Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Sander Furre

Path and Motion Planning for Unmanned Surface Vehicles subject to the International Regulations for Preventing Collisions at Sea

Master's thesis in Cybernetics and Robotics
Supervisor: Konstantinos Alexis
Co-supervisor: Mihir Dharmadhikari
June 2022

NTNU
Norwegian University of Science and Technology

Sander Furre

Path and Motion Planning for Unmanned Surface Vehicles subject to the International Regulations for Preventing Collisions at Sea

Master's thesis in Cybernetics and Robotics
Supervisor: Konstantinos Alexis
Co-supervisor: Mihir Dharmadhikari
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis explores a method to enable mission planning and collision avoidance for Unmanned Surface Vehicles in vast coastal environments without human intervention. The designed and implemented guidance system follows the hybrid robotic paradigm, with a global static mission planner and a local collision avoidance strategy in a dynamic window. The mission planner searches for a safe and optimized path in the continuous configuration space of the ownship by utilizing a specialized adaptation of the Hybrid A* algorithm. The search algorithm is facilitated by the support of a vessel model with three Degrees of freedom (DOF), a specialized framed region quadtree builder, and a Voronoi skeleton generator relying on a novel generation technique. The vessel model is used for motion sampling, while the quadtree is utilized for fast collision checking and accurate distance-to-goal approximations through specialized A* graph search. Reasonable margin of safety to hazards in the nominal path is ensured by utilization of the Voronoi field, which requires the Voronoi skeleton. The generated mission plan is tracked utilizing a Line-of-sight (LOS) guidance law, and collision avoidance is handled locally using predictive control behavior selection, supported by simulation and a rules-aware cost function. Both the mission planner and collision avoidance systems are designed to comply with the *International Regulations for Preventing Collisions at Sea*.

The complete guidance system is implemented with modularity in mind using C++ nodes in ROS Noetic, and testing has been conducted in a simplified simulation environment. In testing, the guidance system in general exhibits feasible, optimized, rules-compliant, and collision-free behaviors in mission planning and collision avoidance across a multitude of scenarios. However, some deficiencies do exist. The mission planner, while compliant, struggles with search progression in the presence of some Traffic Separation Scheme elements. Furthermore, the collision avoidance system occasionally does not appropriately state the ownship intent and, in rare cases, fails to act with acceptable predictability and safety margins to obstacle vessels. The thesis concludes that utilizing sample-based motion planning in combination with predictive control behavior selection collision avoidance as a basis for collision-free guidance of maritime vessels, while not without flaws, is a promising combination of technologies with much potential for use in a rules-compliant maritime guidance system for large-scale environments. Furthermore, while the current implementation is by no means perfect and not ready for real-world testing yet, it acts as a proof of viability and a foundation on which future research should be built.

Sammendrag

I denne masteroppgaven utforskes en metode for å muliggjøre ruteplanlegging og kollisjonsunngåelse for overflatefartøy i storskala kystområder uten menneskelig innblanding. Guidingsystemet som er designet og implementert følger en hybridmodell, med en global statisk ruteplanlegger og en lokal strategi for kollisjonsunngåelse i et dynamisk vindu. Ruteplanleggeren søker etter en trygg og optimert rute i det kontinuerlige konfigurasjonsdomenet til fartøyet gjennom å utnytte en spesialisert versjon av algoritmen Hybrid A*. Søkealgoritmens hovedoppgave blir fasilitert av en 3DOF fartøymodell, et spesialisert regionalt Quadtree og en Voronoi skeleton generingsalgoritme. Fartøymodellen blir brukt til å utføre ekspansjon av rutesøket, Quadtreet blir brukt til rask kollisonsjekking og for å approksimere distanse til mål fra en gitt konfigurasjon, og Voronoi skeleton strukturen legger grunnlaget for å utnytte det kunstige Voronoi kraftfeltet som muliggjør relativ sikkerhetsmargin i ruteplanleggingen. Den genererte ruteplanen blir fulgt av en Line-Of-Sight (LOS) følgelov, og kollisjonsunngåelse blir håndtert lokalt ved bruk av predikativ, simulasjonsstøttet kontrollseleksjon og en regel-bevisst kostfunksjon. Både ruteplanleggeren og systemet for kollisjonsunngåelses er utformet for å følge *Konvensjon om internasjonale regler til forebygging av sammenstøt på sjøen*.

Det fullstendige guidingsystemet er implementert med søkelys på modularitet gjennom C++ noder i ROS Noetic og har blitt testet i et forenklet simulasjonsmiljø. Gjennom testing har systemet utvist optimert og kollisjonsfri oppførsel som følger spesifiserte internasjonale sjøregler i varierte scenarier. Likevel er det ikke uten mangler. Ruteplanleggeren klarer å følge regler for trafikkseparasjon, men progresjonen i planleggingen stagnerer i nærheten av enkelte separasjonselementer. Videre utviser ikke kollisjonssystemet alltid tydelig intensjon om vikemanøver i møte med andre skip, og i sjeldne tilfeller opprettholdes ikke tilstrekkelig sikkerhetsmargin og forutsigbarhetskrav til andre skip. Det konkluderes med at Hybrid A* bevegelsesplanlegging og prediktiv kontrollseleksjon som en basis for kollisjonsfri guiding av maritime overflatefartøy i storskala kystmiljø er en lovende teknologikombinasjon. Selv om implementasjonen som følger denne masteroppgaven ikke er perfekt eller klar for sjøprøver enda, så viser den at et slikt system kan fungere i praksis. Implementasjonen og arbeidet i denne masteroppgaven er også tiltenkt som et grunnlag videre arbeid og forskning om ønskelig kan bygge på.

Preface

This master thesis concludes my five years at the Norwegian University of Science and Technology in Trondheim and is carried out in the spring semester of 2022 at the Department of Engineering Cybernetics. It introduces a proposed design and implementation of a complete guidance system and supportive framework for collision-free motion planning in dynamic coastal environments at a large scale. This thesis aims to contribute toward the development of autonomous maritime vessels capable of mission planning and collision avoidance without human intervention by laying a foundation for further practical academic research.

The work in this thesis relies heavily on the use of Electronic Navigational Charts, and I would like to thank Sølvi Tunge at PRIMAR and Gjertrud Røyland and Jostein Thorsen at The Norwegian Mapping Authority for their assistance with obtaining charts from the Norwegian coastline. Furthermore, I would like to express my gratitude to my supervisor Konstantinos Alexis and Co-supervisor Mihir Dharmadhikari, who have guided my research throughout the semester and provided invaluable feedback through bi-weekly meetings. You have provided new knowledge and perspective and been a source of great inspiration.

The guidance system developed in this thesis is built in a Ubuntu 20.04 Docker container and utilizes ROS Noetic as a framework. Core components rely heavily on earlier open-source efforts through the development of open-source libraries and systems. Some notable mentions are the *GBPlanner* system developed at the NTNU Autonomous Robots lab [44] used for building and maintaining graphs, the *Geospatial Data Abstraction Library* [57] utilized to read and manipulate Electronic Navigational Charts, *GeographicLib* [51] utilized to solve problems related to Geodesy and *OpenMP* [14] utilized to enable concurrency within node procedures. As with most software developed in C++, the C++ Standard Library has also been of immense value. Before proceeding, I thus want to express my sincere gratitude to the developers, maintainers, and facilitators of all open-source software used in this thesis. Making this project come to life would not have been possible without the decades of effort you all have contributed.

Sander Furre

Trondheim, June 6, 2022

Contents

List of Tables

List of Figures

Acronyms

API Application Programming Interface. 45

COLAV Collision avoidance. 1–3, 19, 30, 35, 42, 43, 59, 63, 64, 67, 69, 71, 74, 75, 88–92, 96, 97, 109

COLREG International Regulations for Preventing Collisions at Sea. 2–4, 6, 7, 30, 39, 42, 43, 59, 62, 64–71, 75, 76, 78, 89, 90, 92, 95–97, 104

CTRS Conventional, Terrestrial Reference System. 27

DOF Degrees of freedom. i, 21, 27, 36

DSID Data Set Identifier. 48

ECDIS Electronic Chart Display and Information System. 24, 31

ECEF Earth-centered, Earth-fixed. 27, 36

ECI Earth-centered inertial. 36

ENC Electronic Navigational Chart. 1, 2, 7, 24–27, 31, 32, 35, 45, 47–49, 52, 59, 63, 95, 97, 109

ENU East-north-up. 36, 41, 46, 89, 91

GCS Geographic Coordinate System. 36, 42

GDAL/OGR Geospatial Data Abstraction Library. 26, 27, 45, 46, 48, 97

GNC Guidance, Navigation and Control. 4, 18

GNSS Global Navigation Satellite Systems. 19

IHO International Hydrographic Organization. 24–27

IMO International Maritime Organization. 24

LOS Line-of-sight. i, 19, 41, 42, 67, 71, 92, 95

NOAA National Oceanic and Atmospheric Administration. 24, 47

POI Point of Interest. 89, 90, 92

ROS Robot Operating System. 44, 45

RRT Rapidly-Exploring Random Trees. 12

SPA Sense-plan-act. 13

SSA Smallest Signed Angle. 23

TSS Traffic Separation Scheme. 3, 6, 32, 39, 40, 57, 64, 76–78, 95–97, 104, 107

USV Unmanned Surface Vehicle. 1–7, 19, 30–32, 35, 41, 42, 51, 54, 58–62, 64–75, 77, 88–95, 104

Chapter 1

Introduction

1.1 Motivation

Developing robots capable of performing complex tasks without human intervention has for half a century been a research topic with significant focus, rapid progress, and significant technological advances. From its humble beginnings as a topic primarily of academic interest, autonomous systems have, over the last couple of decades, become more capable and started to play an ever-larger role in robotics for commercial and military applications. In maritime robotics, autonomy enables the development of unmanned surface vessels capable of performing a variety of environmental, transportation, and reconnaissance missions. These vessels are commonly known as Unmanned Surface Vehicle (USV)s and can have varying levels of autonomous capabilities. Depending on the autonomous capabilities of the vessel, it is typically controlled, assisted, or supervised by an operator. Many missions will require a vessel to traverse large distances in coastal waters. By introducing a sufficient level of autonomy where the vessel is capable of autonomous rules-compliant and collision-free guidance, a supervising operator can focus attention on other aspects of environmental and reconnaissance missions, such as data analysis. On transportation missions, such as ferry crossings or coastal shipping, the autonomous capabilities can allow the supervising operator to simultaneously manage a fleet of vessels.

For an USV to traverse coastal areas without relying on human intervention, it must be able to make a strategic mission plan and follow this plan in a dynamic environment. A guidance system plays a pivotal role in enabling this ability and typically consists of a global mission planning strategy and a local collision avoidance strategy. Motion planning supported by data from Electronic Navigational Chart (ENC)s can be used to develop the former, ensuring that the USV obtains a path that is not only safe and optimized but also feasible to follow. A mission plan does not necessarily account for dynamic hazards, thus a Collision avoidance (COLAV) system can be required to ensure collision-free guidance at all times. What is more, the COLAV system must ensure compliance with the *"rules of the*

road" for safe mission operations. Defined in the International Regulations for Preventing Collisions at Sea (COLREG)s for the purpose of safe navigation, compliance with these rules is of utmost importance for any USV wishing to operate safely and lawfully in the presence of other vessels.

1.2 Problem formulation and contributions

To contribute towards the development of guidance systems for autonomous USVs, the objective of this thesis is to design, implement and experimentally verify a complete guidance system intended for usage in large-scale coastal environments with other vessels present. In order to be viable, the mission planner must produce a feasible, optimized mission path, be rules-compliant, and require only a minimal amount of information from the supervisor to generate a mission plan with a reasonable computational cost. The COLAV system must be COLREG compliant, mitigate risk and adhere to strict real-time requirements. Moreover, actions taken by both subsystems must be predictable, explainable, and require only limited, readily available a priori and sensory information to operate. This thesis focuses on utilizing state-of-the-art sample-based motion planning in combination with a collision avoidance strategy based on motion primitives in practical applications. The research question on which the thesis is based is the following:

- *Is combining sample-based motion planning and motion primitives collision avoidance in a modular framework viable to enable optimized, predictable and COLREG compliant guidance for an Unmanned Surface Vehicle in vast dynamic environments?*

Building on recent research in sample-based motion planning, a relatively mature concept for collision avoidance and notably the research presented in [21], [22], [41], [44] and [16], the goal is to develop a complete system that is able to adapt to a variety of scenarios and can be used on any USV. Moreover, the system should rely only on novel algorithms and open-source libraries to encourage the use and continued development of the proposed guidance system.

The main contributions of this thesis are:

- A strategy for motion planning in coastal environments of large scale using an adaptation of the Hybrid A* Algorithm supported by a regional framed Quadtree, a Voronoi skeleton, and a novel sequence matching procedure.
- A proposed design for a complete guidance system including COLREG compliant collision avoidance, active nominal path tracking, and utilization of ENCs.
- An implementation of the complete guidance system as a C++ software system utilizing ROS Noetic and several other open-source software libraries.
- An implementation of a standalone ENC data extraction library written in C++ for reading S-57 charts and extracting data relevant for a vessel guidance system.
- A simulation-based evaluation of the real-world applicability, COLREG compliance, strengths, and limitations of the guidance system.

1.3 Assumptions, simplifications and limitations

Some key assumptions and simplifications have been set to ensure development focus and progression. The following is assumed:

- A target tracking system is readily available, updating tracked vessel states at no less than $1Hz$
- The USV is fitted with a navigational unit, providing all necessary ownship data.
- The USV is fitted with a course control autopilot and speed control.

The assumptions must be satisfied when running the guidance system to avoid undetermined behavior. Simplifications specify a set of aspects regarding the guidance system and simulation environment. It should be noted that the guidance system is designed to function in the absence of these simplifications. The following core simplifications have been made during the development:

- The vessel will not be subjected to environmental disturbances in simulation environment or elsewhere.
- All measurements and estimates are noiseless
- Obstacle vessels do not comply with COLREG rules.

The limitations highlight functionality that one might expect from a complete guidance system but that is not present in the design or development of the proposed guidance system. The key limitations are the following:

- Only collision avoidance COLREGs apply in the COLAV system. Thus, there is no way for the COLAV system to intentionally comply with location-specific rules or a Traffic Separation Scheme (TSS).
- The mission planner does not consider weather phenomena when searching for an optimized path.

1.4 Outline

This thesis consists of 7 chapters. Chapter 1 introduces the motivation for the thesis, defines the key research question, and highlights the thesis contributions. Additionally, key assumptions, simplifications, and limitations are highlighted for the reader's convenience. Mission planning and collision avoidance are topics where significant research efforts have been undertaken before, and in Chapter 2 the research efforts considered most relevant for this thesis are reviewed. Thereafter, Chapter 3 covers the relevant theory and research on which all aspects of the guidance system in this thesis are built. Chapter 4 proposes the complete guidance system, while technical details and pseudocode of crucial procedures and novel algorithms are elaborated upon in Chapter 5. Testing results are presented in Chapter 6, with accompanying evaluation and discussion. Finally, Chapter 7 concludes the research effort in this thesis and suggests future work to overcome the limitations of the proposed and implemented guidance system.

Chapter 2

Literature Review

The research focused on robot motion planning has been integral to the development of autonomous systems since the introduction with *Shakey the robot* in 1984 [4]. Decades of research have since come and gone, yet the field remains divided on how to solve the motion planning problem and related path planning problems. Any reader interested in a general overview is recommended to consult Latombe [10] and LaValle [20]. This literature review will focus on contributions to motion planning and collision avoidance considered relevant for autonomous maritime vessels.

2.1 Overview Papers

In order to develop a viable guidance system for USVs, it is not only essential to consider algorithm candidates but also to establish a solid foundation of guidance system terminology, regulatory aspects, and the general overall framework for enabling autonomous operations. In 2020, Vagale, Oucheikh, Bye, Osen and Fossen [53] conducted a comprehensive review of autonomous surface vehicles, focusing on path planning and collision avoidance primarily but also attempting to improve the consistency in this field of research by reviewing terminology, design choices, regulatory aspects and the typical Guidance, Navigation and Control (GNC) structure facilitating autonomous operations for such vessels. The same authors have also compared 45 path planning and collision avoidance algorithms in another paper accompanying the former [52]. While calling it a comparison of path planning methods, some algorithms consider vessel dynamics organically and can be considered motion planning methods. Several algorithms are promising, and a handful has gone through sea trials. However, while some rely on conventional planning methods, most are stochastic and rely on reinforcement learning and other artificial intelligence methods that do not give sufficient insight into internal reasoning. To guarantee COLREG compliance and ensure consistent and predictable guidance behavior, a system with deterministic logic can be beneficial over stochastic solutions. Such a system will inevitably

have several tuning parameters. However, it will not require a historic dataset or a very realistic simulator for training purposes and thus might be more convenient to adapt and deploy. The following review of some exciting and relevant mission planning and collision avoidance strategies focuses on methods that give ample insight into the reasoning and do not require model training.

2.2 Path and Motion planning algorithms

Liu, Song and Bucknall [31] propose a global static path planning strategy called AFMS based on the fast marching method [13] and fast marching square [26]. Like Djikstra's algorithm and focused search methods such as A* and its variations, AFMS relies on a binary grid map. Contrasting conventional grid planning methods, FMS has characteristics of continuity and smoothness [26] and the authors thus argue that a USV capable of tracking a continuous path will be able to follow it [31]. Sea trials of the AFMS method on the *Springer USV* had promising results, and cross-track error was significantly reduced compared to using A* directly [31]. There are some limitations to this approach. For one, it does not entirely accommodate the dynamics of an USV. The *Springer USV* is a small agile vessel with differential drive, making it vastly more maneuverable than any under-acted monohull vessel with a single fixed-pitch propeller and rudder setup. Furthermore, it relies on a binary occupancy map and is only tested for areas up to 500×500 square meters in simulation and sea trials. In addition to the potential difficulty with scaling the approach to handle map sizes of e.g. 100×100 square kilometers, using a flat binary map will not be viable for large mission areas where the Earth's curvature cannot be neglected and geodetic coordinate systems must be used. Thus, the method can be said to be limited to smaller areas, such as harbors.

Shah and Gupta [47] propose a global static mission planner based on speeding up A* on a visibility graph for large-scale path planning in marine environments. The authors use a quadtree world representation and a novel admissible heuristic to facilitate the traversal of vast environments. Furthermore, Electronic Navigational Charts provide sufficient a priori information about the mission region, and a dynamically changing traversal space due to weather and tide factors are considered. While this method proposes a solution to large-scale planning for autonomous vessels, it does not consider the vessel dynamics. Thus, while a path can be well optimized, it is not guaranteed that an arbitrary USV will be able to follow it. Moreover, the ability to traverse challenging geometry such as narrow channels, straits, and bays remain unclear. Additionally, proximity to land at the waypoints appears to be an issue.

Dolgov, Thrun, Montemerlo and Diebel [21] propose a motion planning technique called Hybrid A*, based on combining the core aspects of A* and sample-based motion planning to create smooth, feasible paths for non-holonomic vehicles such as cars. While combining A* with sample-based motion planning loses the A* path optimality guarantee, the path remains optimized. Furthermore, the method is general enough to apply to other robots such as unmanned surface vehicles and can potentially be adapted to work in vast maritime environments. Being a focused search technique, Hybrid A* relies on a world model facilitating search heuristic evaluation. Such an evaluation can be computation-

ally expensive for large environments, thus while it can be used iteratively in small areas to facilitate collision avoidance, this is not considered viable for vast maritime environments. From the perspective of large-scale motion planning for maritime vessels, it must be considered a global, static motion planner that cannot guarantee collision-free operation whenever dynamic obstacles are present. Supposing it can work together with a local collision-avoidance system and scale reasonably with environment size, it can be a potent candidate for mission planning in a guidance system for an autonomous surface vessel. A research effort undertaken by Rothmund [42] indicates that Hybrid A* can be used for maritime applications, not only to create feasible paths but also to comply with TSS and facilitate complex path optimization strategies with weather constraints.

2.3 Collision avoidance algorithms

Chiang and Tapia [40] propose an adaptation of the RRT algorithm [15] with COLREG compliance called COLREG-RRT for non-holonomic surface vessels. It is a complete guidance system solution, not only a collision avoidance strategy. The COLREG-RRT algorithm intends to identify long prediction trajectories and thus aid with collision avoidance in more complex avoidance situations. By relying on joint forward simulation of the unmanned surface vehicle and obstacle vessels, it can accommodate more advanced obstacle behaviors. Consequently, it is not as reactive as the strictly local methods. COLREG-RRT has been tested in both single-ship and multi-ship encounters and compared with MPC-based and APF-based methods. According to the independent algorithm comparison evaluation in [52], its main drawback is that it does not deal with disturbances such as waves and ocean currents. Scaling the system to operate in vast environments while maintaining sufficient real-time capabilities appear difficult at best. Due to the lack of field testing, the possible difficulty of introducing environment disturbances, and uncertainties around scaling, it is considered a highly experimental method where the viability of usage in the proposed guidance system in this thesis is doubtful.

Kuwata, Wolf, Zarzhitsky, and Huntsberger [29] propose using the much more reactive Velocity Obstacles method for COLREG compliant maritime navigation. Velocity obstacle methods generate cone-shaped obstacle regions in the velocity space and allow for COLREG compliance encoding straightforwardly. What is essentially determined is the velocity of the USV to avoid the obstacle cone. Two key advantages of this method are the frequency at which it can run and its robustness to ambiguous COLREG situations. The authors used a relatively small 32-by-128 velocity space grid for their testing in simulations and sea trials and found that the real-time performance was satisfactory with over 20 obstacles present in the space [29]. From the perspective of large-scale missions, the velocity space grid would probably have to be a limited dynamic window that follows the unmanned surface vessel to maintain acceptable real-time performance. Determining how and when such a grid should move to avoid potential critical edge-case situations appears nontrivial but certainly not impossible. In the evaluation in [52], advantages of this method mentioned are the intent the vessels show of following the COLREGs and the ability to safely navigate cluttered environments. In its characterization, [52] describes the method as appropriate for open waters, not congested waters or coastal areas. While

obstacle vessels and the COLREGs encode naturally in velocity space, static obstacles like buoys, land, and shallow waters do not. The collision avoidance method can thus not intentionally avoid static obstacles while complying with COLREGs, risking grounding in areas with limited sea room. Furthermore, while the method takes advantage of the velocity space, it is not inherently aware of the USV dynamics. It can henceforth potentially request behaviors the vessel is incapable of performing.

Loe [22] proposes a local collision avoidance method based on the dynamic window algorithm, with modifications to incorporate COLREGs, improve performance and expand the use of vessel dynamics. Initial sea trials were performed and showed great potential [22]. Several research projects have sprung out of this effort, with Johansen, Perez and Cristofaro [33] introducing predictive control selection and a proposed design for interaction with a mission planner module. This concept was initially only tested in simulations. Hagen expanded on the concept in her master thesis [35], while Hagen, Kufoalor, Brekke and Johansen [41] performed sea trials with the system in the fjord of Trondheim where it showed great potential in real situations. Later research efforts have also been undertaken to expand upon the concept, with Otterholm [43] exploring utilization of ENC hazard interpretation in this collision avoidance context and Kjerstad [45] proposing the utilization of obstacle vessel intentions. While still experimental, the method has gained research traction, showed potential, and proven viable in complex situations in coastal areas.

Chapter 3

Background

3.1 Motion Planning

3.1.1 The motion planning problem

In general, robot motion planning is a problem concerned with finding an optimal feasible sequence of motions to get a robot from an initial configuration q_I to some goal configuration q_G in the presence of obstacles and external forces [10]. The robot can be a vessel or vehicle traveling autonomously, but it can generally be any robot that interacts with some environment. Limiting the scope to motion planning for mobile robots with the purpose of navigation in a two-dimensional world model, motion planning can be formulated as an extension of the 2D Piano Mover's Problem according to Formulation 7.1 in [20]. This formulation is paraphrased below based on [49], with symbolic notation unchanged for convenience to any reader familiar with the original book.

1. A world environment $\mathcal{W} = \mathbb{R}^2$ in which the problem is to be solved must be clearly defined.
2. Solving the problem takes some time, defined by a time interval $T \subset \mathbb{R}$. T can either be bounded such that $T = [0, t_f]$ for some final time $t_f > 0$, or it can be unbounded such that $T = [0, \infty)$
3. There exists a semi-algebraic, time-varying obstacle region $\mathcal{O}(t) \in \mathcal{W} \forall\, t \in T$. The obstacle region is assumed to be a finite collection of rigid bodies. While some undergoes continous, time-dependent rigid-body transformations others can be static.
4. The purpose of solving the motion planning problem is to provide guidance to some semi-algebraic robot $\mathcal{A} \in \mathcal{W}$.
5. The motion planning problem is solved in a configuration space $\mathcal{C}$, the set of all possible transformations that may be applied to the robot. Based on $\mathcal{O}(t)$ two subsets

$\mathcal{C}_{obs} \in \mathcal{C}$ and $\mathcal{C}_{free} \in \mathcal{C}$ are derived, which represent the occupied and unoccupied configurations respectively.

6. Including time to configurations, the state-space $\mathcal{X} = \mathcal{C} \times T$ is introduced[1]. A robot state $x_{\mathcal{A}} \in \mathcal{X}$ is denoted as $x_{\mathcal{A}} = (q, t)$. With the introduction of time the obstacle state-space $\mathcal{X}_{obs}$ can be defined as

$$\mathcal{X}_{obs} = \{(q, t) \in \mathcal{X} | \mathcal{A}(q) \cap \mathcal{O}(t) \neq \emptyset\} \tag{3.1}$$

and $\mathcal{X}_{free} = \mathcal{X} \backslash \mathcal{X}_{obs}$.

7. An initial state $x_I \in \mathcal{X}_{free}$ defines where $\mathcal{A}$ is at $t = 0$.

8. Based on a goal in $\mathcal{W}$, $\mathcal{X}_g \subset \mathcal{X}_{free}$ is designated as the goal region in the state space. One can for example pick a configuration $q_G \in \mathcal{C}$ and let $\mathcal{X}_G = \{(q_G, t) \in \mathcal{X}_{free} | t \in T\}$.

9. The last component is the solution, the path $\tau[0, 1] \rightarrow \mathcal{X}_{free}$. Here, time is scaled to be in the interval $[0, 1]$ The path must be continous and time-monotonic[2] such that $\tau(0) = x_I$ and $\tau(1) \in \mathcal{X}_G$. If it is impossible to calculate τ it must be reported that such a path does not exist.

3.1.2 Configuration space

The configuration space $\mathcal{C}$ is the set of all possible rigid-body transformations that can be applied to the robot [20]. For example, considering an arbitrary triangle-shaped robot navigating a 2D world representation, its configurations will be of the form $q = (x_t, y_t, \theta)$. A robot navigating the physical world $\mathcal{W}$ requires some guidance through a planning strategy. However, the planning problem is typically solved in the configuration space. The reason for this is that the configuration space abstraction layer makes most inherently different motion planning problems solvable by the same planning algorithms [20].

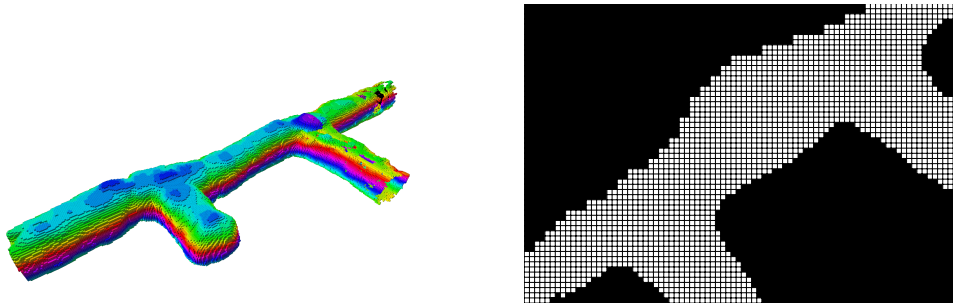
The reader is advised that while all robot configurations are of the form q, not all q are in the configuration space. For the 2D example above, $\theta \pm 2\pi$ yields equivalent rotations. Thus, while $q \subset \mathbb{R}^3$, we write that $\mathcal{C} = \mathbb{R}^2 \times \mathcal{S}^1$, where $\mathcal{S}^1$ is a topological circle [20]. Failing to recognize this property when solving the path planning problem will make the configuration space artificially large and thus add unnecessary planning complexity.

3.1.3 World models

Any autonomous system must sense, interpret and store information about the real world in an internal world model. This world model can be used to discretize the environment, for reasoning and predictions, and evaluations internally in supportive frameworks of the robot. While most world models are application-specific, many are adaptations from a handful of well-established models in path and motion planning. One example is volumetric world models used in mapping libraries like OctoMap [27] and Voxblox [37] and

[1] $\times$ is the Cartesian product

[2] The path is time-monotonic if given $(q_1, t_1) = \tau(s_1)$ and $(q_2, t_2) = \tau(s_2)$ it is guaranteed that $\forall\ s_1, s_2 \in [0, 1]$ the implication $s_1 < s_2 \implies t_1 < t_2$ holds.



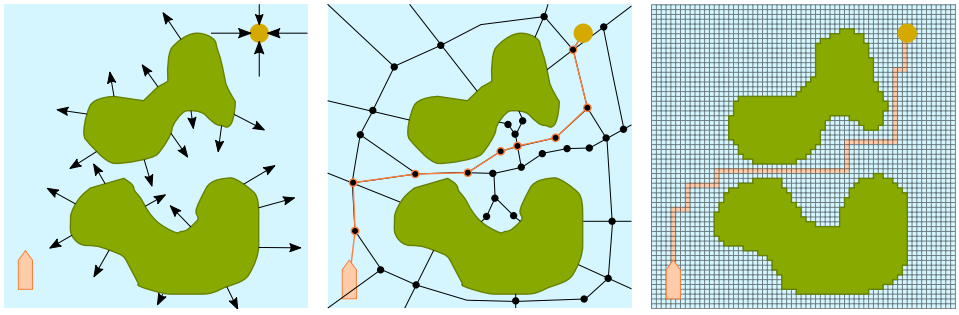
(a) Volumetric world representation of a cave system. Illustration courtesy of Mihir Dharmadhikari who generated it using GBPlanner [44]

(b) A simple uniform occupancy grid world representation of a cave system, manually curated.

Figure 3.1: Example illustrations of world models

notably taken advantage of in drone research projects [44]. In volumetric world models, space is partitioned into a 3D matrix of cubes, as illustrated in Figure 3.1a. These world models typically support dynamic obstacles and combine a priori world knowledge with real-time sensor information. Thus, they often are a potent candidate for motion planning in a three-dimensional environment. From the perspective of motion planning in a two-dimensional world, 3D volumetric world models can be superfluous. Instead, models like occupancy grids and spatial databases can suffice. Using occupancy grids as world models for robotic applications was proposed in 1989 [9], and a plethora of variations have since emerged. Some variations take advantage of quadtrees to generate variable resolution occupancy grids when handling large-scale motion planning [16], while others use uniform grids. One example of a uniform occupancy grid is visualized in Figure 3.1b.

3.1.4 Planning approaches



(a) Potential field

(b) Voronoi diagram roadmap

(c) Cell decomposition

Figure 3.2: Simplified planning approach example illustrations inspired by [10]

Solutions to motion planning problems can generally be put into three main approach categories; potential field methods, roadmap methods, and cell decomposition methods

[10]. Simplified illustrations of all three methods are given in Figure 3.2.

The core idea with potential field methods is to treat the robot $\mathcal{A}$ as a particle in the configuration space affected by a potential field. The field has local variations based on the obstacles in the world $\mathcal{W}$ [10]. Local variation is typically generated by assigning attracting potential to the goal region $\mathcal{X}_G$ and repulsive potential to obstacles in $\mathcal{X}_{obs}$. The potential force dissipation functions can vary for different classes of obstacles and should be considered application-specific. By subjecting the robot particle to the potential field, an artificial force $\vec{F}(q) = -\vec{\nabla}(q)$ is induced [10], showing the direction of the most promising motion. A simplified illustration is given in Figure 3.2a.

Roadmap methods originate from a different concept entirely. Instead of inducing an artificial force, the goal is to capture the connectivity of free space in the world that can later be traversed by graph-search to find the motion sequence necessary to traverse from an initial configuration to a goal configuration [10]. One approach to build the roadmap is to use Voronoi diagrams [10], as is illustrated in Figure 3.2b. Iterative motion planning methods like RRT[15], RRT*[24] and Hybrid A*[21] also exist and incorporate support for non-holonomic robots. Iterative roadmap methods often build the map while the motion planning algorithm searches for a path to the goal configuration. Once the goal is connected to the roadmap, the graph can be searched for the shortest path to the goal from the initial configuration.

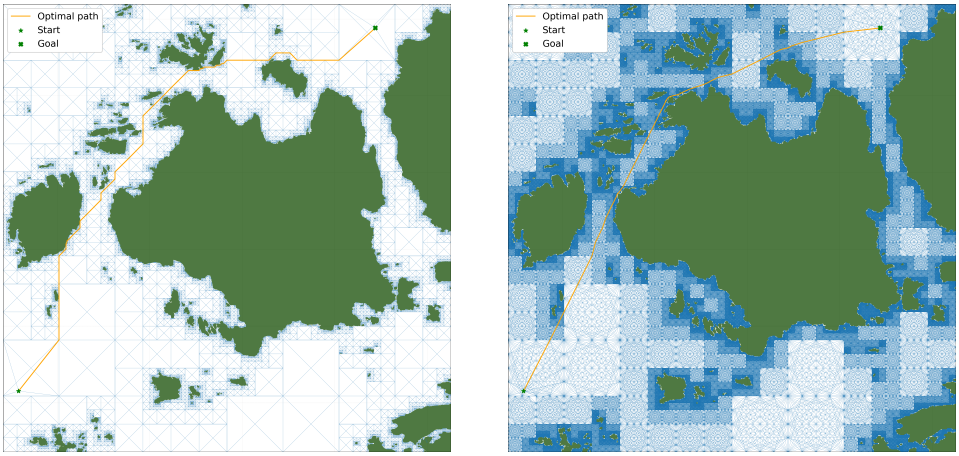
Cell decomposition methods partition the free space $\mathcal{C}_{free}$ in a world $\mathcal{W}$ either exactly or approximately. In approximate cell decomposition, the cells are of a simplified shape such as triangles or quadrilaterals and are a subregion of the free space $\mathcal{C}_{free}$ instead of covering it entirely [10], as shown in Figure 3.2c. Uniform grid decomposition is an example of an approximate technique, and variations of it are often used in small environments. For larger environments, variable-size cell decomposition strategies such as quadtrees can be used to reduce memory requirements [10] and enhance search efficiency. Large-scale planning with quadtree cell decomposition has been featured in several research projects [5], [16], [47], [49]. Similar to roadmap methods, cell decomposition methods rely on building a connectivity graph and applying a graph search technique for shortest-path queries [10]. While the graph in roadmap methods can be built iteratively based on motion sampling, the graph in cell decomposition methods is typically built independently of any robot or vessel. This allows for resolution-optimal paths instead of graph-optimal paths. However, it does not consider vessel dynamics. Therefore, approximate cell decomposition techniques are more appropriate for path planning than motion planning. However, it can support motion planning through distance-to-goal approximations in a sample-based motion planning heuristic.

Potential field methods can be very efficient compared to cell decomposition and roadmap methods and were initially intended for real-time collision avoidance [10]. A significant drawback from the perspective of large-scale motion planning is that such methods are vulnerable to local minima of the potential field as they are essentially gradient descent optimization methods [10]. While solutions to remedy this deficiency do exist, potential field methods employ a search strategy that can be considered local primarily. Meanwhile, roadmap and cell decomposition methods are global strategies and thus can be more suit-

able in large environments. Of the global candidates, cell decomposition methods have been studied extensively [10] and have shown great potential in earlier research projects [5], [16]. However, in recent research, the use of iterative roadmap methods like Rapidly-Exploring Random Trees (RRT)[15], RRT* [24], and Hybrid A* [21] have become well-established for motion planning due to their natural incorporation of nonholonomic constraints through motion sampling. An advantage of Hybrid A* is the heuristic which, similarly to conventional A*, can be used to focus the search and incorporate region-specific knowledge. In vast maritime environments, this can potentially give an advantage over the RRT variations.

3.1.5 Region quadtree

Even when not utilizing a cell decomposition planning strategy, having an approximate cell decomposition of the free space in a mission region can be a valuable tool to facilitate the utilization of a focused search heuristic and for tasks like collision check queries. A quadtree is a hierarchical data structure based on the principle of recursive decomposition of space [8] and typically represents the collision-free space of the mission region as a collection of subregion tiles of variable size in robotic planning applications. Assuming the mission region $\mathcal{R}_{\mathcal{W}}$ is bounded for a time interval T, the process of generating a quadtree can commence. Let $f(\mathcal{R})$ be an occupancy ratio function such that if the occupancy ratio of a region $\mathcal{R}$ is zero or one, then $\mathcal{R}$ is defined as entirely free of obstacles or entirely occupied respectively. An entirely free region is added to the quadtree as a leaf, while an entirely occupied region is discarded. Whenever the occupancy ratio of a region $\mathcal{R}$ is in the open interval $(0, 1)$, the region is decomposed into four quadrants of equal size. Regions are recursively evaluated until all are either discarded or registered as leaf regions. The result is a quadtree decomposition of the free space into the cells of variable size.



(a) Path generated using graph traversal on quadtree. ENC data courtesy of The Norwegian Mapping Authority.

(b) Path generated using graph traversal on framed quadtree. ENC data courtesy of The Norwegian Mapping Authority.

Figure 3.3: Examples of regional quadtree and regional framed quadtree utilization in graph search.

From the perspective of utilizing the quadtree as a tool for evaluating the heuristic in a sample-based search, it can be used to approximate distances such as the distance-to-goal. To achieve this capability, a graph must be built on the quadtree structure and searched to find the shortest path from some initial node in the quadtree to some specified goal node. When compared to fixed-size grid search, quadtrees improve memory efficiency and reduce search complexity [16]. However, being constrained to the tree graph, these improvements come at the cost of distance approximation accuracy. To improve the approximation, the framed region quadtree can be used [16]. By adding additional vertices along the edges of each region $\mathcal{R}$, smoother paths with better optimality characteristics can be found, as shown in Figure 3.3, resulting in more accurate distance approximations. While framed quadtrees improve the paths and approximations found by graph search, they are no miracle cure and typically introduce higher memory utilization to store. Additionally, they are slightly more more computationally expensive to generate and significantly more computationally expensive to search than their non-framed counterparts. For example, the quadtree shown in Figure 3.3a consists of 60768 vertices and is searched with A* to find the graph-optimal path in 197ms. The framed quadtree depicted in Figure 3.3b divides each region edge into four segments and contains 139392 unique vertices. A search query in it took 2237ms using the same software and hardware, indicating that framed region quadtrees can not be used blindly. A variable region edge divisor rule could potentially be used to improve this. In general, any quadtree decomposition can be less efficient to compute and store than uniform grid graphs in worst-case scenarios with high clutter densities [16]. However, for coastal environments, this is unlikely to be an issue.

3.1.6 Robotic paradigms

All autonomous robots rely on sensing, planning, and acting to operate. How these primitives are combined varies, defining various robotic paradigms. A robotic paradigm is essentially a high-level control architecture, and through the last decades of research, three main robotic paradigms have been proposed. These are illustrated in Figure 3.4 and require some introduction.

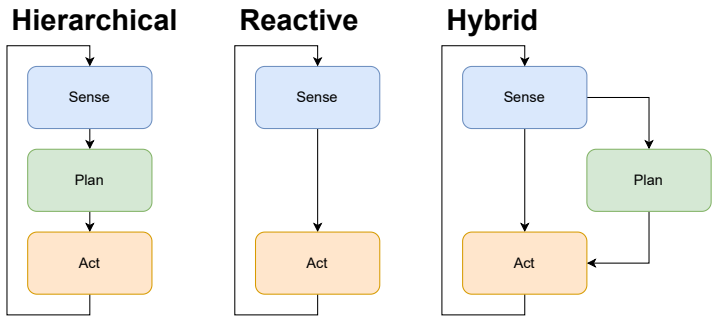


Figure 3.4: Simplified flowcharts for the three main robotic paradigms

In a *hierarchical paradigm*, the robot follows a Sense-plan-act (SPA) loop. By splitting

the continuous-time up into an infinite set of fixed length episodes, this paradigm can be utilized for real-time robot planning problems [38]. Each episode starts with the robot creating a snapshot of the world by combining sensory information with a priori knowledge about the environment. Thereafter, the robot planning system produces a sequence of actions and starts to execute the actions iteratively. Depending on whether the planning system operates synchronously or asynchronously, the world snapshot is recaptured at a fixed or variable frequency. The hierarchical robotic paradigm was first implemented explicitly in *Shakey the Robot* [4] and several robotics research projects have since adapted variations of it.

The *reactive paradigm* is a robotic paradigm for real-time applications. Compared to the hierarchical paradigm, the reactive paradigm connects sensing with acting directly through the use of behaviors. While such a system is real-time capable by design, it lacks planning and reasoning about the world. Thus, it cannot, for example, plan an optimal path or any other similar long-term strategy behaviors. From a motion planning perspective, it is most useful when only local collision avoidance is needed and is found in methods like Velocity Obstacles [29].

In more recent research, the *hybrid paradigm* has been introduced. It combines the hierarchical and reactive paradigm to overcome the issues of real-time performance and lack of global strategy, respectively. Generally, using the hybrid paradigm in a robot planning context takes the form of a multi-stage planner. For example, a global planner can use a global world model to determine the nominal path or, in general, the nominal strategy to get from the initial configuration to the goal. During the execution of this strategy, a local planner Sense-Act subsystem handles real-time control and obstacle avoidance, based on a dynamic window where dynamic obstacles are tracked and avoided for example. The local planner can instruct the robot to deviate from the nominal strategy if so is required for safe operation.

3.1.7 Graph terminology and utilization in motion planning

An arbitrary graph $G = (V, E)$ consists of a set of vertices V, connected by a set of edges E [18]. $\|V\|$ and $\|E\|$ are used to quantify the size of the graph by describing the number of vertices and edges, respectively. A graph edge is defined as an ordered pair $e = (u, v)$, indicating that the vertices u and v are connected by e. In directed graphs, u is considered the source vertex and v the target vertex [18], thus one can traverse from u to v, but not from v to u following this edge. If an edge can be traversed both ways, the graphs are called undirected, and these definitions for u and v are no longer necessary. Any two vertices connected by an edge e are said to be adjacent. Edges commonly have an associated value, which in graph world representations typically is some distance or traversal metric describing the cost of traversing the edge. The value can be based on physical distance, time requirements, fuel consumption, or some combination through a cost function.

For utilization as a tool to approximate shortest-path distances in iterative sample-based motion planning, the graph can follow a quadtree framework. Vertices are then positions or configurations in free space, and edges collision-free transitions between these vertices.

Leaving the associated edge cost as the distance between the vertices results in path distance approximations being available by solving a graph traversal problem.

3.2 Path- and motion planning algorithms

3.2.1 A* Algorithm

When focusing on graph traversal algorithms, the search strategy can be uninformed or informed about the goal vertex. Dijkstra's Algorithm is a well-known example of the former strategy and was first introduced in 1959 [1]. While several variations of Dijkstra's Algorithm exist, none are given information about the goal. Most produce a shortest-path tree consisting of information about the shortest path to any vertex from a source vertex in the graph. While this is useful for many multi-query applications such as digital mapping and IP routing protocols, Dijkstra's Algorithm is inherently inefficient for most robot planning purposes, as many of the shortest paths calculated never will be requested for a single start-to-goal search. By focusing the search in the direction of the goal with a heuristic, efficiency can be improved dramatically. The concept of an admissible heuristic was proposed to achieve this with the introduction of A* in 1968 [2]. By using an admissible heuristic, it can be proven that the calculated path will be graph-optimal [2].

Using the original paper [2] as a reference, the A* algorithm can be described by the following four steps.

1. Let *s* be the initial vertex. Add *s* to a set *open* and calculate the evaluation value using evaluation function $\hat{f}(s)$.
2. In *open*, select the vertex *n* with the smallest evaluation value and resolve any ties arbitrarily except for $n \in Q_G$ which naturally should be prioritized in tiebreaking.
3. If $n \in Q_G$, the algorithm has succeeded in finding a path from *s* to goal region, and the algorithm can be terminated.
4. If $n \notin Q_G$, *n* is marked as closed and its successors are found utilizing successor operator Γ. Thereafter, $\hat{f}(s)$ is run on all successors. If the successor is not marked as closed it is added to the *open* set. If it is marked as closed, but the new evaluation value is lower than when it was closed it is reopened. Finally, return to step 2.

The successor operator and evaluation function are key for the algorithm. Paraphrasing from [2] these functions can be described as follows:

- **Successor operator** Γ:
  The successor operator returns a set of pairs $\{(n_j, c_{ij})\}$ for a given vertex n_i where c_{ij} is the cost associated with traversing the graph edge from n_i to n_j.
- **Evaluation function** $\hat{f}$**:**
  Let an optimal path from vertex *s* through n_i to a goal $n_G \in Q_G$ have a cost defined by the function $f(n)$. The evaluation function $f(n)$ consists of two parts, as shown in Equation 3.2,
  $$f(n) = g(n) + h(n) \tag{3.2}$$

where $g(n)$ is the cost of an optimal path from s to n_i and $h(n)$ is the cost of an optimal path from n_i to $n_g \in Q_G$. The true cost values are not known and must be estimated in order to run the algorithm. Following the approach and notation in [2], the cost function is estimated by use of an evaluation function $\hat{f}(n_i) = \hat{g}(n_i) + \hat{h}(n_i)$, also known as a search heuristic. $\hat{g}(n)$ is commonly known as the *cost-so-far* estimate and it is proposed to use the cost of the path from s to n_i with the lowest cost found so far by the algorithm. This results in $\hat{g}(n_i) \geq g(n_i)$. Estimating $h(n_i)$, commonly known as the cost-to-goal estimate, depends on the graph shape and possible movement directions. For a uniform square grid graph with 4 directions of movement and 8 directions of movement from every vertex it is recommended to use the Manhattan distance L_1 and Diagonal distance L_∞ respectively. For most other scenarios, the direct line distance is often an acceptable estimate. Remark that all of these distance metrics underestimate the true cost-to-goal.

By using the aforementioned estimates for $\hat{g}(n_i)$ and $\hat{h}(n_i)$, the evaluation function will never overestimate the true cost value $f(n_i)$ and is thus admissible[3]. This ensures that the path found is optimal [2]. While the path found will be optimal with regards to the graph, it will typically not be optimal with respect to the world $\mathcal{W}$. In a path- or motion planning context, it can thus be described as a graph-optimal path. How accurately the graph represents the environment then determines how well it approximates the true optimal path.

3.2.2 Hybrid A* Algorithm

Hybrid A* was introduced to solve the motion planning problem for Junior, the Stanford entry in the DARPA Urban Challenge in 2007 [21]. This algorithm is based on the fundamentals of A*, but instead of being graph-based, Hybrid A* is sample-based and is applied to the 3D kinematic state space of the vessel with states given as $q = (x, y, \theta)$ [21]. An illustration comparing Hybrid A* with conventional A* is provided in Figure 3.5.

Using [21] as a reference, the algorithm can be described by the following four steps:

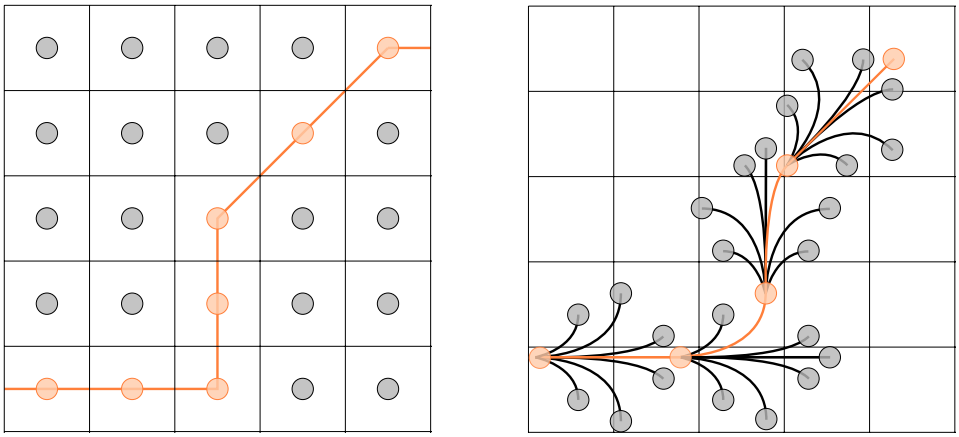
1. Let s be the initial vessel configuration and add it to the exploration frontier $\mathcal{F}$ with zero cost.

2. In $\mathcal{F}$, select the configuration q with the smallest cost value. Any ties should be solved arbitrarily, except for $q \in Q_G$ which should be prioritized.

3. If $q \in Q_G$, the algorithm has succeeded in finding a feasible path from the initial configuration s to the set of goal configurations Q_G and the algorithm can be terminated.

4. If $q \notin Q_G$ it is marked as closed and its successors are found utilizing a hybrid successor operator Γ^*. The cost of each successor q_j is calculated using an evaluation function $\hat{f}^*(q)$. If the successor is not already marked as closed it is added to $\mathcal{F}$

[3] Any reader interested in the proof is advised to consult the Section C in *A Formal Basis for the Heuristic Determination of Minimum Cost Paths* [2, pp. 102–103]



(a) Illustration of A* search

(b) Illustration of Hybrid A* Search

Figure 3.5: Hybrid A* sampling in 3D kinematic state space compared with ordinary A* search. Visualization inspired by [21]

with the evaluation value as its cost. If it is marked as closed, but the new value is lower than when it was closed it is reopened. Finally, return to step two.

It is clear that the general algorithmic procedure is almost identical for A* and Hybrid A*. The difference is in the details, with the successor operator Γ^* and evaluation function $\hat{f}^*(n)$ being the main differentiating factors. Paraphrasing from [21] and introducing new notation, these functions can be described as follows:

- **Hybrid successor operator** Γ^*:
  Given a collection of course and speed correction combinations, a set of candidate trajectories from the current pose represented by vertex q_i is found by simulating the vessel movement for an amount of time T that can be fixed or adaptive. The successor operator returns a set of pairs $\{(q_j, hor_{ij})\}$ for a given configuration vertex q_i and a set of possible control correction combinations. hor_{ij} is the simulation horizon from configuration q_i to q_j and describes the movement between these configurations in detail, which is essential for the evaluation function.

- **Hybrid evaluation function** $\hat{f}^*(q)$:
  The hybrid evaluation function consists of two heuristics; a *non-holonomic-without obstacles* and a *holonomic-with-obstacles*. The former ignores all obstacles but takes the vessel dynamics into account. It is computed by taking the maximum of the straight-line distance and the distance of the shortest path to the goal configuration set Q_G for the robot. In the original implementation, Reeds-Shepp curves are utilized to facilitate this heuristic [21]. The heuristic is admissible, and the effect of it is pruning of search branches that approach the goal with the incorrect heading [21]. The latter heuristic is a dual of the first and computes the distance of the shortest path to the goal from the current configuration q_i discarding the orientation dimension of the configuration, for example using A* on a quadtree graph spanning

the free-space. The purpose of this heuristic is to discover dead-ends in a fast two-dimensional search and guide the significantly more expensive three-dimensional search away from these areas [21]. Because this heuristic is admissible, the maximum of the two heuristics can be used as an evaluation function without loss of admissibility.

3.3 Guidance, Navigation and Control

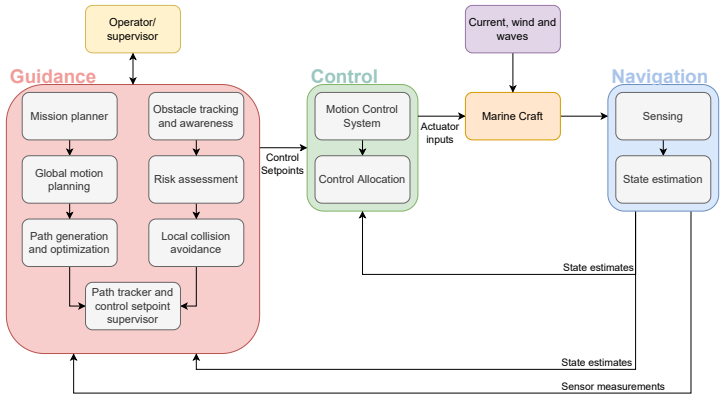


Figure 3.6: Illustration of a complete general GNC system, based on [48] and [53]

Any autonomous vessel required to traverse some environment must incorporate a guidance system, a navigation system, and a control system to function. These three systems are the pillars on which all autonomous mobility rest. When developing a guidance system, it is important to be aware of what is expected from it and how it should interact with navigation and control systems. A figure illustrating a complete GNC system is depicted in Figure 3.6 to accompany the following descriptions.

3.3.1 Guidance Systems

Guidance systems are responsible for calculating how a vessel should act to reach some goal, often by following a path or trajectory from an initial position to a goal position in the presence of environmental disturbances and prediction uncertainties [48]. One way to parametrize a path is by using waypoints, which from the perspective of maritime guidance can be set by an experienced navigator or automatically generated based on a priori map information, environmental factors, and optimization strategies with regards to time and energy consumption.

The vessel can get underway when a global nominal path or trajectory parametrization is made available to the guidance system or generated internally. While the vessel is voyaging, one of the primary responsibilities of the guidance system is path following or trajectory tracking through a guidance law. The difference between path following and trajectory tracking is that while path following is time-invariant, trajectory tracking

is time-dependent, and thus a more demanding and constrained task [48]. The guidance law is used to compute desired course and speed of the vessel to minimize the cross-track error concerning the nominal global path. A plethora of guidance laws exist in path following research, but few are as recognized and widely used as the LOS guidance law [19]. Performing rules compliant and predictable collision avoidance maneuvers whenever necessary is another important task the guidance system has been given responsibility for, as shown in the leftmost block in Figure 3.6. As discussed in Section 2.3 in the literature review, there are many ways in which a COLAV system can be designed. In a modular design based on the hybrid robotic paradigm, the collision avoidance system typically will be a separate module. It can then be given information about the nominal path and nominal control setpoints and issue corrections to the setpoints to ensure collision-avoidance capabilities. In Figure 3.6 it is suggested to use a combined path tracker and control setpoint supervisor to facilitate cooperation between the global mission planner, path tracker, and local collision avoidance system. The reader is advised that while the general layout of Figure 3.6 is based on [48] and [53], considerations such as the setpoint supervisor is not mentioned in these sources.

3.3.2 Navigation systems

In general, navigation systems are responsible for estimating the states of the vessel relevant for the guidance and control systems. For USVs, the pose and twist of the vessel are necessarily estimated [48]. Following the categorization in [48], two main approaches for this estimation task exist:

- **Model-Based**: Systems that determine the vessel pose and twist by utilizing a state estimator based on a mathematical model of the vessel. A Luenberger observer is one type of observer that can be utilized for this purpose [48]. In addition to being provided with a vessel model, the state observer must be given information about the control behaviors of the vessel through low-level sensory data such as propeller RPM and rudder angle.
- **Inertial**: Systems that determine the vessel pose and twist by utilizing only sensory information. Commonly, techniques from sensor fusion such as the Kalman Filter and variations thereof are used to combine sensory information and handle uncertainties to generate state estimates. Accelerometers, gyroscopes, and supporting sensors such as Global Navigation Satellite Systems (GNSS) positioning and dual GNSS compasses are commonly used for this purpose.

It should be noted that while there are two main approaches to navigation, hybrid approaches combining sensory information and model information do exist.

3.3.3 Control systems

A vessel control system is responsible for controlling the actuators of the vessel in order to comply with requests from the vessel guidance system while maintaining vessel stability [48]. The actuators on a marine vessel are connected to effectors such as rudders and propellers that can generate time-varying forces and moments. The general control system

consists of a control law that receives inputs from the guidance system. It thereafter calculates the necessary forces and moments on the vessel τ to comply with the guidance system request. Furthermore, a control allocation scheme is necessary to translate τ to actuator inputs $\mathbf{u}$ such as desired propeller RPM and desired rudder angle. In the state-of-the-art, there are two main approaches to control systems according to Fossen [48]:

- **Motion Control**: Control systems where PID designs combined with successive loop closure are used to control relevant vessel states.

- **Advanced Motion Control**: Control systems where mathematical models of vessels are used in the control loop. Optimal and nonlinear control theory is commonly used for advanced motion controllers. However, a vessel model combined with a PID controller can also be used for feedback linearization control. Assuming the vessel model is sufficiently accurate, feedback linearization, as the name suggests, makes the system one wishes to control linear from the perspective of the PID controller. Thus, it enables a much more predictable and intuitive tuning effort and can make the overall controller significantly more performant.

3.4 Surface Vessel Modelling

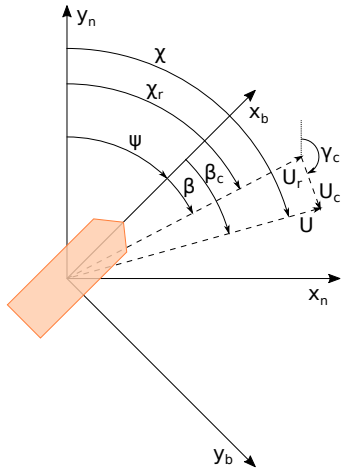


Figure 3.7: Horizontal ocean current triangle, illustration inspired by [48]

Having sufficiently precise surface vessel models for advanced motion control systems, some guidance systems, and all simulation environments is essential. The 1st-order Nomoto model given in the frequency domain in Equation (3.3) is one of the simplest surface-vessel models [48]. This transfer function from rudder angle δ to yaw rate r is not very realistic, but it is linear, and one only has to determine the gain parameter K and time constant T in order to use it.

$$\frac{r}{\delta}(s) = \frac{K}{Ts+1} \tag{3.3}$$

While the 1st-order Nomoto model can be used for some control systems and simple positional approximations, it is often not sufficiently realistic. Guidance systems that rely on accurate vessel horizon predictions and all simulators require more advanced vessel models that capture the vessel dynamics with greater detail. To limit complexity somewhat, a 3-DOF surface vessel model is then a prime candidate for utilization. A general 3-DOF surface vessel model is introduced in [48] and adapted for the Viknes 830 vessel in [22].

3.4.1 The equations of motion

Following the notation in [48], which uses the SNAME naming convention, the vessel's generalized coordinates $\eta = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$ are given in an inertial, Cartesian world frame $n = (x_n, y_n, z_n)$. Furthermore, the velocity of the vessel $\nu = \begin{bmatrix} u & v & r \end{bmatrix}^T$ is defined in the vessel body frame $b = (x_b, y_b, z_b)$. A complete overview of all relevant variable names in the SNAME convention is given in Table 3.1. Additionally, the relevant notation is visualized in the horizontal current triangle in Figure 3.7.

DOF	Description	Force/Moment	Linear speed/Angular rate	Position/Angle
1	Motions in the x_b-direction (surge)	X	u	x^n
2	Motions in the y_b-direction (sway)	Y	v	y^n
6	Rotation about the z_b-axis (yaw)	N	r	ψ

Table 3.1: The relevant notation of SNAME (1950) from [48]

The equations of motion are given by Equation (3.4) and Equation (3.5),

$$\dot{\eta} = \mathbf{R}_{\mathbf{b}}^{\mathbf{n}}(\psi)\nu \tag{3.4}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu = \tau + \tau_w \tag{3.5}$$

where $\mathbf{R}_{\mathbf{b}}^{\mathbf{n}}(\psi)$ defined in Equation (3.6) transform body coordinates to world coordinates.

$$\mathbf{R}_{\mathbf{b}}^{\mathbf{n}}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

The matrices in Equation (3.5) characterize the vessel dynamics and require further detailing. In order to do so, Equation (3.5) is first expanded to get Equation (3.7), and the relative velocity of the vessel $\nu_r = \nu - \nu_c$ is introduced to emphasize that this model is valid in the presence of ocean current $\nu_c = [u_c, v_c, 0]^T$. Note that in order to make the model as simple as possible, the simplifications from [22] can be applied. Thus, the body reference frame origin is placed to coincide with the center of gravity of the vessel. Moreover, the orientation of the body is set such that it coincides with the longitudinal, lateral, and normal symmetry axes of the vessel.

$$\underbrace{\mathbf{M}_{RB}\dot{\nu}_r + \mathbf{C}_{RB}(\nu_r)\nu_r}_{\text{ridig-body forces}} + \underbrace{\mathbf{M}_A\dot{\nu}_r + \mathbf{C}_A(\nu_r) + \mathbf{D}\nu_r + \mathbf{D}_n(\nu_r)\nu_r}_{\text{hydrodynamic forces}} = \tau + \tau_w \tag{3.7}$$

Starting with the rigid-body forces, $\mathbf{M}_{RB}$ is the rigid-body inertia matrix and $\mathbf{C}_{RB}(\nu_r)$ represents the rigid-body Coriolis and centripetal forces. Under the simplifying assumptions in [22], these matrices are given by Equation (3.8) and Equation (3.9) respectively.

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{3.8}$$

$$\mathbf{C}_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix} \tag{3.9}$$

where m is the vessel mass and I_z is the moment of inertia about the z-axis of the vessel. Continuing with the hydrodynamic forces, $\mathbf{M}_A$ is the inertial matrix for added mass, a hydrodynamical phenomenon that can be interpreted as pressure-induced forces and moments due to a forced harmonic motion of the vessel [22]. The added mass matrix is defined as given in (3.10).

$$\mathbf{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix} \tag{3.10}$$

The second component affecting the hydrodynamic forces is the added Coriolis $\mathbf{C}_A(\nu)$ defined as shown in Equation 3.11

$$\mathbf{C}_A(\nu) = \begin{bmatrix} 0 & 0 & -Y_{\dot{v}}v + \frac{Y_{\dot{r}}r + N_{\dot{v}}}{2}r \\ 0 & 0 & X_{\dot{u}}u \\ Y_{\dot{v}}v + \frac{Y_{\dot{r}}+N_{\dot{v}}}{2}r & -X_{\dot{u}}u_r & 0 \end{bmatrix} \tag{3.11}$$

for the Viknes 830 vessel. The last two matrices in the hydrodynamic forces are damping matrices and account for effects such as skin friction and vortex shredding [48]. Damping effects are inherently complicated to model, thus some simplifications must be made. Following the simplifications in [22], it is thus proposed that the linear damping matrix $\mathbf{D}$ is defined as shown in Equation 3.12

$$\mathbf{D} = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} \tag{3.12}$$

and the nonlinear damping term as given in Equation (3.13).

$$\mathbf{D}_n(\nu_r)\nu_r = -\begin{bmatrix} X_{|u|u}|u_r|u_r + X_{uuu}u_r^3 \\ Y_{|v|v}|v_r|v_r + Y_{vvv}v_r^3 \\ N_{|r|r}|r|r + N_{rrr}r^3 \end{bmatrix} \tag{3.13}$$

By defining $\mathbf{M} = \mathbf{M}_R B + \mathbf{M}_A$, $\mathbf{C}(\nu) = \mathbf{C}_R B + \mathbf{C}_A$ and $\mathbf{D}(\nu) = \mathbf{D} + \mathbf{D}_N(\nu)$, Equation (3.5) is retrieved from Equation (3.7). For the vessel modelling, the forces and moments

induced by environmental disturbances like waves and wind τ_w are not detailed further. However, the forces and moments produced by the vessel effectors and controlled by vessel actuators $\tau = [\tau_X, \tau_Y, \tau_N]$ are an essential part of the vessel model. τ_X and τ_Y represent the forces applied along the longitudinal and lateral axes of the Viknes 830 vessel respectively, while the moment τ_N is applied about the vessel yaw axis. The Viknes 830 model assumes that low-level control loops for the actuators exist, and thus that the force from them can be applied directly without any further modelling [22]. The Viknes 830 model has two actuators, a motor attached to a fixed-pitch propeller generating a longitudinal force F_x and a rudder generating a lateral force F_y [22]. In total, the inputs can be defined as given in Equation 3.14

$$\tau = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ l_r F_y \end{bmatrix} \tag{3.14}$$

where l_r is the length of the lever arm. For a vessel, the lever arm is not a physical arm, but instead the distance from the vessel CoG to the *point of attack* of the rudder force [22].

3.4.2 Low-level controller

In order to make the vessel model useful for simulation and to test a guidance system on it without having to handle force and torque calculations in other modules, low-level controllers are important. This way, when looking at the system as a whole, the simulated vessel can easily be replaced with a real vessel as long as that vessel has a tuned autopilot with the same input interface as the simulated vessel. In this thesis, two low-level controllers for speed and course from [22] are utilized and given below in Equation (3.15) and Equation (3.16) respectively.

$$F_x = \overbrace{-X_u u - X_{|u|u}|u|u - X_{uuu}u^3 - mrv}^{\text{model feed forward}} + \overbrace{K_{p,u}m(u_{sp} - u)}^{\text{P feedback}} \tag{3.15}$$

$$l_r F_y = \underbrace{K_{p,\psi} I_z(\Gamma(\psi_{sp} - \psi) + T_{d,\psi}(\dot{\psi}_{sp} - r))}_{\text{PD feedback}} \tag{3.16}$$

The speed controller takes advantage of feedback linearization in order to remove the non-linear terms from the model. In a real-world scenario, the feed-forward term must be considered carefully as any modeling errors or vessel state uncertainties can result in the feed-forward term making the overall system more difficult to control[4]. For the purpose of this thesis, where the vessel model is known and the vessel state is not distorted, this controller will linearize the vessel model and make it easy to obtain decent control performance with only an additional proportional feedback controller.

Observe that the heading controller is a classic proportional-derivative (PD) controller not utilizing feed-forward of any kind. The function $\Gamma : \mathbb{R} \rightarrow\!< -\pi, \pi]$ is the Smallest Signed Angle (SSA) function described in [48] and ensures that the vessel always turns in

[4] Any reader interested in further details on feedback linearization or other advanced motion control methods is advised to consult [48]

the correct direction to reduce the heading error. Moreover, it should be noted that while Equation (3.16) includes the derivative of the course setpoint, $\dot{\psi}_{sp}$, it is set to zero in many applications.

3.5 Electronic Navigational Charts

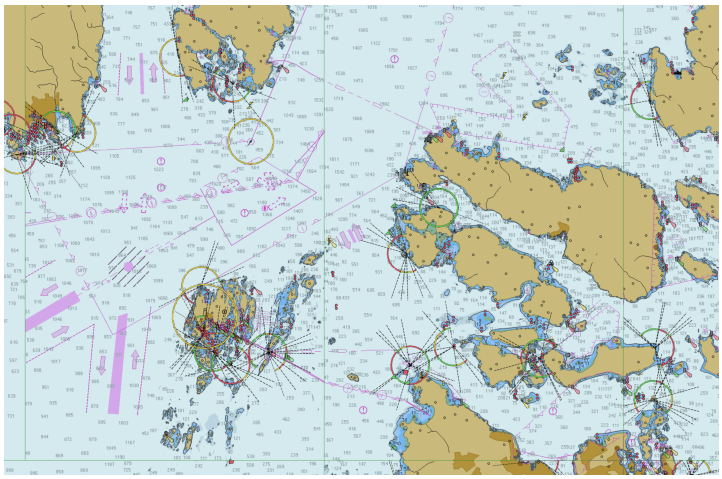


Figure 3.8: Illustration of part of ENC chart generated using OpenCPN [59]. ENC data courtesy of The Norwegian Mapping Authority

ENCs are spatial databases that contain all chartered features relevant for safe maritime navigation. Part of an ENC is shown in Figure 3.8. The intended use for the electronic charts is to supply data to an Electronic Chart Display and Information System (ECDIS). An ECDIS is an instrument to display charts electronically to a navigator, in compliance with a set of requirements specified by the International Maritime Organization (IMO) [17], [36]. While the ENCs are published by different national hydrographic offices, such as *The Norwegian Mapping Authority* for the Norwegian coast and *National Oceanic and Atmospheric Administration (NOAA)* for the coast of the USA, they must comply with the S-57 ENC Product Specification and be certified by the International Hydrographic Organization (IHO) [36]. The intended utility of ENCs is to replace paper charts and aid the navigator. However, the potential value for autonomous operations is also clearly substantial.

3.5.1 The S-57 Product Specification

The purpose of having a Product Specification for ENCs is to ensure that charts produced by different hydrographic offices have the same structure and can be used to supply data to systems such as ECDIS without any risk of misinterpretation or loss of information. The specification is published by the IHO and documented in *Transfer Standard for Digital Hydrographic Data* [17]. The standard intends to enhance uniformity in navigational

charts and guarantee that all information necessary for safe navigation is present. Furthermore, the product specification uses a structure that makes it possible to interpret ENCs for specialized software projects such as autonomous guidance systems.

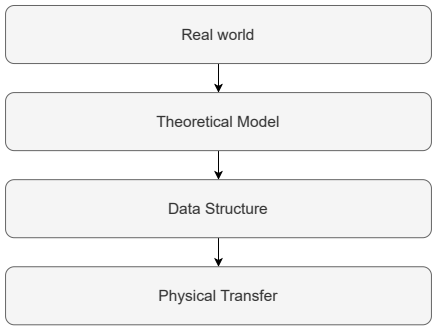


Figure 3.9: Structure layers according to S-57, illustration based on [17].

There are four layers of data transfer in the S-57 product specification, as shown in Figure 3.9. The first layer is the physical world in which some operation, such as navigation of an unmanned vessel, is to be performed. In order to perform this operation, the natural world must be represented by a simplified and descriptive theoretical world model. The model is then translated and stored in named constructs such as records and fields in a spatial data structure. In order to physically transfer this data structure, it must be encapsulated in a file. The S-57 product specification uses the ISO/IEC 8211 interchange format for this encapsulation [17].

The model layer consists of objects considered relevant for hydrographic tasks. The objects define real-world entities by combining spatial subobjects and features with descriptive attributes. The spatial subobject described the geometry and the geographical position of the entity. While the specification allows both for vector, raster, and matrix geometry, only the former has been defined by IHO as of yet [17]. Spatial vector objects can have at most two dimensions. However, a third dimension can be represented through an object attribute if, for example, depth data is to be represented. Object features contain only non-locational descriptions of the entity and are divided into four types [17]:

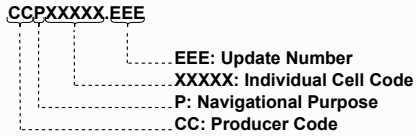
- **Meta feature**: Feature object which contains information about other objects
- **Cartographic feature**: Feature object which contains information about the cartographic representation of the entity.
- **Geo feature**: Feature object which carries the descriptive characteristics of a real world entity
- **Collection feature**: Feature object which describes the relationship between other objects.

The purpose of object features is to encapsulate all information about an object that is not related to its spatial characteristics.

The data structure layer in the S-57 specification translates the objects describing real-world entities into named constructs such as records and fields through a set of rules and constraints [17]. If one is to produce ENCs or develop a driver to read the data structure, these rules and constraints are explained in rigorous detail in [17]. However, be advised that open-source drivers for interpreting S-57 data structures are available through open-source projects such as Geospatial Data Abstraction Library (GDAL/OGR) [57].

S-57 data structures follow a naming convention specified by the IHO S-57 publication [17]:

S-57 Naming Convention



- The first two characters identify the producer. This list is given in Annex A to Appendix A of the IHO Object Catalogue [17].
- The third character indicates the navigational purpose (see clause 2.1)
- The fourth to eight characters are used for the cell code. This code can be used in any way by the producer to provide the unique filename. If characters other than numbers are used only uppercase letters are allowed.

Navigational purpose code overview:

Subfield content	Navigational purpose
1	Overview
2	General
3	Coastal
4	Approach
5	Harbour
6	Berthing

It is essential to know the naming convention when obtaining ENCs, notably because it gives insight into the level of detail and type of information one can expect to extract from the data structure. For example, the level of detail is commonly significantly higher for Approach ENCs compared to Overview ENCs.

3.5.2 Electronic Navigational Charts in autonomous missions

As can be seen in Appendix A of the S-57 Product Specification [17], the complete object catalog is large, and the level of detail in the object attributes is comprehensive. Because the specification requires all objects that can be relevant for maritime navigation to be present, the significance of the object types varies greatly, and some depend on mission type and vessel size. While a handful of projects, to a limited extent, have classified which objects in the S-57 specification are relevant for autonomous missions of unmanned surface vehicles [42], [43], [46], no research effort dedicated to evaluating the relevance of all

objects in the specification catalog has been undertaken. Additionally, no official guidelines from the IHO exist for this purpose. The lack of guidelines and the limited research highlight that the S-57 standard was not intended for autonomous operations when proposed. Due to the importance of the matter, an overview of objects considered relevant for autonomous missions can be found listed in Table B.1 and Table B.2 in Appendix B. It should be noted that, even with a sufficiently rigorous evaluation of relevant objects, not all information in the relevant objects can be used. This is because some objects include information that depends on complex reasoning and human-level situational awareness. Thus, while the S-57 specification guarantees sufficient information for safe navigation for human-crewed vessels, this guarantee cannot be extended to autonomous vessels without careful, dedicated research efforts, which are yet to be embarked upon.

3.6 Geodesy

If one is to guide a vessel towards a goal in a vast maritime environment, the curvature of the Earth will have significance and must thus be taken into consideration by utilizing Geodesy. Following the classical definition of Geodesy by Friedrich Robert Helmert, *"geodesy is the science of the measurement and mapping of the Earth's surface"* [25]. Similar to any other science, Geodesy has evolved and advanced since its origin. It can now be described as the science of accurately measuring the properties of the Earth in general. This includes the geometric shape of Earth, its orientation in space, and the gravitational fields it causes [25]. If one is to perform global positioning on Earth, this is done in geodetic coordinates associated with a datum. A geodetic datum is a description of a geodetic coordinate system for the Earth's body and is commonly expressed using a 3-DOF translation, a 3-DOF rotation, and a scaling factor. Furthermore, since ellipsoidal coordinates typically are used in geodetic reference systems, the datum must also include the ellipsoid parameters [25]. One use-case of the datum is to translate map positions to positions on the curved surface of the Earth, and while some datums such as WGS84 [30] are global, others like OSBG36 are local.

3.6.1 World Geodetic System 1984

WGS84 is a Conventional, Terrestrial Reference System (CTRS), also known as an Earth-centered, Earth-fixed (ECEF), coordinate system and geodetic datum, released and revised by the United States National Geospatial-Intelligence Agency [30]. It represents Earth as an oblate spheroid, which is defined as an ellipse that is rotated about its minor axis to form the ellipsoid. The four defining parameters for WGS84 are given in Table 3.2. All producers of ENCs are required to use WGS84 as the horizontal datum to comply with the S-57 Product specification [17]. If one uses other charts that use another datum, be advised that transforming to WGS84 can easily be achieved using software libraries like GDAL/OGR [57].

Parameter	Notation	Value	unit
Semi-major Axis	a	6378137.0	m
Flattening factor	f	$\frac{1}{298.257223563}$	N/A
Nominal Mean Angular Velocity of the Earth	ω	$7.292115x10^{-5}$	$\frac{1}{s}$
Geocentric Gravitational Constant	GM	$3.986004418x10^{14}$	$\frac{m^3}{s^2}$

Table 3.2: WGS84 Defining Parameters from [30]

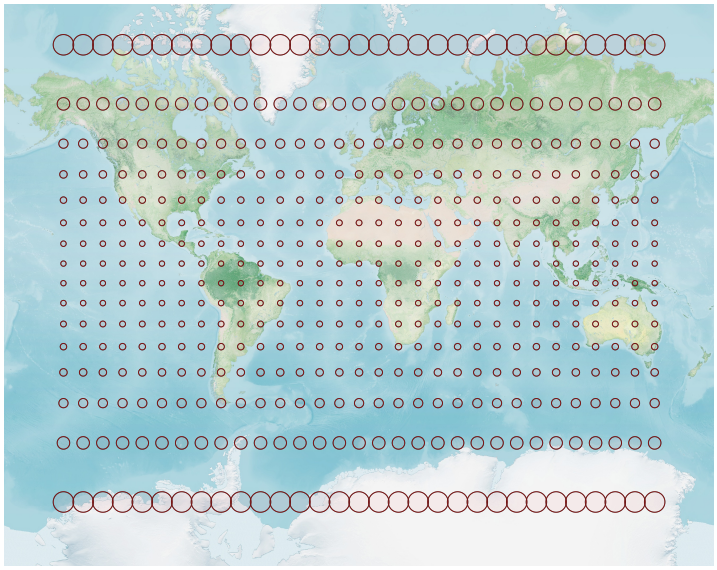


Figure 3.10: Mercator projection illustration with Tissot's indicatrix[7]. Raster from the *Blue Marble* collection, courtesy of NASA. Reprojected from Equirectangular to Mercator using QGIS [58]. Tissot's indicatrices of 150km radius generated using the *Indicatrix mapper* QGIS plugin with 10 degrees resolution

3.6.2 The Mercator Projection

Geodetic datums like the WGS84 define three-dimensional space. In order to represent the reference frame of the datum on a two-dimensional plane, a projection is necessarily utilized. Any projection will distort the relative positions on a map, and the Mercator Projection is no exception. A cylinder is wrapped around the Earth's ellipsoid in this projection, and Earth's surface is projected onto this cylinder. The projection distorts significantly on a global scale, as can be seen from Figure 3.10 with the help of Tissot's indicatrix[7]. Suppose one is to project a local map of extent sufficiently limited such that the curvature of Earth is negligible. In that case, a local linear projection can be used instead of the Mercator projection. For example, using normalized gradients of Earth's curvature about a chosen point in the geodetic reference system as unit vectors, the approximation yields a Cartesian coordinate that tangents the geodetic reference system at the chosen point. Be advised that this linear projection can only be used with reasonably low curvature error for

any operation in the plane that remains in the proximity of the projection origin.

3.6.3 The inverse geodetic problem

For motion planning over large distances, neither a global projection like the Mercator nor a local projection will generally suffice for accurate and safe guidance of the vessel. Instead, a geodesic coordinate system must be used. This proposes a challenge, as most established path- and motion planning techniques assume operation in a Cartesian coordinate system. This assumption is often used for most focused search techniques when distances are calculated for a heuristic or a graph edge cost evaluation. The straight-line distance Euler formula can no longer be used in a geographical coordinate system. Instead, the system must rely on lengths of geodesics, which are the shortest path between two points on the Earth's surface [28].

In order to find the length of a geodesic, the inverse geodetic problem must be solved [28]. Several methods have been proposed to approximate the distance and thus the solution to the inverse geodetic problem. One such method is the Haversine algorithm, which computes what is called the *great-circle distance* between two points using the formula given in Equation (3.17) [34]

$$d_{haversine} = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\theta_2 - \theta_1}{2}\right) + \cos\left(\theta_1\right)\cos\left(\theta_2\right)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (3.17)$$

where r is the radius of the sphere. θ and λ are the latitude and longitude of a point on the sphere. Remark that Haversine ignores all ellipsoidal effects and thus assumes a spherical Earth. Another, more accurate approximator is the Vincenty Algorithm which uses the iterative method of Helmert (1880) to solve the inverse problem [28]. While the Vincenty Algorithm is more accurate than Haversine [34], it fails to converge for nearly antipodal points [28] and was not designed with modern computers in mind. Karney's algorithms for geodesics [28] revisits the geodesic methods of Helmert (1880) on which Vincenty's Algorithm is based, but improves three main aspects;

- **Improved accuracy**:
  Accuracy matching modern computer precision by retaining sufficient terms in series expansion.
- **Guaranteed Convergence**:
  Converges for all pairs of points, no matter if they are antipodal or not.
- **Avoiding numerical differentiation**:
  Karney's algorithms for geodesics calculate differential and integral properties of the geodesic, avoiding the need for numerical differentiation in methods like approximating the inverse geodetic problem solution.

The work of Karney is collected in his software library GeographicLib [51], which is open-source. Any reader interested in the theoretical and technical details of GeographicLib is advised to consult [28] and [51] respectively.

Chapter 4

Method

4.1 Solution concept and system design

The primary purpose of an USV guidance system is to get the vessel from an initial pose to a goal position safely, efficiently, reliably, and within COLREG rules compliance. In order to achieve this, the guidance system must have the following abilities:

- **Global path generation**:
  Generate a global path from the initial pose to the goal pose or position. This path must be safe, optimized, rules-compliant, and feasible for the vessel to follow.
- **Path tracking**:
  Given an arbitrarily parameterized path, the guidance system must be able to provide the vessel autopilot system with control setpoints that facilitates tracking of the nominal path with reasonable frequency response and accuracy. For example, these setpoints can be course and speed over ground setpoints if that is what the vessel autopilot supports.
- **COLAV**:
  Given that the vessel is following some global nominal path, the guidance system must be able to instruct the vessel to deviate from the nominal path and optionally alter its speed whenever necessary to avoid a collision or mitigate damage potential with other vessels tracked by some arbitrary target tracking system. The avoidance behavior must be COLREG compliant and clearly state the vessel intent for any observer on the obstacle ship or elsewhere.

One way to solve the motion planning problem and incorporate these abilities is to use a divide-and-conquer strategy and design three main subsystems, each responsible for one of the abilities. In addition to the three main subsystems, a subsystem facilitating the utilization of geospatial data from Electronic Navigational Charts is necessary to ensure

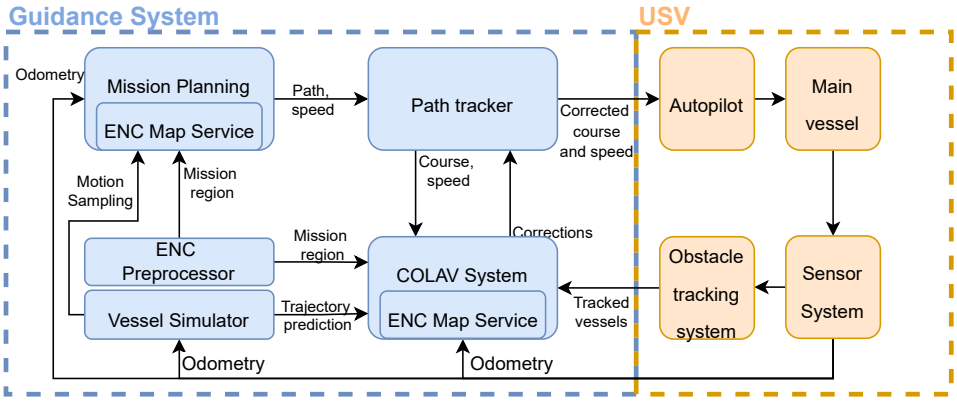


Figure 4.1: A high-level visualization of a complete guidance system including the USV.

collision-free paths and rules compliance. Moreover, a vessel simulation subsystem is necessary to facilitate sample-based motion planning and predictive collision avoidance. The vessel simulation subsystem should be designed to both be an efficient tool internally and also be able to facilitate a traditional simulation environment when evaluating system performance. A complete system is visualized in Figure 4.1. The guidance system is the focus of this thesis. Thus, other necessary components that must be present to enable simulation-based testing, such as target tracking and vessel autopilots, must be simulated.

4.2 Electronic Navigational Chart Manager

For an USV to undertake any mission, knowledge about the surrounding mission environment is of utmost importance. Information about the static environment and the rules and restrictions within it can be obtained by interpreting Electronic Navigational Charts. The ENCs in this thesis follow the S-57 format and are primarily designed to be interpreted by human navigators using ECDIS navigational instruments. While designed to provide all information necessary for safe navigation [36], some information encapsulated in the ENCs require human-level intuition and situational awareness. When interpreting ENCs using artificial intelligence, this results in some information being unusable for autonomous navigation and thus makes extraction and interpretation of available information even more critical. The proposed ENC Manager has two primary responsibilities, reflecting the proposed overall system depicted in Figure 4.1:

- **Pre-processing: Information extraction and interpretation**
  Given a specified mission region extent and a collection of ENC chart tiles, the ENC manager must pre-process the mission region by extracting the relevant information from the minimum required amount of ENC chart tiles. The tiles must be stitched together to a detailed spatial database retaining all information and a simplified overview database containing the information most frequently used for collision avoidance and path safety. Furthermore, additional data structures such as

a region quadtree and a Voronoi skeleton of free space relevant for motion planning and real-time performance during mission execution must be generated.

- **Map service: Readily available environment information during missions**
  Given a pre-processed mission region, the manager must provide information about the environment to other systems that require it. This can be done using an adaptation of spatial queries. The map service can do some calculations to respond to the query, but most of the computational complexity should be handled in pre-processing. This is essential for efficient motion planning and enabling adequate real-time performance in collision avoidance.

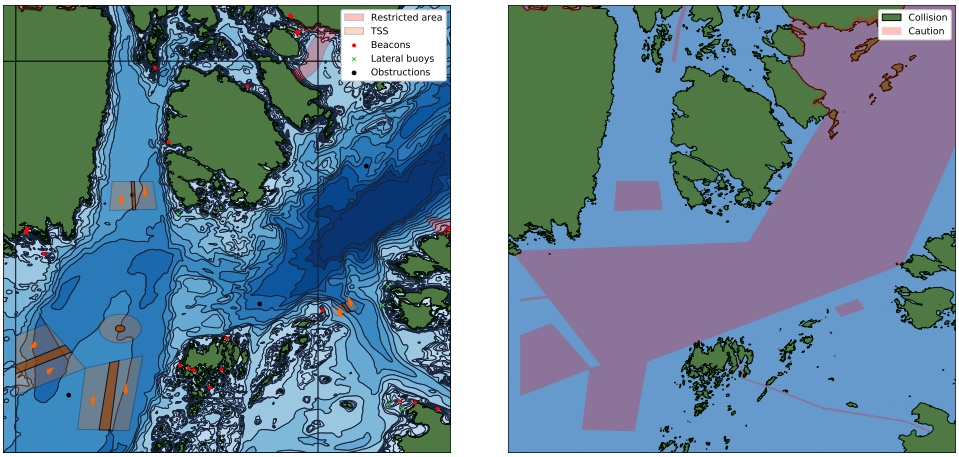
4.2.1 Information extraction from Electronic Navigational Charts

In the S-57 ENC standard, there are many different object classes. The ones considered relevant for autonomous navigation of smaller vessels are given in Table B.1 and Table B.2 in Appendix B and can be classified into *hazards* and *cautions* in the simplified spatial database. *Hazards* are defined as areas or objects that always must be avoided by the USV, and *cautions* are areas that may require action by the USV and potentially will restrict its movement in some way. Some objects, such as land and bridge pillars, are always hazardous. Others, such as TSS objects, always should be considered caution objects due to their paramount importance to safe navigation. However, objects such as shallow areas cannot be classified without knowledge about vessel dimensions. Thus vessel properties such as draft, height, width, and length must be considered during classification. With this, a detailed spatial database like the one visualized in Figure 4.2a can be used to generate a simplified spatial database, the result of which is visualized in Figure 4.2b.

For autonomous missions, the USV must have sufficient knowledge about the static environment at its disposal. Thus, Electronic Navigational Charts with appropriate detail levels should be used. However, one issue arises from this; ENCs are organized in tiles, and the extent of the tile is commonly inversely proportional to the level of detail in the chart [36]. Multiple chart tiles have to be used for a large mission region if one wants to retain as much information about the mission region as possible, which can be achieved by stitching together the tiles to cover the entire mission region with a detailed map. During the information extraction, the system should also identify any subregion that is part of the mission region but not covered by available ENCs. It should be up to the other systems using the map service to determine how to handle unknown areas to ensure system applicability. However, it is recommended to treat such areas as hazards to avoid voyaging into uncharted waters.

4.2.2 Interpreting and representing free space in the mission region

Coastal environments often consist of several large obstacle-free areas and other areas of varying obstacle density. One way to represent the free space is to use uniform grids. However, for vast areas where large subregions are entirely free, this is inherently inefficient with regard to memory utilization and computational cost. Instead, region quadtrees can represent the traversable space in the mission region. Region quadtrees allow for efficient



(a) Complete ENC-based map. Only some symbols visualized. ENC data courtesy of The Norwegian Mapping Authority

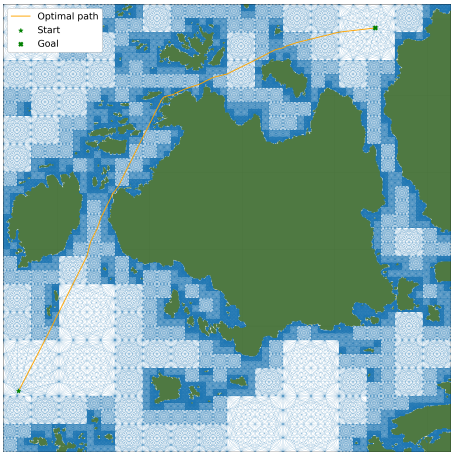
(b) Simplified and reduced map. ENC data courtesy of The Norwegian Mapping Authority

Figure 4.2: Visualization of a detailed and simplified spatial database built based on Electronic Navigational Charts. The simplified map can be used for fast queries, while the complete map can be used whenever interpretation with attributes and situational context is required.

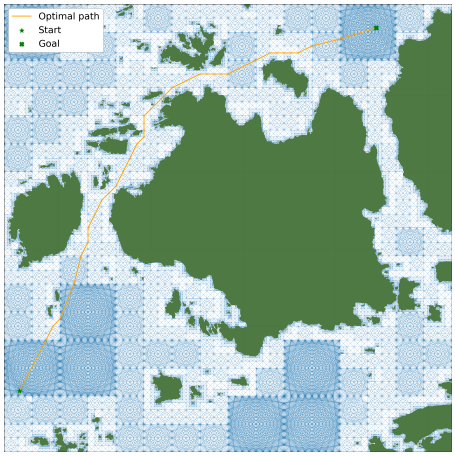
storage of large free regions while maintaining sufficient resolution in narrow straits, bays, and similar areas.

Looking at the quadtree as a graph for utilization by graph search techniques, the conventional quadtree significantly limits the options for the graph search compared to uniform high-resolution grids, with worse path and optimality characteristics. To remedy this problem, framed region quadtrees can instead be utilized as proposed in Subsection 3.1.5. Using a fixed region edge divisor rule can be wasteful regarding memory utilization and computational cost if set too low and sacrifice path optimality if set too high. Instead, a variable divisor rule based on maximum distance can be used, where every region edge is split such that the maximum distance between edge vertices is fixed. A variable divisor rule can give significant efficiency improvements, but it must be considered to what extent the path optimality and distance-to-land approximations worsens.

Comparing both methods, Figure 4.3 indicates that the path becomes a slightly worse approximation of the actual shortest path when using a variable divisor rule and a maximum distance of 300 meters compared to a fixed divisor value of 4 in this coastal map. Some key metrics can support this and compare adaptive edge divisor framed region quadtrees with framed region quadtrees and non-framed quadtrees. These are given in Table 4.1 and show that while the max distance quadtree only has slightly more vertices than the non-framed quadtree, the path is much closer to the fixed-divisor path, which undoubtedly is the best shortest-path approximation of the three by inspection of Figure 3.3b. While this comparison is not extensive enough to conclude on a general basis, it indicates that using an adaptive division rule based on max distance between frame vertices can benefit effi-



(a) Framed region quadtree with fixed divisor and divisor value set to $d = 4$. ENC data courtesy of The Norwegian Mapping Authority.



(b) Framed region quadtree with variable divisor and max distance $dist = 300$ meters. ENC data courtesy of The Norwegian Mapping Authority.

Figure 4.3: Graph search comparison of framed region quadtree with fixed and variable divisor rule

ciency in shortest-path approximations in coastal environments. It is therefore proposed to use such a strategy in the quadtree builder.

Table 4.1: Efficiency and approximation comparison of non-framed, and two different framed quadtrees

Metric name	Non-framed	Framed, fixed divisor	Framed, variable divisor
Quadtree vertices	60768	139392	62114
Path length [m]	17305.5	16034.4	16356.0
Search time [s]	0.197	2.237	0.167

Another benefit of region quadtrees is that they can be used for spatial collision queries. If a quadtree leaf region can be found for a given spatial point that contains this point, then it is guaranteed that the point is in the traversable part of the mission region. This can be used for fast collision checking when performing sample-based motion planning or real-time collision avoidance based on predictive control action assessment. Therefore, it is of significant importance that the quadtree system in the guidance supports not only graph traversal but also leaf region depth search.

4.2.3 Generating a free-space Voronoi skeleton for the mission region

An autonomous vessel on a coastal mission will typically have to navigate relatively open waters and narrow straits, channels, and bays while keeping sufficient distance to land and other hazards. The shortest path from some initial configuration to some goal position will often go much closer to land than what can be considered safe, and the search procedure should thus be informed about the distance to land to ensure sufficient safety margin in the

generated path. One way to achieve this in focused search techniques would be to use a conventional distance field in the search heuristic to penalize path candidates close to land. A key issue with this method is that it would penalize distance to land similarly regardless of the surroundings, which would make it expensive for the USV to pass through narrow straits or go into small bays. If one instead uses an artificial field based on the relative distance to land, this problem can be overcome. The Voronoi field, introduced in [21], is one such field and the Voronoi field strength depends on both distance to land and distance to the Voronoi skeleton of the environment free-space. In order to use the Voronoi field, the Voronoi skeleton of free space must therefore be built. There are several ways in which this can be done. One alternative is to generate a generalized Voronoi Diagram using the hazard polygon edge vertices and prune the diagram to obtain the skeleton. This approach allows for the use of well-established methods for generating the Voronoi diagram, such as the Fortune Sweep Algorithm [6], and thus ensures that an accurate Voronoi skeleton can be generated assuming an appropriate pruning strategy is developed..

4.2.4 Mission map service

The mission planner and collision avoidance system must efficiently check information related to the vessel's spatial position and future positions to traverse the environment safely and in an optimized manner. If the mission region information is stored in a spatial database, it can be retrieved by spatial database queries.

However, certain continuous information such as distance to nearest hazard or strength of the Voronoi field at a specific continuous spatial position must be calculated on-demand based on a combined knowledge about the mission region and the vessel position. The spatial query structure can still be utilized, making information retrieval more intuitive as all information about the environment is retrieved utilizing a unified interface. An alternative design for a ENC Manager in a guidance system could take a decentralized approach, only providing raw information to other systems. While not every system utilizes all the information available in the map service, there is significant overlap. Duplicate functionality in such a guidance system would hurt code maintainability and robustness, especially in the long term. Thus, the centralized approach where a map service class object is given to both the mission planner and the COLAV system, as shown in Figure 4.1, is considered to be the best option.

4.3 Mission Planner

The mission region in which an USV operates is often large, for example $100 \times 100 km^2$. In order to do autonomous mission planning in a reasonable amount of time, it is thus necessary to balance computational complexity and path optimality. The path generated by any sample-based motion planning technique will generally never be truly optimal. Instead, it can for some methods converge towards the optimum as computational time tends towards infinity. This is not necessarily an issue, as it can be argued that it is more important that a path is safe, feasible for the vessel to follow, optimized, and found within a reasonable amount of time instead of optimal. It is therefore proposed to base the mission

planner on a novel specialized version of the Hybrid A* algorithm, where modifications are made to overcome the practical challenges of motion-sampling, search efficiency and rules compliance in vast dynamic coastal environments. In the following, the challenges and proposed solution designs are described.

4.3.1 System of reference

Because the mission region can be large, the Earth's curvature must be considered in mission planning. This can be achieved by using a Geographic Coordinate System (GCS) with geodetic coordinates to describe the pose of the vessel and the path it should follow. From the perspective of a sample-based motion planner like Hybrid A*, this is problematic, as the equations of motion of the vessel used for sampling typically are expressed in a Cartesian coordinate system. There are several ways in which this problem can be solved. One approach is to define the equations of motion in the Earth-centered inertial (ECI) coordinate system and convert the result first to ECEF and then from ECEF to geodetic coordinates using, for example, any of the methods collected and compared by Zhu [12]. A more straightforward approach is to use a local geodetic coordinate system, which is created by attaching the origin of a East-north-up (ENU) system to a geodetic location. Assuming the distance the vessel will travel within this system of reference is short enough such that the curvature of the Earth is negligible, this will not introduce any significant error. In sample-based motion planning using methods such as Hybrid A*, simulation of vessel dynamics is performed when vertex successors are to be determined. By controlling the simulation duration and iteratively attaching the ENU coordinate system origin at the current vertex from which successors are to be determined, it can be ensured that the distance traveled always is within some radius deemed acceptable for negligible induced Earth curvature error.

4.3.2 Successor operator design

In a sample-based motion planner like Hybrid A*, the design for the successor operator is key for search efficiency and motion planning capability. There are two main characteristics to consider when designing the successor operator; simulation detail and simulation duration. The equations of motion utilized and the integration method define the level of detail to a significant extent. In this thesis, it is proposed to use a simplified 3DOF vessel model. Using a 6DOF vessel model and adding support for current, wind, and wave disturbances would result in more detailed motion sampling. However, this would require knowledge of more vessel parameters, data from an accurate ocean weather prediction model, including wave and wind predictions, and result in higher computational cost in integration, which would reduce search efficiency. The guidance system is intended to be used in coastal areas, and assuming the supervising operator ensures not to conduct missions in dangerous weather, the simplified model is considered sufficient. With the vessel model determined, some reflection around simulation duration is necessary. A longer simulation duration can result in rapid progression towards the goal position. However, in the context of Hybrid A* motion-sampling, where a maneuver correction only is controlled at the start of a simulation sample, it sacrifices maneuverability compared to a shorter simulation duration. To strike a balance between maneuverability and search progression, one

could rely on mission-specific tuning of the simulation time. However, this goes against the goal of developing a general guidance system for all coastal environments and can result in much inconvenience for the supervising operator. Another option is to allow for multiple course correction changes along the trajectory. However, this will increase the computational cost of motion sampling exponentially and can thus not guarantee that the search progression overall would improve compared to just using a single correction and a shorter simulation time. A third option is to develop a novel new method for adaptive simulation time in coastal environments. One such method is proposed in Subsection 4.3.4.

4.3.3 Heuristic design

In order to focus the search towards the goal, Hybrid A* uses a distance evaluation function taking the maximum of two heuristics. The former *non-holonomic-no-obstacles* heuristic uses Reeds-Sheep curves to prune the search such that the goal configuration is reached with the correct heading [21]. The proposed guidance system in this thesis does not take goal orientation into account, and thus only the latter *holonomic-with-obstacles* heuristic is used from the original method. The *holonomic-with-obstacles* heuristic should calculate the length of the shortest collision-free path to the goal in 2D, not taking vessel dynamics into account. An approximate cell decomposition graph search is a promising candidate to facilitate the implementation of this heuristic due to its low computational complexity. The traversable space in the mission region must first be decomposed into cells to use this search technique. One alternative would be to use a uniform grid with a high cell resolution where each grid tile is either considered free or occupied. The resulting shortest path will then be resolution optimal and tend towards the true optimal as the cell resolution tends towards infinity. In practice, such a decomposition strategy is infeasible to compute and store in memory. Moreover, even for reasonable cell resolution, this method would be inefficient for most coastal environments due to large unoccupied areas typically being present in such environments.

Instead, region quadtrees can be used to take advantage of the coastal free-space subregions. However, there is no such thing as free lunch, and the quality of the distance approximation will, in many cases, be significantly worse compared to a high-resolution grid decomposition. What is more, for a set of vertex successors, the distance approximations can be inconsistent with the actual, unknown distance-to-goal values. This inconsistency can be detrimental to the progress of the search algorithm, as more promising candidates in the Hybrid A* search frontier then potentially end up being prioritized lower for exploration. Framed region quadtrees can be utilized to remedy this issue while maintaining several efficiency improvements from region quadtrees. A static graph traversal algorithm can be utilized to find the shortest path in the quadtree graph. A* has excellent properties for this kind of problem and is therefore considered a prime candidate as it generally examines the smallest possible amount of vertices to find the optimal path in the graph [2]. With this, a proposed design for the *holonomic-with-obstacles* heuristic is complete. In order to increase search efficiency, the distance evaluation function can be scaled by a constant $k_{dist} \geq 1$ to prioritize progress towards the goal. However, making the search greedy can come at the cost of path optimality. Therefore, care must be taken in tuning

and evaluating such a constant to maintain sufficient exploration.

Using only the aforementioned distance evaluation functions for frontier prioritization would, for most paths, result in a proximity issue where the resulting path goes as close to land as possible. While the artificial buffering of environment obstacle geometries in pre-processing ensures that this does not result in grounding, it stands to reason that a safer route with a reasonable distance to static obstacles should be prioritized. To improve path safety, it is proposed to utilize the Voronoi field strength through an evaluation function added to the heuristic additively to ensure safety locally. This enables the generation of optimized paths with reasonable safety margins, assuming adequate tuning.

4.3.4 Adapting Hybrid A* to large-scale environments

Hybrid A* was in the original paper used in environments of a relatively small extent [21]. In order to improve efficiency in vast coastal environments, some additional features that extend beyond the original implementation should be utilized. The geometrical characteristics of coastal environments have already been taken advantage of in generating the aforementioned framed region quadtrees. They can also be utilized for search efficiency through coarse but fast collision check queries. In Hybrid A*, pose candidates are sampled by simulating the vessel for some time. In the original paper, this time is fixed [21]. This makes sense for a car in an area such as a parking lot where the clutter density is relatively coherent. However, in a coastal mission region there are narrow straits, bays cluttered with obstacles, and large open areas. Any pose-to-goal search can potentially traverse through all these different kinds of subregions. A fixed simulation time would then need to be small enough to allow for sufficient maneuverability at the expense of search efficiency in open areas where the search time ideally should be much longer to progress towards the goal more rapidly. By introducing an adaptive simulation time for the motion sampling that depends on the environment in the vicinity of the sample origin, maneuvering precision can be retained in areas where it is required. Simultaneously, progression can be rapid whenever possible because the motion sampling and heuristic calculations only are done a limited necessary amount.

In order to improve search efficiency further, one can make design changes to improve the *holonomic-with-obstacles* heuristic. Using the method described in Subsection 4.3.3 will result in repeated use of A* on the framed region quadtree graph. In order to avoid having to re-explore the graph-optimal path repeatedly, a strategy should be employed to take advantage of the search history in subsequent heuristic evaluations. One way to do this could be to only search at most once from every quadtree leaf region center and approximate the distance from a candidate configuration to the goal based on the leaf region in which the candidate is located. While enabling a dramatic reduction in the cost of evaluating the *holonomic-with-obstacles* heuristic, this solution has several significant drawbacks. The most prominent drawbacks are approximation consistency concerning placement within a leaf region and region extent. A better alternative is to search directly from every pose candidate but utilize parts of previous searches whenever possible. In order to achieve this, one can use sequence matching, a novel technique designed specifically for this thesis. Sequence matching is based on maintaining a search tree of graph-optimal paths that is iteratively built for each A* query to take advantage of previous searches and exploit

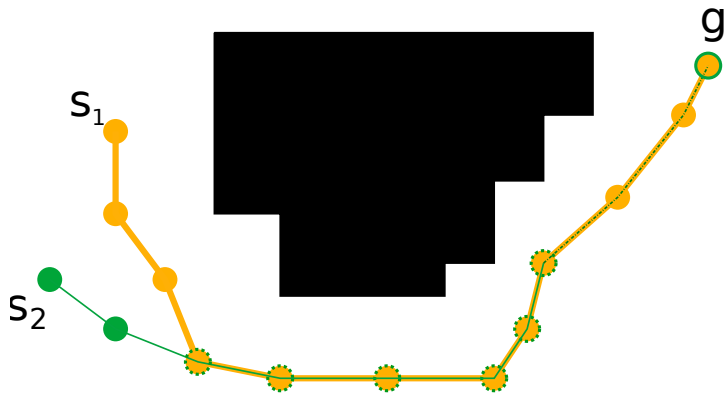


Figure 4.4: Illustration of sequence matching concept in the second query to a shortest path evaluator.

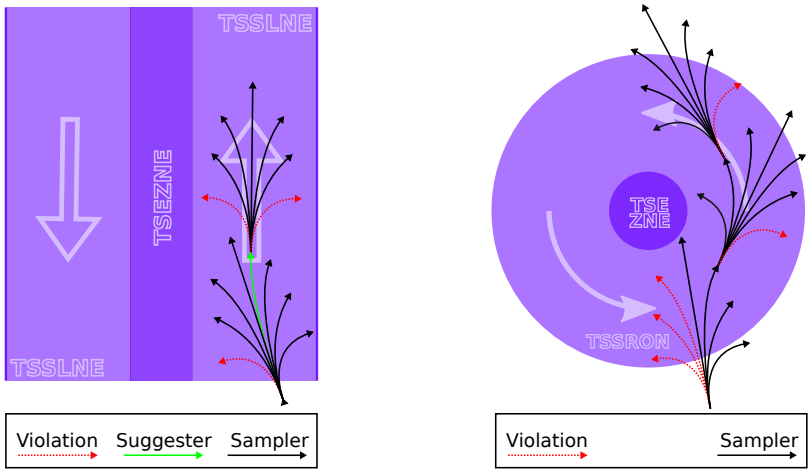
the fact that the goal position never changes. For an ongoing A* search, it can then be checked if the search has started to follow a path that is already in the search tree. If true, the search has attached to the search tree. The search can then be fast-forwarded because the graph-optimal path to the goal is already known from the attachment point onwards. Figure 4.4 offers a simplified illustration of how sequence matching is used to improve search efficiency from the second search iteration. In the first search iteration, from s_1 to g, the root of the search tree was established based on a graph-optimal path. When the second search iteration commences from s_2 to g, the A* heuristic results in the best path candidate starting to follow the path in the search tree eventually. Every new vertex in the frontier is evaluated to check if this best path candidate continues to follow the optimal path tree. If the sequence following the optimal path tree becomes sufficiently long, the search is interrupted, and the optimal path tree is used to provide the remaining part of the path towards the goal.

4.3.5 Incorporation of COLREG compliance

From the perspective of mission planning, the main COLREG rule with which compliance must be ensured is Rule 10, which describes how a vessel should interact with TSS objects. The aspects of this rule most relevant for a minimum level of compliance are given in the description of the rule in Appendix A, with supporting illustrations given in Figure A.2 and proposed strategies to comply illustrated in Figure 4.5.

In order to avoid crossing a separation zone, defined by the *tsezne* in Table B.2, it is proposed to interpret such objects as hazards. The proposed region quadtree search heuristic guiding the Hybrid A* search effort will guide the search clear of such areas, ensuring compliance. While this is a hard constraint from the perspective of the mission planner, it will only affect the nominal path and not any collision avoidance behavior.

Compliance with traffic separation lanes is handled differently. Instead of using only the hard constraint strategy used for zones, it is proposed to use a hard constraint, a soft con-



(a) Lane Strategy

(b) Roundabout Strategy

Figure 4.5: Illustration of proposed strategies to comply with relevant Traffic Separation Schemes.

straint affecting the Hybrid A* search heuristic, and a compliant course suggester combined. In a traffic lane, the course required to follow the general traffic flow is given. Thus, when it is detected that the mission planner frontier has reached a traffic lane, any configuration candidate in the opposite direction of the general traffic flow can immediately be discarded. All remaining candidates can thereafter be compared by evaluating the vessel's course if that candidate was to be followed and the defined course of the traffic lane. The evaluation result can generate a scaling factor for the distance heuristic, which is one nominally. Because the alterations of course in the mission planner sampling strategy should be coarse for search progression, the soft constraint can result in oscillatory behavior while following the traffic lane. It is proposed to utilize a course suggester, which determines the course alteration necessary to align the vessel and lane course for the current vertex retrieved from the search frontier to eliminate this oscillation. This course alteration is then utilized in addition to the fixed set of course alterations when motion sampling is performed to remedy the lack of high-resolution maneuverability in the motion sampling. The compliance strategy is visualized in Figure 4.5a.

The last type of TSS object with which compliance is considered necessary for minimum TSS compliance overall is the roundabout object, defined by the object *tssron* in Table B.2. Compared to the conventional road roundabout, a roundabout at sea does not have clearly defined lanes where one should reside while traversing around it. Thus, to facilitate compliance and route safety, the most important aspect is to ensure that roundabouts are traversed counterclockwise and that the roundabout's center is not crossed. The latter is ensured by compliance with *tsezne* objects, while the former requires a different strategy. It is proposed that for a given set of sampled configuration candidates, one should use knowledge of the corresponding candidate path segment and the roundabout center to evaluate if following the path segment from the current configuration to the sampled candidate results in a clockwise or counterclockwise motion with regards to the roundabout.

One way to do this is to take advantage of the vector cross product in a horizontal plane. In the horizontal plane, the cross product between two vectors is given as described in Equation 4.1

$$\vec{AB} \times \vec{AC} = (|AB||AC| \sin(\theta)\hat{z} \tag{4.1}$$

where $\hat{z}$ is a unit vector normal to the horizontal plane. Making a flat tangential plane by linear ENU projection centered in the roundabout center (R) and projecting the current configuration (V) and candidate configuration (C) to this horizontal plane, one can determine if going from V to C is clockwise or counterclockwise with respect to R by taking the cross product $\vec{RV} \times \vec{RC}$ and evaluating the scaling factor of the resulting $\hat{z}$. If the scaling factor is negative, the motion is clockwise, and if it is positive, it is counterclockwise. In the edge case where the scaling factor is zero, there is no rotation, meaning C is located on the extension of $\vec{RV}$. Be advised that to evaluate the scaling factor directly, Equation 4.2 can be utilized.

$$scaling_{cross} = (V_x C_y) - (V_y C_x) \tag{4.2}$$

4.4 Path tracking

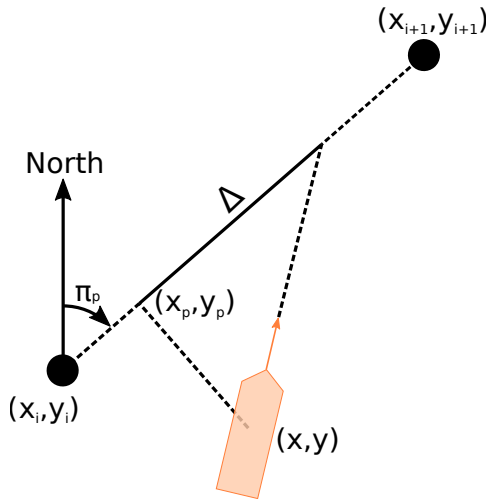


Figure 4.6: Illustration of LOS guidance law inspired by [48]. All coordinates are in the same global inertial coordinate system.

The path generated by the motion planning system in the mission planner must be followed by the USV. In order to achieve this, a path tracker must take the path parametrization as an input and output control values to the USV autopilot system. One path tracker that is frequently used for maritime applications is the LOS Guidance Law illustrated in Figure

4.6. The LOS guidance law computes for a given cross-track error on a path segment between two waypoints the desired course angle χ_d as shown in (4.3). The following explanation of the LOS guidance law is based on an earlier project [49], which follows the theory and notation in [48].

$$\chi_d = \pi_p - \tan^{-1}(\frac{1}{\Delta} y_e^p) \tag{4.3}$$

The equation for desired course χ_d given in Equation 4.3 requires some explanation. Δ is the lookahead distance and y_e^p is the crosstrack error expressed in the path-tangential coordinate system p aligned with the line segment from waypoint (x_i^n, y_i^n) to (x_{i+1}^n, y_{i+1}^n) in an inertial, Cartesian world frame n_i. Calculating y_e^p can then be done by means of a simple coordinate transformation as shown in Equation 4.4.

$$\begin{bmatrix} x_e^p \\ y_e^p \end{bmatrix} = \mathbf{R}_p^n(\pi_p)^T \begin{bmatrix} x^n - x_p^n \\ y^n - y_p^n \end{bmatrix} \tag{4.4}$$

In the equation, $\mathbf{R}_p^n(\pi_p) \in SO(2)$, (x^n, y^n) is the USV position in the inertial world frame n and π_p is the rotation of frame p with regards to frame n, calculated using 4.5.

$$\pi_p = atan2(y_{i+1}^n - y_i^n, x_{i+1}^n - x_i^n) \tag{4.5}$$

The LOS path tracker outlined above conventionally requires waypoints defined in Cartesian coordinates. The mission planner method proposed in Section 4.3 uses a GCS and thus the path is parameterized with geodetic coordinates. Without some conversion, it is thus incompatible with the conventional LOS based path-tracker outlined above. One method to overcome this would be to change the guidance law to work in a geodetic frame of reference. However, implementing this can be impractical and make tuning less intuitive. Another option is to use a sequence of Cartesian frames attached to each waypoint and use the frame related to the latest visited waypoint when tracking the path segment towards the next waypoint. Assuming that the geodesic distance between waypoints is limited, this method will only introduce negligible error due to the Earth's curvature.

4.5 Collision avoidance system

A COLAV system is responsible for COLREG compliant collision avoidance and damage potential mitigation during mission execution. In order to maintain a high level of modularity in the guidance system, the COLAV subsystem should be entirely independent of the mission planner and path tracker. This can be achieved by using the interface design proposed in [33], where the COLAV system is informed about the setpoints suggested by the path tracker to follow the nominal path and produces setpoint corrections based on this to fulfill its responsibilities. This interface is used in Figure 4.1 and allows all main subsystems to be completely invariant of how other subsystems function internally. This

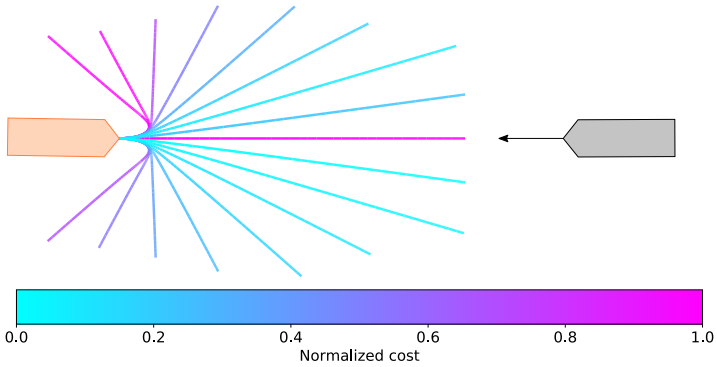


Figure 4.7: Collision avoidance based on predictive control selection concept proposed in [33] in a single-obstacle head-on scenario.

high level of modularity improves system maintainability. Furthermore, it offers flexibility in making changes to subsystems or replacing subsystems entirely. With the responsibility and external interface of the COLAV system defined, what remains is to determine the collision strategy to employ internally.

A plethora of COLREG compliant collision avoidance strategies exist. The ones considered most promising for this project are introduced in Section 2.3. Of these methods, the local collision avoidance strategy proposed by [33] based on the research by [22] is considered to be the most optimal method for the proposed guidance system due to its versatility, predictability, and the ability for incremental improvements.

This method is based on defining a fixed set of motion primitives[1], obtaining the predicted trajectory for the vessel for every combination using a vessel model, and choosing the motion primitive combination candidate with the best trajectory. Evaluating the trajectory candidates can be quantified utilizing a cost function where COLREG compliance, damage mitigation, vessel actuator wear, and path optimality are taken into account. An illustration to highlight how one iteration of the control selection looks is provided in Figure 4.7. For maritime vessels, the motion primitive combinations can, for example, be combinations of course offsets and speed multipliers. While this method is conceptually simple and can be computationally cheap, it is versatile, can handle multiple-vessel avoidance, and enables quite sophisticated behavior such as reducing damage potential when a collision is inevitable. Furthermore, in addition to being verified in a simulation environment [33], sea trials based on the concept have shown that it can perform with uncertainty in measurements and predictions [41].

[1] In the context of the proposed COLAV method, the motion primitives are a collection of course corrections and speed multipliers

Chapter 5

Implementation

5.1 Software

The guidance and collision avoidance system introduced in this thesis depends on a collection of third-party open-source software libraries and middleware. In order to give a complete description of the implemented system, it is considered necessary to highlight these dependencies.

5.1.1 ROS

The system developed in this thesis utilizes Robot Operating System (ROS) Noetic as its middleware to provide a framework for communication and standardization. ROS is a distributed framework with message-based communication between nodes. From the perspective of the developer, system features are designed as nodes, collections of which are organized in packages [39]. Nodes can be written in different programming languages and interact using messages and service calls. Message-based communication and service calls are critical to the implementations in this project and are what enable systems modularity.

Communication in ROS is message-based in a peer-to-peer network [39]. The network is controlled by a *roscore* node, which keeps track of communication channel topics and their publishers and subscribers. When a subscriber node receives a message, this asynchronously triggers a callback function. While topics can have multiple publishers and subscribers connected to them simultaneously, the communication channels are all one-to-one between the main modules in the implemented guidance system except for a simulated target tracker.

5.1.2 OpenMP

OpenMP is a programming library with runtime routines and compiler directives to enable shared-memory parallelism [14]. Key advantages of using OpenMP instead of other options like *pthread* include lower overhead and minimal need for code modification. While OpenMP is highly versatile and can take advantage of modern multi-core processors in many different scenarios, the system proposed in this thesis will primarily use it in scenarios where nested for-loops are used. Such nested loops occur primarily when building and querying the mission region database, when building the Quadtree and when querying the Voronoi skeleton. All asynchronous behavior related to the general guidance system, such as callbacks and running multiple subsystems concurrently, are handled by the ROS middleware.

5.1.3 GDAL/OGR

The Geospatial Data Abstraction Library (GDAL) is a software library consisting of drivers for translating and editing geospatial data formats [57]. It includes the OGR Simple Features Library to handle geospatial vector data and is therefore often denoted as GDAL/OGR. The library is widely used by spatial data applications, with three notable mentions being QGIS, MapServer, and Google Earth. For vector data, the library presents a unified abstract data model [57]. Through a C++ Application Programming Interface (API), it can be used to load different types of geospatial vector files such as S-57 ENCs and SQLite databases as *ogr::DataSource* objects. These objects all have the same interface and thus can be read and manipulated in precisely the same way [57].

In addition to being used for converting and modifying geospatial databases, GDAL/OGR can also be used for many geospatial queries such as collision checking, geometry processing, and frame conversions. Core functions relevant for the implemented system are described briefly below, based on the documentation available [57] and a research effort by Otterholm [43].

- *OGRBoolean OGRGeometry::Intersects (OGRGeometry *)*: The core of all collision checking in this thesis. Can for any two georeferenced geometries determine if they intersect. Internally, the function uses a variation of the Dimensionally Extended Nine-Intersection Model (DE-9IM) through the GEOS Geometry Engine [50]. DE-9IM is an efficient model for determining spatial relations such as intersection and containment between geometries [11].

- *OGRGeometry* OGRGeometry::UnionCascaded()*: Responsible for dissolving geometrically overlapping geometries in map pre-processing by taking the geometric union of all layer features and utilizing cascading for efficiency. Once more, GEOS [50] is utilized by GDAL/OGR to achieve this.

- *int OGRCoordinateTransformation::Transform(int nCount,double* x, double* y, double* z)*: Used by the Geotf library described in subsection 5.1.4 to efficiently and accurately transform 3D points from a source to a destination georeferenced frame of reference.

- *OGRBoolean OGRGeometry::Distance (OGRGeometry *)*: Used in this thesis whenever the shortest distance between geometries or layers of geometries are to be found. Can for any two georeferenced geometries determine the shortest distance between the geometries using GEOS [50].

Any reader interested in developing their own software based on GDAL/OGR is advised to consult the GDAL/OGR library documentation [57] and GEOS documentation [50], as the overview above only is intended to give the reader a brief overview of some of the most advanced features of GDAL/OGR utilized by this thesis and its library dependencies.

5.1.4 Geotf

Geotf is a C++ software library for frame conversion of geodetic data developed by the ETH Zürich Autonomous Systems Lab [54]. Using this library, conversion between multiple georeferenced frames can be performed on a high abstraction level. Furthermore, the library supports mapping between georeferenced frames and ROS TF frames. While the library is not supported by any published paper or other work, this is no cause for concern as it relies on GDAL/OGR for all backend calculations and conversions.

The Geotf library is utilized by creating a realization of the *GeodeticConverter* class. This converter object has several functions in its public interface, and the ones relevant to this thesis are described below:

- *bool addFrameByEPSG(name, id)*: Add a coordinate frame by its EPSG identifier. If for example one wants to add a WGS84 reference frame, which is used by services such as GPS, the EPGS code is 4326. The function creates a *OGRSpatialReference* object.
- *bool addFrameByENUOrigin(name, lat, lon, alt)*: Add an ENU frame where the origin is georeferenced by a geodetic position. Creates a *OGRSpatialReference* object initialized to use the WGS84 reference system. Furthermore, an orthographic projection[1] is added to the spatial reference, centered at sea level at the geodetic position.
- *bool convert(input_frame, input, output_frame, output)*: Convert a position in one georeferenced input frame to an output frame, given that both frames are already defined. In order to achieve this, the spatial references of both frames are used to initialize a *OGRCoordinateTransformation* object if such an object for the relevant transformation is not already present.

5.1.5 Odeint

Odeint is a modern C++ header-only library for solving initial value problems of ordinary differential equations numerically [23]. For the purpose of the implemented guidance system, this is used for both real-time simulation and trajectory prediction. Trajectory prediction is used both in the Mission Planner and the collision avoidance system and thus it is of monumental importance that the library has efficient steppers implemented and that

[1] An orthographic projection is a 2D projection of 3D space

these steppers are implemented with performance in mind. Odeint offers many steppers, including efficient, adaptive stepsize Runge-Kutta steppers [23] and it is thus ideal for the guidance system[2].

5.1.6 GeographicLib

GeographicLib is a C++ software library that can be used for solving problems in the field of Geodesy [51], such as the inverse geodesic problem. The library focuses on result accuracy but has also proven to be robust in the sense that the solution to the inverse problem is always found. This is a significant improvement compared to Vincenty's method which fails to converge, and thus fails to find a solution, for the inverse problem for nearly antipodal points [28].

5.1.7 JC_Voronoi

JC_Voronoi is a C++ header-only library for creating 2D Voronoi diagrams from a set of points [60]. It is used in this thesis as a foundation to create Voronoi skeletons of the mission region. It should be noted that while the implementation is sparsely documented, it is an implementation of the well-known and well-documented Fortune's sweep algorithm [6].

5.2 Electronic Navigational Chart Manager

The Electronic Navigational Chart Manager[3], or ENC Manager for short, is implemented as a pre-processor and a map-service class using C++. Additionally, a standalone C++ ENC extraction library *enc_extract_lib*[4] is developed to provide some of the underlying functionality for the pre-processor. The classes take advantage of the ROS Noetic framework for parameter handling and logging of debug messages. Compared to the other subsystems in the guidance system, the ENC manager does not utilize message-based communication or service calls. Instead, realizations of the classes are owned by other subsystems that require them. While this does come at the expense of extra memory utilization, it prevents computational and transactional overhead associated with sending static data over the ROS network and was thus considered a worthwhile compromise.

Mission region pre-processing is performed as a three-stage procedure, as shown in Figure 5.1. In the following, this procedure is described.

5.2.1 Information extraction from Electronic Navigational Charts

Before initiating the pre-processing procedure, electronic navigational charts covering the desired mission region must be obtained. If one is to perform a mission along the US coastline, NOAA have charts easily accessible online. For missions along the Norwegian

[2]Please note that other libraries do exist and that no library performance comparison has been conducted.
[3]Made available through the open-source *USV Guidance system* GitHub repository [55]
[4]Made available through the separately developed open-source library *enc_extract_lib* GitHub repository [56]

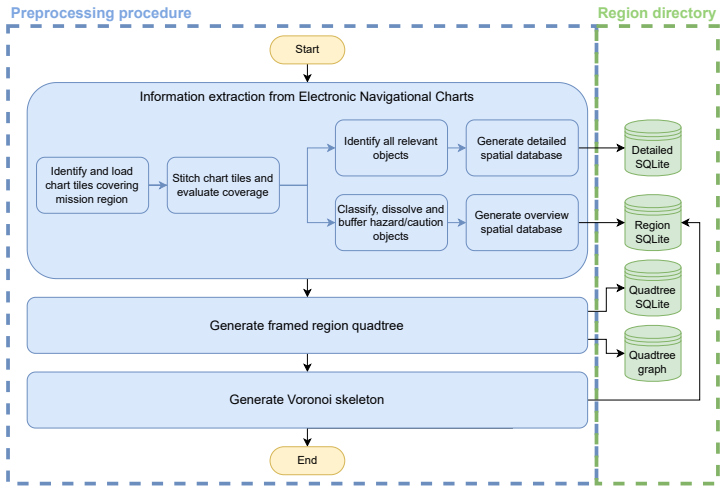


Figure 5.1: Visualization of proposed multi-stage pre-processing procedure for Electronic Navigational Charts.

coastline, access must be requested from The Norwegian Mapping Authority. Obtained chart tiles go through an automated registration process where metadata is extracted to be used by the library. This ingestion process is implemented as a Python script in the *enc_extract_lib*. For every chart to ingest, the ingestion procedure uses the S-57 driver from GDAL/OGR to open the chart and read the Data Set Identifier (DSID) [17] and the extent of the tile. The filename and metadata attributes are then appended to an index file, and the ENC is stored by the program in non-volatile memory together with all other registered ENCs.

Assuming all relevant ENCs have been ingested, a mission region has been defined, and vessel dimensions are specified, pre-processing can commence. Following the flowchart in Figure 5.1, all mission region chart tiles are loaded, stitched, cut, and evaluated for coverage to identify any subregion where there is no map data available. Two spatial databases are then generated; a detailed and a region overview database. The former is generated by adding all objects considered relevant[5] to a SQLite database to provide additional details when the information in the region overview is not enough to make decisions during mission planning and execution. The latter database is generated by iteratively going through all relevant objects, classifying them as either hazard objects, caution objects, or irrelevant. The geometric properties of the hazards and cautions are stored in a geospatial SQLite database consisting of a hazard layer and a caution layer. The geometries in each layer are thereafter dissolved using cascading such that any objects that overlap in the layer become one. The dissolved objects are finally extended with a buffer region based on the vessel dimensions, and the SQLite database is saved for use by the ENC Manager Map Service during mission planning and execution.

[5]The relevant objects are given in Table B.1 and Table B.2 in Appendix B

5.2.2 Interpreting and representing free space in the mission region

The responsibilities of the ENC manager with regards to pre-processing do not end with the extraction and interpretation of ENC information. It is also responsible for building a framed region quadtree of the free space in the mission region. This quadtree is not only used as a graph structure for the Hybrid A* *holonomic-with-obstacles* search heuristic but also for determining for any arbitrary point which region it is inside if any. The latter purpose is quite similar to how point quadtrees are used [32], with the difference being that the free space in the environment dictates how the region quadtree is structured instead of a point cloud. In order to build the quadtree, two key questions must be answered:

1. How should it be determined that a region is obstacle-free?
2. Given that a region is obstacle-free, how should graph vertices be placed and interconnected?

For an arbitrary region $\mathcal{R}$ the occupancy region can be calculated using the GDAL/OGR library to extract geometry overlap and calculate an occupancy ratio based on the area of overlap divided by the area of $\mathcal{R}$. Calculating the overlap geometry between a quadtree region and all polygons in a mission region is computationally expensive. A two-stage process is therefore used to calculate the occupancy to lower computational cost. First, all polygons with which the region possibly overlaps are found by checking for envelope intersection. The polygon envelopes and the quadtree region candidates are all quadrilaterals. Therefore, this intersection checking is computationally cheap. In the second stage, the occupancy is found by evaluating the overlap with polygons whose envelope intersected the region.

A region is considered part of the free space only when it has an occupancy ratio of zero. Whenever such a region is found, a procedure must be run for placing and interconnecting graph vertices. The procedure is described by Algorithm 1 and the interconnection logic is illustrated in Figure 5.2.

Algorithm 1 Add region vertices for a free region.

function $addRegionVertices(G, \mathcal{R})$

Require: $length_{max} \geq 0$

1: $divisor_{\mathcal{R}_{NS}} \leftarrow \mathbf{getDivisorValue}(\mathcal{R}, NS, length_{max})$
2: $divisor_{\mathcal{R}_{EW}} \leftarrow \mathbf{getDivisorValue}(\mathcal{R}, EW, length_{max})$
3: $points_{\mathcal{R}} \leftarrow \mathbf{getRegionFramePoints}(\mathcal{R}, divisor_{\mathcal{R}_{NS}}, divisor_{\mathcal{R}_{EW}})$
4: **for all** $point_{\mathcal{R}} \in points_{\mathcal{R}}$ **do**
5: **if** $point_{\mathcal{R}} \notin G$ **then**
6: $\mathbf{addGraphVertex}(point_{\mathcal{R}})$
7: $\mathbf{connectEdges}()$
8: **end if**
9: **end for**

In Algorithm 1, $divisor_{\mathcal{R}_{NS}}$ and $divisor_{\mathcal{R}_{EW}}$ determine how many line segments the north/south and east/west edges of the region should be split into, respectively. In order to facilitate the variable divisor concept described in Subsection 4.2.2, a $length_{max}$ value

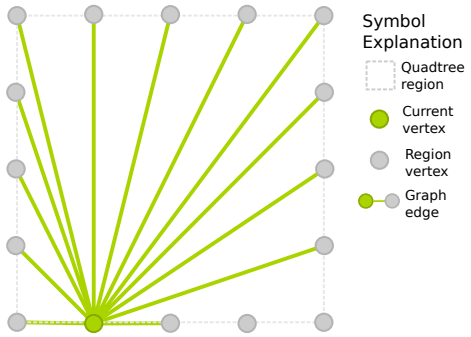


Figure 5.2: Interconnecting vertices in framed regional quadtree. Illustration courtesy of [49]

must be set. It specifies the maximum geodesic length between frame vertices, and by being a function of this value the divisor enforces this limit.

Having established the procedure for determining the occupancy of a region and handling vertices in free regions, what remains is to describe the building procedure tying these parts together to create a framed region quadtree. This is described in Algorithm 2. The reader is advised that the $splitRegion(region_{current})$ procedure splits a region $\mathcal{R}$ into four sub-regions using geodetic coordinates, not Cartesian coordinates.

Algorithm 2 Build framed region quadtree Algorithm. Adapted from [49]

function $buildFramedQuadtree(map, G)$

$\mathcal{R}_{root} \leftarrow$ **getMissionRegion**(map)
while existsRegionToCheck() **do**
  $\mathcal{R}_{current} \leftarrow$ **getRegionToCheck**()
  if regionExtentBelowThreshold$(\mathcal{R}_{current})$ **then**
    $continue$
  end if
  $f_{\mathcal{R}} \leftarrow$ **getOccupancyRatio**$(\mathcal{R}_{current})$
  if $f_{\mathcal{R}}$==1.0 **then**
    $continue$
  else if $f_{\mathcal{R}} == 0.0$ **then**
    addRegionVertices$(G, \mathcal{R}_{current})$
  else
    splitRegion$(\mathcal{R}_{current})$
  end if
end while

Observe in Algorithm 2 that when regions to check get below some area, as evaluated by **regionExtentBelowThreshold**$(\mathcal{R}_{current})$, they are discarded. What can be considered a reasonable area threshold depends on the ownship vessel dimensions. It is proposed that the minimum extent should be chosen small enough such that the quadtree reaches

into any narrow channel, bay, or harbor in which the USV could safely travel.

5.2.3 Generating a free-space Voronoi skeleton for the mission region

In order to generate a Voronoi skeleton of the free-space in the mission region, a Voronoi diagram based on the boundary vertices of the obstacle polygons can be used as a starting point. One fast and reliable way to generate a Voronoi diagram based on a set of points is to use the Fortune's Sweep algorithm [6]. An efficient implementation of this technique is made available through the *jc_voronoi* library published under the MIT license on GitHub by Mathias Westerdahl [60]. Looking at the main algorithm of the Voronoi skeleton generator, given in Algorithm 3, this library is used in the $\mathbf{buildVoronoiDiagram}(sites)$ procedure to build the diagram on which pruning is thereafter performed to extract the Voronoi skeleton.

Algorithm 3 Voronoi skeleton generator

function $buildVoronoiSkeleton(obstacles)$

1: $sites \leftarrow \mathbf{getEdgeVertices}(obstacles)$
2: $diagram \leftarrow \mathbf{buildVoronoiDiagram}(sites)$
3: $edges_{candidate} \leftarrow \mathbf{getCandidateEdges}(diagram)$
4: $point_{map} \leftarrow \mathbf{associatePointsEdges}(edges_{candidate})$
5: $edges_{skeleton} \leftarrow \mathbf{pruneEdges}(edges_{candidate}, point_{map})$
6: $\mathbf{addToRegionDataset}(edges_{skeleton})$

The algorithm requires some further explanation. $\mathbf{getCandidateEdges}(diagram)$ retrieves all edges that do not collide with obstacle polygons or are partially outside the mission region while $\mathbf{associatePointsEdges}(edges_{candidate})$ associates with every vertex in the Voronoi diagram, all edges this point is connected to. This is crucial for the novel edge pruning algorithm $\mathbf{pruneEdges}(edges_{candidate}, point_{map})$ further detailed in Algorithm 4.

$\mathbf{pruneEdges}(edge_{candidates}, point_{map})$ is a novel algorithm developed for this thesis to generate the required Voronoi skeleton supporting structure. Iteratively, edges that are only connected to other edges at one end are evaluated using a distance ratio described by the formula given in Equation (5.1)

$$ratio = \frac{length(edge)}{|d_1 - d_2|} \tag{5.1}$$

where d_1 and d_2 are the distance to the nearest obstacle from the two edge endpoints. Ratios closer to 1 indicate that the edge is almost normal to the nearest land polygon edge. Any Voronoi skeleton edge would be approximately parallel and have a rato $\gg 1$. Thus, edges with a distance ratio $ratio \leq \beta_{voronoi}$ where $\beta_{voronoi} \geq 1$ is a tuneable parameter should be removed[6]. This pruning rule is prone to false positives far from obstacle polygons and in the presence of complex obstacle geometries. It is made far more

[6]The value of $\beta_{voronoi}$ used in testing is given in Table C.3

Algorithm 4 Prune Voronoi diagram edges

function $pruneEdges(edges_{candidates}, point_{map})$

1: **while** $\mathbf{pruningEffective}()$ **do**
2: $edges_{prunecandidate} \leftarrow \emptyset$
3: **for all** $association_{point,edges} \in point_{map}$ **do**
4: **if** $\mathbf{getEdgeCount}(association_{point,edges}) == 1$ **then**
5: $\mathbf{push}(edges_{prunecandidate}, \mathbf{getEdges}(association_{point,edges}))$
6: **end if**
7: **end for**
8: **for all** $edge_{prunecandidate} \in edges_{prunecandidate}$ **do**
9: $ratio \leftarrow \mathbf{getDistanceRatio}(edge_{prunecandidate})$
10: **if** $ratio \leq \beta_{voronoi}$ **then**
11: $\mathbf{removeEdge}(edge_{prunecandidate})$
12: **end if**
13: **end for**
14: **end while**
15: **return** $\mathbf{getSurvivingEdges}()$

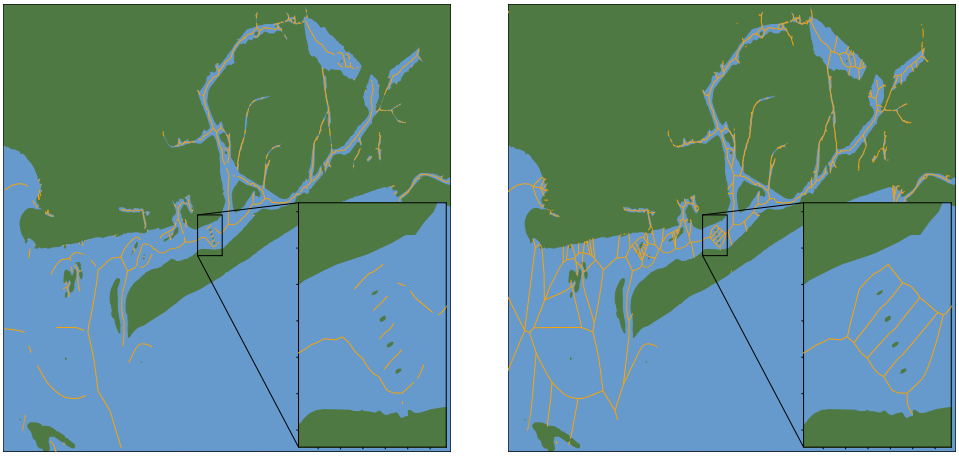
conservative by only considering partially connected edges. Removing partially connected edges must be done iteratively because a set of new edges can become partially connected with each removed edge. When no more edges are removed by the pruning rule, pruning is no longer effective and the process can be stopped. The effectiveness is monitored by the $\mathbf{pruningEffective}()$ procedure. The iterative method is far less prone to false positives in removal detection than using the pruning rule once on all edges. Thus a more accurate Voronoi skeleton is built overall, as can be seen in the comparison in Figure 5.3.

5.2.4 Mission Map Service

The mission map service is the second component of the ENC manager, and all knowledge other modules have about the static environment is retrieved through it by these modules owning manager objects. There are three main aspects the other subsystems need to know about the static environment:

1. **Intersection**: Does a point or collection of line segments collide with any spatial object?
2. **Distance**: What is the distance to the nearest spatial object of a certain type?
3. **Distance context**: How close to land is it reasonable to go in an arbitrary area?

The first aspect, *intersection*, is handled by retrieving the appropriate layer in the mission region database and for every feature in this layer, checking for intersection using the *OGRBoolean OGRGeometry::Intersects (OGRGeometry *)* procedure described in Subsection 5.1.3. Knowledge about *distance* is handled similarly but is based around *OGRBoolean OGRGeometry::Distance (OGRGeometry *)* and uses a copy of the pre-processed mission region database stored in RAM for improved efficiency. Furthermore, the distance



(a) Skeleton generated using pruning rule evaluated on all edges at the same time over a pre-processed mission region around Jamaica Bay Estuary in New York State. ENC data courtesy of NOAA.

(b) Skeleton generated using pruning rule evaluated iteratively on partially connected edges over a pre-processed mission region around Jamaica Bay Estuary in New York State. ENC data courtesy of NOAA.

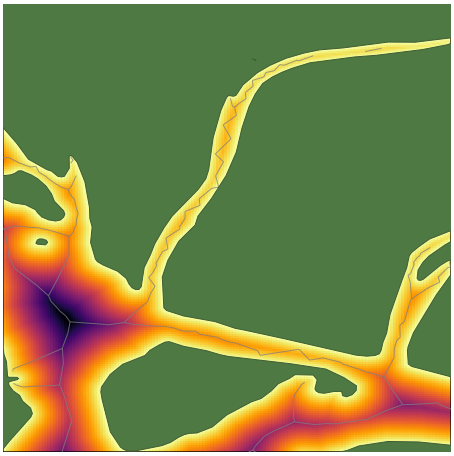
Figure 5.3: Comparison of Voronoi skeleton built pruning all edges once and pruning iteratively.

function has an optional saturation feature.

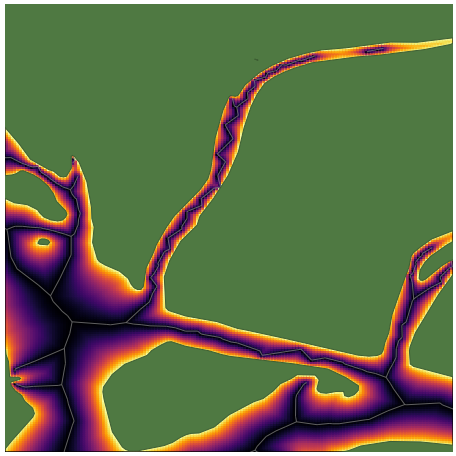
When it comes to providing knowledge about how close to land it is reasonable to go, there are many ways this could be handled. For this thesis, the Voronoi field introduced in [21] is utilized. This artificial potential field is suitable because it will return large values close to land in large open areas but return smaller values for a similar distance to land in narrow channels and through narrow straits. This assures that it does not become artificially expensive to pass through such areas, which would result in worse optimality characteristics of any path found with Hybrid A* or similar methods. The Voronoi field is constructed by utilizing knowledge about the distance to the nearest hazard d_h and the distance to the Voronoi skeleton d_v as shown in Equation (5.2)

$$f_v = \frac{\alpha_{voronoi}}{\alpha_{voronoi} + d_h} \frac{d_v}{d_v + d_h} \frac{(d_h - sat_{distance})^2}{(sat_{distance})^2} \tag{5.2}$$

where $\alpha_{voronoi}$ is a tuneable parameter determining the falloff rate of the field, and $sat_{distance}$ is the maximum distance to land that can occur. The chosen values for both are given in Table C.3. The saturation feature of the distance function is utilized to ensure this in practice. To illustrate the value of using the Voronoi field, Figure 5.4 compares the standard distance field and the Voronoi field in a map with relatively narrow channels. Observe that the field value reflects the aforementioned reasonable distance-to-land concept.



(a) Distance field visualized over a pre-processed portion of Jamaica Bay Estuary in New York State. ENC data courtesy of NOAA



(b) Voronoi field field visualized over a pre-processed portion of Jamaica Bay Estuary in New York State. ENC data courtesy of NOAA

Figure 5.4: Comparison of artificial distance field and Voronoi field. Both are visualized using an *inferno* colormap with field strength represented by warmer color.

5.3 Mission Planner

The Mission Planner[7] is implemented in ROS Noetic as a C++ node. Communication with the host vessel is message-based, while information about the static environment is handled by database query through an ENC manager map service object owned by the Mission Planner internally. This design choice is made to avoid sending large amounts of static data over the ROS network, with the potential disadvantage of increased memory consumption if many Mission Planners are spawned. However, there is only one Mission Planner for every USV. Thus this is not considered to be an issue.

The Mission Planner is requested to find a path either from the current USV configuration or a custom configuration through a call to a ROS service provided by the Mission Planner. The path is found using a novel specialized version of Hybrid A*, designed and implemented specifically for USV mission planning in large-scale coastal maritime environments.

5.3.1 Motion planning with Hybrid A*

Motion planning in this thesis is achieved by utilizing the Hybrid A* Algorithm and motion-sampling is facilitated by the Viknes 830 model from [22]. The Hybrid A* implementation is based upon the underlying theory described in Subsection 3.2.2 and design choices highlighted in Section 4.3, with added features for improved search efficiency in large-scale maritime environments as proposed in Subsection 4.3.4. Motion sampling

[7] Made available through the open-source *USV Guidance System* GitHub repository [55]

is developed using object-oriented programming, where the vessel model implementation from [35] is adapted to work with the Odeint C++ library [23] for integration of the equations of motion. Changes have also been made to the internal structure and external interface of the vessel model, to allow usage as a traditional simulator without sacrificing performance when performing motion-sampling.

Whenever the Mission Planner receives a mission plan service request, it searches for a path using the aforementioned specialized version of Hybrid A*, the behavior of which is described in Algorithm 5.

Algorithm 5 Hybrid A* Search algorithm

function $search()$
Require: $s_{start} \neq nullptr$
Require: $s_{goal} \neq nullptr$
Require: $\boldsymbol{\chi} \neq \emptyset$
1: $\mathbf{addToFrontier}(s_{start}, 0)$
2: **while** $\mathcal{F} \neq \emptyset$ **do**
3: $s_{current} \leftarrow \mathbf{getFromFrontier}()$
4: $r_{current} \leftarrow \mathbf{getLeafRegionContaining}(s_{current})$
5: **if** $r_{current} == \mathbf{getLeafRegionContaining}(s_{goal})$ **then**
6: $break$
7: **end if**
8: $distance \leftarrow \mathbf{getDistance}(s_{current}, s_{goal})$
9: $simTime \leftarrow \mathbf{adaptiveSimulationTime}(s_{current}, distance)$
10: **if** $\mathbf{inTssLane}(s_{current})$ **then**
11: $\mathbf{courseSuggester}(\mathbf{tssLaneOrientation}(s_{current}), \boldsymbol{\chi})$
12: **end if**
13: **for all** $\chi \in \boldsymbol{\chi}$ **do**
14: $s_{candidate} \leftarrow \mathbf{getCandidateState}()$
15: $hor_{candidate} \leftarrow \mathbf{simulateVessel}(s_{candidate}, \chi, simTime)$
16: **if** $\mathbf{similarClosed}(s_{candidate})$ **or** $\mathbf{collision}(s_{candidate}, hor_{candidate}, r_{current})$ **or** $\mathbf{tssViolation}(hor_{candidate})$ **then**
17: $continue$
18: **end if**
19: $s_{next} \leftarrow \mathbf{getNextVertex}(s_{candidate})$
20: $cost \leftarrow \mathbf{getCost}(s_{next})$
21: **if** $\mathbf{explored}(s_{next}) ==$ **false or** $cost < \mathbf{costSoFar}(s_{next})$ **then**
22: $priority \leftarrow \mathbf{heuristic}(s_{current}, s_{next}, cost)$
23: $\mathbf{addToFrontier}(s_{next}, priority)$
24: **end if**
25: **end for**
26: **end while**
27: **return** $\mathbf{reconstructPath}()$

Looking at the $\mathbf{search}()$ algorithm described in Algorithm 5, some symbols and procedures require further explanations. Starting with the notation, $\boldsymbol{\chi}$ is a set of course correc-

tions, and the frontier $\mathcal{F}$ is a priority queue where vertices with the lowest cost have the highest priority. Most procedures in the algorithm are considered self-explanatory if one is familiar with the original Hybrid A* Algorithm from [21]. However, a handful are unique to this thesis and should thus be detailed further, the first of which is the $\mathbf{collision}(\dots)$ procedure described in Algorithm 6.

Algorithm 6 Collision checking algorithm

function $collision(s_{candidate}, simHorizon, r_{current})$
1: **if** $r_{current} == \mathbf{getLeafRegionContaining}(s_{candidate})$ **then**
2: **return** $false$
3: **end if**
4: **if** $\mathbf{getLeafRegionContaining}(s_{candidate}) == nullptr$ **then**
5: **return** $true$
6: **end if**
7: $simHorizonGeodetic \leftarrow \mathbf{transformToGeodetic}(simHorizon)$
8: **if** $\mathbf{intersects}(simHorizonGeodetic, HazardLayer)$ **then**
9: **return** $true$
10: **else**
11: **return** $false$
12: **end if**

$\mathbf{collision}(s_{candidate}, horizon_{candidate}, r_{current})$ has two stages, a fast check stage and a thorough check stage. The former is represented by checking the candidate $s_{candidate}$ vertex against the region quadtree representing free space in the mission region. If $s_{candidate}$ lies within the same obstacle-free region as the current vertex $s_{current}$, then the path between the configurations must be collision-free. Furthermore, if $s_{candidate}$ does not lie within a collision-free region at all, then there must be a collision along the simulation horizon. Doing region checks in the quadtree is computationally cheap but not conclusive for all scenarios. For scenarios where the first stage is inconclusive, the procedure proceeds to the second thorough-check stage. After transforming the simulation horizon leading up to $s_{candidate}$ to geodetic coordinates, the map service from the ENC manager is used to check for any intersection. The collision checking can be considered a pruning of candidates and is necessary to ensure that all segments in the search tree are collision-free. By checking for collision before evaluating the heuristic, the heuristic procedure will not be called more times than what is necessary. This is key for search efficiency, as the heuristic, on average, is the most computationally expensive procedure of candidate evaluation.

It is emphasized that the search is done in a geodesic coordinate system using the WGS84 datum. Thus the pose in the vessel state $s_{candidate}$ must be transformed to a local Cartesian coordinate system if one is to use a traditional 3DOF surface vessel model to do motion sampling. From the perspective of the Hybrid A* search procedure in Algorithm 5 this technicality is transparent by design because the implemented simulator has the necessary support implemented. It silently creates a local frame of reference with $s_{current}$ as the origin before integrating the equations of motion. After the integration, the resulting pose is transformed into the global geodetic coordinate system and returned to the caller.

Algorithm 7 Hybrid A* Heuristic

function $heuristic(s_{current}, s_{next}, cost)$

1: $d_{voronoi} \leftarrow \mathbf{voronoiField}(s_{next})$
2: $c_{tssOrientation} \leftarrow \mathbf{tssOrientationPenalty}(s_{current}, s_{next})$
3: $d_{goal} \leftarrow \mathbf{getGridDistance}(s_{next}, s_{goal})$
4: **return** $cost + c_{tssOrientation} + d_{voronoi} + k_{dist} d_{goal}$

The heuristic of the Hybrid A* search implemented in this thesis and described in Algorithm 7 is inspired by the original but leaves out the *non-holonomic-no-obstacles* and makes some other additions to the search heuristic. One such addition is the utilization of the Voronoi field, where the Voronoi field strength $d_{voronoi}$ is added to the heuristic to penalize being unreasonably close to land. Additionally, the distance scaling factor k_{dist}[8] is introduced to enable tuning for increased search progression. Moreover, a cost penalty $c_{tssOrientation}$ for not following desired orientation in TSS lanes is added.

$\mathbf{getGridDistance}(s_{next}, s_{goal})$ calculates the distance of the graph-optimal path from $s_{current}$ to s_{goal} in the framed quadtree using A*. While significantly less expensive than the 3D iterative search, this pose-to-goal search query must be performed repeatedly and is typically the most computationally expensive part of the heuristic procedure. In order to improve efficiency, one should use a method to take advantage of earlier graph search queries in subsequent calls to the procedure. In this thesis, a novel method called sequence matching is in Subsection 4.3.4 proposed to improve efficiency. The adapted A* algorithm implementing this concept is described in Algorithm 8, with the $\mathbf{followingStoredPath}(s_{current})$ sequence matching procedure illustrated in Figure 4.4 and further detailed in Algorithm 9.

Observe in Algorithm 8 that with the exception of the code needed for sequence matching, this is an ordinary implementation of A*. Sequence matching requires that a search tree is built. This is facilitated by updating a lookup table where all directed edges in the search tree are stored. This is done using the $\mathbf{updateLookupTable}()$ procedure.

The sequence matching itself is described in Algorithm 9. Observe that when the search is at a vertex $s_{current}$ and calls this procedure, the procedure checks how much of the $cameFrom$ vertex sequence matches the established search tree. This is done backward iteratively from $s_{current}$ towards the initial configuration q_I. For every iteration, it is checked if the directed edge from $\mathbf{cameFrom}(s_{current})$ to $s_{current}$ is in the established search tree using $\mathbf{edgeInLookupTable}(\dots)$ and if it follows the tree in the direction towards the goal configuration using $\mathbf{correctPathDirection}(\dots)$. Any reader interested in technical details regarding sequence matching is advised to refer to the implementation source code [55].

A second modification of the Hybrid A* algorithm proposed in Subsection 4.3.4 is using adaptive simulation time in motion sampling based on distance to goal and distance to land. While there are several ways in which such functionality can be implemented, the implemented guidance system utilizes a simple function to calculate the simulation time

[8] The value for k_{dist} used i testing is given in Table C.2.

Algorithm 8 A* Search with sequence matching

function $searchAstar()$
Require: $s_{start} \neq nullptr$
Require: $s_{goal} \neq nullptr$
1: **addToFrontier**$(s_{start}, 0)$
2: **while** $\mathcal{F} \neq \emptyset$ **do**
3: $s_{current} \leftarrow$ **getFromFrontier**()
4: **if** $s_{current} == s_{goal}$ **then**
5: **updateLookupTable**()
6: **return reconstructPath**()
7: **end if**
8: **if followingStoredPath**$(s_{current})$ **then**
9: **updateLookupTable**()
10: **return reconstructPathFromLookup**()
11: **end if**
12: $neighbors \leftarrow$ **getConnectedNeighbors**$(s_{current})$
13: **for all** $s_{neighbor} \in neighbors$ **do**
14: $cost \leftarrow$ **costSoFar**$(s_{current})$ + **getEdgeWeight**$(s_{current}, s_{neighbor})$
15: **if costSoFar**$(s_{neighbor}) == NULL$ **or** $cost <$ **costSoFar**$(s_{neighbor})$ **then**
16: **setCostSoFar**$(s_{neighbor}, cost)$
17: $priority \leftarrow$ **costSoFar**$(s_{neighbor})$ + **heuristic**$(s_{neighbor}, s_{goal})$
18: **addToFrontier**$(s_{neighbor}, priority)$
19: **setCameFrom**$(s_{neighbor}, s_{current})$
20: **end if**
21: **end for**
22: **end while**

and achieve this behavior. This function is given in Equation (5.3) and utilizes a two-stage strategy.

$$t_{sim} = \min(\max(\frac{d_{land}}{v_{usv}}, t_{sim,u,min}), \max(k_{t,approach}\frac{d_g}{v_{usv}}, t_{sim,a,min})) \tag{5.3}$$

When far from the goal, simulation time is determined by distance to land, with a lower saturation $t_{sim,u,min}$ to ensure sufficient progress in narrow passages. As the USV approaches the goal, simulation time is eventually determined by the the distance to goal instead to ensure sufficient maneuverability to get to the goal without having to make additions to the iterative search tree far back. A lower saturation $t_{sim,a,min}$ is once more utilized. Moreover, a scaling factor $k_{t,approach}$ determines when the USV is sufficiently close to switch to the approach stage. Appropriate tuning ensures that simulation time never gets larger in the approach stage than in the underway stage of the search. The values used are given in Table C.2.

Algorithm 9 A* Sequence matching

function $followingStoredPath(s_{current})$

Require: $len_{match} \geq 1$

```
len_sequence ← 0
while len_sequence ≤ len_match do
    if !edgeInLookupTable(cameFrom(s_current), s_current) or
    !correctPathDirection(cameFrom(s_current), s_current) then
        return False
    end if
    s_current ← cameFrom(s_current)
    len_sequence ← len_sequence + 1
end while
return True
```

5.4 Collision avoidance system

The COLAV system[9] is implemented in ROS Noetic as a C++ node. While most all communication is message-based, information about the static environment is obtained through ownership of a ENC manager map service. The COLAV system is responsible for calculating course and speed offsets based on temporal ownship and obstacle information in order to comply with the relevant COLREGs and is built upon the concept of simulation-based control behavior selection originally proposed in [33]. The implementation by Hagen [35] has been used as a foundation. However, the structure, interface, integration method, and collision checking method are unique to this thesis.

5.4.1 Simulating and comparing control action combinations

Given a set $\boldsymbol{\chi}_{corr}$ of course offsets, a set of speed offset multipliers $\mathbf{u}_{corr}$ and a measured USV state x, the COLAV system is responsible for generating a list of trajectory and state predictions for the ownship. This is achieved by solving the equations of motion of the 3DOF surface vessel model defined in 3.4.1 using the 4^{th} order Runge Kutta method with an adaptive step size for a limited, pre-defined time horizon T. It is advised to ensure that T is kept sufficiently large, such that action to prevent collision is taken early. The simulation is facilitated by the Odeint library [23]. The main algorithm in the COLAV system is called $\mathbf{getBestControlOffset}()$ and the pseudocode for it is given in Algorithm 10.

5.4.2 Evaluating risk and COLREG Compliance

The centerpiece in Algorithm 10 is the cost function $\mathbf{costFunc}(\dots)$, which is an implementation based on [35] of the cost function proposed in [33]. Due to its importance for the nominal operating condition of the collision avoidance system, a thorough explanation of it is in order. The explanation below paraphrases from [33], with some additional explanations. While consistent with the notation introduced in the original paper, some

[9] Made available through the open-source *USV Guidance System* GitHub repository [55]

Algorithm 10 Function to determine best control offset for vessel to avoid collision

function $getBestControlOffset()$
Require: $states_{obstacle} \neq \emptyset$
Require: $state_{usv} \neq \emptyset$
Require: $setpoints_{usv} \neq \emptyset$
Require: $combinations_{offset} \neq \emptyset$
Require: $T \neq 0$
1: $horizons_{obstacle} \leftarrow \textbf{predictObstacleTrajectories}(states_{obstacle})$
2: **for all** $combination_{offset} \in combinations_{offset}$ **do**
3: $\quad horizon_{usv} \leftarrow \textbf{simulateHorizon}(state_{usv}, combination_{offset}, T)$
4: $\quad cost \leftarrow \textbf{costFunc}(horizon_{usv}, horizons_{obstacle}, combination_{offset})$
5: $\quad \textbf{storeControlOffsetCost}(combination_{offset}, cost)$
6: **end for**
7: **return** $\textbf{bestOffset}()$

further notation is also introduced for convenience. The chosen values for all parameters mentioned below are given in Table C.1.

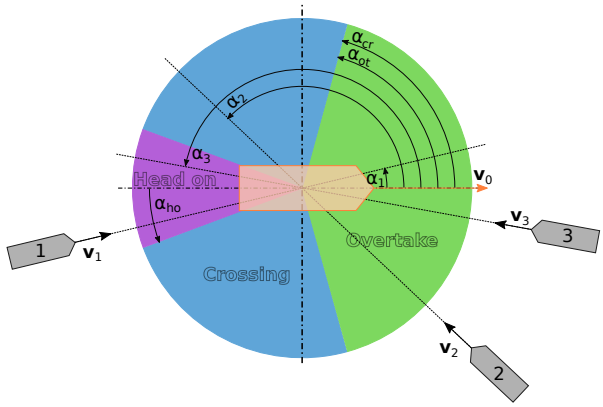
For an arbitrary scenario, a set of vehicle-obstacle states are defined by the cost function. These are described below, with supporting illustrations provided in Figure 5.5.

- **CLOSE**:
  If $d^k_{0,i}(t) \leq d^{cl}_i$, where $d^k_{0,i}(t)$ is predicted distance between USV and obstacle i at time t in scenario k, the USV is said to be close to obstacle i. COLREG compliance is only checked for obstacles i with this relation to the USV.

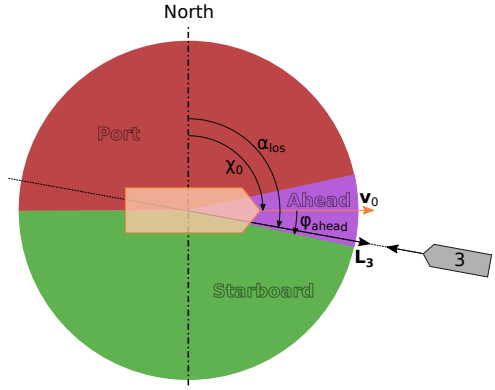
- **OVERTAKEN**:
  If the inequality
  $$\vec{v}^k_0(t) \cdot \vec{v}_i(t) > \cos(\alpha_{ot}) \left|\vec{v}^k_0(t)\right| |\vec{v}_i(t)| \tag{5.4}$$
  is satisfied, the USV is defined as overtaken by obstacle i at time t. Using $\phi_{ot} = 68.5°$ was suggested in the original paper [33]. Satisfying the inequality in Equation 5.4 is equivalent to the obstacle velocity vector translated to the USV origin pointing into the *Overtake* sector. Observe in Figure 5.5a that this is true for obstacle 1 and that ϕ_{ot} determines the size of the sector.

- **STARBOARD**:
  The USV has obstacle i on it's starboard side at time t if the bearing angle of $\vec{L}^k_i(t) > \chi_0$, where $\vec{L}^k_i(t)$ is a line-of-sight unit vector pointing from the USV to obstacle i and χ_0 is the course of the USV. This is visualized in Figure 5.5b, where the bearing angle of $\vec{L}^k_i(t)$ is denoted ϕ_{los}.

- **HEAD-ON**:
  If the inequalities
  $$\vec{v}^k_0(t) \cdot \vec{v}_i(t) < -\cos(\phi_{ho}) \left|\vec{v}^k_0(t)\right| |\vec{v}_i(t)| \tag{5.5}$$
  $$\vec{v}^k_0(t) \cdot \vec{L}^k_i(t) > \cos(\phi_{ahead}) \left|\vec{v}^k_0(t)\right| \tag{5.6}$$



(a) Sectors for evaluation of ownship-obstacle state relation by velocity. Note that α is used in place of ϕ in the angle symbols.



(b) USV course and relation to direction of LOS vector to obstacle ship.

Figure 5.5: Using line-of-sight vector and velocity vectors to evaluate discrete ownship-obstacle relationship states. The concept is used in earlier research efforts [33], [35], [41] but the illustrations are made for the concept explanation in this thesis

are satisfied, obstacle i is head on relative to the USV at time t. Observe that vessel 3 in Figure 5.5a satisfies the first inequality and note that ϕ_{ho} controls the size of the *Head-on* sector. $\phi_{ho} = 22.5°$ was proposed originally [33]. The evaluation of the second inequality is visualized in Figure 5.5b. Because $\vec{L}_3$ is within the *Ahead* sector, the second inequality is also satisfied and thus the USV is in a *HEAD-ON* state with obstacle 3. The size of the *Ahead* sector is controlled by ϕ_{ahead}, with $\phi_{ahead} = 15.0°$ found to work well in [35].

- **Crossed**:
  If the inequality
  $$\vec{v}_0^k(t) \cdot \vec{v}_i(t) < \cos(\phi_{cr}) \left|\vec{v}_0^k(t)\right| |\vec{v}_i(t)| \tag{5.7}$$
  is satisfied, obstacle i is crossed by the USV at time t. This is equivalent to the

velocity vector of the obstacle translated to the origin of the USV pointing in the *crossing* sector, visualized in Figure 5.5a. Similar to the head-on and overtake sectors, the crossed sector size is determined by an angle. $\phi_{cr} = 68.5°$ was suggested in [33]. Note that the *crossed* sector overlaps with the *head-on sector*. Sectors can overlap, and this is accounted for in the utilization of these binary states in the cost function.

With these states defined, assembling the cost function can commence. In the original paper [33], the cost function is described as *associated hazard*. The associated hazard for scenario k at time t_0 given by Equation (5.8)

$$\mathcal{H}^k(t_0) = \max_i \max_{t \in D(t_0)} (\mathcal{C}_i^k(t)\mathcal{R}_i^k(t) + \kappa_i \mu_i^k(t)) + f(P^k, \chi_{ca}^k) + g(P^k, \chi_{ca}^k)) \tag{5.8}$$

The discrete time steps of the simulated prediction horizon is given by $D(t_0) = \{t_0, t_0 + T_s, \ldots, t_0 + T\}$, where T_s is the timestep and T is the prediction horizon end time. The functions making up the associated hazard require further explanation, starting with the *collision risk factor* $\mathcal{R}_i^k(t)$ defined in Equation (5.9)

$$\mathcal{R}_i^k(t) = \begin{cases} \frac{1}{|t-t_0|^p} \left(\frac{d_i^{safe}}{d_{0,i}^k(t)} \right)^q & d_{0,i}^k(t) \leq d_i^{safe} \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

where d_i^{safe} defines the maximum distance at which the risk factor is evaluated relative to an obstacle i and is allowed to be variable for different obstacle vessels based on their characteristics and tracking uncertainty. Care must be taken to make d_i^{safe} sufficiently large to ensure compliance with COLREG Rule 16 as described in Appendix A. Moreover, note that the exponent $q \geq 1$ also is important to ensure sufficiently early action is taken. The last variable to highlight in the collision risk factor function is the exponent p, which from Equation (5.9) describes how risk is weighted as a function of time to collision.

In Equation (5.8), the collision risk factor is scaled by a *collision cost* $\mathcal{C}_i^k(t)$ based on the kinetic energy of a potential impact between the USV and obstacle i as given in Equation (5.10). This scaling is is intended as a tool for damage mitigation in an event where a collision is inevitable, and allows flexibility by introducing the obstacle dependent factor K_i^{col}.

$$\mathcal{C}_i^k(t) = K_i^{col} \left| \vec{v}_0^k(t) - \vec{v}_i^k(t) \right|^2 \tag{5.10}$$

Having established how collision cost is handled, what remains is quantifying the cost of violating the COLREGs described in Appendix A. To achieve this, a *binary violation indicator* $\mu_i^k(t)$ can be utilized, following the logic given in 5.13. Observe that the aforementioned binary USV-obstacle relationship states are utilized to enable this logic.

$$\mu_i^k(t) = \text{RULE14 or RULE15} \tag{5.11}$$
$$\text{RULE14} = \text{CLOSE \& STARBOARD \& HEAD-ON} \tag{5.12}$$
$$\text{RULE15} = \text{CLOSE \& STARBOARD \& CROSSED \& !OVERTAKEN} \tag{5.13}$$

The function $f(P^k, \chi_{ca}^k)$ is as seen in Equation (5.14) a penalty function. The purpose of introducing this function is primarily to do tie-breaking. When similar costly control offset combinations exist, this function ensures that the nominal path is followed if it is one of the candidates. Moreover, it ensures that path changes are not made frequently by penalizing any change in correction using penalty functions δ_P and δ_χ. The tuning parameters k_P and k_χ influence the priority of sticking with the nominal path. An additional benefit of $f(\cdot)$ is that it in general results in the COLAV being conservative with course and speed changes, assuming appropriate tuning.

$$f(P^k, \chi_{ca}^k) = k_P(1 - P) + k_\chi \chi_{ca}^2 + \delta_P(P - P_{last}) + \delta_\chi(\chi_{ca} - \chi_{ca,last}) \tag{5.14}$$

To complete the cost function $\mathcal{H}^k(t_0)$, what remains is to define the *grounding penalty function* $g(\cdot)$. This function uses the intersection functionality in the ENC manager map service. The grounding cost is altered in the implementation to only be evaluated after the rest of the cost function for the minimum amount of candidates. Furthermore, the cost is scaled with an estimate of time until impact. This is done to ensure that even in the scenario where all control candidates result in collision or grounding, the option resulting in the lowest associated hazard is chosen.

Chapter 6

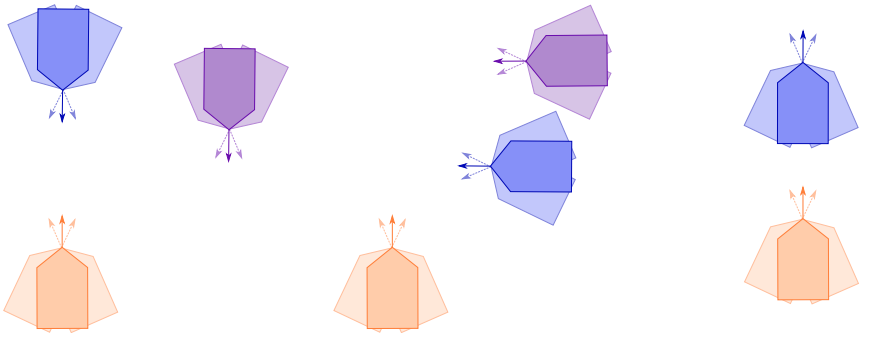
Results and discussion

A divide and conquer strategy is utilized to test the proposed guidance system's viability. First, the collision avoidance capabilities concerning safe distance and COLREG compliance are tested for isolated collision scenarios. Thereafter, the mission planner is tested for compliance with TSS objects, path quality, and search efficiency at scale with the intent to verify the capabilities and identify limitations of the proposed mission planner design. Lastly, the entire system is tested together in a hypothetical full-scale mission scenario.

Critical parameters for the simulation-based testing of the proposed system are given in Appendix C. Model parameters are obtained from work done by Loe [22] and COLAV parameters are based on the tuning efforts by Hagen [35]. All tuning parameters have been found empirically and are generally chosen to balance efficiency, collision avoidance performance, and path quality. The reader is advised that the *USV Guidance System* implementation version used during testing is tagged as *v1.0.0* in the GitHub repository [55] for the convenience of any reader interested in recreating the results from the following system evaluation.

6.1 Isolated collision avoidance scenarios

Head-on, crossing, and overtake collision avoidance scenarios are considered most relevant for COLREG compliance and are thus selected for evaluation. In order to mitigate possible interference from any undiscovered implementation errors, collision avoidance is tested in isolated scenarios. Furthermore, to quantify robustness in identifying and avoiding collisions, the isolated collision avoidance test scenarios are run for 100 Monte Carlo Iterations. Both the initial pose of the USV ownship and the obstacle vessels are subject to Gaussian multivariate noise to ensure sufficient variation across the iterations. For each vessel, the initial configuration is given by Equation 6.1



(a) Head-on scenario **(b)** Crossing scenario **(c)** Overtake scenario

Figure 6.1: Isolated collision avoidance testing scenarios

$$q_I(i) = \begin{bmatrix} x_i \\ y_i \\ \psi_i \end{bmatrix} \tag{6.1}$$

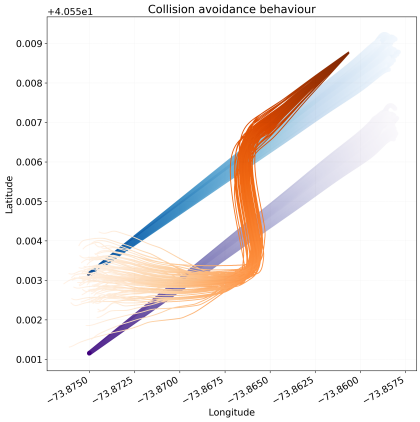
where the ownship has id $i = 0$ and $q_I(i) \sim \mathcal{N}(\boldsymbol{\mu}(i), \boldsymbol{\Sigma}(i))$ and the goal configuration is constant and given by $q_G(i)$. The matrices $\Sigma(0)$ and $\Sigma(1,2)$ given in Equation 6.2 were used as covariance matrices for the USV and obstacles, respectively.

$$\Sigma(0) = \begin{bmatrix} 5e^{-4} & 0 & 0 \\ 0 & 5e^{-4} & 0 \\ 0 & 0 & 0.3925 \end{bmatrix} \quad \Sigma(1,2) = \begin{bmatrix} 2e^{-4} & 0 & 0 \\ 0 & 2e^{-4} & 0 \\ 0 & 0 & 0.3925 \end{bmatrix} \tag{6.2}$$

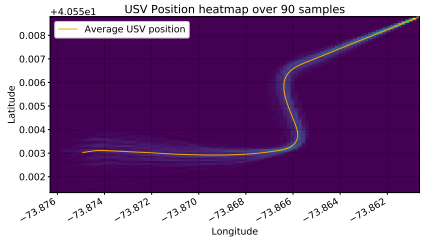
The obstacle vessels are set as non-cooperative to approximate the worst case collision avoidance scenarios where only the action taken by the USV can ensure that collision is avoided. Collision avoidance behavior in each iteration will be categorized as compliant if all relevant rules given in Appendix A are followed.

6.1.1 Head-on scenario

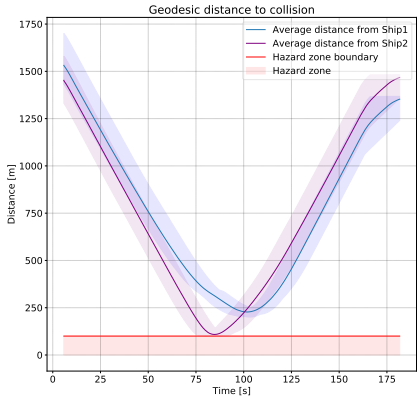
The head-on scenario is set up according to the definition of such a scenario given in Rule 14 of the COLREGs, described in Appendix A, but with two obstacle vessels instead of just one to challenge the robustness of the collision avoidance system. The USV and one non-cooperative vessel are on nearly reciprocal courses as illustrated in Figure 6.1a. In order to comply with Rule 14, the USV must alter course starboard such that the vessels pass on the port side of each other. Compliance with Rule 8 must also be satisfied, thus the course alterations must be large and taken sufficiently early such that the intent of the ownship is clearly stated. To increase scenario complexity, a second obstacle vessel is on a parallel course with the first, and the USV must therefore alter course further to avoid both vessels in rules compliant fashion.



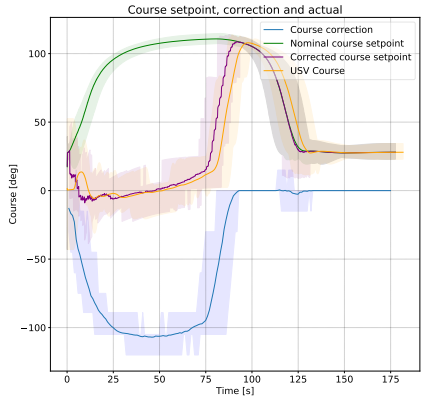
(a) Collision avoidance behaviour



(b) Position heatmap



(c) Geodetic distance to collision



(d) Course values internally and externally

Figure 6.2: Results from Head-on Monte Carlo simulations exhibiting compliant behavior.

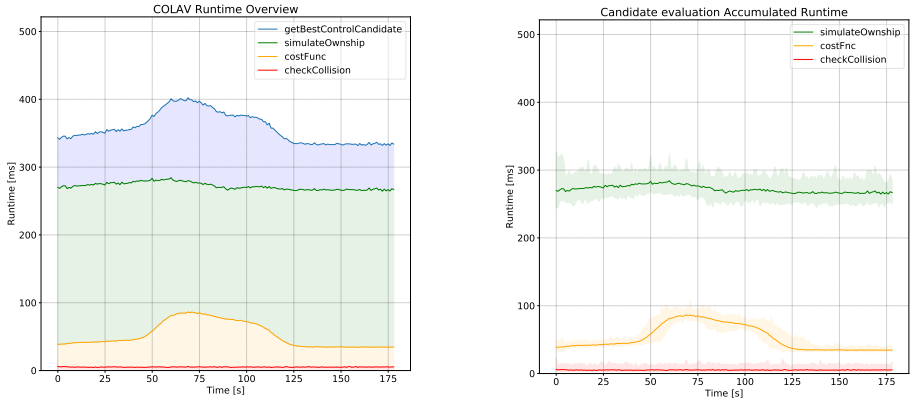
Running this scenario for 100 Monte Carlo Iterations, 90 iterations show the USV acting in compliance with the COLREGs, and 10 iterations include some non-compliant aspects in the maneuver. The compliant iterations will be evaluated first, with the results visualized in Figure 6.2 and Figure 6.3.

Observe in Figure 6.4a that while there is significant variation in the starting position of the ownship and the obstacle vessels, the USV turn to starboard and passes both obstacles port side, ensuring compliance with COLREG Rule 14. While some $q_I(0)$ result in the ownship being on nearly reciprocal courses with at least one obstacle vessel, other initial configurations resulting in more ambiguous situations are also correctly identified as head-on collision scenarios. This behavior is desirable, as Rule14(c) in Appendix A state that one shall assume that the situation exists if one is in any doubt. It thus indicates that

the collision avoidance system with the chosen cost function and tune is conservative, consistent, and robust when identifying COLREG situations.

Collision avoidance can be considered as a two-stage process, with a *deviate* and a *return* stage. In the first stage, the ownship deviates from the nominal path to avoid collision with one or multiple vessels. In order to comply with Rule 8 of the COLREGs, the deviation must start sufficiently early such that the passing happens at a safe distance, and course alterations must be sufficiently large to clearly state the intent of the USV to the obstacle vessel. While the ownship always starts more than $1400m$ from the nearest obstacle vessel, it is clear from Figure 6.2d that the collision avoidance system immediately takes action and alters the course starboard. Furthermore, as can be seen in Figure 6.2c, with the safe distance lower limit defined at $100m$, the USV does not enter this hazard zone. However, please observe that the ownship gets as close as possible to the hazard zone of vessel two, indicating that the collision system with its current tune strikes an acceptable compromise between path optimality and collision avoidance capability. Regarding the course correction, a shortcoming of the implemented system becomes apparent. Observe in Figure 6.2d that the course correction gradually increases towards 90° starboard on average, while it ideally should request the largest necessary correction immediately. Before evaluating how this affects the COLREG compliance, it is considered worthwhile to discuss why this phenomenon occurs. The COLAV system predicts the trajectory of the ownship and obstacle vessels for a predefined amount of time. While the prediction of the ownship considers the vessel dynamics, it does not expect the LOS nominal course setpoint to change. This assumption is not valid for course corrections except for the trivial zero correction option. When the vessel is instructed by the COLAV system to deviate from the nominal path, the LOS path tracker changes the nominal course to try to get the ownship back to this path. Thus, to maintain the necessary course correction starboard to avoid a collision, the absolute value of the demanded course correction gradually increases. Because the collision avoidance system runs at only 1Hz, this internal struggle can be observed in the USV odometry as course oscillations until an equilibrium is found, as can be seen in Figure 6.2d. Thereafter, the course gradually is altered back to port until both vessels are clear, at which point the USV initiates the *return* stage with a large course correction to port. While not ideal with regards to clearly stating intent, looking at both the position of the vessel in Figure 6.2a and Figure 6.2b it does appear to sufficiently clearly show that the USV is taking action to avoid the incoming vessels. This is supported by the course of the vessel shown in Figure 6.2d. Thus, it can be argued that while it is not perfect, the collision avoidance behavior does state intent adequately to be compliant with Rule 8 of the COLREGs.

Before looking at the non-compliant Monte Carlo iterations, the runtime of the collision avoidance system is analyzed to evaluate real-time performance. In Figure 6.3a, the runtime of the three main functions that make up the control candidate selection process is visualized together with the procedure itself. Observe that the control candidate selection takes between $\sim 350ms$ and $\sim 400ms$, with most of the time spent predicting the ownship trajectory through simulation. The computational cost of the simulation depends on the vessel model, the integration method, and the simulation duration. With a collision-avoidance system running at $1Hz$ in mind, the chosen model, integrator, and simulation

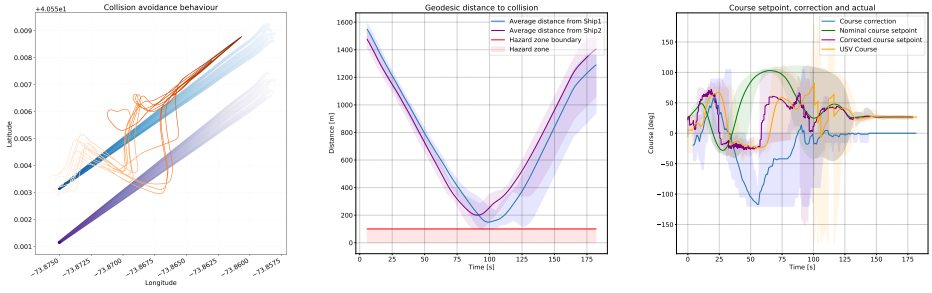


(a) Runtime overview of key procedures

(b) Runtime averages and ranges for key procedures

Figure 6.3: Runtime data retrieved from all 100 Head-on Monte Carlo iterations.

time of $300s$ appear to strike an acceptable balance between complexity and efficiency. The computational complexity of the cost function is fascinating, as a significant increase in cost can be seen between $\sim 50s$ and $\sim 100s$ into the simulation iterations. Comparing with the distance to obstacle visualized in Figure 6.2c, observe that this is on average when at least one obstacle is within the distance of the USV where the COLREGs are considered. COLREG compliance is evaluated for every timestep of the prediction horizon. When the ownship is closer to the obstacles, a more significant portion of the prediction horizon will be evaluated on average. It should be noted that the cost function evaluation will scale with both the number of obstacles and steps in the prediction horizon of the ownship. For the isolated collision avoidance scenarios, collision checking only makes up a marginal part of the overall computational cost. This is by design, as collision checking only checks the minimum number of candidate trajectories. When there is no collision, it thus only checks the best candidate.



(a) Collision avoidance behaviour

(b) Geodesic distance to collision

(c) Course values

Figure 6.4: Results from Head-on Monte Carlo iterations with non-compliant behaviour observed

Finally, the non-compliant Monte Carlo iterations should be investigated for the head-on scenario. Using Figure 6.4 as a reference, observe that most all iterations start with the USV initially turning port instead of the expected turn starboard. Later, the USV turn to starboard and ends up passing either between the vessels or with both to its port side. While Rule 14 is satisfied for the latter cases, Rule 8 of the COLREGs is not due to an ambiguous statement of intent. In an attempt to identify any pattern in the initial pose of the USV making it more likely for the non-compliant behavior to occur, the initial pose of each iteration is visualized in Figure 6.5. Green arrows indicate initial poses resulting in compliant behavior, and red arrows initial poses resulting in non-compliant behavior. From the samples in this collection, no such pattern was found.

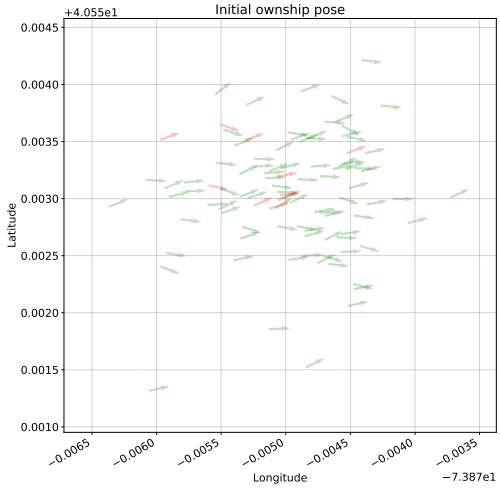
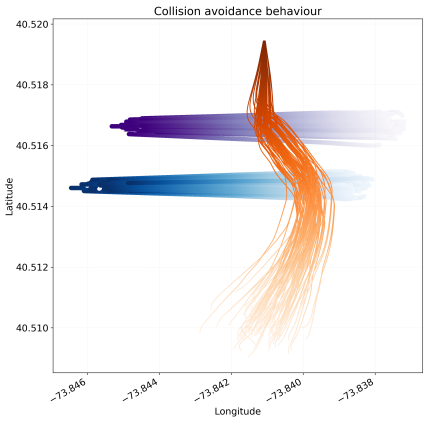
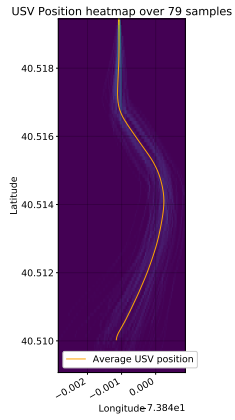


Figure 6.5: Initial pose of ownship in Monte Carlo Iterations

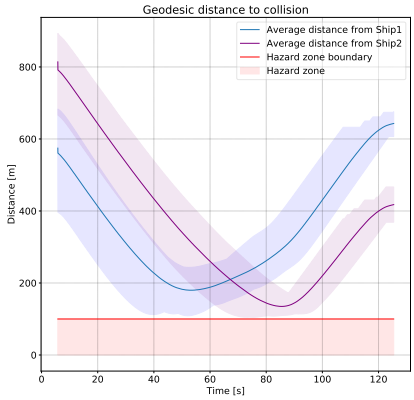
The non-compliant iterations highlight a weakness of the collision avoidance system. Looking at the method utilized to detect COLREG violation in Subsection 5.4.2, whenever the trajectory the USV is following is violating Rule 14, it can get out of the situation either by taking action such that the *STARBOARD* or the *HEAD-ON* ownship-obstacle relation state becomes false. There can be maneuvers port that result in breaking the latter. Thus, from the perspective of the collision avoidance system, these maneuvers are legitimate and rule compliant. However, in practice, they clearly are not compliant. Further tuning of the COLAV cost function could potentially make this type of behavior less commonplace, but note that significant tuning efforts were needed to get to the current compliance ratio. Furthermore, it is worth noting that the USV never enter inside the hazard zone of any obstacle vessel, as can be seen in Figure 6.4b. Thus, safety margins are ensured for all Monte Carlo Iterations, which inspires confidence in the overall safety of the collision avoidance system.



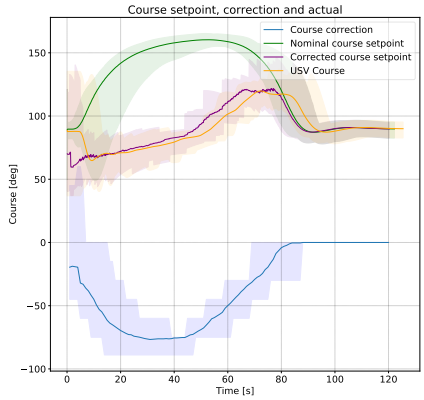
(a) Collision avoidance behaviour



(b) Position heatmap



(c) Geodetic distance to collision



(d) Course values internally and externally.

Figure 6.6: Results from Crossing Monte Carlo simulations exhibiting compliant behavior.

6.1.2 Crossing scenario

In the crossing scenario, illustrated in Figure 6.1b, the USV has two non-cooperative obstacle vessels at it's starboard side. Rule 15 of the COLREGs dictate that the USV must then stay well clear. There is sufficient clearance to pass behind the obstacle vessels in the isolated collision avoidance scenario and passing behind the obstacles will therefore be the compliant maneuver. With regards to the initial pose of the ownship and obstacles, the covariance matrices given in Equation 6.2 are used.

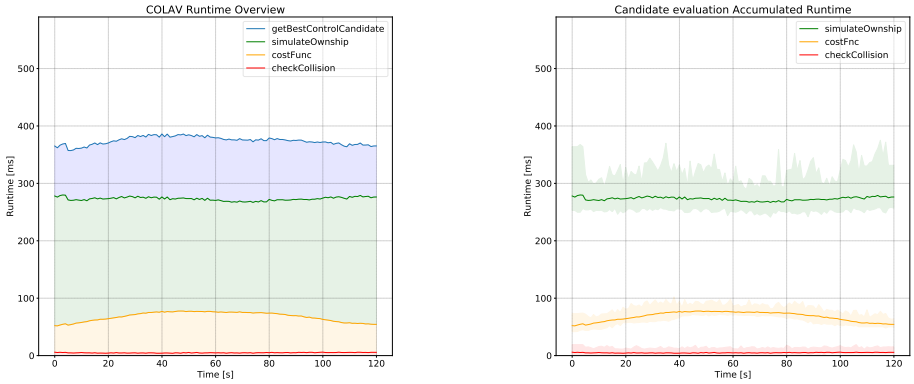
Like the head-on scenario, the crossing scenario has been run for 100 Monte Carlo Iterations. The compliance ratio was worse than in the head-on scenario, with 79 compliant and 21 non-compliant iterations. The compliant iterations are evaluated first, with the behavior

illustrated in Figure 6.6.

Observe from Figure 6.6d that the collision avoidance system immediately instructs the USV to turn starboard and that the same gradual correction increase that was observed in the head-on scenario happens, supporting the claim that it is caused by the LOS nominal course setpoint gradually changing as the vessel deviates further from the nominal path. Robustness in the identification of a COLREG situation is once more sufficiently accurate. Some initial configurations result in a direct collision if no action is taken and thus can easily be recognized as COLREG situations. Others would not result in a direct collision but instead result in the USV passing in front of or dangerously close behind the obstacles and are still detected correctly. This highlights a strength of the collision avoidance system. Instead of just relying on relative position or time to collision, the COLAV system uses the velocity vectors of both vessels to detect the *CROSSED* ownship-obstacle state. This strategy appears to be more robust when human intuition cannot be used to interpret the situation.

Compared to the head-on scenario, the USV starts much closer to the obstacles in the crossing scenario, as can be seen by comparing Figure 6.2c and Figure 6.6c. This was done to evaluate to what extent, if any, the COLAV system requests unnecessarily large course corrections. This is important because it is assumed that any deviation from the nominal path will be less optimal for the USV overall. Thus, making small corrections and staying as close as possible to the nominal path will be the optimal collision avoidance strategy. Using Figure 6.6d, observe that even though the obstacle vessels are much closer, an initial correction of $\approx -15°$is still requested on average. Moreover, the USV can stay closer to the nominal path because the obstacles are effectively moving out of the way with time. The course correction observation indicates that the implemented collision avoidance behavior approximates the optimal, and the use of the distance-to-collision metric can quantify the accuracy of the approximation. The ownship should, in this context, travel as close as possible to the obstacles. From Figure 6.6c, observe that while the minimum is close to the hazard zone for both obstacles, the average is not. Thus, while the collision avoidance maneuver approximates the optimal, there exists an opportunity for improvement. Some of the improvement can be made up for with additional tuning of the cost function weights. However, an attempt to achieve this resulted in more oscillation and a lower COLREG compliance ratio in Monte Carlo Simulations. The current tune thus offers an acceptable compromise between stability and maneuver optimality. Please remark that the same collision avoidance tune is used for all isolated testing scenarios.

Due to the added Gaussian noise in the initial pose of the ownship, the *deviate* stage of the maneuver naturally has much spread, as can be seen in Figure 6.6a. Using the position heatmap in Figure 6.6b however, there is a clear convergence tendency towards the average with time, indicating that the collision avoidance behavior has deterministic properties. However, remark that for the last part of the *return* stage, convergence is artificially high. This pattern is a bi-product of the testing setup, as the nominal path for each sample is the direct line from the initial position to the goal position. While the initial position is subject to noise, the goal is not, and thus all the nominal path lines focus at the same point, resulting in an added convergence effect in the heatmap.



(a) Runtime overview of key procedures

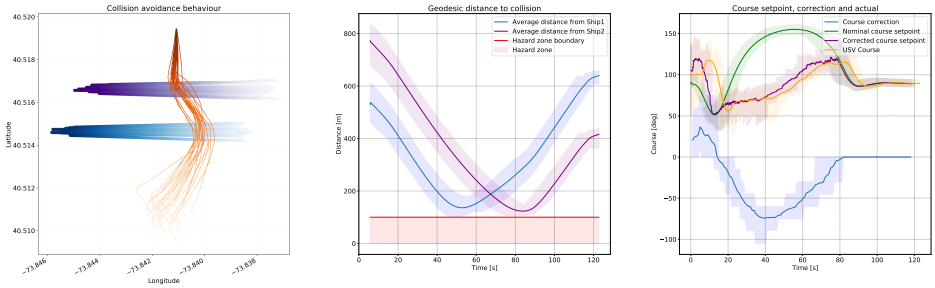
(b) Runtime averages for key procedures

Figure 6.7: Runtime data retrieved from all 100 Monte Carlo iterations of the crossing scenario.

Following the same evaluation process as in the head-on scenario, the runtime is next to be discussed. Runtime for all 100 Monte Carlo Iterations is shown in Figure 6.7. Observe that while the computational cost of simulating the ownship trajectory and evaluating collision is approximately the same as for the head-on scenario, cost function evaluation is higher. Furthermore, observe that for the first half of the collision avoidance behavior, cost function evaluation runtime gradually increases towards a maximum before decreasing in the second half. Comparing the distance to the obstacle in Figure 6.6c and the avoidance behavior in Figure 6.6a, it becomes clear that this change in computational cost likely is due to more prediction trajectory steps falling within the circle of consideration in the first half and naturally less in the second half when the more distant parts of the prediction horizon will be far from the obstacle vessels. Note also that while the collision avoidance scenario is entirely different from the head-on scenario, runtime is approximately the same for getting the best control action overall, as can be seen by comparing Figure 6.7a with Figure 6.3a. This highlights that the collision avoidance system appears invariant to the type of collision avoidance scenario from the perspective of computational cost, which can be considered to be a significant advantage.

The last thing to evaluate for the crossing scenario is what happens during the non-compliant iterations. Most all non-compliant iterations start with the USV turning port and, after some time, turning starboard, resulting in passing behind both vessels and thus complying with COLREG Rule 15 as described in Appendix A. The behavior is shown in Figure 6.8. Once more, it is Rule 8 that the initial part of the maneuver has violated. The initial pose evaluation supported by Figure 6.9 indicates that if the USV start with an initial course significantly port of the course needed to follow the nominal path, it is more likely to be non-compliant.

The sample size of non-compliant iterations is far too small to draw any conclusion. However, looking at how violation of Rule 15 is detected, it appears that for the non-compliant



(a) Collision avoidance behaviour **(b)** Geodesic distance to collision **(c)** Course values

Figure 6.8: Results from Crossing Monte Carlo iterations with non-compliant behavior observed

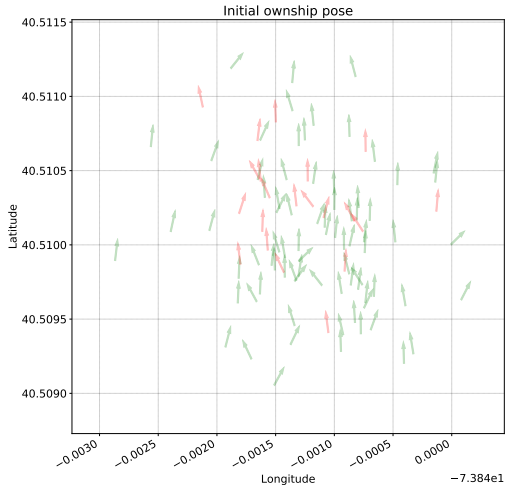
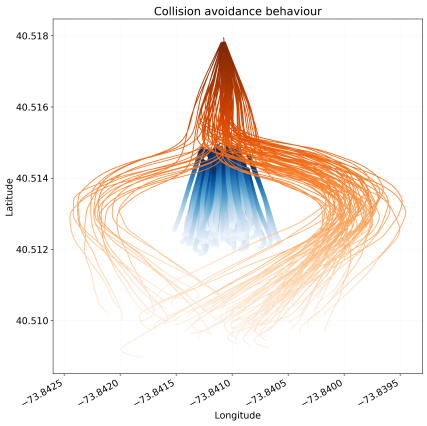


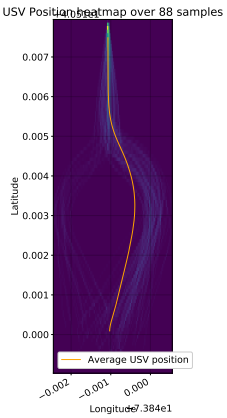
Figure 6.9: Initial pose of ownship in Monte Carlo Iterations

maneuvers, the vessel takes action to get out of the *CROSSING* state described in Subsection 5.4.2. Significant port course corrections are penalized more than similar starboard corrections. Thus, when the course correction has to increase to fight the path tracker, it eventually becomes prohibitively expensive, and the USV therefore turn starboard to get out of the *STARBOARD* state instead. It is becoming more evident that this non-compliant behavior is a deficiency of the collision avoidance system that affects several different collision scenarios for essentially the same reason. Once more, tuning efforts were conducted to mitigate the problem, but balancing robustness and maneuver optimality across all scenarios resulted in no better tune that worked reliably across all collision avoidance scenarios.

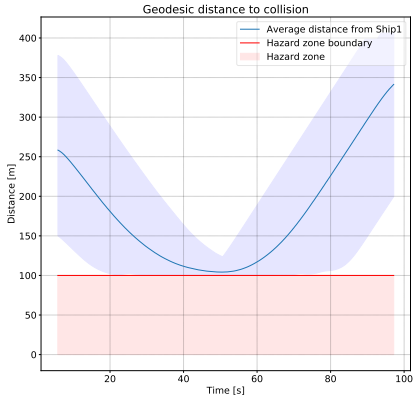
6.1.3 Overtake scenario



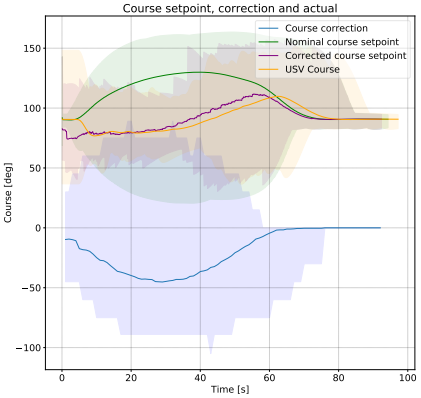
(a) Collision avoidance behaviour



(b) Position heatmap



(c) Geodetic distance to collision



(d) Course values internally and externally

Figure 6.10: Results from Overtake Monte Carlo simulations exhibiting compliant behavior.

In the overtake scenario, illustrated in Figure 6.1c, the USV approaches aft of one slow obstacle ship. In order to comply with COLREG Rule 13, the USV must maneuver to avoid collision and stay clear of the obstacle ship being overtaken as described in Appendix A. The rules do not specify at which side of the obstacle vessel the USV should overtake it. Furthermore, what makes this scenario extra interesting is that while the control candidate cost function does have a *OVERTAKE* state, it does not use this to detect violation of COLREG Rule 13. Violation of COLREG Rule 13 is not evaluated explicitly in the cost function. Any compliant behavior is thus the result of the COLAV system as a whole. The same covariance matrices as before are used for the initial pose of the USV and obstacle vessel.

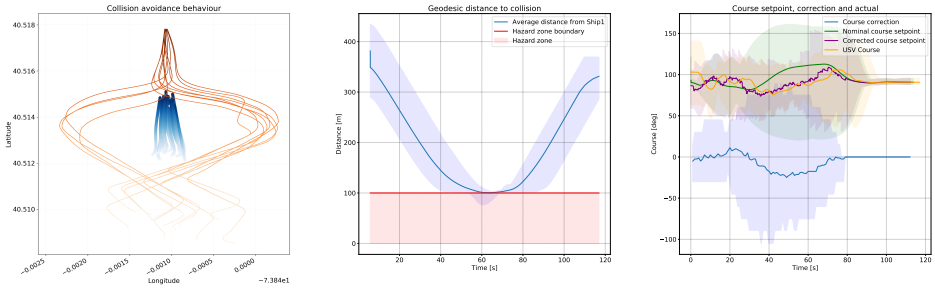
As for the two previous scenarios, overtake has been run for 100 Monte Carlo Iterations. However, to save time during testing, the vessel starts quite close to the obstacle and has its goal not too far ahead of it. The compliance ratio was better than in the crossing scenario but worse than in the head-on scenario, with 88 compliant iterations and 12 non-compliant. The reader is advised that the runtime was almost identical to the previous scenarios and will thus not be evaluated further in this collision scenario discussion.

Using Figure 6.10a and Figure 6.10d, it is clear that the COLAV system immediately takes action to avoid collision. On average, the system demands course correction starboard, passing the obstacle on the port side of the USV. As it is not specified on which side the overtake should happen, the iterations where the COLAV system results in passing the obstacle on the starboard side are also considered compliant. Looking at Figure 6.10a, it appears that, on average, the COLAV system prefers to keep the obstacle vessel on the same side as it is initially. This is expected, as the nominal path then, on average, will pass slightly to that side of the obstacle vessel. Thus, to deviate only the minimum required amount from the nominal path, one wants to stick to that same side of the obstacle if possible. This behavior is likely to be a result of two factors; the initial distance from the obstacle vessel and the tune of the COLAV system. It is desirable because it results in the collision avoidance system approximating the true optimal maneuver with acceptable accuracy. The distance to collision metric in Figure 6.10c supports this, as it shows that the USV gets very close to the hazard zone not only on average but also in the min-max range.

While the *deviate* stage of the maneuver is essential to avoid collision and comply with Rule 8 of the COLREGs, compliance with Rule 13 is determined mainly by the *return* stage. To be compliant and also provide an acceptable approximation of the optimal behavior, the USV must return to the nominal path as soon as possible without being considered in the way of the obstacle vessel. Using the position behavior plot in Figure 6.10a and the course data visualized in Figure 6.10d, it is clear that the USV makes a smooth maneuver, gradually returning to the nominal path. Thus, while difficult to quantify, it does appear to satisfy Rule 13 if one considers that the return maneuver is smooth and that the distance to the obstacle continues to increase during the *return* stage. Furthermore, using the position heatmap in Figure 6.10b, the maneuver appears to have decent deterministic properties, with the USV typically returning to the same spot on the nominal path no matter from where it initially began.

All non-compliant iterations violate Rule 8 of the COLREGs by first altering course port or starboard and then changing to starboard or port, as can be seen in Figure 6.11a and the course in Figure 6.11c. One can argue that Rule 13 is not violated for all but one and that the intent of the USV is less relevant in an overtake scenario. However, it does create uncertainty for the obstacle vessel, and if the obstacle vessel should choose to act on the uncertainty by doing any maneuver, safety could be compromised.

Remark in the distance to the collision plot that there is, in fact, one iteration where the obstacle vessel goes into the hazard zone. One can argue that the safety of the ownship or obstacle was never compromised because the cost of traveling further into the hazard zone grows exponentially, resulting in it being prohibitively expensive for the USV to hit



(a) Collision avoidance behaviour **(b)** Geodesic distance to collision **(c)** Course values

Figure 6.11: Results from overtake Monte Carlo iterations with non-compliant behavior observed.

the obstacle. However, it could result in the obstacle vessel doing an avoidance maneuver. Thus, it violates Rule 13, is not acceptable under any circumstance, and can be considered the first and only instance of a critical failure in isolated collision avoidance testing.

6.2 Mission Planning Scenarios

The Mission Planner is responsible for finding a feasible, safe and optimized path from an initial configuration to some goal position efficiently. In order to evaluate to what extent the mission planner fulfills its responsibility, it must be extensively tested across different mission regions with varying coastal geometry. Additionally, in order to evaluate the safety of the found path properly, compliance with core TSS elements must be considered. In an effort to quantify the overall efficiency and robustness of the Mission Planner, testing across several different environments is performed with metrics such as moving average search progression evaluated. Runtime of key procedures in the mission planning search are also highlighted in an attempt to give a complete picture of planner efficiency.

6.2.1 Traffic Separation Scheme Compliance

A core goal of the proposed guidance system is to facilitate the safety of navigation. In order to achieve this, it is of crucial importance that the Mission Planner makes use of TSS objects in order to comply with Rule 10 of the COLREGs, described in Appendix A. To evaluate compliance with TSS elements, the Mission Planner is tested in a mission area on the west coast of Norway where both Lanes, Separation Zones and Roundabouts are represented. In order to get an undistorted evaluation of how the TSS elements affect search progression, the adaptive simulation time feature of the mission planner is disabled and replaced by a fixed simulation time of $t = 30s$.

The mission starts with an initial configuration q_I and a goal configuration q_G as given in

Equation 6.3

$$q_I = \begin{bmatrix} 5.345468 \\ 59.021870 \\ 3.14 \end{bmatrix} \quad q_G = \begin{bmatrix} 5.345468 \\ 59.021870 \\ 3.14 \end{bmatrix} \tag{6.3}$$

to ensure interaction with both lanes, separation zones and roundabouts. Please note that with the current implementation, the goal orientation is not considered.

Observe from Figure 6.12 that the selected initial configuration q_I results in the search approaching a lane at almost $90°$. While the Mission Planner does allow crossing into the TSS lane at such a sharp angle, observe that it immediately after starts following the required course in the lane, taking advantage of the course suggester and the maneuverability of the USV model. Furthermore, note that the steepest angle at which the USV can enter a lane is a tuneable parameter in the mission planner, thus it could be set to a smaller angle if one was to utilize the planner on a larger vessel with lower maneuverability.

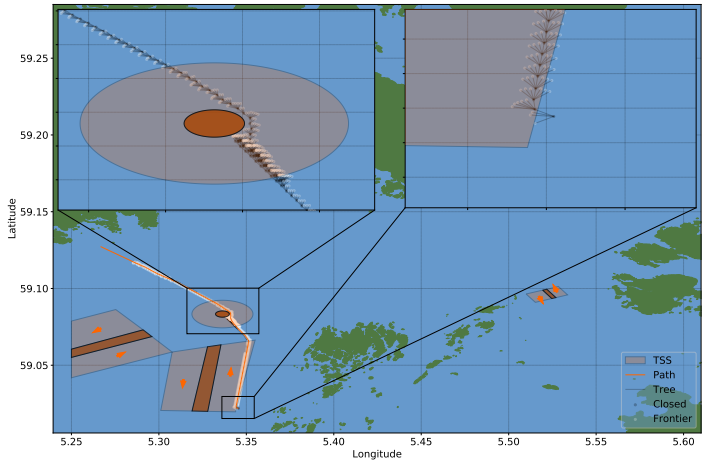


Figure 6.12: Search overview for TSS Mission with Points of Interest (POIs) highlighted.

In the upper left highlight in Figure 6.12, observe how the mission planner handles an upcoming roundabout. The planner indeed exhibits compliant behavior by going around the roundabout counterclockwise. However, the chosen path is not ideal as any other vessel in the roundabout that has a larger turning radius could be crossed by a USV following this path. This deficiency is an unfortunate consequence of the TSS compliance design, where only a hard constraint enforces compliance with roundabout objects. Where an appropriate soft constraint can encourage the desired behavior, utilizing only a hard constraint results in the planner being uninformed about how well it interacts with the roundabout. Thus, any option satisfying the hard constraint while resulting in a minimal increase to the *holonomic-with-obstacles* heuristic will be selected eventually. While one could argue that using only a hard constraint for roundabout compliance is insufficient for acceptable path safety characteristics, adding a soft constraint introduces several new challenges both in design, implementation, and tuning. Based on the goals set for TSS compliance with

regards to roundabouts in the Mission Planner design, the exerted behavior in proximity of roundabouts, while not ideal, complies sufficiently with Rule 18 of the COLREGs and is as expected from the design.

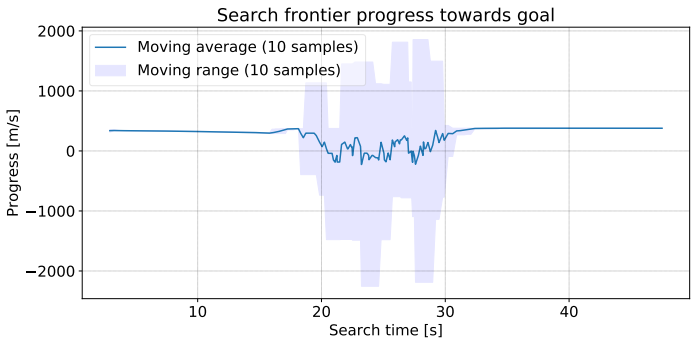


Figure 6.13: Moving average and range of search progression in TSS Mission

In the presence of design deficiencies yielding suboptimal results, it is important not to lose sight of the goal; to ensure compliance without severely reducing search progression and efficiency. The change in remaining distance to the goal position is differentiated numerically to quantify search progression. Observe in Figure 6.13 that moving average search progression is affected by the presence of the roundabout from $\approx 20s$ into the search until $\approx 30s$. However, reduction in search efficiency in the presence of obstacles is an inherent part of Hybrid A* search optimization, and thus some reduction is to be expected. Because the mission planner is not informed about how to traverse the roundabout, similarly to when meeting a hazardous obstacle, it will thus have to go back and evaluate other options. Unlike passing around obstacles, search progression takes a larger hit in the presence of the roundabout due to a conflict of interest between TSS compliance and the distance-to-goal heuristic which focuses on path optimality. The former prohibits the search from going clockwise around the roundabout while the latter would prefer a clockwise rotation to get the path as short as possible. In light of this ongoing tug-of-war-like situation between TSS compliance and path optimality, passing through the roundabout in a sufficiently compliant manner in $\approx 10s$ does appear reasonable. It stands to reason that the effect on progression will depend on how far around the roundabout the vessel has to travel and how much it has to deviate from the optimal path in order to do so. When it comes to TSS lanes, observe that because an informed strategy with a soft constraint is used in combination with a course suggester, entering the lane does not reduce search progression at all. However, please note that the informed strategy only considers how to follow a lane when entering it. Thus, it is probable that if the planning frontier were to be obstructed by a lane in the opposite direction, this would impact search progression negatively.

6.2.2 Large-scale mission planning in a varied coastal environment

A large-scale hypothetical mission is set up in coastal waters between the port of Stavanger and Sauda on the west coast of Norway to test the full capabilities of the mission planner. The initial configuration q_I and goal configuration q_G are both given by Equation (6.4),

$$q_I = \begin{bmatrix} 5.729235 \\ 58.971442 \\ 1.77 \end{bmatrix} \quad q_G = \begin{bmatrix} 6.348977 \\ 59.640291 \\ 0 \end{bmatrix} \tag{6.4}$$

and the geodesic distance between these ports is approximately 81.9 kilometers. The mission region processed for this mission is approximately $80 \times 47km^2$. Any reader familiar with traveling between these ports knows that an efficient route will go through narrow straits and relatively open coastal waters. Thus, it is considered a suitable and realistic mission to evaluate the mission planner's path quality and performance. Path quality is generally related to smoothness, safety, and optimality. In addition to path quality, mission planner performance and robustness are evaluated. To evaluate performance, the search progression metric is utilized once more. Additionally, all core procedures are monitored. Due to the deterministic properties of the designed and implemented mission planner, running Monte Carlo Simulations with Gaussian noise applied to the initial and goal configuration will not be sufficient to test robustness. Instead, robustness should be quantified by running the Mission Planner on various mission regions. A testing effort separate from this mission concerns robustness, thus robustness is not considered in the following.

Figure 6.14 provides an overview of the resulting path from the mission planner with two points of interest highlighted. Observe from the figure that the mission planner guides the search through narrow straits requiring significant maneuvering in POI1 and POI2, even though longer route options in relatively open waters exist. This pattern indicates that the mission planner employs a strategy focused on path optimality and exploits the vessel maneuverability to achieve this. Such behavior is expected, as the scaling factor on the distance heuristic is set relatively high at 1.2. Setting the scaling factor > 1 results in the Hybrid A* sampled path following the graph-optimal A* search in a greedy manner whenever feasible and safe maneuvers to accomplish this exist. Comparing the points of interest in Figure 6.14 and the A* optimal path tree in Figure 6.15 supports this observation. Because the sample-based Hybrid A* planner essentially is attempting to follow the best available optimal path approximation, this greedy strategy benefits the optimality characteristic of the resulting path in the average case. Mission planning is not only about path optimality. It must also be ensured that the path is feasible and safe regarding the risk of collision or grounding. Assuming the vessel model sufficiently accurately captures the vessel dynamics for which one is planning a mission, feasibility is ensured by design. Safety, however, must be ensured through the Hybrid A* search heuristic. The heuristic is calculated by combining the scaled distance-to-goal approximation and the artificial Voronoi field strength additively, where the latter is intended to ensure nominal path safety. While these two components to some extent represent conflicting interests, the Voronoi field does allow passage through narrow straits by design as long as a reasonable distance to land is ensured. To evaluate if this indeed is enough to ensure nominal path

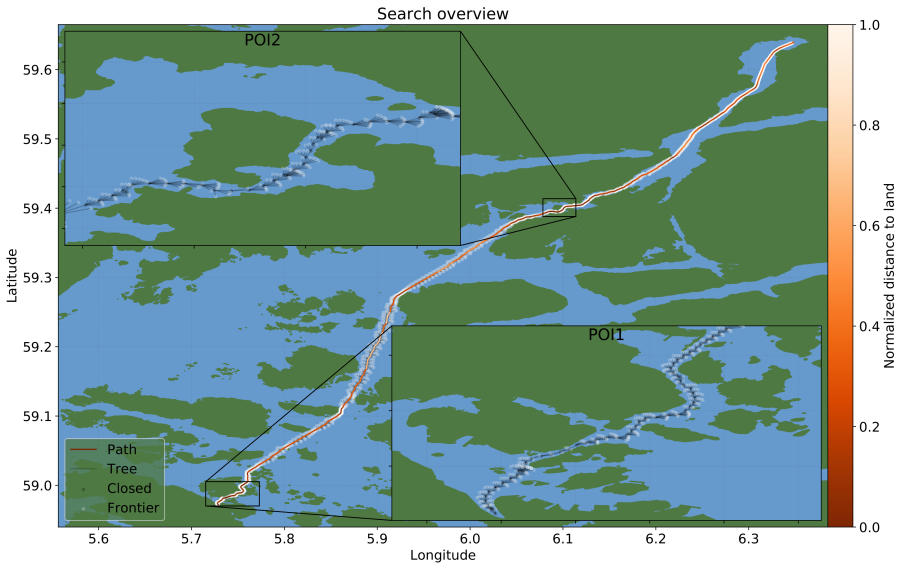


Figure 6.14: Overview of Hybrid A* Search result with two Points of Interest (POI)s. ENC Data courtesy of The Norwegian Mapping Authority

safety in the mission, the three path waypoints closest to land are explored and visualized together with the path and Voronoi field strength in Figure 6.16. The reader is reminded that the Voronoi field has a repelling effect, resulting in a tendency towards lower field values for the sampled motion whenever this does not severely increase path length.

The path waypoint closest to land is shown in Figure 6.16a. Comparing the coordinates of the subregion with the overview in Figure 6.14, it is clear that this strait is crossed early on in the search, shortly after leaving the Port of Stavanger. While it is a narrow strait, the path keeps as much distance to land as possible. Furthermore, the distance to the land geometry is $\approx 34.7m$, which for a vessel the size of a Viknes 830 should be considered well within a safe distance margin. This observation inspires confidence in the safety of the path overall, as no waypoint is closer to any land geometry than this.

The waypoint with the second-highest proximity to land is shown in Figure 6.16b. Once more, the proximity to land occurs when passing through a strait. However, the proximity to land is higher along the path segment between the highlighted and previous waypoint. This highlights a flaw in the evaluation method and a simplification in the mission planner implementation. Only the waypoint vertices and not the edges connecting the vertices are evaluated for land proximity. One could then expect that the actual shortest distance to land can be somewhere else than the selected waypoints entirely. By design, this is highly unlikely for most planning scenarios. The mission planner determines land proximity based on the search tree configuration vertices and not the curved configuration transition edges

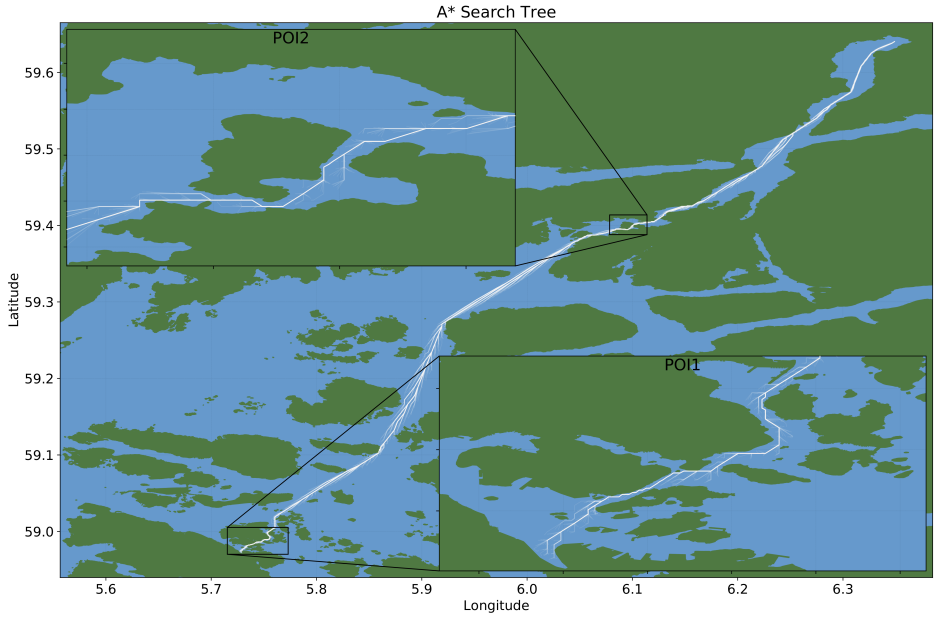
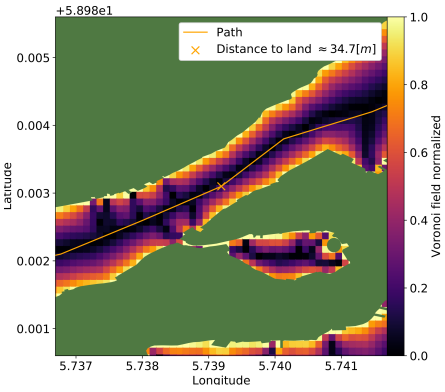


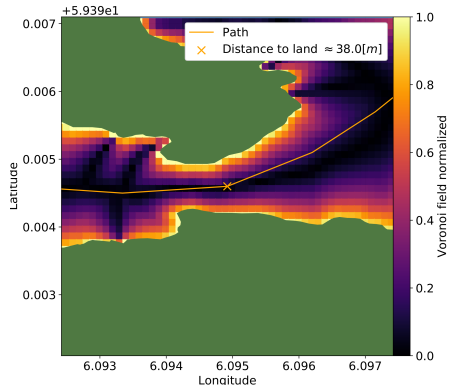
Figure 6.15: A* search tree with highlighted POIs

between vertices. This is done to optimize the planner and improve search efficiency. If the simulation time used in motion sampling was static and large, one could end up with long transition edges close to land, and the aforementioned issue would then likely occur frequently. The implemented adaptive simulation time mitigates this issue because edges between configuration vertices in the motion-sampling graph with adaptive simulation time always are short in proximity to land due to low motion sampling simulation time in such locations. One could argue that the problem could still occur with sufficiently sharp geometries or sufficiently high demanded speed. It would typically be exceptionally rare to come across such hazard geometries in coastal waters where the geometries have been buffered and the desired speed easily can be controlled.

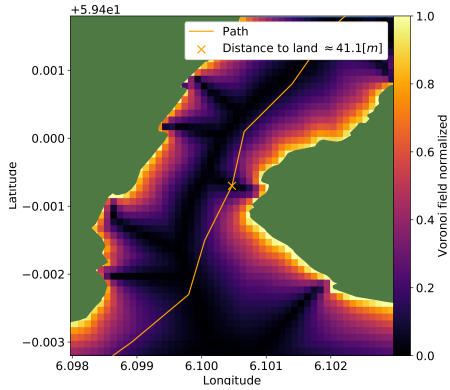
The waypoint third closest to land is shown in Figure 6.16c, and while the two former explored waypoints were in or near straits, this is in a wider channel. Instead of following the middle of the channel, the planner here takes advantage of an artifact of the Voronoi skeleton, which has resulted in a low field value branch heading towards the small bay port of the highlighted waypoint. If only the distance-to-goal approximation heuristic guided the search, the path would tangent land several places, including close to the highlighted waypoint. Observing that with the Voronoi field added to the heuristic, the path is repelled close to where there is no field potential, it can be said that with the current tune, the Voronoi field is quite strict locally. This affects the path optimality negatively but ensures path safety. If willing to sacrifice some safety margin for increased path optimality, one could change the tune. It must then be carefully re-evaluated if adaptive simulation time is still enough to mitigate the consequences of the aforementioned distance evaluation



(a) Voronoi field in proximity of waypoint closest to land



(b) Voronoi field in proximity of waypoint second closest to land



(c) Voronoi field in proximity of waypoint third closest to land

Figure 6.16: Voronoi field and path in proximity of three waypoints with the lowest land distance.

simplification.

Having evaluated the path generated by the mission planner, what remains is to evaluate planning efficiency. Observe in Figure 6.17 that the moving average search progression is positive for the entirety of the search, with significant progress spikes around $\approx 40s$, $75s$ and $100s$. Comparing with the benchmark data for the *simulateVessel* procedure in Figure 6.18, a similar pattern is found. The runtime of *simulateVessel* naturally correlates with the simulation time, which depends on proximity to land. Thus, the progression spikes occur when the search frontier is in relatively open areas of the mission region. This is expected and indicates that the mission planner operates as efficiently as the current implementation permits. While a different planning strategy or an implementation overhaul could improve efficiency, it can be said that the current guidance system has acceptable search progression given the focus on path feasibility, optimality, and safety. In order to complete the efficiency evaluation, the runtime of all key procedures relevant for iterative search frontier

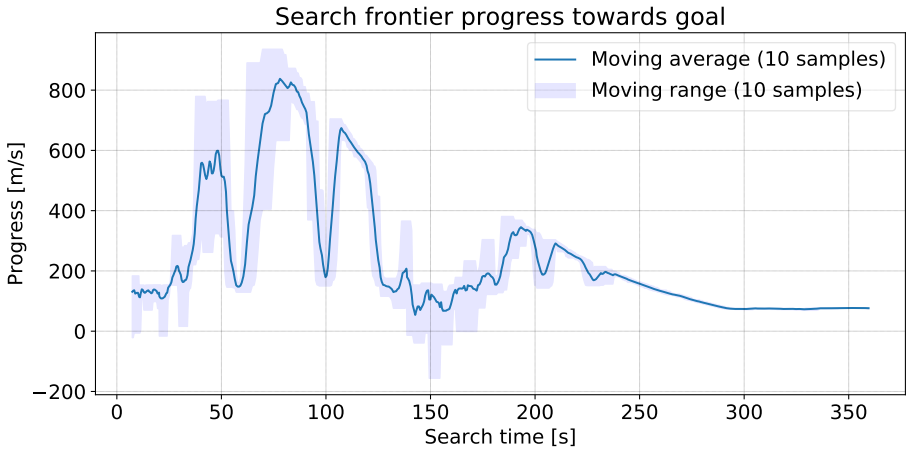


Figure 6.17: Search progression towards goal

expansion should also be considered to evaluate how the area around the frontier affects the efficiency and how the mission planner efficiency would scale with mission size.

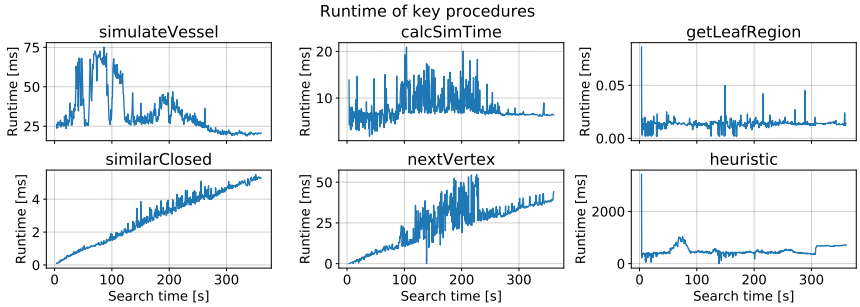


Figure 6.18: Runtime of key procedures. Accumulation to one value for each search frontier iteration.

Figure 6.18 shows the runtime of all the key procedures involved in iterative exploration and expansion of the Hybrid A* search frontier. Please remark that because some procedures are called several times for each iteration and some are called only once, the cumulative runtime of each procedure for each search iteration is utilized in this evaluation. There are several interesting remarks to be made from this data. Observe that *simulateVessel* have large runtime spikes, and that it typically is high when *calcSimTime* is low. This pattern is no coincidence. When determining the adaptive simulation time in *calcSimTime*, the computational cost of a query to the distance-to-land approximation in the map service is on average proportional to the proximity to land, as more hazard geometries will have to be checked there compared to in open waters where many hazards will be beyond the specified max range of the query. Other procedures, such as the *getLeafRegion* procedure,

are not affected by the hazards in a specified range around a spatial query. Instead, runtime depends on proximity to the hazard layer as a whole, as the leaf regions will be smaller in such areas and thus require a deeper search in the quadtree. It is worth noting that leaf region queries in the quadtree are blazing fast, indicating that the quadtree implementation is efficient. The procedures *similarClosed* and *nextVertex* stand out from the rest by having runtimes that, on average, increase linearly with time. Both of these procedures make evaluations based on the search tree, with *similarClosed* depending on the closed vertices and *nextVertex* on the explored vertices. While the runtime of the former remains low for the entirety of the search, the runtime of the latter is significant towards the end of the search. This highlights that while the mission planner has scaled reasonably well with a large mission region, computational cost will take a significant hit for even larger mission regions. The last procedure to evaluate is the *heuristic* procedure. Observe a spike in the runtime in the first call to the procedure before it starts to average significantly lower. To understand why this happens, the three most computationally expensive subprocedures in the heuristic are further explored in Figure 6.19.

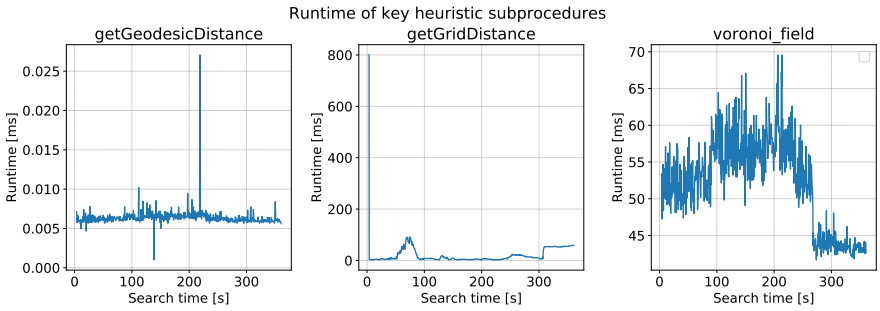


Figure 6.19: Runtime of key *heuristic* subprocedures

Observe in Figure 6.19 that the spike is caused by the initial call to the *getGridDistance* subprocedure. This procedure initially searches for a graph-optimal path from the initial configuration to the goal in the mission-region framed region quadtree graph utilizing A*. Subsequent calls to the procedure take advantage of the novel sequence matching concept added to the A* algorithm, and it is evident from figure 6.20 that this improves runtime dramatically. This observation, combined with the visual inspection of the distance approximation consistency of the A* Search tree visualized in Figure 6.15 indicates that the novel sequence matching method is working as intended, with efficiency improvements far beyond what was initially expected.

In the last phase of the search, observe a sudden and significant increase in runtime. This happens because sequence matching can not fast-forward the search when near the top of the search tree. Thus, the distance approximation falls back to the conventional A* graph search used initially, with a resulting increase in computational cost. The start of the search frontier is much closer to the goal, and therefore less of the graph has to be traversed to find the graph-optimal path in the quadtree. Thus, the runtime increase is lower than for the initial search query. This highlights that while the current design and implementation of sequence matching do not allow for search fast-forwarding close to the

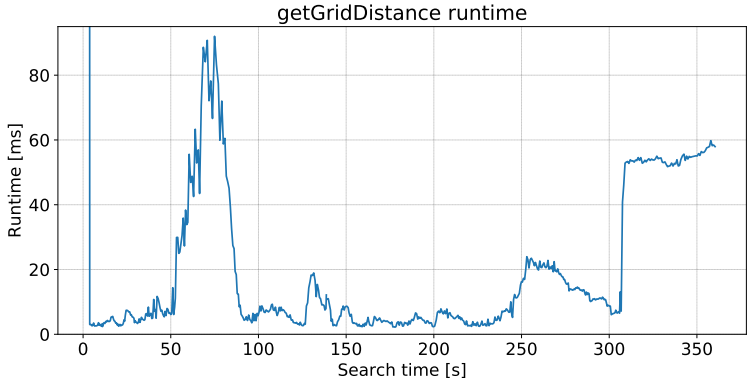


Figure 6.20: Runtime of the *getGridDistance* procedure focusing on runtime after initial spike

goal, this limitation does not significantly reduce search efficiency.

6.2.3 Evaluation of Mission Planner Robustness

The robustness of the mission planner is critical to ensure that it will produce satisfactory paths in different environments. The mission region hazard geometries can vary significantly from one region to another. In an effort to evaluate robustness, the mission planner is tested for four additional challenging missions spanning a total of three different mission regions. While the quantity of data clearly will be insufficient to draw any conclusion on robustness, it will give a strong indication. A general overview of the results is given below, emphasizing the robustness of path quality and safety.

Figure 6.21 highlights the resulting path for a different mission query in the same mission region previously used for the Stavanger to Sauda mission. The mission planner managed to find a path from the initial configuration to the goal region that appears sufficiently smooth. However, observe in the upper leftmost highlight that there is a potential safety issue when crossing under a bridge with pillars. This highlights the potential issue arising from the previously mentioned distance evaluation simplification. While adaptive simulation time was enough to remedy this issue in the Stavanger to Sauda mission, it is not sufficient here, at least with the current tune of the system. Thus, while the generated path is still collision-free, ensuring safety when following it perfectly, it can be argued that the margin of safety is not sufficient and that changes should be made either to the design or implementation of the planner to facilitate improvement. This can be done by sampling the edge between configurations with some defined resolution and querying the Voronoi field for all sampled points.

The second mission is in the same mission region as the previous missions, and the result of mission planning is shown in Figure 6.22. Observe that while the planner guides the path through the middle of narrow straits, it does allow traversing into the potential field when going around would yield a significant detour.

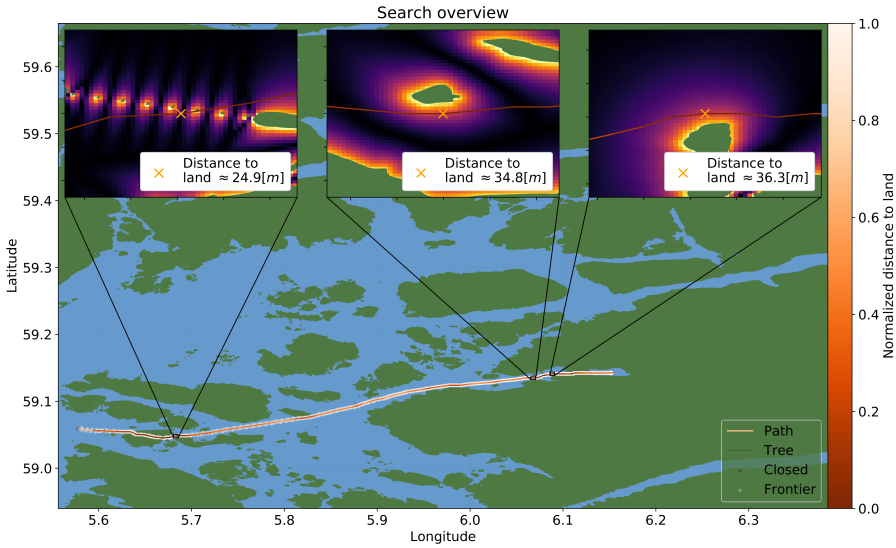


Figure 6.21: First mission on west coast of Norway. ENC data courtesy of The Norwegian Mapping Authority.

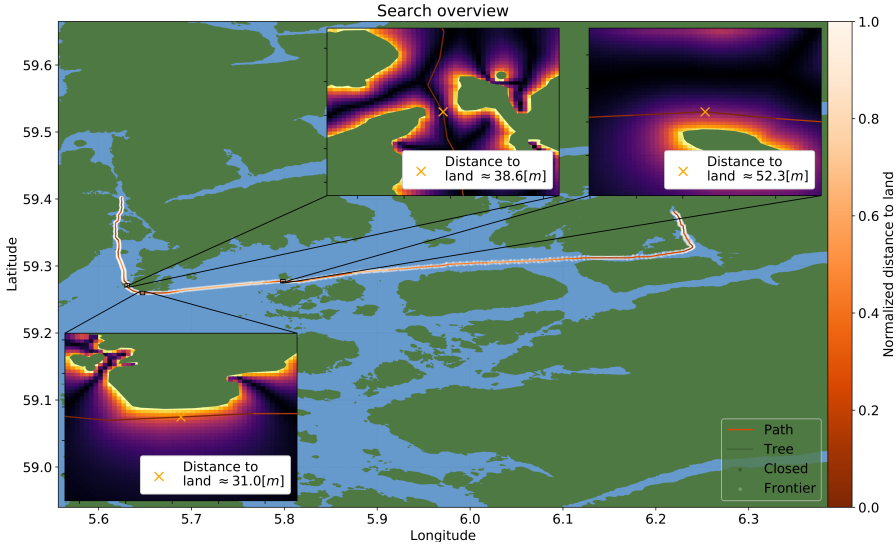


Figure 6.22: Second mission on west coast of Norway. ENC data courtesy of The Norwegian Mapping Authority.

The third mission is in an entirely different mission region, where the planner must find a feasible path traversing around a large island. Here, the planner has to frequently compromise between path optimality and safety, as the shortest path around an island will in

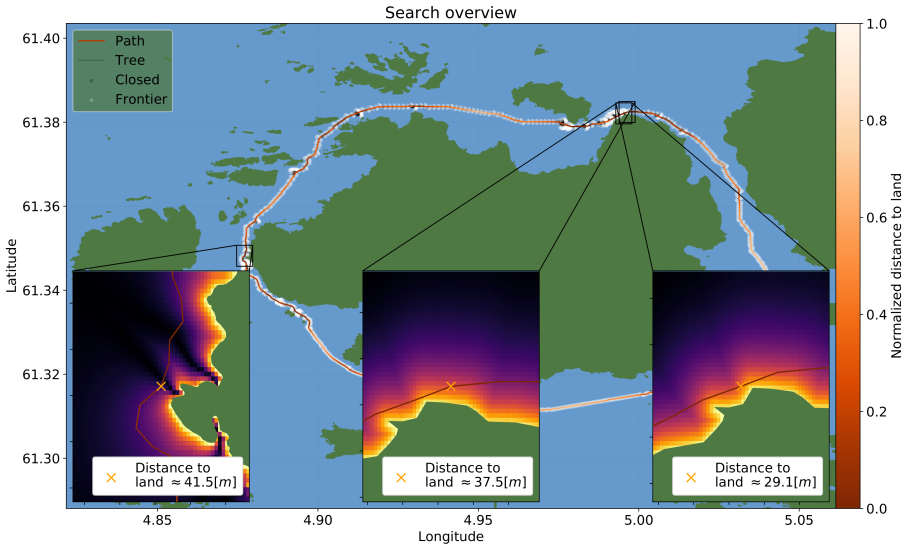


Figure 6.23: Mission travelling along shore around a larger island. ENC data courtesy of the Norwegian Mapping Authority.

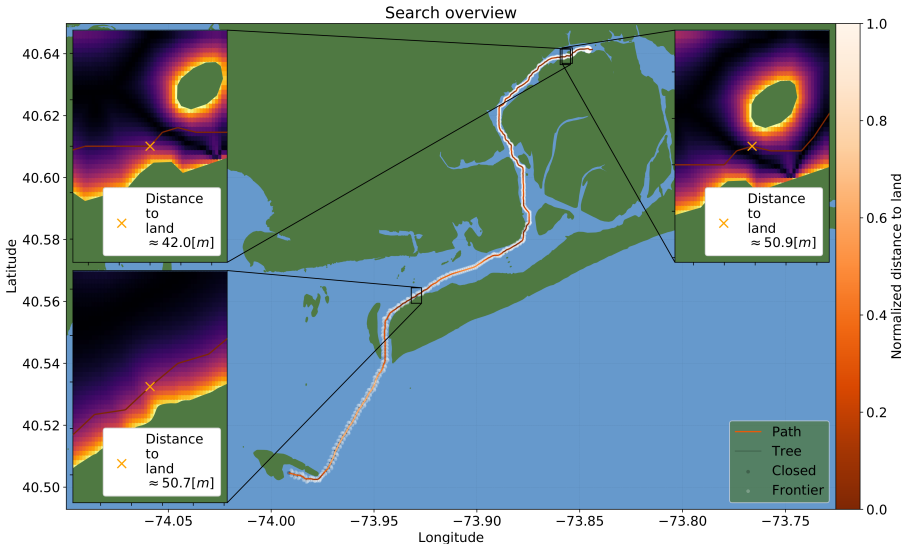


Figure 6.24: Mission travelling upstream in the Jamaica Bay Estuary outside New York City. ENC data courtesy of NOAA.

several locations go as close to land as possible. Observe in Figure 6.23 that while the minimum distance to land is slightly lower than in a majority of the previous missions, it remains reasonably high at $\approx 29.1m$. Furthermore, observe that the mission planner

actively compromises between path optimality and path safety, especially in open areas where the potential field has very high values close to land. This indicates that the proposed design of the mission planner with the utilized tune strikes an acceptable balance between path optimality and safety.

The last mission evaluated is on the east coast of the United States, traveling upstream into Jamaica Bay Estuary outside New York. Similar to the previous missions, the Mission Planner finds a relatively smooth path once more. Safety is also maintained, as the distance to land in the case of highest proximity is just $\approx 42.0m$ as shown in Figure 6.24.

Overall, testing the mission planner across different missions and different mission regions indicates that the employed mission planning strategy is robust with regard to path optimality. While it is typically sufficiently robust regarding safety, there are issues close to small geometries such as bridge pillars. It must be emphasized that the safety robustness ensured by following the nominal path perfectly is not compromised. However, the true USV will likely not be able to follow the path perfectly due to modeling simplifications, environmental disturbances, path tracker performance, and course autopilot performance. Thus, one could argue that path safety is not always satisfactory due to subpar safety margins in the proximity of specific types of geometries. Lowering the Voronoi field falloff rate would typically enforce crossing through bridges at a safer minimum distance and angle of approach. However, in testing, this had a significant impact on path length and search progression.

6.3 Complete mission scenario

In order to evaluate the implemented guidance system as a whole, a complete simulated mission should be run. From the perspective of the mission planner, this is no different from just doing mission planning in isolation, as the Mission Planner must generate a route before the USV starts moving. However, from the perspective of the COLAV system, a full-scale mission will present new challenges. There were no static hazards in the isolated scenarios, and the linear projection in which the subsystem operates was kept constant. In the following, these simplifying conditions will no longer be met. The full-scale mission is a hypothetical transportation mission upstream in the Jamaica Bay Estuary. Underway, the USV will encounter three non-compliant obstacle vessels, all traveling at different velocities. Two vessels will be met head-on, while one vessel will be encountered in a crossing scenario. Due to limited sea room, the scenario has been set up with only single-obstacle encounters. After an initial overview of the results, obstacle avoidance and nominal path tracking will be evaluated.

6.3.1 Results overview

The full mission starts with the mission planner generating a path from an initial configuration q_I illustrated by a pentagon in Figure 6.25 to a goal position q_G marked by a star in the same figure. The geodetic positions are given in Equation (6.5).

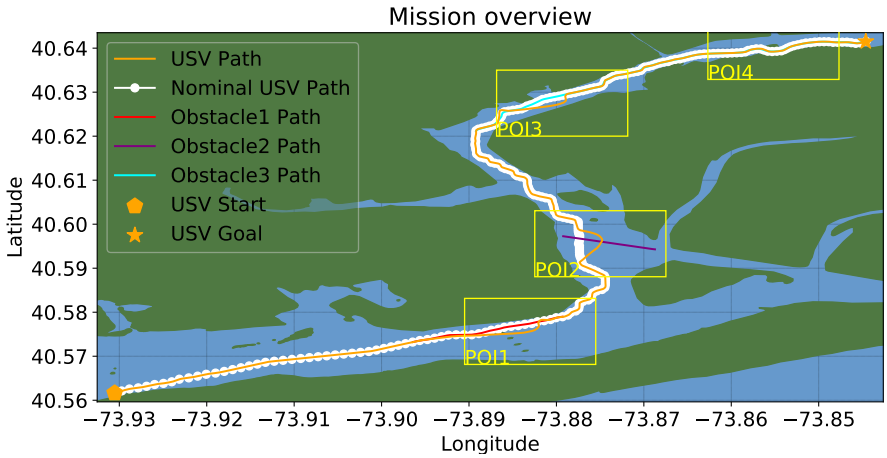


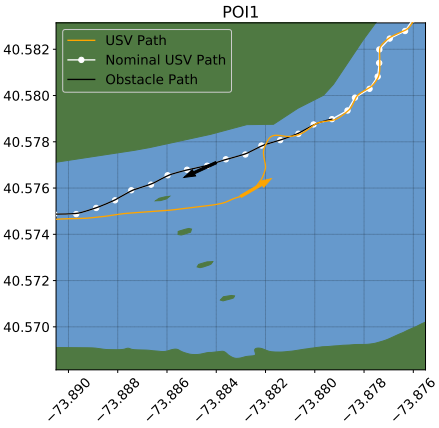
Figure 6.25: Overview of the complete mission in Jamaica Bay Estuary. Four Points of Interest (POI)s are highlighted and will be discussed. ENC Data courtesy of NOAA.

$$q_I = \begin{bmatrix} -73.93053235 \\ 40.561632870 \\ 0 \end{bmatrix} \quad q_G = \begin{bmatrix} -73.844579 \\ 40.641538 \\ 0 \end{bmatrix} \tag{6.5}$$

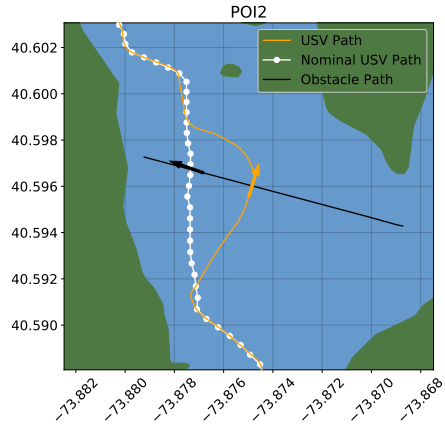
Observe that the Mission Planner generates a nominal path that appears to strike an acceptable balance between path safety and optimality except for the nominal path between Point of Interest (POI)2 and POI3 where it can be argued that there are unnecessary oscillations.

6.3.2 Obstacle encounters

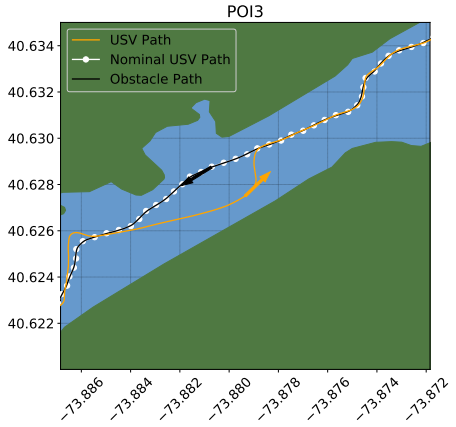
The first POI in Figure 6.25 is a head-on collision avoidance scenario. Figure 6.26a provides a closer look at the scenario, and Figure 6.28a and Figure 6.27a show distance and course values respectively. Observe that collision avoidance in this encounter s performed while crossing under a bridge with pillars that must be avoided. In the real world, such an encounter is critical to get right due to the danger of collision with the bridge pillars. The COLAV subsystem handles this well, deviating from the nominal path and crossing between two other bridge pillars than where the nominal path would prefer. Both the perceived and actual distance to obstacle tangents the specified hazard zone boundary at $180m$, indicating both that the COLAV system can make an efficient collision avoidance maneuver and that the proximity of the bridge does not affect its performance or reliability. It is worth noting that while the COLAV system by design will not suggest a course correction that imminently results in grounding or collision, it does not consider the proximity of static hazards. What is more, observe that the course correction immediately instructs the USV starboard, indicating a clear statement of intent as required for COLREG compliance. There is some oscillation both in the requested correction and the nominal course, a major contributor to which is the iterative linear ENU projection that occurs for every



(a) POI1: Head-on encounter with obstacle vessel.



(b) POI2: Crossing encounter

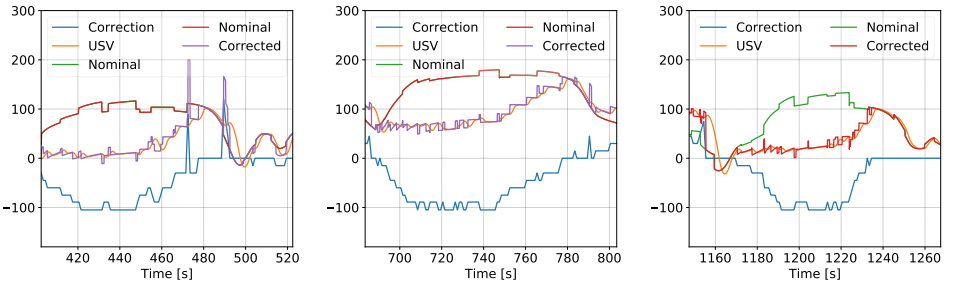


(c) POI3: Head-on encounter

Figure 6.26: Collision avoidance maneuvers in full scale mission

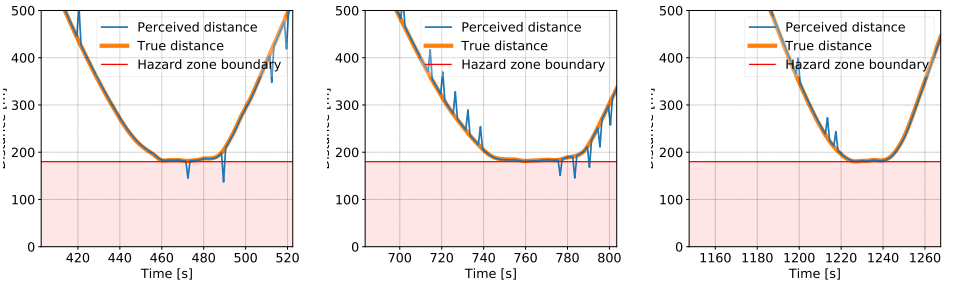
waypoint switch in the patch tracker. However, this oscillation is dampened significantly by the vessel, as any dynamic vessel is a natural low-pass filter. Thus, while the internal oscillation is not ideal and can cause unnecessary rudder wear and tear, it does not significantly affect the actual vessel response.

The second POI in Figure 6.25 is a crossing collision-avoidance scenario. Figure 6.26b provides a closer look at the scenario, while Figure 6.28b and Figure 6.27b show distance and course values. In this scenario, the overall behavior of the USV can be said to be COLREG compliant. However, as can be seen from the course correction values in Figure 6.27b, there is a short period initially where the COLAV system demands a course correction port. While this shares some similarities with the non-compliant behavior in the isolated collision avoidance scenarios in Section 6.1, the effect is much less severe in this



(a) POI1: Course values in USV **(b)** POI2: Course values in USV **(c)** POI3: Course values in USV

Figure 6.27: Course values in USV during collision avoidance maneuvers in full scale mission



(a) POI1: Distance to obstacle in first encounter **(b)** POI2: Distance to obstacle in second encounter **(c)** POI3: Distance to obstacle in third encounter

Figure 6.28: Distance to obstacles in collision avoidance maneuvers in full scale mission

case. Thus it is considered not to break the required statement of intent. Similar to the first obstacle encounter, there is some oscillation in the internal course values of the guidance system. Comparing once more with the isolated collision avoidance scenarios where no waypoint switching took place, the cause of this oscillation can, in large, be contributed to the switching of waypoints due to the internal change of ENU frame of reference that occurs during these switches. With regards to distance, it is clear from Figure 6.28b that the COLAV system allows the USV to deviate as little as possible from the nominal path while complying with the specified hazard zone radius. Furthermore, observe that the perceived distance, calculated using the Euclidean distance formula in the local flat frame of reference, except for some switching instability, is almost identical to the actual geodesic distance, calculated by solving the inverse geodetic problem. This indicates that using a local Cartesian frame of reference is viable when always attached to a waypoint that can be guaranteed to be in proximity of all objects using it. An undesirable consequence is an oscillatory effect dominant in distance approximations and internal course values. However, it can not be dismissed that frame synchronization improvements could remedy this issue.

The third POI in Figure 6.25 is another head-on encounter with a non-compliant obstacle vessel. Once more, distance values are shown in Figure 6.28c and course values are shown in Figure 6.27c. Of the three collision avoidance scenarios in the full mission, this one shows the poorest performance from the USV due to a lack of clear intent statement. The USV makes a large maneuver port initially before the COLAV system changes its mind and instructs starboard correction instead. Thus, while the obstacle is passed in a compliant manner, the maneuver is not entirely COLREG compliant. It is worth noting from the distance values in Figure 6.28c that the safety margin is maintained, so the maneuver can still be considered sufficiently safe. The reader is reminded that even when the USV itself is outside the d^{close} consideration range described in Table C.1, the COLREGs are considered if the predicted trajectory of the USV lies within this consideration range. This is done to ensure that action is taken sufficiently early. However, note that it contributes to the undesirable initial maneuver. In order to eradicate this maneuver, a complete redesign of the COLREG detection in the cost function is warranted based on these findings.

6.3.3 Nominal path tracking

Having investigated the obstacle encounters in the full mission, what remains is to evaluate the nominal path tracking behavior when no obstacle is present. Because the mission planning relies on the same vessel model used for the simulator, one could expect the tracking to be trivial and not a topic relevant for evaluation and discussion. However, path tracking relies on the performance of the LOS path tracking subsystem, and how well it works can thus be evaluated by investigating the nominal path tracking performance. Moreover, as an artificial challenge to compensate for the lack of two separate vessel models, mission planning was done with an intended vessel speed of $5m/s$. In the mission execution, desired vessel speed has been set to $10m/s$. This affects the maneuvering capabilities of the vessel. In the following, nominal path tracking is evaluated for a section of the path. The subregion in which this path is contained is called POI4 and is highlighted in the overview in Figure 6.25.

Figure 6.29 show the nominal and actual path of the USV in POI4. Observe that from visual inspection, path tracking appears satisfactory, with the most significant deviations occurring during larger maneuvers such as when passing by a smaller island in the middle of the channel. To quantify the path tracking performance, the cross-track error metric can be utilized.

The cross-track error is visualized in Figure 6.30, with vertical lines added to show where waypoint switching occurs. Observe that the cross-track error, for the most part, lies between $\pm 10m$, with occasional spikes just outside this range. The spikes occur because of waypoint switching, which can be seen in the figure. The LOS implementation in the guidance system uses two different methods to detect if a waypoint has been reached. The first method is to compare the distance from one waypoint to the next with the along-track value of the USV position. This is used to enable waypoint switching when not following the nominal path. In proximity to the nominal path, the circle of acceptance is also checked to ensure that maneuvering action to accommodate a turn in the path is taken before reaching the waypoint. The spikes in Figure 6.30 are a consequence of the latter method demanding waypoint switching before the waypoint is reached, with the intended

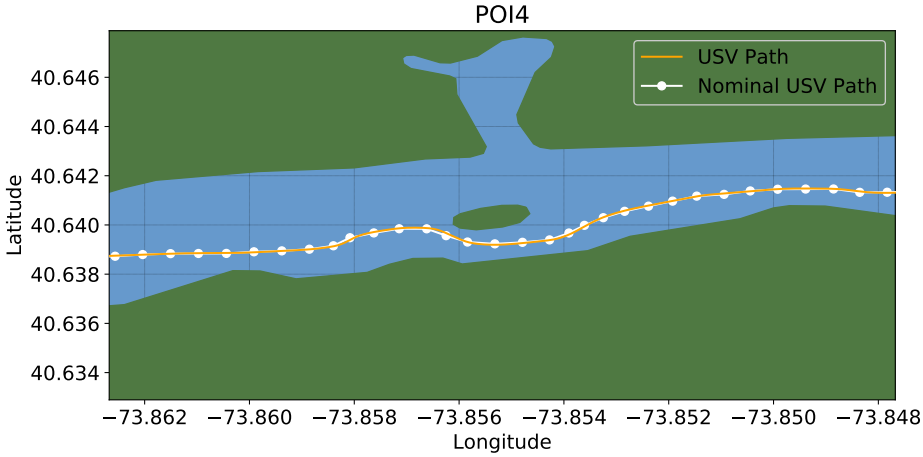


Figure 6.29: Overview of the part of the complete mission used to evaluate nominal path tracking.

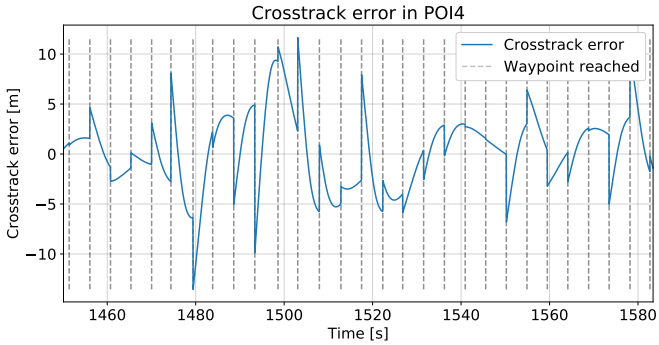


Figure 6.30: Path cross-track error in the highlighted section in POI4

effect of taking maneuvering action early to avoid overshooting the path. The reader is advised that in a simulation environment where there is no significant difference between the vessel model used for motion planning and mission execution, the former switching strategy method would suffice by itself. The latter has been introduced because it is often used in practice and can be beneficial if one wishes to test the system with a different vessel model in the simulator at some point.

Lastly, the demanded and actual course values for the nominal path tracking should be evaluated. Figure 6.31 show these values. The effect of the circle of acceptance waypoint switching makes an appearance here, with occasional high rates in the desired course that the USV naturally can not follow perfectly. However, the overall form indicates that the path generated by the mission planner is feasible for the USV to follow, as was also indicated by the overview in Figure 6.29. While feasible, it can be argued that the path should have been post-processed to make it more smooth as there are many small course correc-

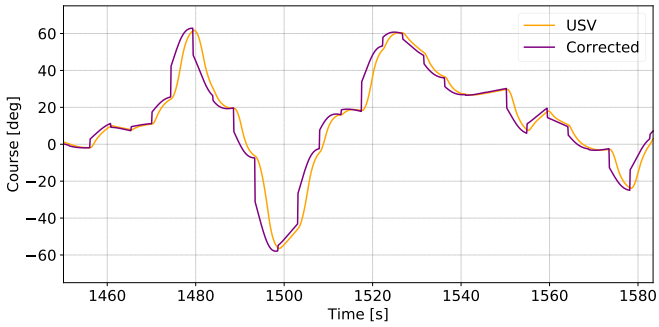


Figure 6.31: Course values in the highlighted section in POI4

tions demanded. Any change in commanded course will yield a response from the USV. In a conventional underactuated monohull vessel with a fixed-pitch propeller and a rudder, such as the simplified version of the Viknes830, this will result in rudder fluctuations. If the changes in the commanded course cannot be justified, this will cause unnecessary wear and tear on the rudder effector, the actuator controlling it, and mechanical supports. Looking at the path in Figure 6.29, it is clear that while the larger changes to the desired course are warranted due to the presence of a small island to be avoided, the smaller changes are not. The fluctuations are caused by trying to follow the nominal path, and the Mission Planner can hence be held responsible for them. There are many contributors to why the path generated by the Mission Planner can require excessive maneuvers. However, the main contributors are the variations in the Voronoi field in narrow channels and the varying distance-to-goal approximation accuracy due to limited quadtree frame resolution. Furthermore, there is no penalty in the mission planner for course alterations. Such an addition is not recommended because it can negatively affect path safety and optimality. Instead, one could post-process the generated path. While allowing for more robust checking due to the availability of context, path safety and optimality changes can also be monitored and, to some extent, controlled when performing post-processing of the path. The lack of such a post-processor can be seen as a limitation in the current mission planner.

Chapter 7

Conclusion and future work

7.1 Conclusion

In this thesis, a COLREG compliant guidance system for maritime vessels operating in coastal environments has been designed and implemented. The system relies on a novel specialization of the Hybrid A* algorithm for mission planning and Simulation-Based Control Behaviour selection to enable real-time collision avoidance. A method to build regional framed quadtrees traversable by A* has been implemented to enable optimized mission planning. Moreover, the Voronoi field has been used to ensure reasonable mission path safety margins. The latter has been enabled by a novel method to build a Voronoi skeleton of free space in a mission region by iterative pruning of partially connected edges in a Voronoi diagram. To make a complete guidance system, a path tracker utilizing the LOS guidance law and enabling robust waypoint switching has also been implemented. What is more, a map-preprocessor and map service have been designed and implemented to provide a priori information based on S-57 ENC charts to the mission planner and collision avoidance subsystems.

Testing and evaluation of the guidance system have been done following a divide and conquer strategy. First, the collision avoidance subsystem was evaluated in isolated collision scenarios with multiple non-cooperative obstacle vessels using Monte Carlo simulation and Gaussian noise for variability. Thereafter, mission planner compliance with TSS lanes, separation zones, and roundabouts was evaluated. Having tested the aspects of the guidance system considered most relevant for COLREG compliance, the mission planner was thereafter evaluated in large-scale mission regions on the west coast of Norway and the east coast of the United States. Finally, a complete simulated mission scenario in Jamaica Bay Estuary was performed and thoroughly evaluated.

The guidance system has shown great potential in avoiding obstacle vessels and guiding the USV ownship from an initial configuration to a goal position safely, efficiently, and

in compliance with key COLREG rules across a variety of environments and scenarios. However, it has not been only smooth sailing. The COLREG compliance rate in collision avoidance scenarios does not meet expectations, indicating that a partial or complete redesign of the COLAV system is warranted. Moreover, the mission planner struggles to progress towards the goal and find acceptable paths in the proximity of TSS roundabouts. Additionally, it can occasionally produce path segments that do not maintain a reasonable distance to land and, in general, lack some method of path post-processing to limit unnecessary maneuvers. What is more, while the guidance system always manages to get the simulated vessel to the goal safely in the testing that has been done, the exclusion of environmental disturbances such as current, wind, and waves leaves questions of robustness in real-world scenarios unanswered. However, the guidance system proves viability for using sample-based motion planning and motion primitives to enable optimized, predictable, and rules-compliant guidance in vast dynamic maritime environments. Additionally, it does propose a design for the system, highlights core technologies to facilitate it, and provides an open-source implementation in ROS Noetic for anyone to utilize. Thus, it lays a foundation on which future work can and should be done.

7.2 Future Work

The guidance system implemented in this thesis is subject to some strict limitations and simplifications which evidently will have to be overcome for any real use outside of academic research. Furthermore, deficiencies of the design found through testing should be addressed. In the following, suggestions for modifications and additions to the implemented guidance system to overcome the main deficiencies are described.

The motion sampling in the mission planner and trajectory predictions in the COLAV system both assume no current, wind or wave motion. This does not have any consequence in the simulation environment used for testing because that too does not support it. However, in a more advanced simulator or real-world sea trials, these environmental disturbances do affect the ownship and should be taken into account when evaluating performance, robustness and safety in the mission planner and COLAV systems. To incorporate such disturbances into the guidance system, one can take advantage of the modular design and simply replace the vessel model with another which can take environmental disturbances into account. A sufficiently fast and reliable source of weather forecast data will also have to be incorporated to feed the model during mission planning and execution. One can also improve the mission planning further by incorporating weather-optimization strategies for improved vessel endurance and enable the ability to operate in challenging weather conditions.

The guidance system has in testing not been subjected to noise or otherwise imperfect measurements or estimates. It is therefore suggested to evaluate how the guidance system performs in the presence of imperfect sensory data and make any necessary changes to remedy it. Another simplifying assumption used in testing is that obstacles do not comply with COLREGs and are not making any large maneuvers when encountered. This has consequences for the COLAV system, which in the implementation expects that the obstacle vessels follow a linear trajectory with a constant velocity during control behaviour predic-

tion. For real-world applications in coastal environments, it is likely to encounter many vessels with high maneuverability that do comply with the COLREGs themselves. Thus, it is suggested that the COLAV design should be altered to facilitate obstacle maneuvers in the prediction by exploiting stated or predicted obstacle intent.

Compliance with TSS objects are only handled in the mission planner in the implemented guidance system, and only lanes, separation zones and roundabouts are considered. While TSS objects are not frequently encountered in coastal environments close to shore, it is worth noting that if with future standard improvements of Electronic Navigational Charts TSS elements are utilized more extensively, a new strategy to handle them should be designed as the current has significant shortcomings with regards to robustness, safety and efficiency. Furthermore, it is also suggested to research how TSS elements can be incorporated in collision avoidance to ensure not only the safety of all vessels but also that flow of traffic is not reduced overall.

From the perspective of the implemented guidance system, some efficiency improvements can also be suggested. With the increasing core and thread count in modern computers it is suggested to improve the guidance system performance further by introducing concurrency in motion-sampling. An effort has been made to optimize the guidance system code and introduce concurrency through OpenMP whenever it was considered beneficial, however, there is still more that can be done both as design measures and implementation measures to further enhance performance. What is more, the performance of mission region preprocessing has not been a primary focus of this thesis. While concurrency and GDAL/OGR specific optimization techniques are used to efficiently build the regional framed quadtree for a given mission region, similar optimization efforts have not been performed in ENC data extraction and Voronoi skeleton generation. Improving the design and implementation of mission region pre-processing is therefore also suggested as future work that can enable the interpretation of more data attributes from ENC charts. Only a limited amount of data extracted from the charts are actually processed and utilized in this thesis, and it is assumed that a lot of untapped potential can be realized through clever interpretation of a broader range of data objects and attributes found in the ENC charts.

Bibliography

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs", vol. 1, no. 1, pp. 269–271, Dec. 1959, ISSN: 0945-3245. DOI: 10.1007/BF01386390.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

[3] International Maritime Organization, "International Convention for the Safety of Life at Sea", Nov. 1974. [Online]. Available: imo.org/en/KnowledgeCentre/ConferencesMeetings/Pages/SOLAS.aspx.

[4] N. J. Nilsson, "Shakey the robot", 1984.

[5] S. Kambhampati and L. Davis, "Multiresolution path planning for mobile robots", vol. 2, no. 3, pp. 135–145, 1986. DOI: 10.1109/JRA.1986.1087051.

[6] S. Fortune, "A sweepline algorithm for voronoi diagrams", *Algorithmica*, vol. 2, no. 1, p. 153, Nov. 1987, ISSN: 1432-0541. DOI: 10.1007/BF01840357.

[7] J. P. Snyder, "Map projections: A working manual", Washington, D.C., Tech. Rep., 1987, Report. DOI: 10.3133/pp1395.

[8] H. Samet, "An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures", in *Theoretical Foundations of Computer Graphics and CAD*, R. A. Earnshaw, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 51–68, ISBN: 978-3-642-83539-1.

[9] A. Elfes, "Using occupancy grids for mobile robot perception and navigation", *Computer*, vol. 22, no. 6, pp. 46–57, 1989. DOI: 10.1109/2.30720.

[10] J. C. Latombe, *Robot Motion Planning*, 1. Edition. Boston: Springer, 1991. DOI: https://doi.org/10.1007/978-1-4615-4022-9.

[11] E. Clementini, P. Di Felice, and P. van Oosterom, "A small set of formal topological relationships suitable for end-user interaction", in *Advances in Spatial Databases*, D. Abel and B. Chin Ooi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 277–295, ISBN: 978-3-540-47765-5.

[12] J. Zhu, “Conversion of Earth-centered Earth-fixed coordinates to geodetic coordinates”, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 3, pp. 957–961, 1994. DOI: 10.1109/7.303772.

[13] R. Kimmel and J. Sethian, “Fast marching methods for computing distance maps and shortest paths”, 1996. [Online]. Available: https://cds.cern.ch/record/303038.

[14] L. Dagum and R. Menon, “OpenMP: an industry standard API for shared-memory programming”, in *Proceedings of IEEE Computational Science and Engineering*, vol. 5, 1998, pp. 46–55. DOI: 10.1109/99.660313.

[15] S. M. Lavalle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning”, *The annual research report*, 1998.

[16] A. Yahja, A. Stentz, S. Singh, and B. Brumitt, “Framed-quadtree path planning for mobile robots operating in sparse environments”, in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 1998, 650–655 vol.1. DOI: 10.1109/ROBOT.1998.677046.

[17] International Hydrographic Organization, “IHO Transfer Standard for Digital Hydrographic Data”, 2000. [Online]. Available: https://iho.int/uploads/user/pubs/standards/s-57/31Main.pdf.

[18] A. L. Jeremy G. Siek Lie-Quan Lee, *The Boost Graph Library: User Guide and Reference Manual*, ser. The C++ In-Depth Series. Addison-Wesley Professional, 2001, ISBN: 978020172914-6.

[19] W. Naeem, R. Sutton, S. M. Ahmad, and R. S. Burns, “A Review of Guidance Laws Applicable to Unmanned Underwater Vehicles”, *Journal of Navigation*, vol. 56, no. 1, pp. 15–29, 2003. DOI: 10.1017/S0373463302002138.

[20] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877.

[21] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Practical Search Techniques in Path Planning for Autonomous Driving”, *AAAI Workshop - Technical Report*, Jan. 2008.

[22] Ø. A. G. Loe, “Collision Avoidance for Unmanned Surface Vehicles”, 2008. [Online]. Available: http://hdl.handle.net/11250/259696.

[23] K. Ahnert and M. Mulansky, “Odeint – Solving Ordinary Differential Equations in C++”, *AIP Conference Proceedings*, vol. 1389, Oct. 2011. DOI: 10.1063/1.3637934.

[24] S. Karaman and E. Frazzoli, *Sampling-based Algorithms for Optimal Motion Planning*, 2011. arXiv: 1105.1186 [cs.RO].

[25] W. Torge and J. Müller, *Geodesy*. De Gruyter, 2012. DOI: doi:10.1515/9783110250008.

[26] J. Gómez, A. Lumbier, S. Garrido, and L. Moreno, “Planning Robot Formations with Fast Marching Square including Uncertainty Conditions”, *Robotics and Autonomous Systems*, vol. 61, pp. 137–152, Feb. 2013. DOI: 10.1016/j.robot.2012.10.009.

[27] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees", *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013. DOI: 10.1007/s10514-012-9321-0.

[28] C. F. F. Karney, "Algorithms for geodesics", *Journal of Geodesy*, vol. 87, no. 1, pp. 43–55, Jan. 2013. DOI: 10.1007/s00190-012-0578-z.

[29] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with colregs, using velocity obstacles", *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2014. DOI: 10.1109/JOE.2013.2254214.

[30] National Geospatial-Intelligence Agency, "World geodetic system 1984, Its definition and relationships with local geodetic systems", 2014. [Online]. Available: https://earth-info.nga.mil/index.php?dir=wgs84&action=wgs84.

[31] Y. Liu, R. Song, and R. Bucknall, "A practical path planning and navigation algorithm for an unmanned surface vehicle using the fast marching algorithm", in *OCEANS 2015 - Genova*, 2015, pp. 1–7. DOI: 10.1109/OCEANS-Genova.2015.7271338.

[32] A. D'Angelo, "A Brief Introduction to Quadtrees and Their Applications", 2016.

[33] T. A. Johansen, T. Perez, and A. Cristofaro, "Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3407–3422, 2016. DOI: 10.1109/TITS.2016.2551780.

[34] H. Mahmoud and N. Akkari, "Shortest Path Calculation: A Comparative Study for Location-Based Recommender System", in *2016 World Symposium on Computer Applications Research (WSCAR)*, 2016, pp. 1–5. DOI: 10.1109/WSCAR.2016.16.

[35] I. B. Hagen, "Collision Avoidance for ASVs Using Model Predictive Control", 2017. [Online]. Available: http://hdl.handle.net/11250/2433779.

[36] International Maritime Organization, "ENCs Production, Maintenance and distribution guidance", May 2017. [Online]. Available: https://iho.int/iho_pubs/standard/S-65/S-65_ed2%5C%201%5C%200_June17.pdf.

[37] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. DOI: 10.1109/iros.2017.8202315.

[38] M. Przybylski and B. Putz, "D* Extra Lite: A Dynamic A* With Search–Tree Cutting and Frontier–Gap Repairing", vol. 2, pp. 273–290, Jun. 2017. DOI: 10.1515/amcs-2017-0020.

[39] Y. Pyo, H. Cho, R. Jung, and T. Lim, *ROS Robot Programming*. ROBOTIS Co.,Ltd., 2017, ISBN: 9791196230715.

[40] H.-T. L. Chiang and L. Tapia, "COLREG-RRT: An RRT-Based COLREGS-Compliant Motion Planner for Surface Vehicle Navigation", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018. DOI: 10.1109/LRA.2018.2801881.

[41] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "MPC-based Collision Avoidance Strategy for Existing Marine Vessel Guidance Systems", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7618–7623. DOI: 10.1109/ICRA.2018.8463182.

[42] S. V. Rothmund, "Ship Path Planning using Navigational Charts with Time-Dependent Weather Constraints", 2018. [Online]. Available: http://hdl.handle.net/11250/2616144.

[43] O. S. Otterholm, "Extracting Mapped Hazards from Electronic Navigational Charts for ASV Collision Avoidance", 2019. [Online]. Available: http://hdl.handle.net/11250/2622322.

[44] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based Subterranean Exploration Path Planning using Aerial and Legged Robots", in *Proceedings of Journal of Field Robotics*, Oct. 2020. DOI: 10.1002/rob.21993.

[45] K. Kjerstad, "Collision Avoidance System for Ships Utilizing Other Vessels' Intentions", 2020. [Online]. Available: https://hdl.handle.net/11250/2780862.

[46] N. Lauvås, "Design and development of a robotic fish tracking vehicle", 2020. [Online]. Available: https://hdl.handle.net/11250/2780997.

[47] B. C. Shah and S. K. Gupta, "Long-Distance Path Planning for Unmanned Surface Vehicles in Complex Marine Environment", *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, pp. 813–830, 2020. DOI: 10.1109/JOE.2019.2909508.

[48] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2. Edition. Hoboken: Wiley, 2021. DOI: 10.1002/9781119994138.

[49] S. Furre, "Agile collision-free path planning for an autonomous surface vehicle", Dec. 2021, [unpublished].

[50] GEOS contributors, *GEOS coordinate transformation software library*, Open Source Geospatial Foundation, 2021. [Online]. Available: https://libgeos.org/.

[51] C. F. F. Karney, *Geographiclib*, Version 1.52, 2021. [Online]. Available: https://geographiclib.sourceforge.io/1.52.

[52] A. Vagale, R. Bye, R. Oucheikh, O. Osen, and T. Fossen, "Path planning and collision avoidance for autonomous surface vehicles II: a comparative study of algorithms", in *Proceedings of Journal of Marine Science and Technology*, Feb. 2021. DOI: 10.1007/s00773-020-00790-x.

[53] A. Vagale, R. Oucheikh, R. T. Bye, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles I: a review", in *Proceedings of Journal of Marine Science and Technology*, vol. 26, Jan. 2021, pp. 1292–1306. DOI: 10.1007/s00773-020-00787-6.

[54] ETH ASL, *Geodetic_utils*, GitHub repository, 2022. [Online]. Available: https://github.com/ethz-asl/geodetic_utils.

[55] S. Furre, *USV Guidance System*, GitHub repository, 2022. [Online]. Available: https://github.com/sanderfu/usv-guidance-system.

[56] S. Furre, *enc_extract_lib*, GitHub repository, 2022. [Online]. Available: https://github.com/sanderfu/enc_extract_lib.

[57] GDAL/OGR contributors, *GDAL/OGR geospatial data abstraction software library*, Open Source Geospatial Foundation, 2022. DOI: 10.5281/zenodo.5884351. [Online]. Available: https://gdal.org.

[58] QGIS Development Team, *QGIS Geographic Information System*, QGIS Association, 2022. [Online]. Available: https://www.qgis.org.

[59] D. Register, *OpenCPN (Open Chart Plotter Navigator)*, 2022. [Online]. Available: https://opencpn.org/.

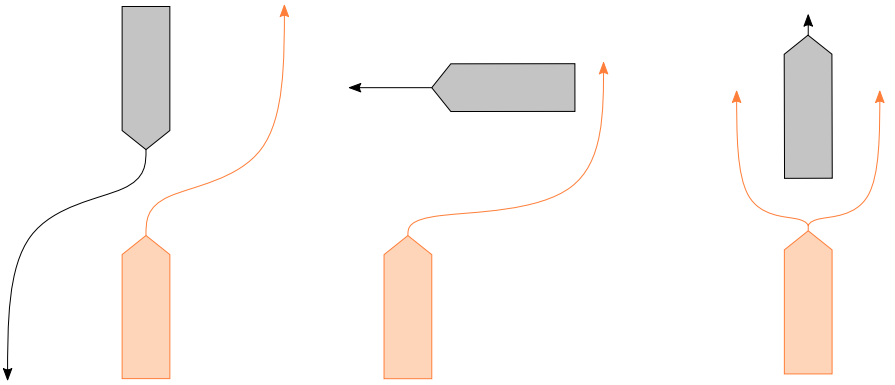
[60] M. Westerdahl, *JC_voronoi*, GitHub repository, 2022. [Online]. Available: https://github.com/JCash/voronoi.

Appendices

Appendix A

Relevant COLREG rules

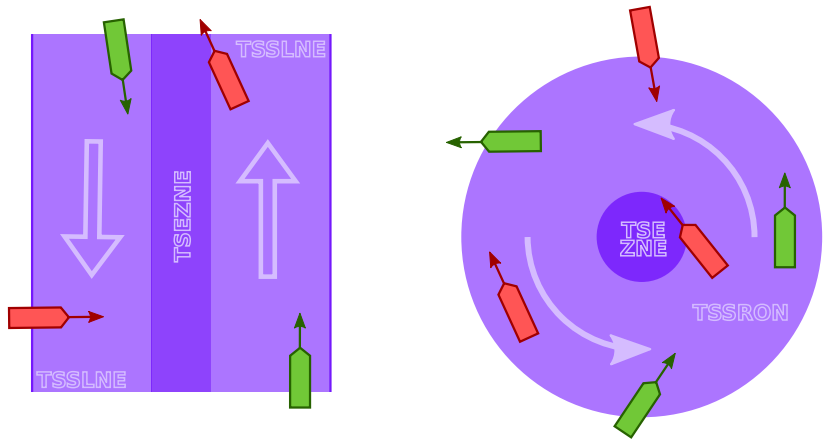
It is important that an USV follows the relevant International Regulations for Preventing Collisions at Sea [3] such that its behavior is consistent with what a ship captain would expect from a manned vessel. This appendix is intended to give a brief overview of the COLREGs most relevant for mission planning and collision avoidance in unmanned surface vehicles of limited size. Figure A.1 is intended to illustrate the most common collision-avoidance COLREG scenarios of relevance and Figure A.2 illustrates how to be compliant with TSS. The rule descriptions below are from [3]. Only the parts considered relevant for autonomous operations are given below.



(a) Head-on scenario **(b)** Crossing scenario **(c)** Overtake scenario

Figure A.1: Illustration of COLREG compliant maneuvers in relevant collision avoidance scenarios

- **Rule 8: Action to avoid collision**
  (b) Any alteration of course and/or speed to avoid collision, shall, if the circumstances of the case admit, be large enough to be readily apparent to another vessel



(a) TSS Lanes and separation zone **(b)** TSS Roundabout and separation zone

Figure A.2: Illustration of compliance (Green) and non-compliance (Red) in the presence of most relevant TSS objects according to Rule 10 of the COLREGs.

observing visually or by radar; a succession of small alterations of course and/or speed should be avoided.

(c) If there is sufficient sea-room, alteration of course alone may be the most effective action to avoid a close-quarters situation provided that it is made in good time, is substantial and does not result in another close-quarters situation.

(d) Action taken to avoid collision with another vessel shall be such as to result in passing at a safe distance. The effectiveness of the action shall be carefully checked until the other vessel is finally past and clear

(e) If necessary to avoid collision or allow more time to assess the situation, a vessel shall slacken her speed or take all way off by stopping or reversing her means of propulsion

- **Rule 10: Traffic separation schemes**
  (b) A vessel using a traffic separation scheme shall:
  - *(i) proceed in the appropriate traffic lane in the general direction of traffic flow for that lane*
  - *(ii) so far as practicable keep clear of a traffic separation line or separation zone*
  - *(iii) normally join or leave a traffic lane at the termination of the lane, but when joining or leaving from either side shall do so at as small an angle to the general direction of traffic flow as practicable.*

- **Rule 13: Overtake situation**
  (a) Notwithstanding anything contained in the Rules of Part B, Sections I and II, any vessel overtaking any other shall keep out of the way of the vessel being overtaken.

  (b) A vessel shall be deemed to be overtaking when coming up with another vessel from a direction more than 22.5° abaft her beam, that is, in such a position with reference to the vessel she is overtaking, that at night she would be able to see only the sternlight of that vessel but neither of her sidelights.

  (c) When a vessel is in any doubt as to whether she is overtaking another, she shall assume that this is the case and act accordingly.

  (d) Any subsequent alteration of the bearing between the two vessels shall not make the overtaking vessel a crossing vessel within the meaning of these Rules or relieve her of the duty of keeping clear of the overtaken vessel until she is finally past and clear.

- **Rule 14: Head-on situation**
  (a) When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other.

  (b) Such a situation shall be deemed to exist when a vessel sees the other ahead or nearly ahead and by night she could see the masthead lights of the other in a line or nearly in a line and/or both sidelights and by day she observes the corresponding aspect of the other vessel.

  (c) When a vessel is in any doubt as to whether such a situation exists she shall assume that it does exist and act accordingly

- **Rule 15: Crossing situation**
  When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.

- **Rule 16: Action by give-way vessel**
  Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear

It should be noted that rule 8 can be challenging to quantify compliance with, however it can be qualitatively discussed in collision avoidance scenarios. Remark that taking *early* and *substantial* action to avoid collision is key for both head-on and crossing scenarios.

Appendix B

Relevant ENC Objects for autonomous operations

All objects in the object catalog of the S-57 product specification are defined in Appendix A of the S-57 Product Specification [17]. While all objects have some relevance for navigation, not all of them are equally important for navigation of unmanned surface vehicles of a limited size such as the Viknes 830 vessel. Furthermore, the significance depends upon what kind of mission the vessel is undertaking. The most important objects are considered to be the ones describing hazardous areas in which the vessel could ground or collide. These objects are marked as hazards in Table B.1

Furthermore, objects describing areas where TSS or other navigational restrictions such as speed limits apply are considered important to ensure that the vessel operates in a safe, predictable and legal manner. All TSS objects are considered cautions and are given in table B.2. Additional objects considered as cautions for current and future development of the guidance system are marked as such in Table B.1. Observe that some objects are both hazards and caution objects in general, with their attributes determining on an object-by-object basis which category they truly correspond to.

Table B.1: Overview of relevant objects in the S-57 product specification. Acronym, object and description details are retrieved from [17].

Acronym	Object	Description	Hazard	Caution
BCNSPP	Special Purpose Beacon	A special purpose beacon is primarily used to indicate an area or feature, the nature of which is apparent from reference to a chart	Yes	Yes
BOYLAT	Lateral Buoy	Used to indicate the port or starboard hand side of the route to be followed. They are generally used for well defined channels and are used in conjunction with a conventional direction of buoyage.	Yes	Yes
BRIDGE	Bridge	A structure erected over a depression or an obstacle such as a body of water.	Yes	No
CTNARE	Caution Area	An area where the mariner has to be made aware of circumstances influencing the safety of navigation.	No	Yes
DEPARE	Depth Area	An area whose water depth is within a defined range of values	Yes	No
FAIRWY	Fairway	The main navigable channel for vessels of larger size. It is also the usual course followed by vessels entering or leaving harbors, called 'ship channel'.	No	Yes
LNDARE	Land area	The solid portion of the Earth's surface, as opposed to sea, water.	Yes	No
OBSTRN	Obstruction	Anything that hinders or prevents movement, particularly anything that endangers or prevents the passage of a vessel. The term is usually used to refer to an isolated danger to navigation.	Yes	Yes
OFSPLF	Offshore platform	A permanent offshore structure, either fixed or floating	Yes	Yes
PILPNT	Pile	A long heavy timber or section of steel, wood, concrete, etc.. forced into the earth which may serve as a support, as for a pier, or a free-standing pole within a marine environment.	Yes	No
PYLONS	Pylon or bridge support	A vertical construction consisting, for example, of a steel framework or pre-stressed concrete to carry cables, a bridge, etc.	Yes	No
RESARE	Restricted Area	A specified area on land or water designated by an appropriate authority within which access or navigation is restricted in accordance with certain specified conditions	No	Yes
SOUNDG	Sounding	A measured water depth or spot which has been reduced to a vertical datum.	Yes	No
UWTROC	Underwater or awash rock	A concreted mass of stony material or coral which dries, is awash or is below the water surface.	Yes	No
WRECKS	Wreck	The ruined remains of a stranded or sunken vessel which has been rendered useless.	Yes	No

Table B.2: Overview of TSS relevant objects, all of which are considered cautions in the overview spatial database. Acronym and object name is consistent with the S-57 object Catalogue [17], and description is paraphrased from it.

Acronym	Object	Description
DWRTCL	Deep water route centerline	Deep water route is area which has been accurately surveyed for clearance of sea bottom and submerged obstacles.
DWRTPT	Deep water route part	The deep water route area.
ISTZNE	Inshore traffic zone	Area between landward boundary of TSS object adjacent coast
PRCARE	Precautionary area	Area where ships must navigate with particular caution. A recommended direction of voyage may be included.
TSELNE	Traffic separation line	Line separating traffic lanes. Can be used for direction separation or ship class separation
TSEZNE	Traffic separation zone	Similar to traffic separation line, but instead of just a line is is an area with some extent.
TSSBND	TSS boundary	Outer limit of traffic lane part or roundabout
TSSCRS	TSS crossing	Area where traffic lines cross
TSSLPT	TSS lane part	Area in which direction of voyage is specified, and thus traffic flow uniform.
TSSRON	TSS roundabout	Point or zone around which traffic should move in counter-clockwise direction

Appendix C

Guidance system core parameters

The implemented guidance system has many parameters that must be set to appropriate values. While the final values below primarily are found empirically, the COLAV parameters in Table C.1 are to a large extent the same as the ones used in [35]. Please note that while essential for predictive control selection and simulation, the vessel model parameters are not given in this appendix. This is done because they are the result of an identification effort done by Loe [22] and thus should be seen in the context of the identification process. Furthermore, please note that while it is important that the vessel model is sufficiently realistic, the hydrodynamic parameters and vessel dimensions depend strictly on the physical vessel and are thus not parameters one modifies to enhance guidance system performance. In the following, the relevant parameters in the COLAV system, the Mission Planner and the ENC manager are summarized.

Table C.1: Collision Avoidance Cost Function Parameters with utilized tune based on [35].

Symbol	Description	Value
P	Collision risk factor time exponent	1.0
Q	Collision risk factor distance exponent	4.0
d^{cl}	Maximum COLREG consideration range [m]	500
d^{safe}	Minimum safe distance to obstacle [m]	100
K^{col}	Collision cost scaling factor	0.5
ϕ_{ahead}	Ahead angle [deg]	15.0
ϕ_{ot}	Overtake angle [deg]	68.5
ϕ_{ho}	Headon angle [deg]	42.5
ϕ_{cr}	Crossing angle [deg]	68.5
κ	COLREG Violation Scaling Factor	2.0
k_p	Velocity correction scaling factor	4.0
k_χ	Course correction scaling factor	0.6
δ_p	Velocity correction change scaling factor	3.5
$\delta_{\chi_{sb}}$	Scaling factor when penalizing change in course correction to starboard	0.9
δ_{χ_p}	Scaling factor when penalizing change in course correction to port	2.2
k_{corr}	Scaling factor when penalizing course correction different from zero	0.6

Table C.2: Parameters in the Mission Planner.

Symbol/Name	Description	Value
$t_{sim,u,min}$	Minimum adaptive simulation time underway [s]	30
$k_{t,approach}$	Scaling factor to determine when to switch to approach	0.015
$t_{sim,a,min}$	Minimum adaptive simulation time in approach [s]	15
r_{prune}	Pruning radius evaluating if vertex explored/closed [m]	20
α_{prune}	Pruning angle evaluating if vertex explored/closed [deg]	10
$r_{tss,roundabout}$	Maximum distance for TSS consideration [m]	3000
k_{dist}	Scaling distance-to-goal approximation in Hybrid A* heuristic	1.20
$k_{tss,lane}$	Scaling factor on course delta between USV and TSS Lane	0.20

Table C.3: Parameters in Electronic Navigational Chart (ENC) Manager.

Symbol/Name	Description	Value
$length_{max,divisor}$	Maximum distance between edge vertices in a Quadtree region [m]	300
$area_{minimum,region}$	Minimum region size [deg^2]	10e-7
$\beta_{voronoi}$	Ratio value in Voronoi diagram pruning	1.5
$\alpha_{voronoi}$	The Voronoi field falloff rate	0.001
$sat_{distance}$	The default distance saturation value [deg]	0.0075

Sander Furre

Path and Motion Planning for Unmanned Surface Vehicles

NTNU
Norwegian University of
Science and Technology