TÉCNICO LISBOA

Trajectory planning problem for detecting radioactive sources using underwater gliders

Ahmed Yehia Zakaria Ibrahim

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Francisco Fernandes Castro Rego
co-Supervisor: Prof. Éric Busvelle

Examination Committee

Chairperson: Prof. João Luís Da Costa Campos Gonçalves Sobrinho
Supervisor: Prof. Francisco Fernandes Castro Rego
Members of the Committee: Prof. David Alexandre Cabecinhas
Prof. Daniel de Matos Silvestre

November 2023

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my family for their support, encouragement, and caring over all these years, for always being there for me through thick and thin and without whom this project would not have been possible. I would also like to thank my supervisor Dr Rego and Professor Busvelle for their unequivocal support during this work. I would like to express my thanks to Professor Batista - the coordinator for Marine Intelligent Robotics (MIR) of which I am a member of - for facilitating the logistics during my stay in Instituto Superior Tecnico.

Last but not least, to all my friends and colleagues who helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

Radioactivity monitoring offers the opportunity to understand its impact on ocean ecosystems in various extreme locations, such as underwater volcanoes, seismic faults, or deep-ocean drilling locations. To expand radioactivity monitoring capabilities, the RAMONES project aims to develop lightweight, high-resolution, power-efficient radiation spectrometers integrated aboard autonomous underwater vehicles, namely underwater gliders, to perform in situ natural and artificial radioactivity measurements in the marine environment. The problem of detecting sources of radioactivity with one or more underwater gliders equipped with radiation spectrometers is a challenging one given the large search areas and the small detection distance of the sensors. To address this problem, in this thesis project, coverage algorithms are devised to generate a path so that gliders can determine the most efficient route to maximize the probability of detecting a radioactive source while avoiding obstacles and ensuring safe navigation through complex environments. The implemented algorithms are Travelling Salesman Problem (TSP), Minimum Spanning Tree (MST), and Optimal Control Problem (OCP). The latter considers the dynamics of the vehicle, that's why, modeling of the vehicle is considered.
Path planning simulations for each algorithm have been implemented using Matlab and an analysis has been done according to the processing time, uncovered areas, length of the traversed path, and time taken during each path. OCP is considered to be more useful if the time to traverse the path is constrained but has a much higher processing time than the other algorithms, and is less suitable when the objective is to cover the area, where a TSP algorithm fares better. Moreover, if the main goal is to obtain a solution fast, MST-based approaches yield smaller processing times.

Keywords

Radioactive sources detection; Autonomous underwater vehicles;Optimal Control;Covering Problem.

Contents

List of Figures

1

Introduction and Motivation

Contents

1.1 Pollutants

Marine habitats are particularly vulnerable to pollution from human activity. Several pollutants have just started to enter the pristine habitats of the Arctic and Antarctic. For many years, industrial waste, radioactive, chemicals (including chemical weapons), wastewater, rubbish, and other waste from terrestrial sources have been dumped in the oceans. Although the dumping of some compounds, such as radionuclides, has been prohibited recently, radioactive sources are nevertheless still present in the environment. For instance, hydrothermal deep-sea vent fauna [3], which was discovered in 1977, is exposed to a particular environment that is loaded with potentially harmful species such as sulfides, heavy metals, and natural radionuclides. It is now well recognized that some of the organisms present in such an environment collect metals during their lives. Although there are not many radionuclide measurements, it appears likely that the communities around hydrothermal vents receive substantial natural radiation exposures. Various archived biological samples gathered on the East Pacific Rise and the Mid-Atlantic Ridge in 1996, 2001 and 2002 were analyzed to be able to determine their uranium contents (U-238, U-235, and U-234) [4]. Moreover, Polonium and Lead were determined in 2 samples collected in 2002 [4]. Vent organisms are characterized by high Uranium, Polonium, and Lead levels compared to what is generally encountered in organisms from outside hydrothermal vent ecosystems. Even though the number of data is low, the results reveal various trends about the site, the location within the mixing zone, and/or the organisms' trophic regime.

The advancement of contemporary remote sensing methods, simpler access to spatial data, data on environmental quality, and developments in analytical chemistry are all advancing our understanding of marine garbage. The size, location, and temporal variability of waste sources and channels, the destiny of waste, and biological and chemical interactions in the environment are all areas where there are still significant knowledge and information gaps [5]. The management of garbage that endangers the oceans frequently faces legal issues, such as the absence of suitable rules, the ambiguity or non-compliance of current laws, and the absence of sufficient policies and strategies.

Figure 1.1 shows the location of ocean garbage patches, the global inventory of radioactive waste disposal at sea (total amounts in petabequerels per region), and chemical weapons dumped at sea.

According to [4], From 1946 through 1993, radioactive waste was thrown into the oceans. The first dump site was situated close to the California coast in the Northeast Pacific Ocean.

The London Convention 1972 came into effect in 1975, outlawing the disposal of high-level radioactive waste at sea. Low-level rubbish disposal in seas and oceans was put on hold for ten years starting in 1983, and since 1993, it has been completely prohibited. However, it should be noted that not all nations followed the restrictions. For instance, the former Soviet Union continued to dispose of radioactive waste by national law in the Arctic Seas and the Northwest Pacific, and this practice involved both low-

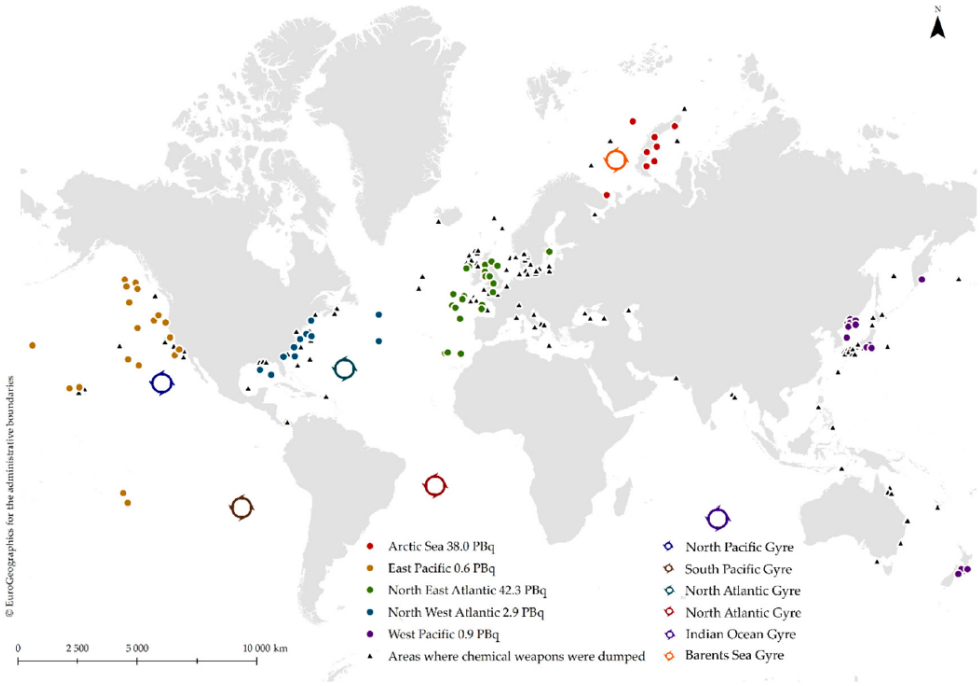

Figure 1.1: The location of ocean garbage patches, the global inventory of radioactive waste disposal at sea (total amounts in petabequerels per region) and chemical weapons dumped at sea

and high-level radioactive waste. There are roughly 8.5×10^4 TBq of radionuclides disposed of in marine ecosystems overall. The maximum decay-corrected inventory of radioactive waste dumped at sea was reached in 1982 and was anticipated to be at 4.5×10^4 TBq; it is currently less than 2.0×10^4 TBq, and it is predicted to be decreased to less than 1.0×10^4 TBq by 2050.

According to [6], the International Atomic Energy Agency (IAEA) maintains a global inventory of anthropogenic radioactive material entering the marine environment, which includes radionuclides from two sources: (i) dumping radioactive waste at sea, and (ii) accidents and losses at sea involving actual or potential releases of radioactive material into the marine environment [6]. The report excludes radioactive materials from nuclear sites on land, inputs from nuclear weapons testing in the past and other military operations, and inputs relating to naturally occurring radioactivity.

According to IAEA, the sources of radioactive material entering the marine environment are as follows: waste dumping at sea including radioactive waste and packaging (nuclear reactor pressure vessels, with and without fuel; solid waste; liquid waste); moreover, radioactive materials enter the sea from accidents and losses (nuclear-powered navy ships and submarines, nuclear weapons, and military vessels capable of carrying such weapons, nuclear-powered civilian ships, nuclear energy sources used

in spacecraft, satellites and acoustic signal transmitters, radioisotope thermoelectric generators, i.e., for lighthouses power supply, sealed radiation sources).

Conducted analyses showed most low- and intermediate-level radioactive waste dumped in the Atlantic and Pacific Oceans and in the Arctic Seas between 1946 and 1993 was located in Atlantic and Arctic dumping sites, the least in Pacific sites, 53.43%, 44.87%, and 1.70% of total activity, respectively. At the North Atlantic, tritium (3H) together with other beta and beta-gamma emitters (90Sr, 134Cs, l37Cs, 55Fe, 58Co, 60Co, 125I, and 14C) accounted for over 98% of the total activity of the waste. The Arctic Seas were dominated by 90Sr and 137Cs, as well as 60Co, 63Ni, and 152Eu, representing 86% and 12% of the total inventory, respectively. Low-level radioactive waste dumped at the Northeast Atlantic sites and the high-level radioactive waste dumped by the former Soviet Union in the Arctic Seas and in the Pacific Ocean account for more than 93% of the activity of all the radioactive material dumped at sea.

Discharges from nuclear facilities in the Baltic Sea region are one of the largest sources of anthropogenic radionuclides in the sea (nuclear power plants and research reactors), the Chernobyl disaster (1986), releases from nuclear reprocessing facilities (Sellafield in England and La Hague in France), and atmospheric nuclear weapons testing (mostly carried out by the US and the USSR in the 1950s and 1960s) [7]

Several nuclear submarine sinkings that occurred after 1963 have been verified. Two of them from the USA Navy—Thresher in 1963 and Scorpion—were in the Atlantic Ocean. The Russian Federation Navy's two submarines perished in the Arctic Seas (K-141 Kursk in 2000, K-159 in 2003). Studies done so far have verified the presence of 137Cs in water and soil samples close to the Komsomolets submarine wreck and 60Co in sediment samples taken in the area of the Scorpion and Thresher submarines. [8], [9]

As previously indicated, nuclear power plant accidents result in radionuclides entering the seas and oceans; the worst of these occurred in 1986 at Chernobyl (now in Ukraine) and Fukushima (Japan, 2011). Initial radiation exposure from the Chernobyl nuclear power plant disaster was caused by short-lived 131I (with a half-life of eight days), while long-lived 137Cs (with a half-life of 30 years) are the main threat to marine ecosystems. Additionally, higher concentrations of two other radionuclides present in the fallout, 134Cs and 90Sr, have been observed in marine ecosystems [10]. Currently, the Baltic Sea's primary marker of anthropogenic radioactivity is 137Cs. Compared to Chernobyl, releases from the Fukushima Daiichi nuclear power plant were considerably smaller (estimated at 10–15% of the Chernobyl value) [11]. Literature sources give divergent data on the total activity released during accidents at both power plants. An extensive analysis, comparison, and compilation of the data from different sources is found in the work by Steinhauser et al. [11].

Eighty-two percent of the 137Cs that entered the Baltic Sea came from the Chernobyl accident, fourteen percent from nuclear weapon testing, and four percent from Sellafield (England) and La Hague (France) facilities that reprocess nuclear material. In contrast, 81% of 90Sr comes from atmospheric nuclear bomb testing and 13% from the Chernobyl emission [8]. Nuclear reprocessing facilities in Western Europe (Sellafield and La Hague) have recently reduced their radioactive output significantly, making them a small supplier of radionuclides [7].

1.2 Ramones Project

As a result of all of the aforementioned issues, RAMONES project [1] was founded. The new H2020-EU FET Proactive Project RAMONES aims to provide innovative and effective approaches for in situ, ongoing, long-term radioactivity monitoring in challenging underwater environments. Advanced underwater radiation measurement equipment uses artificial intelligence and advanced robots to understand coastal locations, develop regulations, and establish standards for environmental sustainability, economic expansion, and human health.

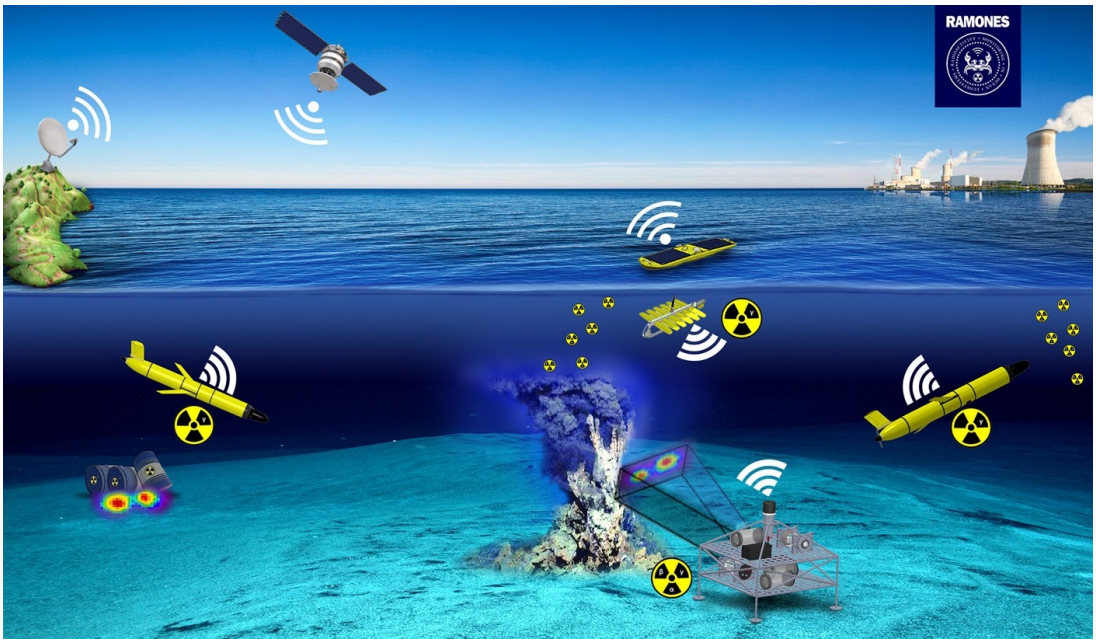

Figure 1.2: A simplified sketch of the RAMONES concept [1]

RAMONES offers a new vision of wisdom-enabled cutting-edge results in both instrumentation and robotic seeing platforms towards a step change in Radioactivity Monitoring in Ocean Ecosystems. RAMONES aims to prove that innovative combination and advancement of recent developments in sensitive environments, low power independent robotic systems, and process modeling propositions, have the eventuality to overcome current limitations and open the window to high temporal and spatial res-

olution aquatic radioactivity measures, in **situ** and in near real-time, forming a game changer in deep-water environmental monitoring.

1.2.1 Goal of the Project

The primary goal of RAMONES is to forge a completely new approach to close the current marine radioactive under-sampling gap and encourage new interdisciplinary research in imperiled natural deep-sea ecosystems.

RAMONES will work hard to provide tools for rapid, long-term deployments, creative robotics, and AI-driven supported methodologies, and scaled-up solutions to communities, decision-makers, and researchers. To respond effectively to both natural and man-made threats, RAMONES will combine SoA technology from diverse disciplines with advanced modeling in a highly synergistic manner. The work of RAMONES will help to shape future policies for the entire world's population.

The RAMONES project aims to design, develop, and validate a new generation of radiation-detecting instruments for marine terrain. These instruments will offer high effectiveness and fine resolution for spectroscopic studies in extreme oceanic environments. The project will use state-of-the-art detectors and innovative designs to create a novel generation of low-power, fast integration-time aquatic instruments for radiation measurements. It will focus on providing tools for long-term, cost-effective deployments, introduce AI-driven methodologies, and offer valuable results to researchers, policymakers, and communities.

Furthermore, RAMONES will contribute to AI development by creating advanced recurrent neural networks and online learning frameworks for time series analysis, abnormal radioactivity detection, and deep learning for identifying radioactivity hotspots in multi-dimensional imaging datasets. The project also aims to develop innovative modeling approaches for various applications, including radiation dose assessment, health concerns, geohazard modeling, and industrial waste radiation. RAMONES will provide forecasts for the likelihood of exceeding radiation thresholds, propose new risk indices, offer policy recommendations, disaster response modeling, and support policy implementation strategies. RAMONES focuses on using gliders which is explained in section 1.3.

1.3 Underwater vehicles: Gliders

Autonomous underwater gliders [12] represent a rapidly maturing technology with a large cost-saving potential over currently available ocean sampling techniques, especially for sustained, month-at-a-time, real-time oceanographic measurements. Underwater gliders move efficiently through the water column by exploiting their ability to change their weight in water.

Due to frequent communications and shallow dives, which reflect frequent changes in buoyancy, there is an increase in power consumption and therefore a reduction in mission length. Currently, the operational endurance of the gliders varies from 3 to 4 weeks for the shallow SLOCUM glider figure 1.3 (max. depth 200m) to several months for the deeper diving gliders Seaglider (max. depth 1000m) and Spray (max. depth 1500m). All three gliders are comparable in size and handling requirements. Their weight in air is approximately 50 kg and their total volume change capacity is between 0.5 and 1% of their total displacement. The horizontal speed relative to the surrounding water is typically around 35 cm/s.

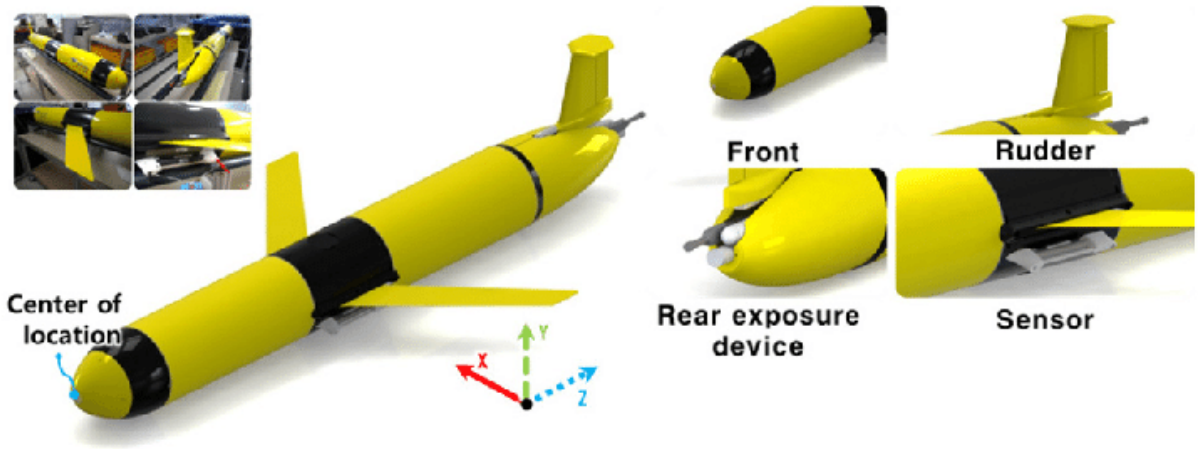

Figure 1.3: SLOCUM glider and components

1.4 The need for Search Algorithms

Searching for such radioactive sources requires a path-planning algorithm to guarantee a high probability of finding radioactive sources. This could be achieved through optimal control approaches mentioned in chapter 2. This can also lead us to an important concept which is motion planning or path planning. Geographic information systems, robotics, computer graphics, and other industries all encounter path-planning or motion-planning problems. The problem consists of finding an optimal route that avoids obstacles and achieves a certain goal. Normally, the path's quality is evaluated based on its Euclidean path length, smoothness, and obstacle-free status. In particular, we consider the coverage problem, which amounts to visiting every point in a defined area within a predefined distance.

Throughout this thesis, several covering algorithms are to be discussed and implemented based on the state-of-the-art planning methods and coverage problem mentioned in chapter 2. These methods can be classified into standard path planning methods e.g. Travelling Salesman Problem (TSP),

Minimum Spanning Tree(MST), etc., and Optimal Control Problems (OCP) as in the Koopman Search Theory [13]. The methodology of each algorithm is discussed in chapter 3 as well as a framework for modeling basic search problems that were established as part of WWII antisubmarine warfare activities and have since been widely used in industrial and tactical applications [14]. Afterward, the mathematical difficulty of the Nomoto Steering Model3.1.1 is studied as well as the new numerical methods that are now available to solve it. These methods are then used to implement a high-dimensional search problem with many searchers, nonlinear searcher dynamics, and control restrictions.

Several implemented algorithms can be found in chapter 4 associated with several meaningful simulations and graphs shown in chapter 5, where a comparison between the aforementioned algorithms is done according to several key points e.g. Length of the traversed path and time taken during each path. For, the MST problem, 2 configurations were utilized which are the square configuration and the Hexagonal configuration to define the path traversed in each algorithm.

2

Literature Review

Contents

2.1 Search Theory Problem

Search theory [15] plays a crucial role in the field of robotics, offering a structured approach to solving problems related to navigation, decision-making, and resource management. One of its primary applications is path planning, particularly for autonomous robots. By employing search theory, robots can determine the most efficient route to maximize the probability of detecting a determined object while avoiding obstacles and ensuring safe navigation through complex environments.

Moreover, search theory enables the optimization of resources in robotics. Robots often have limited resources, such as energy and computational power. Through the principles of search theory, they can plan their actions to minimize energy consumption and maximize operational time. This is of paramount importance in scenarios where robots are deployed in remote or hostile environments and must operate efficiently with constrained resources.

In addition to path planning and resource optimization, search theory assists robots in making informed decisions in ambiguous or dynamic settings. Robotics sometimes entails dealing with partial or imprecise information. In such instances, search theory provides a framework for generating optimal decisions, allowing robots to adapt and choose the best course of action depending on the available data.

Furthermore, search theory is useful in exploration and mapping activities. When robots are entrusted with mapping unknown or partially known settings, this theory guides their exploration, allowing them to efficiently map the surroundings and uncover new areas or places of interest. It is essential in applications such as search and rescue, environmental monitoring, and archaeological exploration.

Search theory can aid in task assignment and resource allocation in multi-robot systems. When many robots collaborate, tasks are divided ideally among the robots, taking their skills and the nature of the jobs into account. Resource allocation is also properly handled, ensuring that shared resources are dispersed in such a way that the system's efficiency and overall performance are maximized.

Several research papers give a taxonomy of search problems that illustrates the distinctions caused by differing assumptions about searchers, targets, and the environment. Some papers, that focus on the search theory applications, are mentioned in the next part.

Starting with [15], which covers recent advances in pursuit-evasion and autonomous search that apply to mobile robotics applications. Figure 2.1 refers to situations in which the target is unaware of the search process and does not attempt to disguise its location or actively elude an approaching

searcher. Search optimization is unopposed, which may apply in operations if the searcher's stealth ability exceeds that of the target. Search games, on the other hand, have relevance to the antagonistic behaviors discussed in the previous section. Many works in this area, however, incorporate discrete time and space, including stationary positioning of a target to impede the searcher's efforts. The following papers illustrate the importance of search theory in different applications as well as the advantages and disadvantages of applying these approaches.

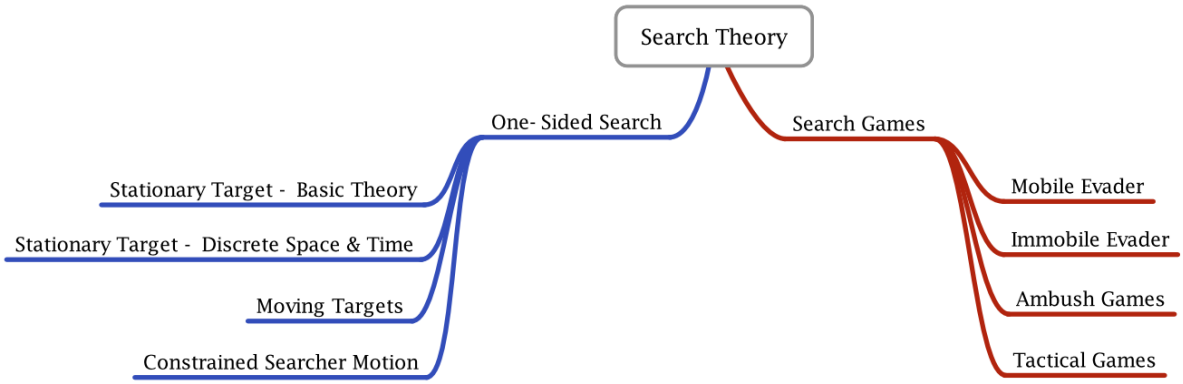

Figure 2.1: Illustration of the partial taxonomy provided by Benkoski et al. [2] for search theoretic problems and application areas previously considered in the operations research and applied mathematics communities

Karaman & Frazzoli (2011) discuss sampling-based algorithms for optimal motion planning in robotics. They present algorithms such as Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRT) that have been shown to work well in practice and have theoretical guarantees of probabilistic completeness [16].

Masakuna and Fukuda Masakuna et al. (2019) [17] address the problem of search by a group of solitary robots. They propose a coordinated search strategy for self-interested robots without prior knowledge about each other. The results demonstrate the effectiveness of the strategy in achieving efficient search.

Ismail & Hamami (2021) [18] conduct a systematic literature review on swarm robotics strategies applied to the target search problem with environment constraints. The review identifies various strategies and highlights their applicability in different research domains, including computational and swarm intelligence.

Yang et al. (2019) [19] propose an extended particle swarm optimization (PSO) based collaborative searching approach for robotic swarms with practical constraints. The approach aims to address the problem of collaborative search for specific targets in an unknown area. The results demonstrate the effectiveness of the approach in achieving efficient search.

Luo et al. (2020) [20] focus on the target search process in swarm robotics. They propose a search approach based on robot chains with an elimination mechanism. The approach utilizes limited percep-

tion and local interaction of robots under a self-organizing mechanism. The results show the effectiveness of the approach in achieving cooperative search.

Zhou et al. (2021) [21] propose a multi-target coordinated search algorithm for swarm robots considering practical constraints. The algorithm addresses the challenges of multi-target search in unknown complex environments. The results demonstrate the effectiveness of the algorithm in achieving efficient search.

2.2 Coverage Problem

The coverage problem in robotics refers to the task of ensuring that one or more robots visit each point in a target area at least once in an optimal manner, considering factors such as time, energy, and resource utilization. This problem is particularly relevant in applications such as environmental monitoring, search and rescue missions, cleaning tasks, agriculture, and surveillance.

To properly complete their tasks, robots must make well-informed decisions on how to navigate and cover an area. Several researchers consider the covering problem as part of the motion planning problem.

Another critical component related to the coverage issue is time efficiency. Identifying the most time-efficient methods for area coverage is critical in search and rescue missions, inspection activities, and surveillance, especially when time is of the essence. Furthermore, in mobile robotic systems, energy is frequently a valuable and restricted resource. Efficient coverage tactics are critical for reducing energy consumption, prolonging the robot's operational time, and improving overall efficacy.

One benefit of this approach is that the sensors mounted on the robots gather data uniformly throughout the area, reducing the risk of overlooking important information in data collection applications like environmental monitoring and exploration. In other cases, such as floor cleaning robots or crop harvesting equipment, complete coverage is required because missing any part of the area is undesirable.

Furthermore, coverage methods are frequently included in path planning. They aid in the generation of pathways that not only cover the area effectively but also take obstacle avoidance and safe navigation into account, especially in dynamic contexts. The coverage problem has several practical applications, including autonomous lawnmowers, vacuum cleaners, agricultural robots, and autonomous cars, all of which require good area coverage to complete their tasks.

Furthermore, as robots gain autonomy, the capacity to make independent coverage decisions becomes increasingly important. This autonomy allows robots to operate with little human involvement, increasing their utility and decreasing the need for ongoing supervision.

In industrial contexts, effective coverage can result in significant cost savings. Painting, welding, and inspection robots benefit from optimized coverage, which reduces production costs and increases

productivity.

In general, the coverage problem becomes more critical and complex in multi-robot systems. Coordination of various robot movements to achieve complete and efficient coverage is a big task that necessitates complex algorithms and coordination methodologies. In essence, the coverage problem drives the growth of automation and autonomy in robotics by underpinning the efficiency, effectiveness, and cost-effectiveness of robotic operations across a wide range of applications.

Several papers have addressed this problem and proposed different approaches and algorithms.

Yehoshua et al.(2016) [22] focused on the robotic adversarial coverage of known environments. They introduced a unique problem where the environment is adversarial, and the robot may be physically harmed. They presented preliminary results and discussed the challenges and potential solutions for this problem.

Batalin & Sukhatme (2004) [23] defined the coverage problem as the maximization of the total area covered by a robot's sensors. They discussed the exploration and deployment of a mobile robot in a communication network and proposed algorithms for efficient coverage.

Galceran & Carreras (2013) [24] conducted a survey on coverage path planning for robotics. They presented a collection of algorithms for complete coverage path planning using a team of mobile robots in unknown environments. They discussed various approaches, including boustrophedon decomposition and cellular decomposition.

Le et al. (2018) [25] proposed a modified A-Star algorithm for efficient coverage path planning in a self-reconfigurable robot with an integrated laser sensor. They aimed to advance coverage path planning in robots used for applications such as cleaning, painting, and mining.

Strimel & Veloso (2014) addressed the coverage planning problem with finite resources. They focused on finding a motion path for a robot that traverses all points in a given area or space [26].

Ku et al. (2019) presented a graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots. They considered scenarios where the mobile robot is equipped with reconfigurable modules. Their approach utilized graph theory to plan efficient coverage paths for reconfigurable robots [27].

Algorithms mentioned by Batalin & Sukhatme (2004) [23] and Le et al. [25] **improved** the efficiency of coverage path planning, however resource constraints and coverage optimization were not studied, which was done by Strimel & Veloso (2014) [26].

2.2.1 Minimum Spanning Tree

A Minimum Spanning Tree (MST) is a fundamental concept in graph theory that is frequently used in robotic path planning to discover an efficient way to connect a set of points in space (usually representing places or waypoints) [28]. MSTs are important in robotics for tasks such as motion planning, exploration,

and connection analysis. Here's an explanation of how MSTs are utilized in robotic path planning. The environment is typically represented in robotics as a graph, with nodes representing points of interest like obstacles, targets, or waypoints and edges representing relationships or distances between these points. The major goal is to figure out how to connect these places while minimizing the overall distance traveled, which is a challenge strongly connected to path planning.

An MST is a graph tree given by a subset of edges that connects all nodes while minimizing the overall sum of edge weights. In robotics, these nodes can represent critical sites that the robot must visit, and the MST acts as a roadmap for linking these spots effectively. Robots can cross the MST's edges to visit all essential places while traveling the least amount of distance.

While MSTs are optimal for connecting all nodes with the least total distance, they may not be optimal for specific path-planning scenarios with extra limitations, such as avoiding obstacles or passing across non-uniform terrain. Additional path planning approaches, such as a potential field or Dijkstra's algorithm, may be employed to move within the edges of the MST, optimizing the path for various criteria, to handle these specific issues.

In dynamic contexts with changing conditions, the MST may need to be computed regularly as new information becomes available. This adaptability is required to account for the movement of barriers or changes in the robot's goals and ensure that the robot operates successfully and efficiently. In conclusion, MSTs are an important component of robotic path planning, as a core element for connecting waypoints and guiding the robot through its surroundings. However, they are frequently combined with other strategies to solve the unique constraints of certain robotic tasks.

2.2.2 Traveling Salesman Problem

An alternative casting for the coverage problem can implemented through the Travelling Salesman Problem (TSP) [29].The TSP belongs to the class of combinatorial optimization problems, where the objective is to find the shortest single path that, given a list of cities (or nodes) and distances between them, visits all the cities only once. Here is an example solution of the TSP in figure 2.2.

2.2.2.A Applications

The TSP has practical applications in diverse fields such as logistics, transportation planning, circuit design, and DNA sequencing. Efficient solutions to the TSP can significantly impact resource utilization and operational costs.

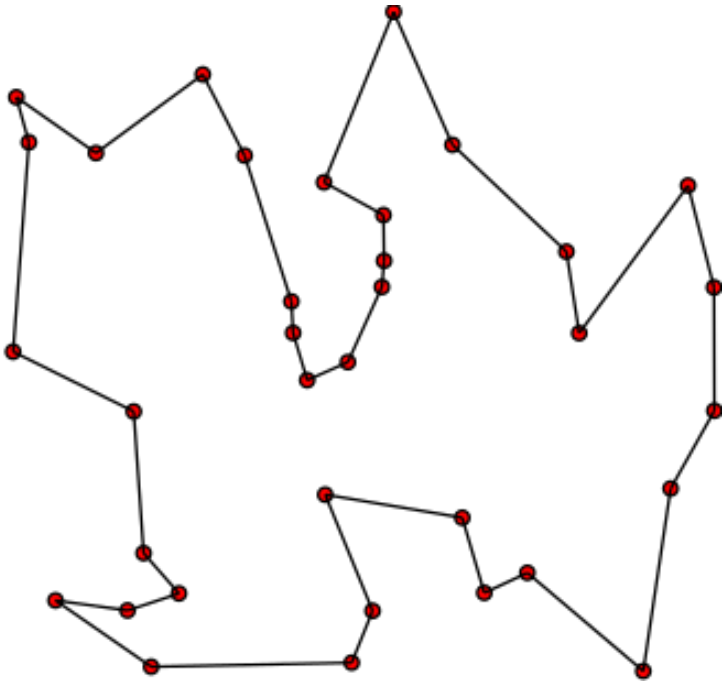

Figure 2.2: Solution of a traveling salesman problem: the black line shows the shortest possible loop that connects every red dot.

2.2.2.B Complexity and Intractability

NP-Hardness: The TSP is classified as NP-hard, indicating that finding an optimal solution in polynomial time is unlikely. This inherent complexity has motivated the exploration of approximation algorithms and heuristic methods to obtain near-optimal solutions.

Decision and Optimization Versions: The decision version of the TSP involves determining whether a tour with a given length exists, while the optimization version seeks the shortest tour.

2.3 Optimal Control Problem

Optimal control problems present themselves in a variety of domains, such as path planning, to determine the control inputs that minimize a given cost function while fulfilling system dynamics and constraints.

There are various types of optimal control problems, depending on the performance index, the type of time domain (continuous, discrete), the presence of different types of constraints, and what variables are free to be chosen. The formulation of an optimal control problem requires the following:

1. a mathematical model of the system to be controlled,

2. a cost function,

3. a specification of all boundary conditions on states, and constraints to be satisfied by states and controls,

4. a statement of what variables are free.

Numerous algorithms for solving optimal control problems are reviewed as Evolutionary Algorithms, collocation, Dynamic Programming, Pontryagin's Maximum Principle, Linear Quadratic Regulator, Model Predictive Control (MPC), Gradient-Based Optimisation Methods, Direct Methods, Stochastic Optimisation, Mixed-Integer Programming (MIP), Simulated Annealing and others.

Each algorithm from the aforementioned ones has its advantages and limitations, and the choice of algorithm depends on the problem characteristics and computational requirements.

2.3.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are population-based optimization algorithms inspired by natural evolution processes. [30] proposed a fast multiobjective genetic algorithm. EAs have been extensively used to solve multiobjective optimization problems, including optimal control problems. Evolutionary algorithms like Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) can be applied to search for optimal control policies. They are beneficial when it comes to dealing with complex and non-linear systems.

2.3.2 Dynamic Programming

Dynamic Programming (DP) is a mathematical optimization method that breaks down a complex problem into simpler subproblems. [31] applied the Conjugate Gradient Approach to solve a nonlinear optimal control problem with model-reality differences. DP is Highly helpful for problems with a discrete state and control space.

2.3.3 Pontryagin's Maximum Principle

Pontryagin's Maximum Principle (PMP) is a powerful tool for solving optimal control problems with continuous state and control variables. [32] considered optimal control problems governed by linear unsteady fluid-structure interaction problems. PMP offers the required circumstances for optimality and aids in the formulation of the control law.

2.3.4 Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) is a popular control technique that solves optimal control problems for linear systems with quadratic cost functions. [33] used the LQR method to solve a heat transfer model in the continuous cast secondary cooling area. LQR provides an analytical solution for linear time-invariant (LTI) systems, however it is not applicable for non linear systems.

2.3.5 Model Predictive Control (MPC)

Model Predictive Control (MPC) is a control strategy that solves an optimal control problem at each time step using a dynamic model of the system. MPC has received a lot of attention because of its capacity to deal with constraints and uncertainty. [34] compared optimization algorithms based on combining FD approximations and stochastic gradients with methods based only on a stochastic gradient for optimal well-control problems.

2.3.6 Gradient-Based Optimization Methods

Gradient-Based Optimization Methods utilize the gradient of the cost function to iteratively update the control inputs. [35] proposed a distributed continuous-time algorithm for constrained convex optimizations using a nonsmooth analysis approach. Gradient-based methods are efficient for problems with smooth cost functions and constraints.

2.3.7 Stochastic Optimization

Stochastic Optimization algorithms incorporate randomness in the optimization process to explore the search space efficiently. [36] presented distributed continuous-time approximate projection protocols for shortest-distance optimization problems. Stochastic optimization methods are useful when the problem involves uncertainties or noisy measurements.

2.3.8 Mixed-Integer Programming (MIP)

Mixed-integer programming (MIP) formulations handle problems with discrete decision variables. [37] discussed production optimization with adjoint models under nonlinear control-state path inequality constraints. MIP algorithms can handle complex constraints and discrete decision-making, making them suitable for certain optimal control problems.

2.3.9 Simulated Annealing

Simulated Annealing is a metaheuristic algorithm inspired by the annealing process in metallurgy. [38] discussed the use of swarm and evolutionary algorithms, including simulated annealing, for energy disaggregation problems. Simulated annealing is effective for exploring the search space and escaping local optima.

2.3.10 Direct Methods

Direct Methods transcribe the optimal control problem into a finite-dimensional nonlinear programming problem. [39] exploited sparsity in direct collocation pseudospectral methods for solving optimal control problems. Direct methods provide a direct solution to the problem but can suffer from high computational complexity for large-scale problems.

2.3.11 Shooting Method

The shooting method is a numerical technique used to solve optimal control problems. Optimal control problems involve finding the optimal trajectory or sequence of controls that minimizes or maximizes a given performance criterion, subject to dynamic system constraints. The shooting method is particularly useful for solving problems where the system dynamics are described by ordinary differential equations (ODEs) and the goal is to find the optimal control inputs. one can start at the initial value of the working conditions and work towards the solution. However, this approach fails to handle boundary value issues since there aren't enough beginning value conditions to solve the ODE and produce a singular result. To address this challenge, the shooting techniques were created.

Several scientific methods will be stated based on the state of the art for each method studied and they will be categorized according to some classifications e.g. direct and indirect Methods, and so on.

According to the references in [40], it is clear that If the differential equations are unstable or very stiff, they will be very sensitive to the initial conditions. Consequently, very good initial estimates are required to start the method. Otherwise, the forward integration can overflow, or the Jacobian matrix used in computing the corrections can be very ill-conditioned. Unfortunately, very good initial estimates are often difficult to obtain without prior knowledge of the solution.
This difficulty can be alleviated by using the multiple shooting method. In this method, the original interval is divided into several (not necessarily equal) subintervals. This decreases the sensitivity of the problem to the initial conditions as the integration is performed over shorter intervals. The concept of the multiple shooting method is readily adapted to parallel processing.

Moving to another paper [41] discussing multiple shooting advantages, the author discusses a novel

flexible multiple-shooting technique, that is developed to greatly improve robustness of convergence, and achieves a reduction in the design space, improved solution accuracy compared with direct methods, large convergence domain with the aid of multiple shooting, flexibility for handling discrete system parameters, and rapid problem convergence that allows more trials of initial guesses to locate global optima.

In [42] the idea of the Combination of direct Multiple Shooting and Collocation was studied. There was a new direct method for trajectory optimization has been developed, allowing for a combination of different types of transcription within one problem. Each phase of a problem can either be discretized using collocation or multiple shooting. For multiple shooting and collocation, several different types of integration and transcription methods are available.

Meng et al. focused on creating an analytical gradient-based hybrid continuation approach to increase the effectiveness of solving the fixed-time minimum-fuel bang-bang control trajectory. The technique facilitates the continuing process and can accelerate the solution of a nonlinear programming problem (NLP) connected to a hybrid multiple-firing scheme. The analytical formulations that have been created are, in order, the Jacobian of the equality constraints concerning the decision variables, the gradient of the performance index about the decision variables, and the decision variables concerning a continuation parameter (thrust magnitude in this work). The last one is used to forecast initial values in a thrust magnitude continuation, whereas the first two are used to solve an NLP with a given thrust magnitude [43].

Shippey and Subbarao worked on fusing the genetic algorithm with the shooting Method in this paper. In their study, they presented a hybrid numerical optimization method using the direct Hermite-Legendre-Gauss-Lobatto collocation method for trajectory synthesis. The transfer trajectory from a geocentric circular orbit to an areocentric circular orbit with the least amount of time was calculated. Using the genetic algorithm and the shooting method, the initial hypotheses for the nonlinear programming problem were found by scouring the state space for workable candidate solutions. The comparison between the obtained and published results is very favorable [44].

Searching for some sources in an environment can be approximated as searching for uncertain targets as expressed in [45], which has implemented the shooting approach principles. The shooting approach has been depicted in the optimal search problem, created in the 1940s, offering a fundamental framework for motion planning when looking for uncertain targets. The optimal search problem examines how to maximize the likelihood of detecting a non-evasive target with uncertain properties, given the capabilities of the detection equipment and some restrictions on searcher trajectories.

2.3.11.A Koopman Search Theory

An example of the shooting method is the Koopman Search Theory algorithm [13], which was first created as part of the Second World War anti-submarine warfare efforts, and has been extensively used in tactical and industrial applications. Further details about the mathematical issue of this model are studied as well as the modern numerical techniques that can now be used to solve it. Then, applying these techniques to a multi-searcher, nonlinear searcher dynamics, and control constraint high dimensional search issue.

Moving to another category, which is the "Collocation Method", there are plenty of papers to be mentioned .

2.3.12 Collocation Method

Starting with Kelly's reference in [46] for studying the Direct Collocation Method, which is used to generate an optimized trajectory. Kelly has presented an overview of numerical trajectory optimization with a special emphasis on direct collocation techniques. These approaches are generally easy to comprehend and successfully address a broad range of trajectory optimization issues. a series of four example problems have been used to demonstrate each new set of topics throughout the paper. The best gait for a bipedal walking robot is computed using Hermite-Simpson collocation after we first use trapezoidal collocation to solve a straightforward one-dimensional toy problem. Some tips for creating well-behaved optimization problems, along with fundamental debugging techniques, are discussed.

A summary of additional strategies for trajectory optimization is towards the end of the study. The well-documented MATLAB code for each of the examples and methodologies is also available as an electronic supplement. The main objective is to support the reader with the tools they need to comprehend and effectively use their direct collocation approaches.

In [47] the orthogonal Collocation Method is introduced, where the Gauss pseudospectral method is used for dealing with nonlinear optimal control issues. At the Legendre-Gauss points, the approach described here performs orthogonal collocation of the dynamics. It is possible to map the costates of the continuous-time optimum control problem to the Karush-Kuhn-Tucker (KKT) multipliers of the nonlinear programming problem (NLP) that results from this type of orthogonal collocation. As a result, the KKT multipliers of the NLP can be used to obtain an accurate estimate of the costate at both the Legendre-Gauss points and the boundary points. The costate mapping at the boundary points is specifically caused by the Legendre-Gauss collocation. The approach is illustrated on a sample problem where it is revealed that extremely precise costates equations are achieved. According to the findings of this study,

direct route optimization and costate estimation can be accomplished using the Gauss pseudospectral approach. It is shown that the errors in the state, costate, and control decrease rapidly as the number of collocation points increases. The results obtained in this paper demonstrate the viability of the Gauss pseudospectral method as a means of obtaining accurate solutions to continuous-time optimal control problems.

Starting with the direct numerical solution of an optimal control problem in which [48] Hargraves and Paris have stated that state variables are represented by cubic polynomials, control variables are linearly interpolated, and collocation is used to solve the differential equations. The Optimal Control Problem (OCP) is converted into a mathematical programming problem, which is then resolved using sequential quadratic programming. The method is demonstrated to be highly effective for several sample problems and is simple to program for a fairly broad trajectory optimization problem. Results are contrasted with conclusions drawn from different techniques.

Hargraves and Paris have presented a trajectory optimization technique that combines mathematical programming and an embedded collocation strategy. This approach has been tested on a wide range of test cases, and it has been discovered that it is competitive in terms of price and robustness. Their objective is to include that approach in a program for generic trajectory optimization.

Moving to 2019, [49] when Patel et al. suggested a technique to increase trajectory optimization accuracy for dynamic robots with intermittent employing orthogonal collocation to make contact Up until recently, mode-scheduling, which requires an a priori understanding of the contact order and cannot produce complex or counter-intuitive behaviors, constituted the majority of trajectory optimization techniques for systems with contacts. By allowing the optimization to create or dissolve contacts as necessary, contact-implicit trajectory optimization methods provide a remedy for this, although their accuracy has been lacking up to this point. To produce trajectories with substantially higher precision, they combined techniques from direct collocation employing higher-order orthogonal polynomials with contact-implicit optimization. The crucial idea is to raise the polynomial representation's order while continuing to operate under the presumption that effect happens throughout one finite element.

Moving to 2000, [50] when Fahroo and Ross developed a novel approach at that time to be able to solve the trajectory optimization problem using mixed approaches. Using a direct method and an indirect collocation technique, the Bolza problem that arises in trajectory optimization is resolved. The time histories of the state and control variables are approximated using piecewise continuous polynomials in traditional collocation methods. The dynamic constraints across the subintervals whose endpoints are the collocation points are then satisfied using a numerical integration method.

Global orthogonal polynomials, such as Legendre polynomials, are utilized in the pseudo spectral approaches as opposed to standard collocation methods to approximate the state and control variables. A differentiation matrix forces the state equations precisely at the collocation locations. The co-state equations are also roughly calculated using the indirect method. They demonstrated that the fundamental structure of the original equations may be preserved while using this method to approximatively satisfy the necessary criteria obtained from the Pontryagin minimal principle. As a result, the Jacobians are few and have a clear structure, which makes efficient numerical implementation possible. For precisely resolving optimal control issues, this method offers an alternative to conventional indirect methods (like indirect shooting) and is simple to apply in conjunction with direct methods. To illustrate the value of this approach, a numerical example is provided.

In [51] Ross and Fahroo provided a mathematical framework that distinguishes or blends the various approaches. This framework is facilitated by distinguishing a transformation from the discretization rather than bundling them together. Two clear layers emerge with two types of convergence notions. A Covector Mapping Principle is enunciated which facilitates the definition of a Complete Method. Many competing claims and issues can now be resolved. A surprising conclusion that can be drawn from this is that only a few fundamental methods for trajectory generation and optimization of complex dynamical systems have been studied in the literature.

In [52], Bhattacharya and Gavrilova are presenting an algorithm based on the Voronoi diagram to calculate the best path between the source and the destination when there are simple polygonal barriers that are disjoint. Furthermore, the path's quality is assessed according to its total length, smoothness, and obstacle-clearance. They also offered a thorough explanation of the dynamic updates and maintenance algorithm for Voronoi diagrams. The experimental results of this technique shows great potential of finding an optimal path.

2.4 Voronoi Diagram Approach

The division of a plane with n points into convex polygons such that each polygon has precisely one generating point and that each point in a particular polygon is closer to its generating point than it is to any other creates a Voronoi diagram. Applications for the Voronoi diagram and its dual, the Delaunay triangulation, are numerous and include collision detection [52].

Compared to several algorithms e.g. optimal control approaches, the Voronoi Diagram approach has numerous advantages e.g. simplicity, versatility, low computational power, and efficiency. For instance, the Voronoi diagram as a roadmap has a big O Notation of $O(nlogn)$ faster than a quadratic order

$O(n^2)$. Big O notation is a mathematical notation used to describe the performance or complexity of an algorithm. It provides an upper bound on the growth rate of an algorithm's time or space complexity as a function of the input size.

2.5 Voronoi Diagram

Voronoi diagram is a representation of how the space is partitioned among the random points while considering polygonal obstacles. This diagram can be useful for path planning or other applications where spatial partitioning is required.

The following example generates a Voronoi diagram for a set of random points in a bounding box, considering polygonal obstacles. The Main Components of the aforementioned algorithm are depicted in the following flowchart 2.3 as follows:

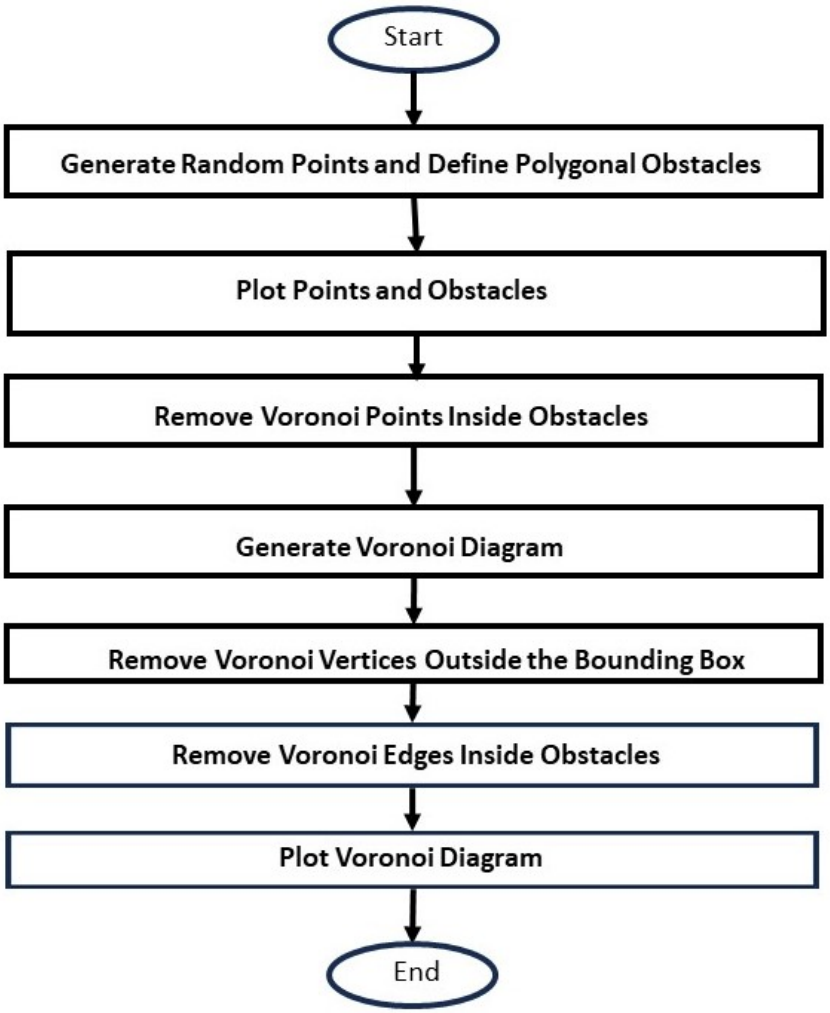

Figure 2.3: Flowchart of Voronoi Diagram

The generated Voronoi diagram 2.4 is presented as follows:

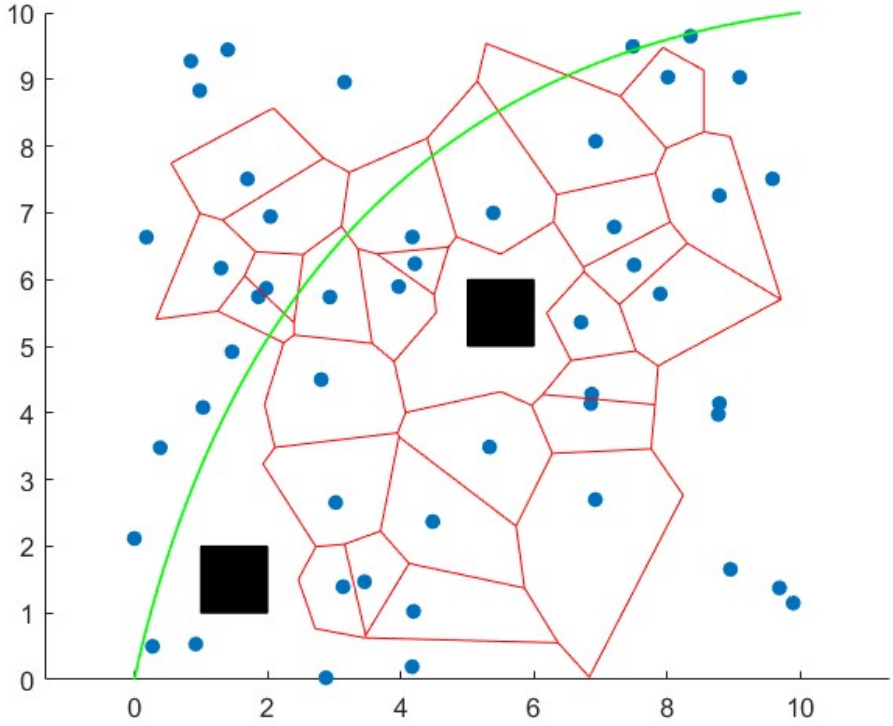

Figure 2.4: Voronoi diagram generated with 50 random points (blue points), Voronoi cells (red polygons), and 2 obstacles (black squares), navigation path (green)

The red polygons in 2.4 represent the Voronoi cells or regions associated with each of the Voronoi vertices. Each red polygon outlines the region of space that is closest to a particular vertex (a random point or a point on the boundary of an obstacle) in the set of random points.

Important terminologies for the Voronoi path planning approach are listed below

1. Voronoi Vertex: Each Voronoi vertex represents a significant point in the space, which is usually one of the random points you generated or a point on the boundary of an obstacle.

2. Voronoi Cell: The Voronoi cell associated with a vertex is the region of space that is closer to that vertex than to any other vertex. It is the set of points for which the associated vertex is the nearest point.

3. Space Partitioning: The Voronoi diagram effectively divides the entire space into regions, where each region belongs to one of the Voronoi cells. These regions are outlined by the red polygons.

In path planning, the Voronoi diagram 2.4 can be used to determine paths that navigate through the space while staying within the regions associated with the random points and avoiding collision with any

of the obstacles. Paths that stay within the Voronoi cells are guaranteed to be the shortest paths to their respective associated points.

So, in summary, the red polygons in the Voronoi diagram represent regions in the space that are closer to specific points (the Voronoi vertices) than to any other point in the set, and they provide a way to partition the space based on proximity to these points.

3

Methodology

Contents

3.1 Modeling

System modeling is needed for the optimal control approach explained in part 3.2. Throughout this thesis, an underwater autonomous glider is approximated to an Unmanned Surface Vehicle (USV), which is extensively studied in the literature. For this research, it has been decided to follow the method studied in [53], which is devoted to Mine Countermeasures (MCM) search missions. For instance, to develop a model for these vehicles, it is assumed that USVs conduct search missions at a constant velocity, without aggressive maneuvers, and therefore exhibit simple planar motion at the sea surface (pitch, roll, and heave motions are zero) Assuming also that no sway slipping is performed. The equations of motion (EOM) can be easily modeled by kinematics as in equations (3.1) to (3.4). Using figure 3.1, let the state variable pair $(x(t), y(t))$ be the vehicle's position (**vertical** and **horizontal** respectively) in meters north and east from an inertial reference frame, $\psi(t)$ be its heading angle in radians measured clockwise from north, and $r(t)$ is its turn rate in radians per second. If the vehicle travels at constant forward velocity V meters per second, the state–space Equation Of Motion (EOM) for the state vector $\overrightarrow{x(t)} \equiv [x(t), y(t), \psi(t), r(t)]^T$ and control input u(t) are

$$\dot{x(t)} = V cos\psi(t), \tag{3.1}$$

$$\dot{y(t)} = V sin\psi(t), \tag{3.2}$$

$$\dot{\psi(t)} = r(t), \tag{3.3}$$

$$\dot{r(t)} = \frac{1}{T}(Ku(t) - r(t)). \tag{3.4}$$

Equation 3.4 implements a first-order approximation to the well-known Nomoto model for ship-steering equations, after applying the Laplace transform $\mathcal{L}\{f(t)\}$ to the differential equation 3.4 assuming zero initial conditions, a simple transfer function in **S-domain** between rudder deflection angle $u(s) = \delta_r(s)$ and turn rate r(s) that "is the most popular model used for ship autopilot design due to its simplicity and accuracy" [[53], p. 5]. Note that a transfer function is a relation between output Y(s) and input U(s) in the **S or Frequency** domain as follows: $G(s) = \frac{Y(s)}{U(s)}$. The Laplace transform $\mathcal{L}\{f(t)\}$ is a tool for analyzing Linear Time-Invariant systems (LTI), providing a way to analyze their behavior in the frequency domain.

3.1.1 Nomoto Steering Model

A Nomoto Steering model [53] is derived as follows:

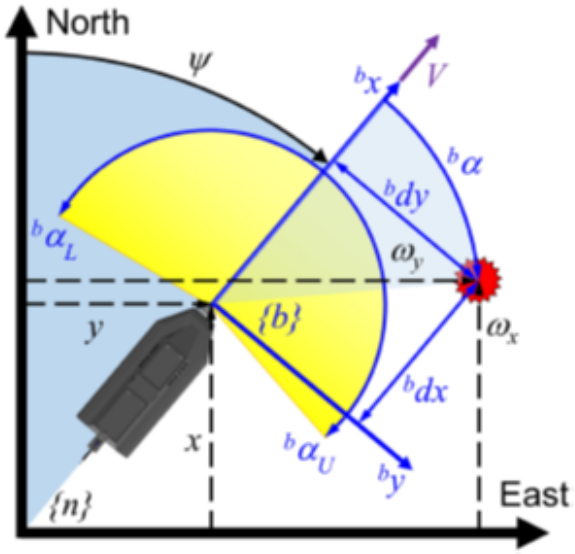

Figure 3.1: Model: Horizontal Plane Geometry of a USV

$$\begin{bmatrix} m - Y_{\dot{v}} & -Y_{\dot{r}} & 0 \\ -N_{\dot{v}} & I_{zz} - N_{\dot{r}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v}(t) \\ \dot{r}(t) \\ \dot{\psi}(t) \end{bmatrix} + \begin{bmatrix} -Y_v & mu_0 - Y_r & 0 \\ -N_v & -N_r & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} v(t) \\ r(t) \\ \psi(t) \end{bmatrix} = \begin{bmatrix} Y_{\delta_r} \\ N_{\delta r} \\ 0 \end{bmatrix} \delta_r(t) \tag{3.5}$$

where

$v(t)$ sway velocity in the y-axis direction;

$r(t)$ yaw rate, a control variable driven from equation 3.3;

$\psi(t)$ yaw angle;

$\delta_r(t)$ rudder angle;

m vehicle's mass;

I_{zz} vehicle's moment of inertia about the z-axis;

Y hydro. coefficients producing sway forces;

N hydro. coefficients producing yaw moments.

In general, control inputs and state variables (and their derivatives) produce nonlinear hydrodynamic forces and moments. It is common practice, however, to approximate these effects by multiplying each contributing variable with a linearized hydrodynamic coefficient. In equation (3.5), Y and N denote coefficients that produce sway forces and yaw moments, respectively, while subscripts denote their corresponding control input e.g. $\delta_r(t)$ or state variable e.g. $v(t)$. Considering no sideslip ($v(t) = 0$), equation 3.5 can be arranged to

$$\begin{bmatrix} I_{zz} - N_{\dot{r}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{r}(t) \\ \dot{\psi}(t) \end{bmatrix} + \begin{bmatrix} -N_r & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} r(t) \\ \psi(t) \end{bmatrix} = \begin{bmatrix} N_{\delta_r} \\ 0 \end{bmatrix} \delta_r(t) \tag{3.6}$$

3.1.2 Assumptions

For the sake of simplicity, some assumptions were considered as follows:

1. Zero stream current
2. Vehicle traveling at a constant speed
3. The Problem is considered to be a simple planar motion at the **sea surface** (i.e. pitch, roll, and heave motions are zero).
4. yaw rate can be directly controlled (Nomoto time constant is considered to be very large)

3.2 Optimal Control Problem: Optimal motion planning for searching of uncertain Targets

Searching for radioactive sources can be considered as searching for uncertain targets as expressed in [45].

For the sake of constructing a performance criterion for the optimal search problem, the probability of target detection must be modeled. According to the Koopman search theory [13], an exponential probability of detection model is derived which has since become ubiquitous in optimal search literature.

Given the position of a searcher at $p1(t)$ and a target at $p2(t)$, this instantaneous rate of detection is a function $\eta(p1(t); p2(t); t)$ such that the probability of detection in a quite small interval $[t, t + \Delta t]$ is independent from previous time intervals and given by the quantity $\eta(p1(t); p2(t); t)\Delta t$. The rate function $\eta(p1(t); p2(t); t)$ is chosen to model the qualities of sensor equipment such as acoustic and sonar sensors, which have fast enough sweep rates to be modeled as continuous processes. An example of a detection rate function using the Poisson Scan Model is shown in figure 3.2.

Taking into account the previous assumptions, an explicit formula for the probability of target detection is derived. If we denote the probability of non-detection with the function P(t), the independence of the time intervals creates the following difference equation:

$$P(t + \Delta t) = P(t)[1 - \eta(p1(t), p2(t), t)\Delta t]$$

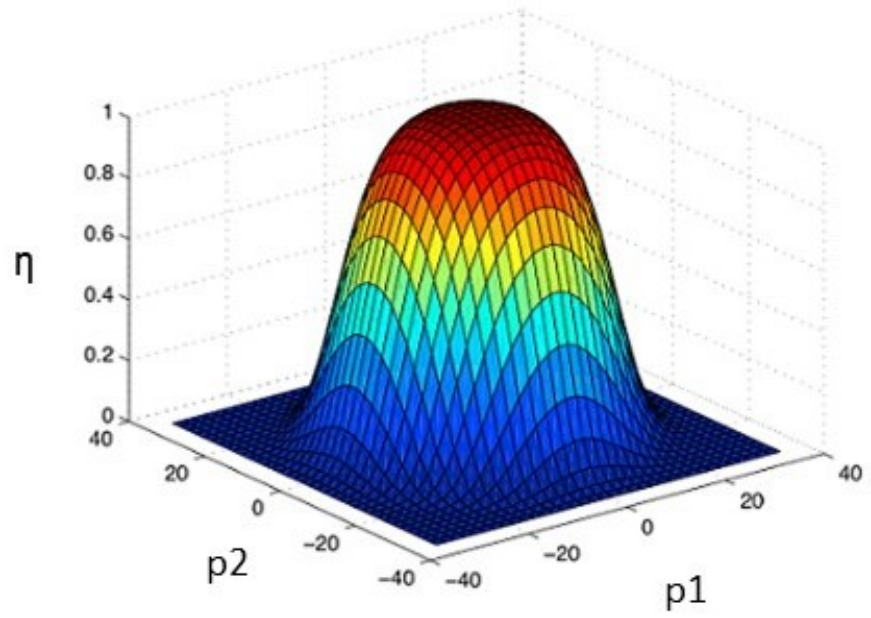

Figure 3.2: Example detection rate function η: Poisson Scan Model

where

- η(p1(t),p2(t),t)·Δt: This term represents the probability of detection during the time interval Δt. It's the product of the instantaneous detection rate and the duration of the time step.
- 1−η(p1(t),p2(t),t)·Δt: This is the probability of non-detection during the time interval Δt. It complements the probability of detection, indicating the likelihood that the target remains undetected during the time step.
- P(t+Δt): This is the updated probability of non-detection at the next time step t+Δt. It's determined by multiplying the current probability of non-detection P(t) by the probability of non-detection during the time step.

As $\Delta t \to 0$ this has an exponential solution:

$$P(t) = e^{-\int_0^t \eta(p1(\tau),p2(\tau),\tau)d\tau} \tag{3.7}$$

Using similar techniques, a range of probability for multiple searchers and targets can be obtained.

These include the likelihood of finding every target and, in the worst case, the likelihood of finding none of the targets. Generally, this can be expressed as follows:

$$P(t) = G\left(\int_0^t \eta(p1(\tau), p2(\tau), \tau)d\tau\right).$$

Throughout this thesis, the model of conditionally deterministic motion is taken into consideration. That model assumes that the motion of the targets is given entirely by a function of time and a parameter ω. This parameter is an element of a bounded parameter space $\Omega \subset \mathbb{R}^n$ and has a known probability density function $p : \Omega \to \mathbb{R}$.

Considering a conditionally deterministic target motion $y(t|\omega) = h(t|\omega)$ leads to a probability quantity which is itself a random variable, i.e. P(t,ω). A natural performance metric is to minimize the expectation of this random variable over a time interval [0, T]. This will create such a cost function class:

$$J = \int_\Omega G\left(\int_0^T \eta(p1(t), u(t), p2(t, w), w)dt\right)p(w)dw \tag{3.8}$$

Using the derivations in the reference paper [45], Equation (3.8) can be rewritten in another way to perform discretization if needed but it is outside the scope of this thesis.

3.3 Minimum Spanning Tree

A Minimum Spanning Tree (MST) [28] is a fundamental concept in graph theory, particularly in the field of optimization. Given a connected, undirected graph $G = (V, E)$ with a set of vertices V and edges E, an MST is a tree that spans all vertices in V while minimizing the total edge weight.

3.3.1 Problem Formulation

Let $G = (V, E, w)$ be a weighted graph, where $w : E \to \mathbb{R}^+$ assigns a non-negative weight to each edge. The goal is to find an MST T with the minimum total weight:

$$\text{minimize} \quad \sum_{e \in T} w(e)$$

subject to the constraint that T forms a tree that spans all vertices in V.

3.3.2 Prim's Algorithm

One algorithm to find the MST is Prim's algorithm [28], which starts with an arbitrary vertex and greedily grows the tree by adding the smallest-weight edge that connects a vertex in the tree to a vertex outside the tree. The process continues until all vertices are included in the MST.

The algorithm can be described using the following steps:

1. Initialize an empty set T to represent the MST.
2. Choose an arbitrary vertex v_0 and add it to T.
3. While T does not span all vertices:
   (a) Select the smallest-weight edge (u, v) such that $u \in T$ and v is outside T.
   (b) Add vertex v and edge (u, v) to T.

The algorithm maintains a set T that grows into the MST.

3.3.3 Kruskal's Algorithm

Another algorithm for finding the MST is Kruskal's algorithm, which is based on sorting the edges by weight and greedily adding edges to the MST while avoiding cycles.

The algorithm can be summarized as follows:

1. Sort all edges E in non-decreasing order of weight.
2. Initialize an empty set T to represent the MST.
3. Iterate through the sorted edges and add each edge (u, v) to T if adding it does not form a cycle.

The algorithm builds the MST incrementally by adding edges to the growing set T.

3.3.4 Equations and Notations

Here are some equations and notations used in the context of Minimum Spanning Tree:

$$G = (V, E, w) \quad \text{(Weighted graph)} \tag{3.9}$$

$$w(e) \quad \text{(Weight of edge } e\text{)} \tag{3.10}$$

$$T \quad \text{(Minimum Spanning Tree)} \tag{3.11}$$

$$\sum_{e \in T} w(e) \quad \text{(Total weight of edges in } T\text{)} \tag{3.12}$$

The total weight of edges in the Minimum Spanning Tree (T) is the objective function to be minimized.

3.4 Traveling Salesman Problem (TSP)

The Traveling Salesman Problem (TSP) [29] is a classic optimization problem in the field of combinatorial optimization. It involves finding the shortest possible tour that visits each city exactly once and returns to the original city. TSP can be formulated as an Integer Linear Programming (ILP) problem. In the ILP formulation, binary decision variables are introduced to represent whether an edge (the connection between two cities) is included in the tour or not.

3.4.1 Problem Statement

Consider a set of n cities $C = \{1, 2, \ldots, n\}$, and let d_{ij} represent the distance between cities i and j. The objective of the TSP is to minimize the total tour length L for a given permutation of cities.

3.4.2 Decision Variable

Introduce a binary decision variable x_{ij} for each pair of cities i and j to indicate whether the tour includes the direct route from city i to city j. The variable x_{ij} is defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if the tour includes the route from city } i \text{ to city } j \\ 0, & \text{otherwise} \end{cases}$$

3.4.3 Objective Function

The total tour length L can be expressed as the sum of distances traveled between consecutive cities in the tour:

$$L = \sum_{i=1}^{n} \sum_{j \neq i} d_{ij} x_{ij} \tag{3.13}$$

3.4.4 Constraints

To ensure a valid tour, the following constraints are imposed:

$$\sum_{j \neq i} x_{ij} = 1, \quad \forall i \in C \tag{3.14}$$

$$\sum_{i \in C} x_{ij} = 1, \quad \forall j \in C \tag{3.15}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \text{for all non-empty proper subsets } S \subseteq \{1, 2, \ldots, n\} \tag{3.16}$$

Equation (3.14) ensures that each city is visited exactly once, and Equation (3.15) ensures that the salesman leaves each city exactly once. Equation (3.16) ensures Subtours elimination which prevents cycles that do not include all cities.

3.4.5 Binary Decision Variable Constraint

To handle the binary decision variable x_{ij}, we add the following constraint:

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in C, i \neq j \tag{3.17}$$

3.4.6 Complexity of TSP

The TSP is known to be a Non-deterministic Polynomial-time (NP-hard) problem, meaning that no known algorithm can solve all instances of the problem efficiently. Various heuristic and approximation algorithms are commonly employed to find near-optimal solutions in practice.

3.4.7 Solution of a TSP

For the Traveling Salesman Problem (TSP), there can be multiple solutions or sequences of solutions, especially if there are multiple paths that result in the same optimal total distance. The goal of the TSP is to find the tour with the minimum total distance, and there might be more than one tour that achieves the same optimal distance.

In mathematical terms, if there are multiple Hamiltonian cycles in the graph (cycles that visit each node exactly once), and all of them have the same total weight (sum of edge weights or distances), then there are multiple optimal solutions to the TSP. A Hamiltonian cycle is visiting each node exactly once and returning to the starting node.

The number of possible solutions grows rapidly with the number of cities, and finding all possible solutions is generally not practical for large instances of the TSP. Instead, optimization algorithms are used to efficiently find a single optimal or near-optimal solution. While there may be multiple optimal solutions to the TSP, the focus is typically on finding any one of these optimal solutions efficiently.

4

Implemented Algotihms

Contents

This chapter discusses the implemented algorithms that are responsible for achieving the path planning concept which mainly relies on covering the environment as much as possible, avoiding any obstacles, and reaching the goal if possible. Some algorithms e.g. the Traveling Salesman Problem (TSP), and Minimum Spanning Tree (MST) have various configurations e.g. hexagonal and square routes to increase the covered area. Some also take into consideration the dynamics of the vehicle e.g. Optimal Control Problem (OCP) achieving more optimality. In all of the aforementioned algorithms, 3 basic steps are done to build the environment needed for implementing the algorithms.

Firstly, a random map is generated with polygons with the help of the Matlab function $rng(N)$, where N is the map Number. These polygons represent obstacles in the environment. The size and number of polygons are set under control by various parameters like Average obstacle radius and Maximum number of vertices.

Secondly, polygon intersection is considered, as in equation 4.3, it is guaranteed that the created polygons do not collide with one another, preventing obstacles from overlapping. If two polygons are found to be included within each other, one of them is ignored.

$$b = (x_1 - x_2)(y_2 - y_1) - (x_1 - x_2)(y_2 - y_1) \tag{4.1}$$

where (x_1, y_1) and (x_2, y_2) are the endpoints of one line segment, and ($x_{1'}$, $y_{1'}$) and ($x_{2'}$, $y_{2'}$) are the endpoints of the other line segment.

$$a = (x'_1 - x_2)(y'_2 - y'_1) - (x'_1 - x'_2)(y'_2 - y'_1) \tag{4.2}$$

The intersection point (x, y) is calculated as follows:

$$\zeta = \frac{a}{b} \tag{4.3}$$

If 0 $<\zeta <$1, then an intersection exists.

The Euclidean distance between two points (x_1, y_1) and (x_2, y_2), which can be expressed as:

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.4}$$

Thirdly, the map is represented by a graph. The graph's edges show the possible connections between the nodes, which are valid locations in the surrounding space. The adjacency matrix is computed by the algorithm by utilizing the intersections of the obstacles and nodes.

4.1 Minimum Spanning Tree (MST) Algorithm

This algorithm is responsible for generating a random map filled with randomly placed obstacles and finding a minimal spanning tree (MST) for path planning [54].

The **minspantree** built-in Matlab function is used to calculate the graph's minimal spanning tree. The connected path represented by this MST minimizes the overall distance while covering every vertex. An optional visualization of the MST in the main figure is also included in the code. The aforementioned algorithms find the tree that connects all vertices with the minimum total edge weight. The solution of an MST is a collection of vertices defined at a distance ***d*** from each other, where these vertices are located outside the polygons. In figure 4.1, two possible configurations for the path are implemented which are the square and hexagonal configurations where the vertices are chosen to be at a ***d*** distance from each other in the form of a hexagon and the same way for a square configuration.

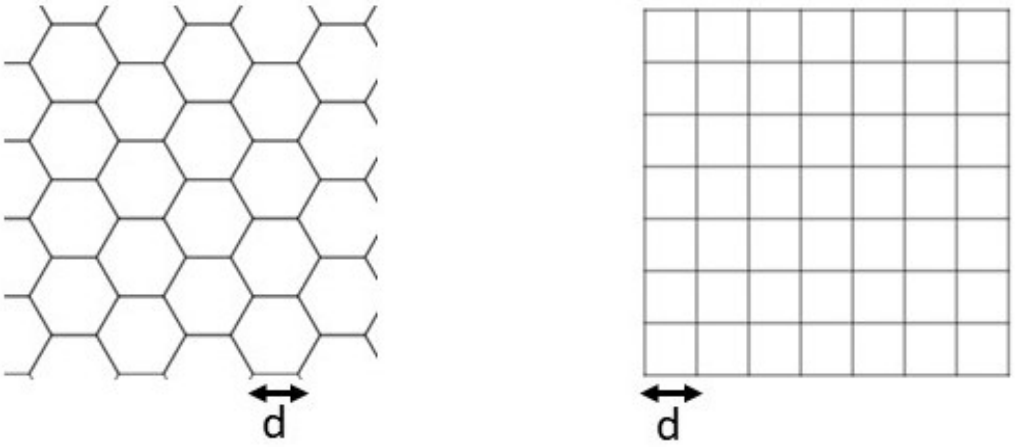

Figure 4.1: MST Hexagonal grid (left) and square grid(right) configurations

In the hexagonal configuration, the vertices are governed by the following equations:

$$
\begin{aligned}
v_1 &= (i \cdot \frac{3}{2} \cdot d, j \cdot \frac{\sqrt{3}}{2} \cdot d) \\
v_2 &= (v_1 + (d, 0)) \\
v_3 &= (v_1 + (\frac{d}{2}, \frac{\sqrt{3}}{2} \cdot d)) \\
v_4 &= (v_1 + (-\frac{d}{2}, \frac{\sqrt{3}}{2} \cdot d)) \\
v_5 &= (v_1 + (-d, 0)) \\
v_6 &= (v_1 + (-\frac{d}{2}, -\frac{\sqrt{3}}{2} \cdot d))
\end{aligned}
$$

In these equations, *i* and *j* represent the grid coordinates of the hexagon, and *d* is the spacing between the hexagons. The equations provide the coordinates of the six vertices of the hexagon.

In 4.2, the randomly generated map number 492 is generated with obstacles and path. For the sake of analysis and debugging, the MST is transformed into this Morphological dilatation in figure 4.4. Then, computing a line that gives contour which gives a path to be traversed as in figure 4.5.

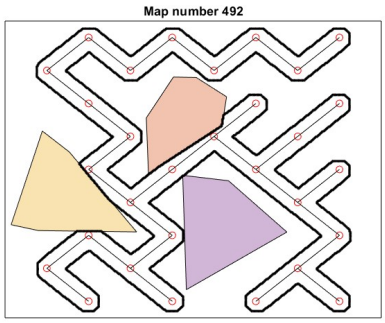

Figure 4.2: Main figure: Map and Path

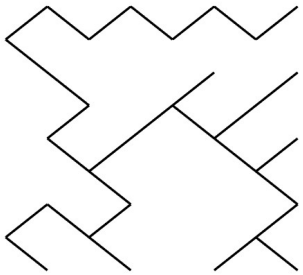

Figure 4.3: Minimal covering Tree

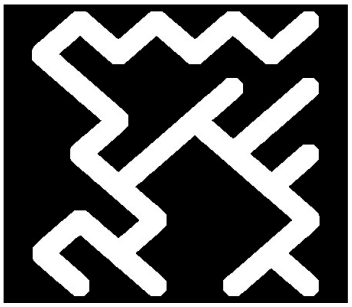

Figure 4.4: Morphological dilatation

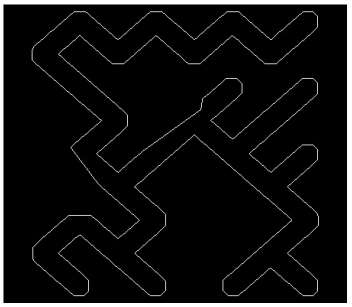

Figure 4.5: Generated Contour

4.1.1 Path Tracing

The script forms a path by traveling from one of the tree's nodes to its neighboring nodes, tracing a path through the resulting minimal-spanning tree. In the main figure, the path is represented by a succession of black dots. The path is traced by iteratively moving from one point to the next. The code tracks the change in position (dx, dy) to move to the next point, which can be expressed as:

$$dx = x_{\text{next}} - x_{\text{current}} \tag{4.5}$$

$$dy = y_{\text{next}} - y_{\text{current}} \tag{4.6}$$

4.1.2 Output and Visualization

The script provides information about the length of the generated path and visualizes it on the main figure, showing the path through the polygons(obstacles).

4.2 Traveling Salesman Problem (TSP) Algorithm

This algorithm is responsible for implementing the Traveling Salesman Problem (TSP) with additional features e.g. obstacle avoidance [55]. Similar to MST, vertices are chosen at a specified distance but here it is ***0.5*d*** instead of d. Here is a breakdown of the algorithm as below:

4.2.1 Initialization

In this part, point generation and obstacle definition are implemented via the following steps:

1. Generating random points within a bounding box.
2. Defining polygonal obstacles based on input parameters.
3. Filtering out points that are included within the dimensions of the obstacles.

4.2.2 Graph Construction

In this part, a graph is created with the filtered points as nodes. Then using 4.4, the distance between the nodes is calculated. In order to find all paths from one node to another in a graph, a helper function (”AllPath”) is defined. AllPath accepts inputs as node numbers and adjacency matrix and generates nodes of the path that are visited one time at most, applying the main concept of a TSP.

4.2.3 TSP Problem Formulation

In this part, the TSP problem is formulated as follows:

1. Set up the ILP problem to solve the Traveling Salesman Problem (TSP).
2. Defining constraints to ensure that each node is visited exactly once.

The objective function in TSP seeks to minimize the total distance traveled while maintaining the *Hamiltonian cycle* which is visiting each node exactly once and returning to the starting node. This can be represented as follows:

$$\text{Minimize} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \cdot x_{ij} \tag{4.7}$$

Where:

N represents the number of nodes (locations to visit).

c_{ij} represents the distance or cost between nodes i and j

x_{ij} is a binary decision variable, where $x_{ij} = 1$ if traveling from node i to node j occurs, and 0 otherwise.

4.2.4 Initial TSP Solution

This step is responsible for solving the TSP problem and computing an initial solution.

4.2.5 Subtour Elimination

Several Subtours elimination is done by first identifying and eliminating subtours in the solution using linear inequality constraints as in equation 3.16. Then, repeat the optimization process until there is only one subtour remaining.

4.2.6 Plotting and Visualization

The optimal path is plotted in chapter 5. As well as taking into consideration the performance criteria used.

4.3 Optimal Control Problem (OCP)

The Nomoto Steering Model parameters are chosen according to the following table 4.1:

Design Parameter	Value
Nomoto Gain Constant,K	0.5 1/s
Nomoto Time Constant,T	5.0 s
Velocity,V	2.5 m/s

Table 4.1: Design Parameters for a USV Model

For this algorithm, Matlab is also used for generating some simulations for the optimal Control Problem alongside Casadi [56] optimization toolbox and Ipopt [57], a library for large-scale nonlinear optimization implementing an interior point method for nonlinear programming, which is suitable for solving both convex and nonconvex nonlinear optimization problems.

The following algorithm solves an optimal control problem (OCP) using direct multiple-shooting [45]. The problem involves finding an optimal path for a robot navigating through a map with obstacles while avoiding collisions and minimizing a cost function.

1. Initialization

    - The algorithm begins by setting up the problem's parameters, including the map size, obstacle positions, control intervals, and various constants.
    - It defines the optimization problem using the CasADi library, specifying decision variables (state trajectory and control trajectory), an objective function to be minimized, dynamic constraints, control constraints, and boundary conditions.

2. Solving the NLP (Nonlinear Programming Problem): In section 2, there are numerous techniques and solvers stated to solve the programming problem, but in order to solve a nonlinear programming problem (NLP) in a computationally efficient way, the IPOPT solver is chosen to solve the defined in the optimization setup.

3. Post-processing: The code post-processes the optimization results to calculate and visualize various quantities related to the solution.

4. Plotting Several plots to visualize the results, including gamma functions, trajectories, probability of not detecting, and control inputs.

4.3.1 Main components of the aforementioned algorithm

- Optimization variables, and dynamic constraints to find the optimal state and control trajectories.

```
% ---- decision variables ---------
Cellpos1 = repmat(xrange,length(yrange),1);
Cellpos2 = repmat((yrange)',1,length(xrange));
X = opti.variable(3,N+1); % state trajectory
pos  = X([1 2],:);
pos1 = X(1,:);
pos2 = X(2,:);
yaw = X(3,:);
U = opti.variable(N);   % control trajectory (throttle)

% ---- dynamic constraints --------
f = @(x,u) [speed*cos(x(3));speed*sin(x(3));u]; % dx/dt = f(x,u)
```

- An objective function that considers the cost associated with navigating through the map, avoiding obstacles, and minimizing a cost related to gamma functions.

```
% ---- objective          ---------
dt = T/N; % length of a control interval
cost = 0;
for i=1:length(yrange)
    for j=1:length(xrange)
        indic = 0;
        for k=1:(N+1)
            distsq=(pos1(k)-Cellpos1(i,j))^2+(pos2(k)-Cellpos2(i,
               j))^2;
            indic = indic+dt*gamma_distsq(distsq);
        end
        cost = cost + apf_disc(i,j)*exp(-indic);
    end
end

opti.minimize(cost);
```

- Integration using Runge-Kutta 4 for dynamics propagation.

```
for k=1:N % loop over control intervals
   % Runge-Kutta 4 integration
   k1 = f(X(:,k),         U(k));
   k2 = f(X(:,k)+dt/2*k1, U(k));
   k3 = f(X(:,k)+dt/2*k2, U(k));
   k4 = f(X(:,k)+dt*k3,   U(k));
   x_next = X(:,k) + dt/6*(k1+2*k2+2*k3+k4);
   opti.subject_to(X(:,k+1)==x_next); % close the gaps
end
```

- Control constraints to limit the control inputs. Boundary conditions for the initial position and yaw angle of the robot.

```
% ---- control constraints --------
opti.subject_to(rmin<=U<=rmax);% control is limited
% ---- boundary conditions --------
```

```
opti.subject_to(pos(:,1)==pos0);    % start at position 0 ...
opti.subject_to(yaw(1,1)==yaw0);    % start at yow 0 ...
```

- Post-processing to visualize the results, including trajectory plots and probability maps.

Here the resulting simulations show various explanations about placing the obstacles randomly on a defined map, and a trajectory is created achieving obstacle avoidance as much as possible.

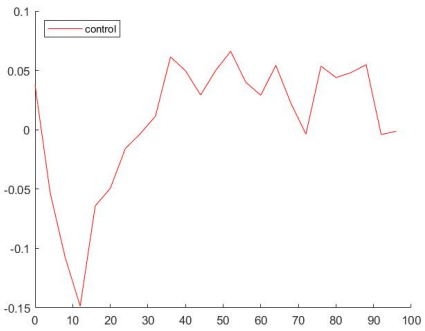

Figure 4.6: Control Signal (yaw angle)

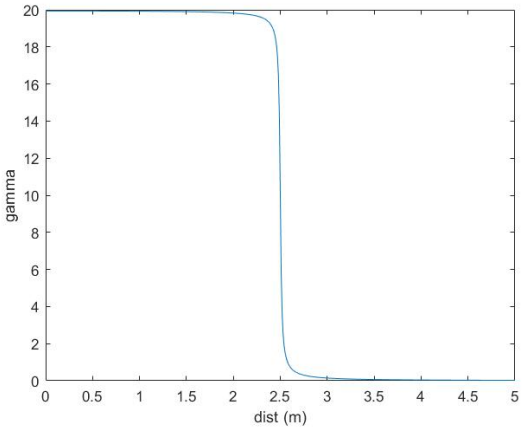

Figure 4.7: Gamma against distance

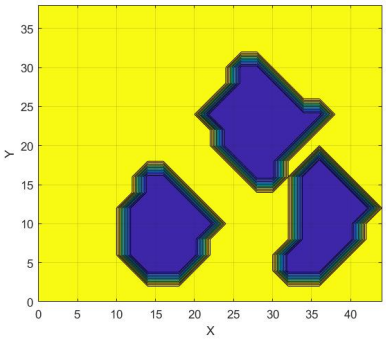

Figure 4.8: Random Obstacles placed

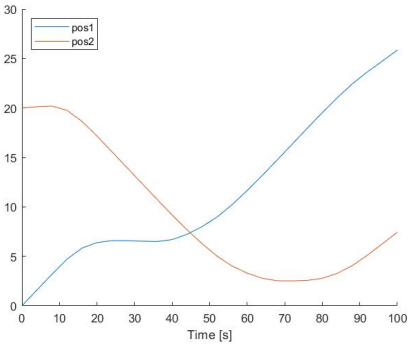

Figure 4.9: Position1 against Position2

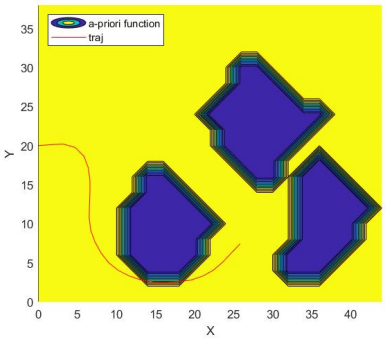

Figure 4.10: Generated Trajectory with priori distribution

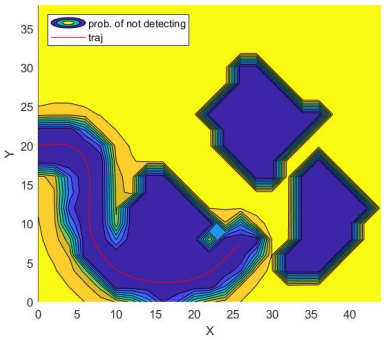

Figure 4.11: Probablity of not detecting knowing the location of the source

5

Results

Contents

5.1 Traveling Salesman Problem

The solutions and obstacles (polygons) are visualized on the same graph.

To have performance criteria, path computation is performed simply by computing the time it takes to traverse the optimal path, considering a specific speed. Moreover, computing the final positions (first coordinate and second coordinate) and the associated time. Here is the generated path for the randomly generated maps 493 and 292 using the Matlab function $rng(N)$.

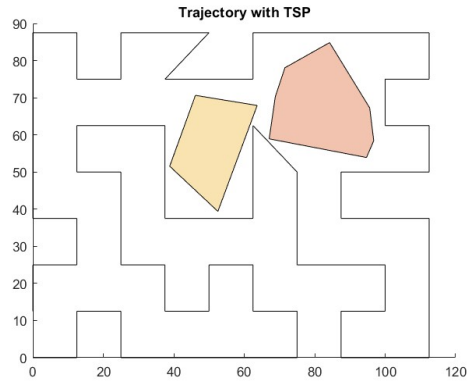

Figure 5.1: TSP map 493

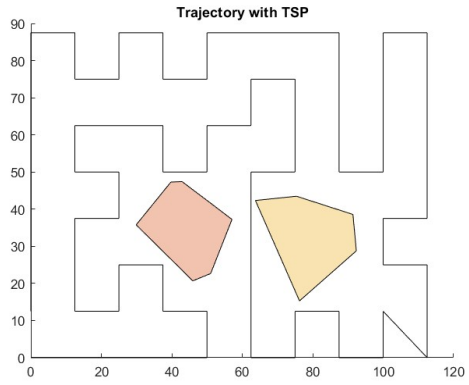

Figure 5.2: TSP map 292

5.2 Minimum Spanning Tree

Here is the generated path for the randomly generated map, starting the script with $rng(493)$ and $rng(292)$. In figures 5.3 to 5.6, a collision between the path and obstacles could be observed, however, after zooming in, it is a border-overlapping. Border-overlapping is considered a drawback of this implementation.

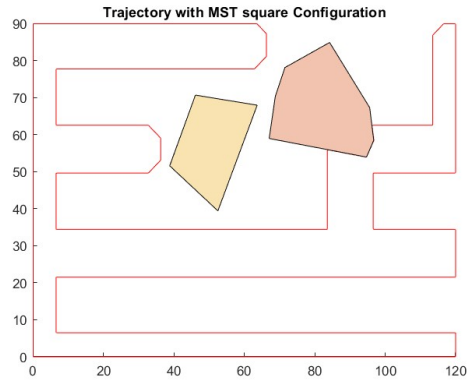

Figure 5.3: MST with square configuration map 493

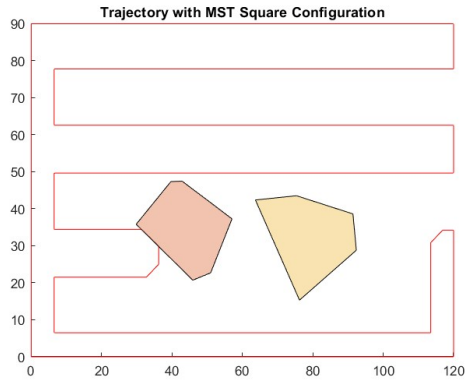

Figure 5.4: MST with square configuration map 292

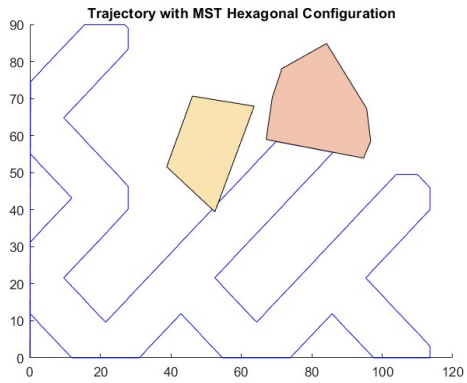

Figure 5.5: MST with hexagonal configuration map 493

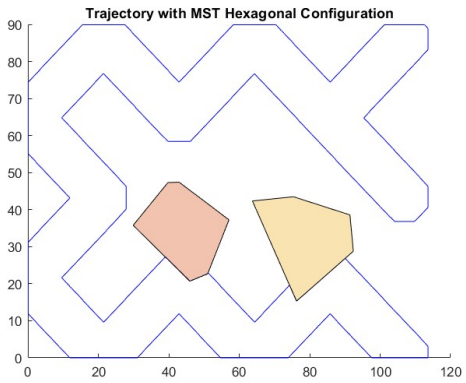

Figure 5.6: MST with hexagonal configuration map 292

5.3 Optimal Control Problem

Here is the generated path for the randomly generated map, using $rng(493)$. It can be observed that there are a lot of uncovered areas due to a large radius of detection.

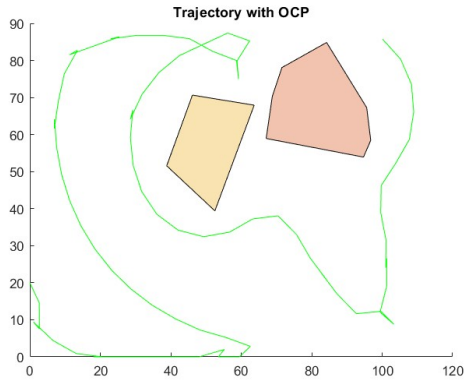

Figure 5.7: OCP Path map 493

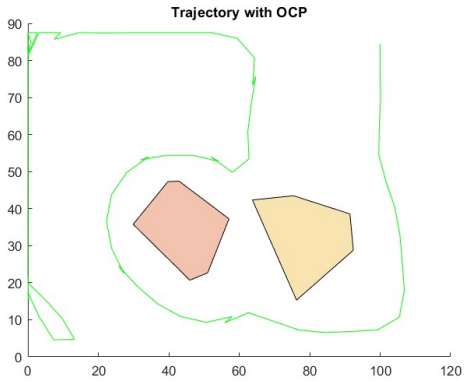

Figure 5.8: OCP Path map 292

Note: OCP failed to give a proper solution in some maps e.g. 101 due to a failure of the solver.

5.4 Discussion

In this chapter, the results of maps 493 and 292 are discussed concerning several performance criteria taken into consideration e.g. processing time of each algorithm, uncovered area, traversed path, and traversed distance. Note: due to the available computational resources, only 2 maps were analyzed as an example. First, let's start with plotting all paths together as shown in 5.9 and 5.10.

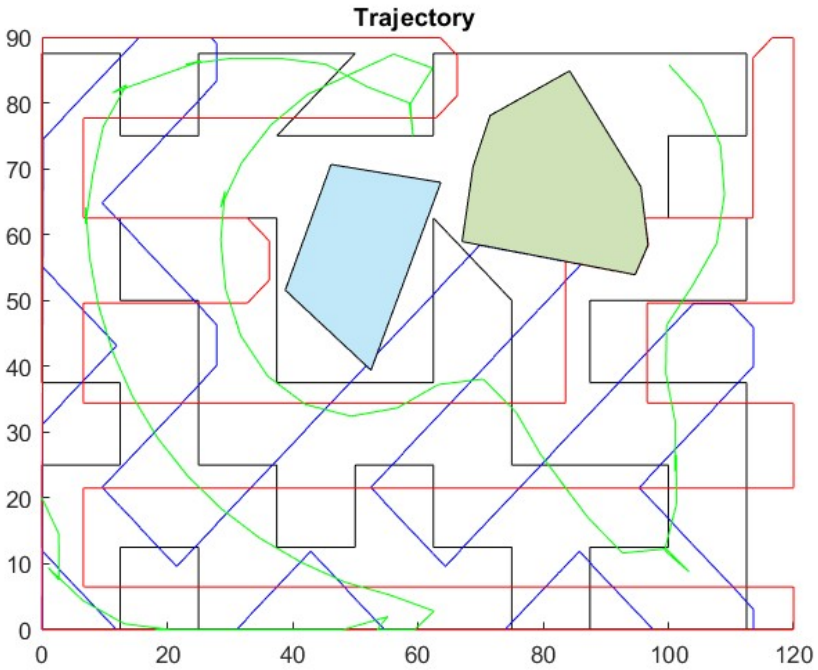

Figure 5.9: Map493: Paths of all algorithms are plotted

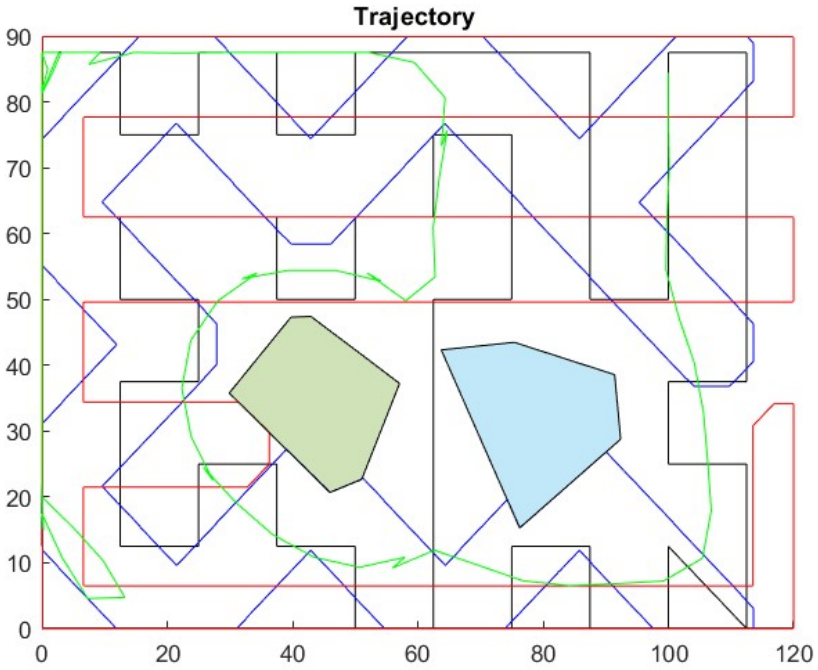

Figure 5.10: Map292: Paths of all algorithms are plotted

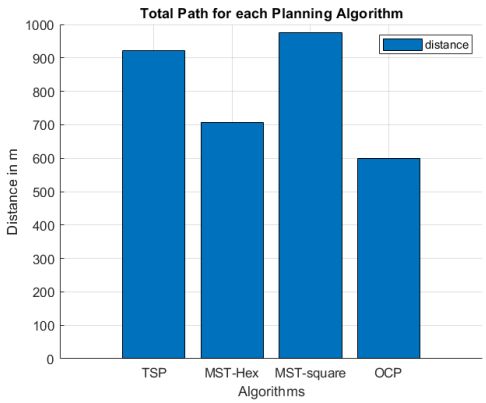

Figure 5.11: Map 493: Traversed distance taken for each algorithm

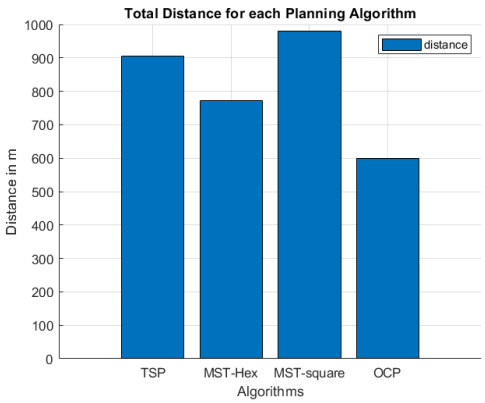

Figure 5.12: Map 292: Traversed distance taken for each algorithm

The processing time graphs are as follows:

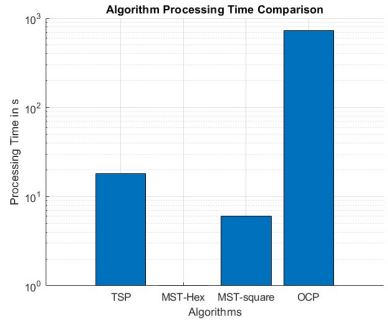

Figure 5.13: Processing time taken for each algorithm map 493

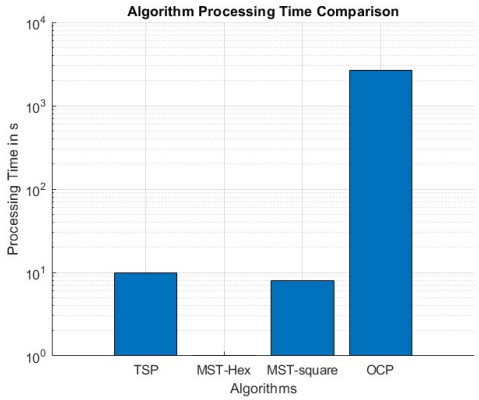

Figure 5.14: Processing time taken for each algorithm map 292

It is observed that the MST square algorithm takes the longest time to traverse the path unlike OCP, which is preset to 1500s. OCP has the shortest traversed distance, unlike MST square with the largest traversed path.

From figure 5.13, one can see that there is a prominent drawback of OCP which is the processing time with an average time between the 2 maps of about 2550s, which is the maximum processing time compared to the other algorithms while the MST Hexagonal holds the record for the least processing time with about 1s.

5.4.1 Uncovered Areas

The uncovered areas are shown in the environment for each algorithm using map 493. The uncovered areas are represented with dots.

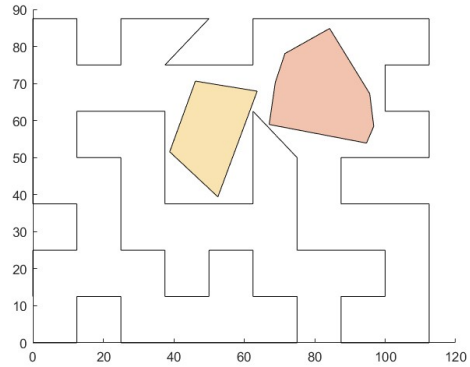

Figure 5.15: Map493: Uncovered Areas with TSP

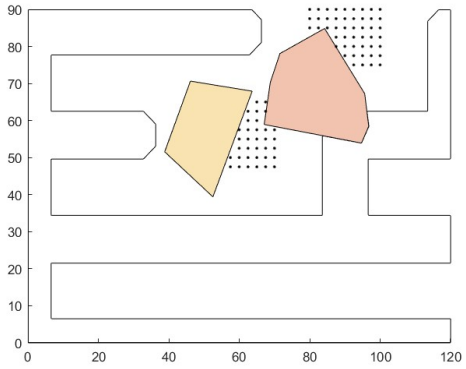

Figure 5.16: Map493: Uncovered Areas with MST square configuration

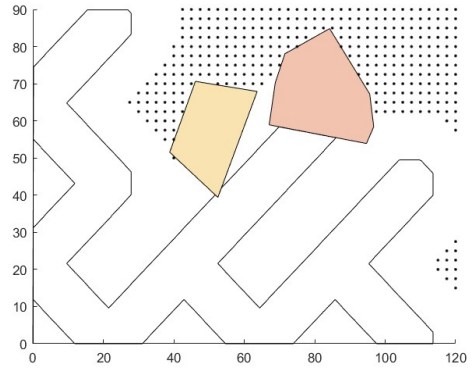

Figure 5.17: Map493: Uncovered Areas with MST Hexagonal configuration

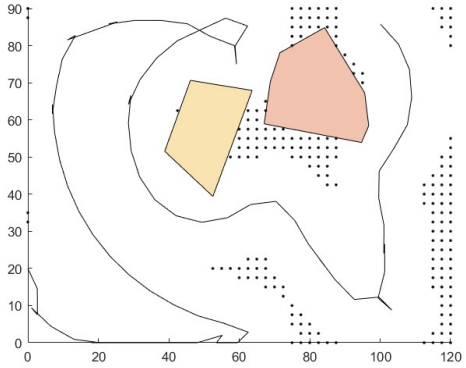

Figure 5.18: Map493: Uncovered Areas with OCP

Likely the uncovered areas for map 292 are also shown as follows:

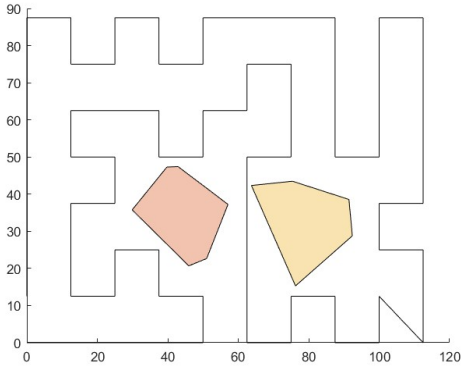

Figure 5.19: Map292: Uncovered Areas with TSP

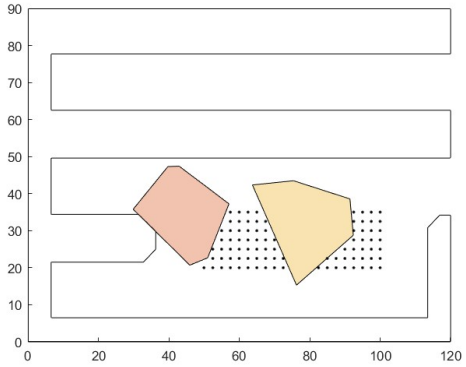

Figure 5.20: Map292: Uncovered Areas with MST square configuration

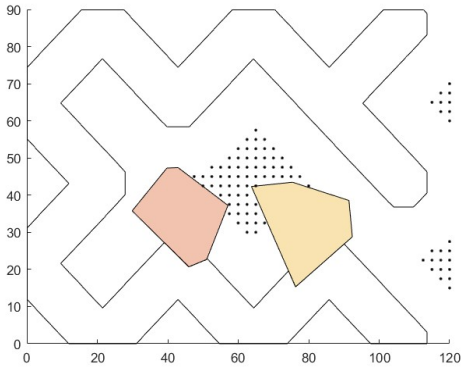

Figure 5.21: Map292: Uncovered Areas with MST Hexagonal configuration

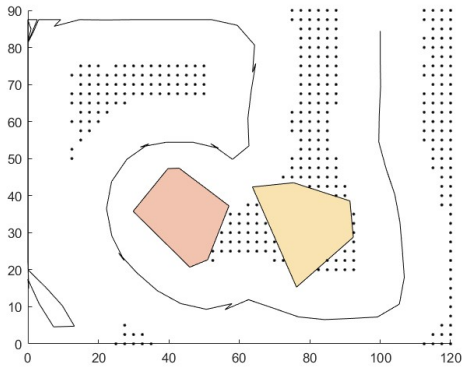

Figure 5.22: Map292: Uncovered Areas with OCP

Using the uncovered areas graphs and 5.23, one can deduce that there is a trade-off between uncovered area and traversed path. The larger the traversed path means less uncovered area. Starting from figure 5.15 to figure 5.22, there are some intersections between the path and some obstacles which means failure to compute a suitable path.

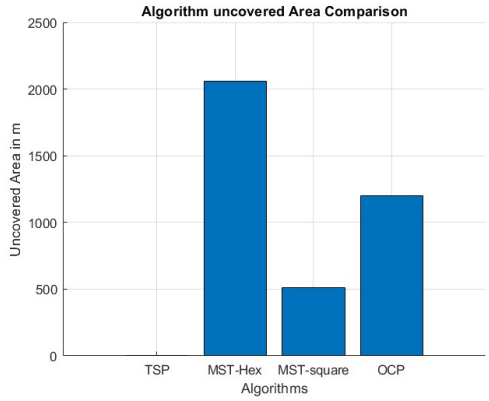

Figure 5.23: uncovered Areas for each algorithm for map 493

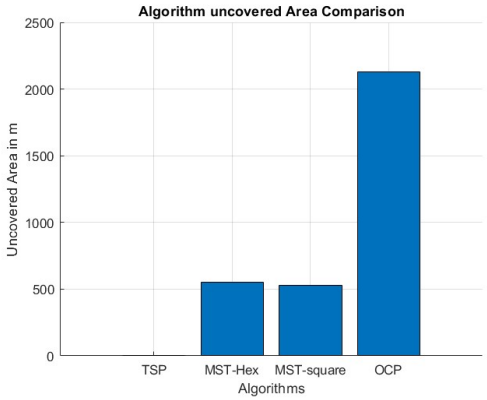

Figure 5.24: uncovered Areas for each algorithm for map 292

It is clear that the uncovered areas for the OCP are more in map 292 than 493 and this is clear from graphs 5.18 and 5.22.

In a nutshell, OCP is considered to be more useful if the time to traverse the path is constrained but has a much higher processing time than the other algorithms, and is less suitable when the objective is to cover the area, where a TSP algorithm fares better. Moreover, if the main goal is to obtain a solution fast, MST-based approaches yield smaller processing times.

OCP is more flexible to adapt to other situations where different detections could be done for diversified applications with the help of different sensors being fitted on the vehicle, where different sensors can change the simulations which was not studied in this thesis. It has been observed that there are fewer missing spots as the radius of detection is reduced but that comes with a disadvantage with more computational power since it increases the coverage area. OCP has a fixed traversed time with a drawback of more uncovered area.

The OCP approach considers the dynamics of the vehicle, unlike other algorithms. However, there is a drawback for the OCP which is having quite noticed uncovered areas, which is controlled by the parameter ”radius of detection”. As the radius increases, this will lead to more processing time but more covered area to solve the problem of high covered areas in graphs 5.18 & 5.22.

6

Conclusion and Future Work

Contents

A summary of the thesis and future developments are discussed in this chapter.

6.1 Conclusion

Trajectory generation is an important problem as part of the RAMONES project for which underwater gliders are used to monitor underwater radioactivity, which requires significant covering and traversing. Coverage algorithms are devised to generate a trajectory that maximizes the probability of detecting a radioactive source and covers the working environment as much as possible in particular through the solution of a generalized optimal control problem (OCP) with an exponential detection model. To have a fair comparison with another algorithm found in the literature review, the Travelling Salesman Problem (TSP) and Minimum Spanning Tree (MST) were implemented to analyze the trajectory generated and compare it with the OCP concerning some performance criteria e.g processing time, uncovered areas, traversed time and distance of the trajectory. OCP is considered to be more useful if the time to traverse the path is constrained but has a much higher processing time than the other algorithms, and is less suitable when the objective is to cover the area, where a TSP algorithm fares better. Moreover, if the main goal is to obtain a solution fast, MST-based approaches yield smaller processing times.

6.2 Future Work

Due to time constraints, an image processing algorithm could not be continued but it is crucial to identify and label obstacles from a Google Maps image.

Applying the Voronoi-based coverage path planning approach can lead to good results. Huang et al. [58] propose a method, for planning the paths of mowers. The approach utilizes an algorithm based on Voronoi diagrams to navigate through irregular environments. By dividing the environment into regions and generating paths that cover each region the robot can effectively cover the area while avoiding obstacles.

Bibliography

[1] "Ramones project." [Online]. Available: https://ramones-project.eu/

[2] S. J. Benkoski, M. G. Monticino, and J. R. Weisinger, "A survey of the search theory literature," *Naval Research Logistics (NRL)*, vol. 38, no. 4, pp. 469–494, 1991.

[3] S. Charmasson, P.-M. Sarradin, A. Le Faouder, M. Agarande, J. Loyen, and D. Desbruyères, "High levels of natural radioactivity in biota from deep-sea hydrothermal vents: a preliminary communication," *Journal of Environmental Radioactivity*, vol. 100, no. 6, pp. 522–526, Jun. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0265931X09000356

[4] K. Sjoeblom and G. Linsley, "Sea disposal of radioactive wastes: The London Convention 1972," *IAEA Bulletin*, vol. 36, no. 2, pp. 12–16, 1994, place: International Atomic Energy Agency (IAEA) INIS Reference Number: 25064645.

[5] J. Dabrowska, M. Sobota, M. Świader, P. Borowski, A. Moryl, R. Stodolak, E. Kucharczak, Z. Zieba, and J. K. Kazak, "Marine Waste—Sources, Fate, Risks, Challenges and Research Needs," *International Journal of Environmental Research and Public Health*, vol. 18, no. 2, p. 433, Jan. 2021, number: 2 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1660-4601/18/2/433

[6] I. A. E. Agency, "Inventory of Radioactive Material Resulting from Historical Dumping, Accidents and Losses at Sea," 2015, publisher: IAEA. [Online]. Available: https://www.iaea.org/publications/10925/inventory-of-radioactive-material-resulting-from-historical-dumping-accidents-and-losses-at-sea

[7] J. Herrmann, T. Ikaeheimonen, E. Ilus, G. Kanisch, M. Luning, J. Mattila, S. Nielsen, I. Osvath, and I. Outola, "Radioactivity in the Baltic Sea, 1999-2006 HELCOM thematic assessment," 2009, place: Finland INIS Reference Number: 41087107.

[8] Q.-H. Hu, J.-Q. Weng, and J.-S. Wang, "Sources of anthropogenic radionuclides in the environment: a review," *Journal of Environmental Radioactivity*, vol. 101, no. 6, pp. 426–437, Jun. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0265931X08001392

[9] P. Oelgaard, “Accidents in nuclear ships,” Denmark, Tech. Rep. 87-550-2266-9, 1996, nKS-RAK–2(96)TR-C3 INIS Reference Number: 28026137.

[10] S. Saremi, M. Isaksson, and K. C. Harding, “Bio accumulation of radioactive caesium in marine mammals in the Baltic Sea – Reconstruction of a historical time series,” *Science of The Total Environment*, vol. 631-632, pp. 7–12, Aug. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048969718306831

[11] G. Steinhauser, A. Brandl, and T. E. Johnson, “Comparison of the Chernobyl and Fukushima nuclear accidents: A review of the environmental impacts,” *Science of The Total Environment*, vol. 470-471, pp. 800–817, Feb. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004896971301173X

[12] R. Bachmayer, N. Leonard, J. Graver, E. Fiorelli, P. Bhatta, and D. Paley, “Underwater gliders: recent developments and future applications,” in *Proceedings of the 2004 International Symposium on Underwater Technology (IEEE Cat. No.04EX869)*, Apr. 2004, pp. 195–200.

[13] B. O. Koopman, “The Theory of Search. II. Target Detection,” *Operations Research*, vol. 4, no. 5, pp. 503–531, 1956, publisher: INFORMS. [Online]. Available: https://www.jstor.org/stable/167139

[14] D. V. Chudnovsky and G. V. Chudnovsky, *Search theory: some recent developments*. Crc Press, 2023.

[15] T. H. Chung, G. A. Hollinger, and V. Isler, “Search and pursuit-evasion in mobile robotics: A survey,” *Autonomous robots*, vol. 31, pp. 299–316, 2011.

[16] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.

[17] J. Masakuna, S. Utete, and S. Kroon, “A coordinated search strategy for solitary robots,” 2019.

[18] Z. Ismail and M. Hamami, “Systematic literature review of swarm robotics strategies applied to target search problem with environment constraints,” *Applied Sciences*, vol. 11, p. 2383, 2021.

[19] J. Yang, X. Wang, and P. Bauer, “Extended pso based collaborative searching for robotic swarms with practical constraints,” *IEEE Access*, vol. 7, pp. 76 328–76 341, 2019.

[20] Y. Luo, G. Ye, Y. Wang, L. Xie, X. Wang, S. Zhang, and X. Yan, “Toward target search approach of swarm robotics in limited communication environment based on robot chains with elimination mechanism,” *International Journal of Advanced Robotic Systems*, vol. 17, p. 172988142091995, 2020.

[21] Y. Zhou, A. Chen, X. He, and X. Bian, “Multi-target coordinated search algorithm for swarm robotics considering practical constraints,” *Frontiers in Neurorobotics*, vol. 15, 2021.

[22] R. Yehoshua, N. Agmon, and G. Kaminka, “Robotic adversarial coverage of known environments,” *The International Journal of Robotics Research*, vol. 35, pp. 1419–1444, 2016.

[23] M. Batalin and G. Sukhatme, “Coverage, exploration and deployment by a mobile robot and communication network,” *Telecommunication Systems*, vol. 26, pp. 181–196, 2004.

[24] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous Systems*, vol. 61, pp. 1258–1276, 2013.

[25] A. Le, P. Veerajagadheswar, V. Sivanantham, and R. Mohan, “Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor,” *Sensors*, vol. 18, p. 2585, 2018.

[26] G. Strimel and M. Veloso, “Coverage planning with finite resources,” 2014.

[27] P. Ku, R. Mohan, N. Nhân, and A. Le, “Graph theory-based approach to accomplish complete coverage path planning tasks for reconfigurable robots,” *IEEE Access*, vol. 7, pp. 94 642–94 657, 2019.

[28] R. Graham and P. Hell, “On the history of the minimum spanning tree problem,” *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[29] “Travelling salesman problem,” Dec. 2022, page Version ID: 1129661278. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Travelling_salesman_problem&oldid=1129661278

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE Transactions on Evolutionary Computation*, 2002.

[31] S. L. Kek, W. J. Leong, S. Y. Sim, and K. L. Teo, “Application of conjugate gradient approach for nonlinear optimal control problem with model-reality differences,” *Applied Mathematics*, 2018.

[32] L. Failer, D. Meidner, and B. Vexler, “Optimal control of a linear unsteady fluid–structure interaction problem,” *Journal of Optimization Theory and Applications*, 2016.

[33] Y. Wang, X. Luo, and S. Li, “Optimal control method of parabolic partial differential equations and its application to heat transfer model in continuous cast secondary cooling zone,” *Advances in Mathematical Physics*, 2015.

[34] X. Yan and A. C. Reynolds, “Optimization algorithms based on combining fd approximations and stochastic gradients compared with methods based only on a stochastic gradient,” *Spe Journal*, 2014.

[35] X. Zeng, P. Yi, and H. Ye, “Distributed continuous-time algorithm for constrained convex optimizations via nonsmooth analysis approach,” *IEEE Transactions on Automatic Control*, 2017.

[36] Y. Lou, H. Ye, and S. Wang, “Distributed continuous-time approximate projection protocols for shortest distance optimization problems,” *Automatica*, 2016.

[37] P. Sarma, W. H. Chen, L. J. Durlofsky, and K. Aziz, “Production optimization with adjoint models under nonlinear control-state path inequality constraints,” *Spe Reservoir Evaluation & Engineering*, 2008.

[38] S. Ghorbanpour, T. Pamulapati, and R. Mallipeddi, “Swarm and evolutionary algorithms for energy disaggregation: Challenges and prospects,” *International Journal of Bio-Inspired Computation*, 2021.

[39] M. Patterson and A. V. Rao, “Exploiting sparsity in direct collocation pseudospectral methods for solving optimal control problems,” *Journal of Spacecraft and Rockets*, 2012.

[40] D. J. Wirthman, S.-Y. Park, and S. R. Vadali, “Trajectory optimization using parallel shooting method on parallel computer,” *Journal of Guidance, Control, and Dynamics*, vol. 18, no. 2, pp. 377–379, 1995, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/3.21397. [Online]. Available: https://doi.org/10.2514/3.21397

[41] Y. Gao and C. Kluever, “Low-Thrust Interplanetary Orbit Transfers Using Hybrid Trajectory Optimization Method with Multiple Shooting,” in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, ser. Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, Aug. 2004. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2004-5088

[42] P. Gath and K. Well, “Trajectory optimization using a combination of direct multiple shooting and collocation,” Aug. 2001.

[43] Y. Meng, H. Zhang, and Y. Gao, “Low-Thrust Minimum-Fuel Trajectory Optimization Using Multiple Shooting Augmented by Analytical Derivatives,” *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 3, pp. 662–677, 2019, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.G003473. [Online]. Available: https://doi.org/10.2514/1.G003473

[44] K. Subbarao and B. M. Shippey, “Hybrid Genetic Algorithm Collocation Method for Trajectory Optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 4, pp. 1396–1403, 2009, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.41449. [Online]. Available: https://doi.org/10.2514/1.41449

[45] C. L. Walton, Q. Gong, I. Kaminer, and J. O. Royset, "Optimal Motion Planning for Searching for Uncertain Targets," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 8977–8982, Jan. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016430302

[46] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, Jan. 2017. [Online]. Available: https://epubs.siam.org/doi/10.1137/16M1062569

[47] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao, "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 6, pp. 1435–1440, 2006, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.20478. [Online]. Available: https://doi.org/10.2514/1.20478

[48] C. Hargraves and S. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/3.20223. [Online]. Available: https://doi.org/10.2514/3.20223

[49] A. Patel, S. Shield, S. Kazi, A. M. Johnson, and L. T. Biegler, "Contact-Implicit Trajectory Optimization using Orthogonal Collocation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2242–2249, Apr. 2019, arXiv:1809.06436 [cs]. [Online]. Available: http://arxiv.org/abs/1809.06436

[50] F. Fahroo and I. Ross, "Trajectory optimization by indirect spectral collocation methods," in *Astrodynamics Specialist Conference*, ser. Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, Aug. 2000. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2000-4028

[51] I. M. Ross and F. Fahroo, "A Perspective on Methods for Trajectory Optimization," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, ser. Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, Aug. 2002. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2002-4727

[52] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*. IEEE, 2007, pp. 38–47.

[53] S. Kragelund, C. Walton, I. Kaminer, and V. Dobrokhodov, "Generalized Optimal Control for Autonomous Mine Countermeasures Missions," *IEEE Journal of Oceanic Engineering*, vol. 46, no. 2, pp. 466–496, Apr. 2021, conference Name: IEEE Journal of Oceanic Engineering.

[54] M. Peasgood, C. M. Clark, and J. McPhee, “A complete and scalable strategy for coordinating multiple robots within roadmaps,” *IEEE Transactions on Robotics*, vol. 24, pp. 283–292, 2008.

[55] D. G. Macharet and M. F. M. Campos, “A survey on routing problems and robotic systems,” *Robotica*, vol. 36, pp. 1781–1803, 2018.

[56] “CasADi,” accessed: 29/10/2023. [Online]. Available: https://web.casadi.org/

[57] A. Wächter, “Ipopt documentation,” https://coin-or.github.io/Ipopt/, accessed: 29/11/2023.

[58] K.-C. Huang, F.-L. Lian, C.-T. Chen, C.-H. Wu, and C.-C. Chen, “A novel solution with rapid voronoi-based coverage path planning in irregular environment for robotic mowing systems,” *International Journal of Intelligent Robotics and Applications*, vol. 5, no. 4, pp. 558–575, 2021.