rescue operations". In: *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2013, pp. 1–4. DOI: 10.1109/SSRR.2013.6719323 (cit. on p. 21).

[16] M. M. Marques et al. "Use of multi-domain robots in search and rescue operations — Contributions of the ICARUS team to the euRathlon 2015 challenge". In: *OCEANS 2016 - Shanghai*. 2016, pp. 1–7. DOI: 10.1109/OCEANSAP.2016.7485354 (cit. on p. 21).

[17] M. M. Marques et al. "Chemical and radiological detection using UAV's with ATEX compliance: Proof of concept in port and maritime incident-based scenarios". In: *OCEANS 2018 MTS/IEEE Charleston*. 2018, pp. 1–5. DOI: 10.1109/OCEANS.2018.8604601 (cit. on p. 21).

[18] N. R. Council. *Autonomous Vehicles in Support of Naval Operations*. Washington, DC: The National Academies Press, 2005. ISBN: 978-0-309-09676-8. DOI: 10.17226/11379. URL: https://nap.nationalacademies.org/catalog/11379/autonomous-vehicles-in-support-of-naval-operations (cit. on p. 21).

[19] S. Gupta, M. Ghonge, and P. Jawandhiya. "Review of Unmanned Aircraft System (UAS)". In: *International Journal of Advanced Research in Computer Engineering Technology* 9 (2013-04). DOI: 10.2139/ssrn.3451039 (cit. on p. 21).

[20] *Uncrewed Surface Vessels*. en. URL: https://www.maritimerobotics.com/platforms/category/usv (visited on 2024-01-03) (cit. on p. 21).

[21] M. Liang and D. Delahaye. "Drone Fleet Deployment Strategy for Large Scale Agriculture and Forestry Surveying". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 4495–4500. DOI: 10.1109/ITSC.2019.8917235 (cit. on p. 22).

[22] M. Momeni et al. "Coordinated routing system for fire detection by patrolling trucks with drones". In: *International Journal of Disaster Risk Reduction* 73 (2022), p. 102859. ISSN: 2212-4209. DOI: https://doi.org/10.1016/j.ijdrr.2022.102859. URL: https://www.sciencedirect.com/science/article/pii/S2212420922000784 (cit. on p. 22).

[23] R. Alyassi et al. "Autonomous Recharging and Flight Mission Planning for Battery-Operated Autonomous Drones". In: *IEEE Transactions on Automation Science and Engineering* 20.2 (2023-04), pp. 1034–1046. ISSN: 1558-3783. DOI: 10.1109/tase.2022.3175565. URL: http://dx.doi.org/10.1109/TASE.2022.3175565 (cit. on p. 22).

[24] C. Ramirez-Atencia et al. "Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms". In: *Soft Computing* 21.17 (2017-09), pp. 4883–4900. ISSN: 1433-7479. DOI: 10.1007/s00500-016-2376-7. URL: https://doi.org/10.1007/s00500-016-2376-7 (cit. on pp. 22, 29).

[25] G. Bello-Orgaz et al. "GAMPP: Genetic Algorithm for UAV Mission Planning Problems". In: *Intelligent Distributed Computing IX*. Ed. by P. Novais et al. Cham: Springer International Publishing, 2016, pp. 167–176. ISBN: 978-3-319-25017-5 (cit. on p. 22).

[26] R. Shivgan and Z. Dong. "Energy-Efficient Drone Coverage Path Planning using Genetic Algorithm". In: *2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR)*. 2020, pp. 1–6. DOI: 10.1109/HPSR48589.2020.9098989 (cit. on p. 22).

[27] H. Deng et al. "A Distributed Collaborative Allocation Method of Reconnaissance and Strike Tasks for Heterogeneous UAVs". In: *Drones* 7.2 (2023). ISSN: 2504-446X. DOI: 10.3390/drones7020138. URL: https://www.mdpi.com/2504-446X/7/2/138 (cit. on p. 22).

[28] H. Deng et al. "A Distributed Collaborative Allocation Method of Reconnaissance and Strike Tasks for Heterogeneous UAVs". In: *Drones* 7.2 (2023). ISSN: 2504-446X. DOI: 10.3390/drones7020138. URL: https://www.mdpi.com/2504-446X/7/2/138 (cit. on p. 22).

[29] F. Yan et al. "A Hierarchical Mission Planning Method for Simultaneous Arrival of Multi-UAV Coalition". In: *Applied Sciences* 9.10 (2019). ISSN: 2076-3417. DOI: 10.3390/app9101986. URL: https://www.mdpi.com/2076-3417/9/10/1986 (cit. on pp. 22, 29).

[30] S. Poudel and S. Moh. "Task assignment algorithms for unmanned aerial vehicle networks: A comprehensive survey". In: *Vehicular Communications* 35 (2022), p. 100469. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2022.100469. URL: https://www.sciencedirect.com/science/article/pii/S221420962200016X (cit. on p. 22).

[31] T. Huang et al. "Multi-UAV Mission Planning Method". In: *2020 3rd International Conference on Unmanned Systems (ICUS)*. 2020, pp. 325–330. DOI: 10.1109/ICUS50048.2020.9274958 (cit. on p. 22).

[32] J. Song, K. Zhao, and Y. Liu. "Survey on Mission Planning of Multiple Unmanned Aerial Vehicles". In: *Aerospace* 10.3 (2023). ISSN: 2226-4310. DOI: 10.3390/aerospace10030208. URL: https://www.mdpi.com/2226-4310/10/3/208 (cit. on p. 22).

[33] H. Aziz et al. "Task Allocation Using a Team of Robots". In: *Current Robotics Reports* 3.4 (2022-12), pp. 227–238. ISSN: 2662-4087. DOI: 10.1007/s43154-022-00087-4. URL: https://doi.org/10.1007/s43154-022-00087-4 (cit. on p. 29).

[34] S. Khetarpal. *Dividing a polygon in any given number of equal areas*. 2021-12. URL: https://www.khetarpal.org/polygon-splitting/ (cit. on p. 63).

.1 Missions

This section contains a list of all the missions performed by the Navy according to Reference Model for Interoperability of Autonomous Systems by Mario Marques [1].

The missions performed by the Navy can be separated into two main categories:

.1.1 Civilian

- **Monitoring:** monitor the behavior of crowds, traffic, animals, tree growth, pollution, and air sampling;
- **Scientific Exploration:** validation of geological surveys, helping researchers gain a deeper understanding of the environment;
- **Agriculture and Forestry:** monitoring of crops and pesticide spraying services;
- **Aerial Photography:** taking photographs and filming various events;
- **Engineering and Construction:** performing inspections on power transmission lines and high-voltage towers;
- **Navigation:** identification of reference points that aid navigation;

- **Law Enforcement:** incident surveillance, security, drug enforcement, and search for missing people;
- **Fire Service and Hazardous Materials Operations:** fire detection, incident control, and dangerous material handling;
- **Emergency Medical Services:** provision of medical assistance through the transportation of first aid kits and other medical material;
- **Support River Authorities:** river monitoring, flood, and pollution control;
- **Meteorological Services:** weather forecast through the analysis of samples;
- **Wildlife and Fisheries Management:** animal and fisheries protection. Marine environmental protection;
- **Site Security:** monitor pipelines or other installations to keep them secure from tampering;
- **Surveying:** Geographical, geological, and archaeological survey;
- **Disaster Response and Damage Assessment:** disaster control, cooperation in search and rescue operations, and visual assessment of damaged areas;
- **Marine Environmental Protection:** oil spill response, identification or removal of marine debris, among others;
- **Disaster Management:** Real-time communication assistance, assessment and mapping of damage, pre- and post-event monitoring;
- **Environmental Survey and Measurement:** cloud and aerosol measurements, measuring of carbon dioxide flux, water vapor and total water measurements, coastal ocean observations, O2 and CO2 flux measurements, estimation of glacier and ice sheet dynamics, vertical profiling, heating rates, ice sheet thickness and surface deformation, measuring of cloud properties, physical oceanography, meteorology, and support of atmospheric chemistry;
- **Focused Observations in Extreme Weather:** observations and recording of information on extreme weather events like hurricanes;

.1.2 Military

- **Intelligence** - gathering, analysis, protection and dissemination of information about the enemy.
- **Reconnaissance** - inspection or exploration of an area to gather information.
- **Mine Countermeasures (MCM)** - operations in minefields as an off-board sensor.

- **Anti-Submarine Warfare (ASW)** - robust tracking of quiet diesel electric submarines.
- **Inspection/Identification (ID)** support for Homeland Defense (HLD), Anti-Terrorism / Force Protection (AT/FP), and Explosive Ordnance Disposal (EOD) needs.
- **Oceanography/Hydrography** - collection of environmental data that directly supports anti-submarine, mine, amphibious, strike, special and expeditionary warfare;
- **Communication/Navigation Network Codes (CN3)** - connectivity across multiple platforms, as well as navigation assistance on demand;
- **Payload Delivery** - clandestine method of delivering logistics to support a variety of other mission areas. The missions supported include MCM, CN3, ASW, Oceanography, Special Operations Forces Support, and Time Critical Strike (TCS);
- **Influence Activities (IA)** - deception, deterrence and disruption of enemies;
- **Time Critical Strike (TCS)** - delivery of ordnance to a target with sensor-to-shooter delay measured in seconds, rather than minutes or hours;
- **Maritime Security** - security of allied domestic ports, waterways, and protection of ship and maritime infrastructures (piers, docks, anchorages, warehouses) against a spectrum of threats from conventional attacks to special warfare or specifically targeted terrorist attacks;
- **Surface Warfare** - armed engagement of threats in open waters, as well as littoral warfare;
- **Special Operations Forces (SOF) Support** - missions involving unconventional warfare, counter-terrorism, reconnaissance, direct action and military assistance;
- **Electronic Warfare (EW)** - means of deception, jamming, and warning of electronic attacks;
- **Maritime Interdiction Operations (MIO) Support** - diversion, disruption, delaying, or destruction of the enemy's merchant marine trade. Drug interdiction and alien migrant interdiction operations;
- **Aerial Warfare** - engagement of aerial threats;
- **Transport Cargo or Passengers** - delivery of cargo or passengers in dangerous environmental conditions;
- **Extraction/Insertion** - payload extraction/insertion from/to a specific target or location;
- **Surveillance** - monitoring the behavior of people, objects, or processes for conformity with expected or desired norms;

- **Search and Rescue (SAR)** - search for, and provision of aid to people who are in distress or imminent danger;
- **Combat Maritime Piracy** - combat violence or plunder on the high seas or in the air, for private ends, using aircraft or vessels;
- **Analysis of Damage** - collection of data (images and video) related to disasters, or the effects of attacks;
- **Border Patrol** - monitoring, regulation or control the movement of people, animals and goods into or out of a country;
- **Battlefield Management** - improving of the Command and Control capabilities of a force commander in the field;
- **Operations in Hazardous Environments** - operations where humans couldn't operate (CBRN environments, high-pressure environments, crumbling buildings);