Doctoral theses at NTNU, 2023:200

Simon Vinding Blindheim

Risk-aware decision-making and control of autonomous ships

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Simon Vinding Blindheim

Risk-aware decision-making and control of autonomous ships

Thesis for the Degree of Philosophiae Doctor

Trondheim, June 2023

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

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-7102-1 (printed ver.)
ISBN 978-82-326-7101-4 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

ITK-report 2023-14-W

Doctoral theses at NTNU, 2023:200

Printed by NTNU Grafisk senter

Summary

Autonomous agents able to operate without human supervision require decision-making capabilities that predict outcomes based on internal and external models, sensors and data. However, research on the topic of incorporating risk awareness into such decision-making and control for autonomous ships is today limited. The purpose of this thesis is thus to contribute to accelerate and expand the knowledge within the field of risk-aware decision-making and control of autonomous ships, by developing novel models, algorithms and software tools, and demonstrating proof-of-concept results through simulation.

The thesis is divided into three main parts. The first part presents an application programming interface (API) for electronic navigational charts (ENC), which directly assists and enhances the actual process of effectively developing new autonomous planning and control systems with anti-grounding capability for both open research efforts as well as future industry applications. The second part examines how model predictive control (MPC) may be utilized for risk-aware autonomous control and path planning, based on the concept of risk using the ENC API and a ship model. The third part considers particle swarm optimization (PSO) as an alternative to nonlinear programming (NLP) in MPC, in order to ultimately further improve the risk-aware decision-making capabilities through the addition of simultaneous autonomous machinery management.

The first part introduces and demonstrates the ENC API as a programming tool, which may be used during development of algorithms and methods related to both manned or unmanned (autonomous or remotely operated) maritime surface vessels in future works.

The second part considers the use of MPC for autonomous ships, by minimizing costs based on the concept of risk. Chapter 3 presents a proof of concept demonstration of an MPC approach for joint autonomous path planning, risk-aware decision-making, and dynamic real-time emergency management. Chapter 4 develops a methodology which transforms the results of System-Theoretic Process Analysis (STPA) of the path planning problem with anti-grounding into mathematical equations and inequalities, which may be formulated and systematically structured into an optimal control problem to be solved numerically by the MPC approach.

In order to enable the use of non-smooth terms and discrete variables in the constraints and the cost function to be minimized, the last part considers the sampling-based PSO method as an alternative approach to MPC. Chapter 5 demonstrates an online and risk-aware waypoint re-planning approach, which displays analogous behavior to that of the MPC method. Chapter 6 utilizes the flexibility inherent to PSO by combining model-based predictions for autonomous machinery management (AMM) with the sampling-based risk-aware path re-planning method. Here, PSO essentially replaces NLP in the planning part of the problem, but is nevertheless used in a receding horizon procedure similar to an MPC approach. This elevates the ability of the autonomous navigation system (ANS) to even further minimize fuel consumption and risks through joint AMM and risk-aware path planning.

Sammendrag

Autonome agenter i stand til å operere uten menneskelig tilsyn krever evner for beslutningstaking som kan forutse utfall basert på interne og eksterne modeller, sensorer og data. Forskning på å inkorporere risikobevissthet inn i slik beslutningstaking og kontroll for autonome skip er derimot i dag begrenset. Målet med denne avhandlingen er dermed å bidra til å akselerere og utvide kunnskapen innen risikobevisst beslutningstaking og kontroll av autonome skip, ved å utvikle nye modeller, algoritmer og programvare, samt demonstrere konseptutprøving gjennom simulering.

Avhandlingen er delt inn i tre hoveddeler. Første del presenterer et grensesnitt for applikasjonsprogrammering for elektroniske sjøkart, som direkte bidrar til å forbedre selve prosessen for å effektivt utvikle nye autonome systemer for planlegging og kontroll med hensyn på anti-grunnstøting både for åpen forskning så vel som for fremtidige industriapplikasjoner. Andre del undersøker hvordan modellprediktiv kontroll (MPC) kan bli utnyttet for risikobevisst autonom kontroll og baneplanlegging, basert på et konsept av risiko ved bruk av programmeringsgrensesnittet og en skipsmodell. Tredje del betrakter partikkelsverm-optimalisering (PSO) som et alternativ til ulineær programmering i modellprediktiv kontroll, for å ytterligere øke systemets kapabilitet for risikobevisst beslutningstaking ved å inkludere simultan autonom maskineristyring eller kontroll.

Den første delen introduserer og demonstrerer grensesnittet for elektroniske sjøkart som et programmeringsverktøy, som kan bli brukt under utvikling av algoritmer og metoder relatert til både bemannede og ubemannede (autonome eller fjenstyrte) maritime overflatefartøy.

Del to omhandler bruk av MPC for autonome skip, ved å minimere koster basert på konseptet risiko. Kapittel 3 demonstrerer en konseptutprøving for en MPC-basert tilnærming for autonom baneplanlegging, risikobevisst beslutningstaking, og dynamisk sanntidshåndtering av nødssituasjoner. Kapittel 4 utvikler en metodologi som transformerer resultatene av en systemteorietisk prosessanalyse (STPA) i baneplanleggingsproblemet med anti-grunnstøting til matematiske likninger og ulikheter, som kan bli formulert og systematisk strukturert som et optimalingseringsproblem som vil løses numerisk av MPC-metoden.

For å kunne utnytte både ikke-kontinuerlige og diskrete variabler i begrensningene kostfunksjonen som skal minimeres, tar siste del for seg den prøvetakingsbaserte metoden PSO som en alternativ tilnærming til MPC. Kapittel 5 demonstrerer en dynamisk og risikobevisst metode for omplanlegging av banepunkter, som resulterer i en oppførsel analog til MPC-metoden. Kapittel 6 utnytter fleksibiliteten i PSO ved å kombinere modellbaserte prediksjoner for autonom maskinerihåndtering (AMM) med den prøvetakingsbaserte risikobevisste metoden for omplanlegging av baner. Her erstatter i prinsippet PSO ulineær programmering i planleggingsdelen av problemet, men blir likevel brukt i en prosedyre med dynamisk horisont likt som i en MPC-tilnærming. Dette øker det autonome navigasjonssystemets (ANS) evne til å i en større grad minimere drivstofforbruk og risiko gjennom kombinert AMM og risikobevisst baneplanlegging.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU), Trondheim.

This work has been conducted at the Department of Engineering Cybernetics and Centre for Autonomous Marine Operations and Systems NTNU, as part of the ORCAS project. Professor Tor Arne Johansen from the Department of Engineering Cybernetics was the main supervisor, and Professor Ingrid Bouwer Utne and Asgeir Johan Sørensen from the Department of Marine Technology were co-supervisors. The work was funded by the Research Council of Norway (grant number 280655), Kongsberg Maritime and DNV GL, and Centre for Autonomous Marine Operations and Systems (AMOS) at NTNU funded by the Research Council of Norway (grant number 223254).

Acknowledgments

This thesis would not have come into existence without the inexhaustible and consistent support and patience of a significant group of people dear to my heart.

My main supervisor Tor Arne has inspired and encouraged me to push on and avoid any and all grounding obstacles through many years with constant exceptional and personal guidance, and for that I would like to express my deepest gratitude. I would also like to thank my co-supervisor Ingrid for our essential and rewarding counseling sessions during my stay in Trondheim, as well as the continued support throughout a stormy sea of lengthy e-mails. There were times when life made sure to test my commitment, and you both invariably and promptly responded with considerate and reassuring words which always kept me floating through the rocky shallows. My co-supervisor Asgeir also served as a fundamental driving force for the project as a whole, and I thank you all for making all these years truly worthwhile.

I would furthermore like to thank my colleagues in the ORCAS project, Børge, Thomas and Tobias for all of their joyful encouragement and valuable inputs across a multitude of project meetings, classes, seminars and writing sessions. The countless hours we spent together will be remembered as a big mess of liveliness, hard work, priceless bad humor, and great pride.

The importance of my two closest childhood friends, Joakim and John, may not be overstated. This is just the latest note on our shared story, as we have powered through both master's degrees and PhD's together. You have once again continued to support me during my candidate period - as you have helped me progress and improve during most of my existence. For this, I'm extremely grateful.

Moreover, profound thanks must also be issued to those who provided me with genes and a wonderful upbringing; my parents Verner and Hildegunn, and my little sister Fride. You are still only a call away whenever I get lost, and I thank you for your support all of my years.

Lastly, I would like to thank my wife and best friend Nina. Words are insufficient to describe the fire you have ignited in my core. Your unconditional love and support motivates me to better myself every day, and I hope my work these last years may help provide for our new family. This thesis is dedicated to you.

Contents

Abbreviations

2D	two-dimensional
3D	three-dimensional
AMM	autonomous machinery management
AMMS	autonomous machinery management system
ANS	autonomous navigation system
APF	artificial potential fields
API	application programming interface
ASV	autonomous surface vessels
BBN	Bayesian belief network
CC	controller constraints
COLREGs	the International Regulations for Preventing Collisions at Sea
CSV	comma-separated values
DG	diesel generator
DoF	degrees of freedom
DP	dynamic positioning
EA	evolutionary algorithms
ENC	electronic navigational charts
ETA	estimated time of arrival
FGDB	Esri File Geodatabase
GIF	graphics interchange format
GIS	geographic information systems
HIS	hydrographic information systems
HSG	hybrid shaft generator
IMO	International Maritime Organization
IPOPT	Interior Point OPTimizer
LOA	length overall
LoA	level of autonomy
LOPP	loss of propulsion power
LoS	line-of-sight

LQR	linear-quadratic regulator
MASS	maritime autonomous surface ships
MCR	maximum continuous rating
ME	main engine
MEC	mechanical
ML	machine learning
MPC	model predictive control
MSO	machinery system operation
MUMPS	MUltifrontal Massively Parallel sparse direct Solver
NE	North-East
NLP	nonlinear programming
NTNU	Norwegian University of Science and Technology
OCP	optimal control problem
ORCAS	Online Risk management and risk Control for Autonomous Ships
ORE	online risk estimation
PhD	Philosophiae Doctor
PID	proportional–integral–derivative
PSO	particle swarm optimization
PTI	power take in
PTO	power take out
PyPI	Python Package Index
ROC	remote operations center
RPN	risk priority numbers
RRT	rapidly-exploring random trees
SAS	situational awareness system
SC	safety constraints
SCA	supervisory control actions
SI	safety inequalities
SRC	supervisory risk control
STAMP	Systems-Theoretic Accident Model and Processes
STPA	System-Theoretic Process Analysis
SV	slack variables
TTG	time-to-grounding
UCA	unsafe supervisory control actions
USV	unmanned surface vessel
UTM	Universal Transverse Mercator
WP	waypoint

Chapter 1

Introduction

"I just want the future to happen faster. I can't imagine the future without robots."

— Nolan Bushnell

1.1 Motivation

The maritime industry is subject to a technology and business transition towards an increased level of autonomy (LoA). This is motivated by the prospects of lower manning or unmanned ships supported by onshore operational centers to reduce construction and operational cost, as well as improved safety and environmental impact. This thesis is part of the project Online Risk management and risk Control for Autonomous Ships (ORCAS), which endeavors to develop novel technological solutions for online (real-time) risk management and risk control of autonomous ships.

Increasing the LoA for maritime vessels is considered a powerful measure to improve safety and environmental impact at sea, and to reduce the number of human injuries and fatalities [1]. However, advancements in the technological systems onboard, the operations, and the interactions with the environment, also increase the complexity of the systems [2]. Thus, the requirements for the risk awareness and risk control capabilities of the autonomous systems grow with this increased complexity. Without proper risk-aware decision-making capabilities and safety measures in place, the risk may not be found acceptable by commercial companies, regulatory bodies, or the public.

LoA can in general be characterized as a set of metrics that describe detailed aspects of an autonomous system and operation, including operator dependency, communication structure, human-machine interface, a dynamic online risk management system, intelligence, planning functionalities, and mission complexity.

They are defined by the International Maritime Organization (IMO) as [3], [4]:

1. *Degree one* | **Ship with automated processes and decision support**
   Seafarers are on board to operate and control shipboard systems and functions. Some operations may be automated and at times be unsupervised but with seafarers on board ready to take control.
2. *Degree two* | **Remotely controlled ship with seafarers on board**
   The ship is controlled and operated from another location. Seafarers are available on board to take control and to operate the shipboard systems and functions.
3. *Degree three* | **Remotely controlled ship without seafarers on board**
   The ship is controlled and operated from another location. There are no seafarers on board.
4. *Degree four* | **Fully autonomous ship**
   The operating system of the ship is able to make decisions and determine actions by itself.

Industry leaders within classification and specifications for compliance with regulatory frameworks state that for an autonomous system, the action planning or decision-making is performed by an algorithm based on necessary condition detection and analysis, as well as a predefined ship mission and a set of rules [5]. Established navigational rules such as the International Regulations for Preventing Collisions at Sea (COLREGs) do however not cover every possible situation, and the complete set of rules must consequently be formulated (or learned, e.g. through machine learning (ML)) such that any circumstance, commonly anticipated and unexpected alike, may be handled safely by the autonomous system. Moreover, recent scoping by the IMO to assess the way forward for regulating maritime autonomous surface ships (MASS), included clarifying core terminology related to LoA [4]: A key issue with respect to addressing the functional and operational requirements of the remote operations center (ROC) was identified, which subsequently is directly dependent on explicit and well-defined capabilities of the onboard systems of the autonomous ships commanded by the ROC. In this context, the methods presented in this thesis contribute through the development, demonstration and discussion of dynamic online risk management and improved onboard risk-aware decision-making for MASS, for the purpose of achieving LoA Degrees three and four in the future.

Advanced control systems, such as ANS and path or voyage planners, introduce new types of failures, due to this increased complexity and unforeseen interactions in system design and functionality. If personnel are not onboard to operate, the ship needs to have safe and reliable onboard control systems to be able to maneuver safely in the seaway. Shutting down and re-mobilizing the ship operation due to hazards caused by automatic or autonomous systems are not economically viable, nor acceptable from a risk perspective. Increased LoA in complex maritime operations may additionally support the human operator in supervision and decision-making, and reduce human workload. With reduced human operator intervention, and eventually presence onboard, it is important to provide early warnings (prediction) of potential deviations outside the operating envelope to enable reconfiguration of the system. Moreover, high LoA may reduce the ability for the human operator to intervene when necessary [6]. If sufficient system integrity cannot be ensured, highly autonomous and unmanned ships will not be realized on a broader scale.

The theme of this thesis as defined by the ORCAS project is supervisory risk control (SRC), focused on frameworks for online decision-making under uncertainty. Though hazardous events for autonomous ships in general include technical failures, mal-operation, loss of position, fires, extreme weather conditions, malicious attacks, hijacking, collision, and grounding, this thesis explicitly deals with grounding avoidance and (proactively) handling unexpected machinery faults that lead to loss of propulsion or steering capabilities in challenging scenarios. The over-arching goal is to propose a general formulation of mission objectives and constraints for the operation of autonomous ships, for online high-level decision-making and consequence analysis. Specifically, the decision problems include modifications to the mission plan or route and the selection of operational modes, in order to mitigate risk. Numerical implementations of the methods based on simulation case studies considering automatic sailing systems and propulsion control of autonomous ships are subsequently used to evaluate the developed algorithm's performance, with respect to uncertainty from various weather conditions, unexpected faults, overall mission requirements, and navigation hazards such as grounding obstacles.

1.2 Literature review

This section presents an overview of the most relevant research on risk-aware decision-making and control of autonomous ships, to provide sufficient background on which the research questions may be formulated. Each subsequent chapter in this thesis will more specifically address research efforts related to each topic.

Risk management is essential for robust decision-making, and for safe and cost-efficient design and operation of complex systems. With this in mind, SRC considers autonomous systems with risk management capabilities [7]. A systematic control theoretic approach to decision-making and control under uncertainty is to use optimization to determine optimal control actions, and to predict their consequences through model-based simulations or digital twins [8]. When the control decisions are updated more or less continuously based on new information received in real time, the implementation is usually referred to as MPC. It is currently a widely used and powerful control technology that has been highly successful in the optimization and automatic control of advanced industrial systems [9]. Moreover, MPC facilitates the use of a nonlinear dynamic vehicle model, including environmental forces, hazards and operational constraints for SRC, as well as objectives in terms of a cost function with constraints in an optimization problem. It is an attractive method for optimizing safety, efficiency, and emissions, and has been successful in the process industries during recent years due to its ability to mathematically capture the specifications, performance objectives, and constraints of the control system in a unifying framework.

However, the system- and application-specific data is greatly dependent on specifications of mission success criteria, acceptable risks and regulatory requirements, as well as multiple uncertain factors and parameters. The selection of the control system parameters and objectives is consequently a challenging task, and the choices made with respect to these uncertainties and varying requirements strongly influence the performance of the system. A set of scenarios may in this regard be used to represent typical realizations of the uncertain parameters and factors, and is exemplified in a recent application of this approach in the context of collision-avoidance for autonomous ships [10]. It is proposed that this approach can be adapted to more general automatic sailing systems for autonomous ships.

Another significant challenge is how to appropriately and accurately structure and define the control problem such that the criteria for acceptable risk, mission success, and regulatory requirements are met. STPA is a relatively new hazard analysis technique based on an extended model of accident causation, which focuses on system functions and couplings, rather than component failures [11]. The assumption that "safety is a control problem" implicitly reflects dynamic properties of systems, but has not been applied for risk management of autonomous systems and online risk control. Its feasibility for verification of maritime systems has however been demonstrated [12], [13], and STPA is thus considered an appropriate base approach which may be utilized to standardize numerical optimal control formulations.

In contrast to the gradient-based MPC approach, the sampling-based PSO method is able to deal with non-smooth or mixed-integer control and planning problems utilizing discrete variables [14]. This is especially useful when considering more complex problems such as navigation in large non-convex environments involving grounding obstacles. Moreover, the cost or fitness function may in this way be defined using disjoint or non-smooth terms which would otherwise prove intractable for numerical optimization. Recent works related to anti-grounding for autonomous surface vessels have explored usage of PSO for distance-based path or waypoint planning, energy efficiency, grounding risk, and safety [15]–[19]. It is nevertheless proposed that there is significant potential for improvement in the achieved control and decision-making performance, if the control problem is enhanced to include simulated grounding risks using ship dynamics, machinery mode management considerations, and more effective utilization of well-structured ENC data. This sampling-based PSO approach may ultimately be used in a receding horizon fashion similar to the method employed in MPC, and even to utilize independent MPC solvers as internal sub-processes for PSO state predictions and optimization in future works.

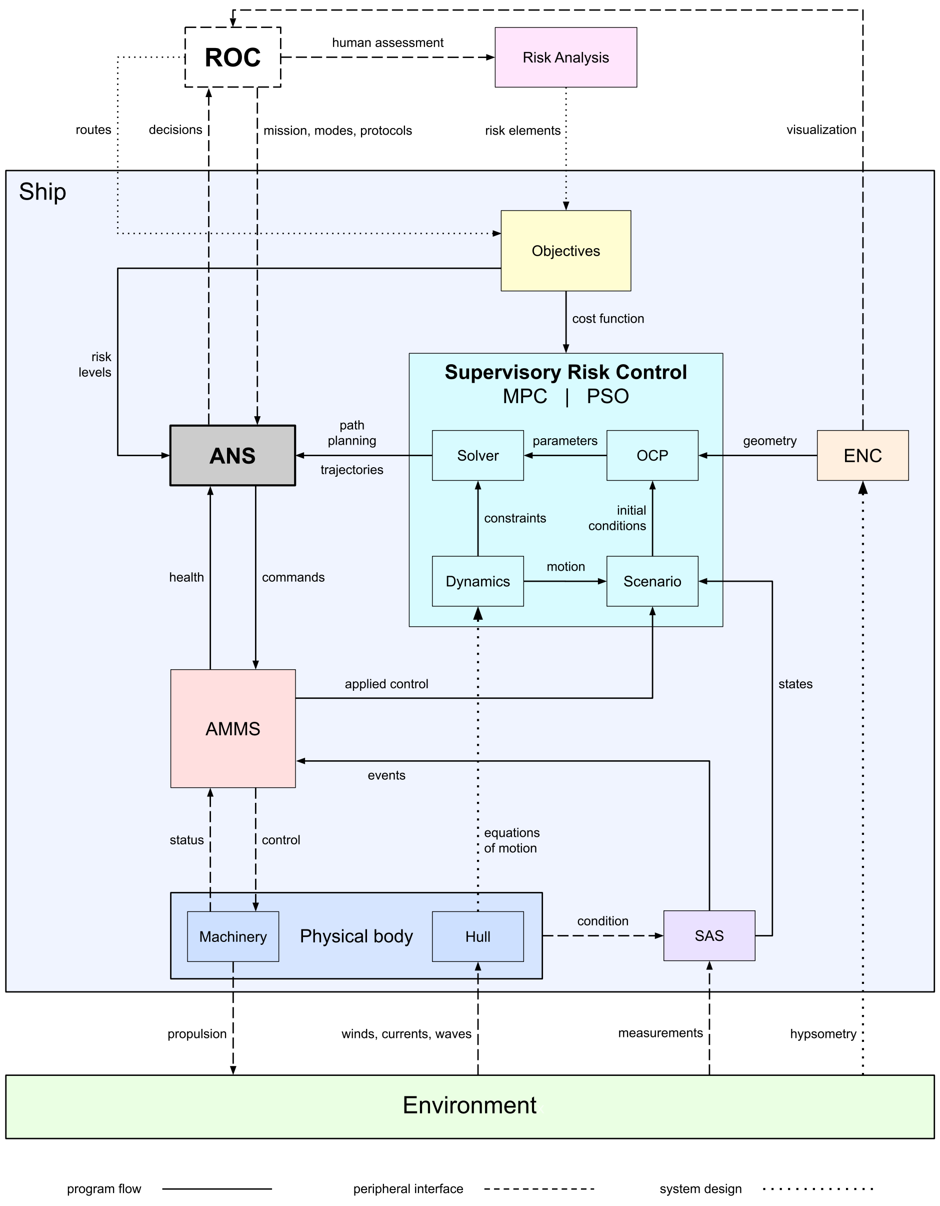


Figure 1.1: Overview of the scope considered in this thesis, which is focused on the SRC and its inputs (objectives, ENC and risk awareness).

1.3 Research questions

The topics of this thesis are risk-awareness, decision-making, path planning, and optimal control for autonomous ships. The scope of the thesis with respect to considered systems and interfaces is visualized in Figure 1.1. The top-level is the ROC, from which humans supervise the autonomous sailing process as planned and carried out by the ANS. Additionally, the ROC performs and provides risk analysis and routes to the Objectives module, which defines the cost function to be supplied to the SRC path planner, which is the main focus of this thesis. The SRC bases its costs on input geometry from ENC, states from a situational awareness system (SAS), and the equations of motion of the physical system. Lastly, the control commands issued by the ANS are applied to the ship machinery through the autonomous machinery management system (AMMS), ultimately driving the ship safely towards its given target destination.

During the initial work on introductory topics within the fields of path planning and optimal control for surface vessels, there was noted an apparent lack of standard development tools or simulation environments in which one can quickly establish e.g. an ownship in a chosen scenario for rapid prototyping, experimentation and evaluation. It seemed like fellow colleagues all started from scratch and developed their own software platforms designed for their specific environment or ENC needs in parallel, despite the evident similarities and strongly connected research topics between them.

This lack of available software was also reflected in the literature. Although recent research utilizing ENC data has made significant progress on the topics of decision support [20]–[22], path planning [23]–[25], and collision avoidance in restricted waters [26]–[29], no comprehensive or unified platforms for open-source work with ENC currently exists. Thus, considerable additional efforts were put into the development of an open-source application programming interface for ENC which, if shared, could help accelerate research on autonomous ships. The accompanying research question is defined as follows:

RQ 1 How can the utilization and visualization of ENC data be made fast and user-friendly in the research and development process for autonomous ship control and decision-making?

Though the available literature shows that MPC is a promising method for heading control, path following and collision avoidance [30]–[33], the systems discussed had in general assumed standard conditions or a limited scope within which they were allowed to make autonomous decisions, like e.g. path following during the transit phase of an autonomous ferry [34]. However, little research currently exists on the topic of autonomous planning which explicitly considers the ability to dynamically handle uncertain (weather) disturbances and unexpected faults. Thus, the second research question is proposed:

RQ 2 How can autonomous ships **a)** be proactively controlled such that grounding hazards are predicted and avoided on a voyage while subject to uncertain disturbances, and **b)** be prepared for unexpected machinery faults such as loss of steering or propulsion?

It is furthermore useful to define a systematic approach which can formulate and structure the resulting objectives or constraints of established risk analysis methods such as STPA [35] safely into a numerical optimal control problem:

RQ 3 How can the results of qualitative risk analysis methods such as STPA be utilized and systematically structured into a numerical optimal control problem for autonomous ship navigation, such that grounding hazards are identified, mathematically formulated and avoided through numerical optimization?

Due to PSO being based on sampling rather than gradient search [14], the approach is suitable for ENC, which in general lead to non-smooth cost functions. Several works on utilizing PSO for MASS with respect to static obstacles were found in the literature. However, these either do not consider the more complex risks associated with close proximity to (grounding) obstacles [15], [16], or may be structurally improved with regards to the fitness (cost) function and utilization of ENC data [17]–[19], which leads to the research question:

RQ 4 How can the planned routes generated by established voyage planners be modified to give a navigation path during complex real-time transit conditions, such that unexpected hazards and changing environments are accounted for en route?

Finally, the inherent versatility of using sampling-based PSO may be exploited e.g. by taking into account additional sub-systems such as AMM during real-time path following for autonomous ships. Previous work [36] proposes a model-based control system which automatically selects the optimal machinery mode in order to minimize grounding risk and fuel consumption as a stand-alone approach. Thus, it is proposed that improved fuel efficiency and risk-aware decision-making may be achieved by combining risk-based online route re-planning with discrete optimization variables for real-time machinery management modes and route selection:

RQ 5 How can model-based predictions for ship dynamics and onboard machinery management systems be combined with a sampling-based path planning algorithm and established path following algorithms in order to reduce overall risks and expected mission costs?

1.4 Contributions overview

This thesis consists of four parts, based on five peer-reviewed publications.

Part I presents one article on the development of an API for ENC in Chapter 2.

Part II is comprised of Chapters 3 and 4, which considers MPC and STPA for risk-aware autonomous ship emergency management.

Part III deals with PSO for risk-aware autonomous route re-planning and AMM in Chapters 5 and 6.

Part IV offers concluding remarks in Chapter 7.

The rest of Chapter 1 presents an overview of the contributions of each publication.

Chapter 2 - Application programming interface

[37] **S. Blindheim** and T. A. Johansen, "Electronic Navigational Charts for Visualization, Simulation, and Autonomous Ship Control," *IEEE Access*, vol. 10, pp. 3716–3737, 2022. DOI: `https://doi.org/10.1109/access.2021.3139767`

This chapter presents an open-source ENC visualization and manipulation API implemented in Python, with emphasis on accessibility and simplicity for the purpose of providing an accessible and open-source API for displaying and managing spatial bathymetry or other related sea-faring data for research and development. The current version of the package provides tools for displaying marine polygon data such as ships, ocean depths, reefs, and shallows, using the transverse Mercator projection. Additionally, polygon- and point-based transformation and calculation methods for application development based on spatial geometry, path planning and numerical optimal control are implemented. Usage of the spatial methods are demonstrated by examples involving high-level path or trajectory planning, optimization, and assisted decision-making for autonomous and remote-controlled ships.

Chapter 2 contributes towards RQ 1 by providing an open-source API which enables researchers to access, visualize and utilize ENC data to more efficiently develop methods for autonomous ships in future works.

Chapter 3 - Autonomous ship emergency management

[38] **S. Blindheim**, S. Gros, and T. A. Johansen, "Risk-Based Model Predictive Control for Autonomous Ship Emergency Management," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 524–14 531, 2020, 21st IFAC World Congress. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1456

In this chapter, a dynamic risk-based decision-making algorithm based on MPC is constructed through the use of heuristic objectives, capable of planning suitable vessel trajectories in emergency situations. Nonlinear programming using the direct multiple-shooting method implemented with the CasADi framework is considered [39], and the resulting control performance of several emergency scenarios is analyzed using simulation. The developed MPC algorithm with independent risk cost terms is capable of both generating suitable trajectories below a certain risk threshold, and to engage the safety systems appropriately.

Chapter 3 contributes towards RQ 2 by demonstrating a dynamic risk-based navigation algorithm with inherent emergency management capabilities, which avoids grounding hazards both during normal conditions and when machinery faults unexpectedly occurs, such as impaired steering and loss of propulsion.

Chapter 4 - Risk-based predictive supervisory control

[40] **S. Blindheim**, I. B. Utne, and T. A. Johansen, "Risk-Based Supervisory Control for Autonomous Ship Navigation," *Journal of Marine Science and Technology*, pp. 1–25, 2023. DOI: 10.1007/s00773-023-00945-6

This chapter proposes a novel method for transforming the results of qualitative risk analysis into a numeric optimal control problem for autonomous ship navigation. STPA considers safety as a control problem, which makes it feasible for revealing hazards related to autonomous systems. Such hazards should be considered in the design of control algorithms and when optimizing decisions during operations to improve safety. One of the challenges with STPA, however, is that it only brings forward qualitative results, which are impossible to use directly for MPC. Thus, general principles for how the results from STPA can be transformed into a quantitative and computationally tractable optimization problem are suggested, solved by a MPC-based decision-making algorithm for autonomous navigation. The proposed method is demonstrated and evaluated by simulating an autonomous ship navigating in a coastal environment.

Chapter 4 contributes towards RQ 3 by developing a step-wise methodology which takes the results of an STPA and transforms them into mathematical equations and inequalities, which subsequently are structured into an optimal control problem to be solved numerically by an MPC approach.

Chapter 5 - Dynamic risk-aware path planning

[41] **S. Blindheim** and T. A. Johansen, "Particle Swarm Optimization for Dynamic Risk-Aware Path Following for Autonomous Ships," *IFAC-PapersOnLine*, 2022, IFAC CAMS. DOI: https://doi.org/10.1016/j.ifacol.2022.10.411

This chapter presents the use of PSO for dynamic risk-aware path planning and following during autonomous surface navigation in a maritime environment with polygonal grounding obstacles. Although recent research on control and local or global path planning for MASS is considerable, few consider the concept of risk during dynamic path following along a pre-planned path. The proposed method introduces risk-based terms in the PSO fitness function for dynamic (online) adjustment or re-planning of intermediate waypoints during path following of a pre-planned route, where the control of the vessel in this work is left to a standard line-of-sight (LoS) proportional–integral–derivative (PID) controller. The results are compared to the performance of an analogous implementation of risk-aware MPC using a gradient-based solver. The suggested method yields adequate performance similar to that of the MPC algorithm.

Chapter 5 contributes towards RQ 4 by presenting a more flexible online path re-planning method utilizing PSO, which may employ the use of established path following or guidance algorithms after re-planning. This alternative method based on PSO is compared to the previous MPC approach of Chapter 3 and Chapter 4, and demonstrates analogous behavior, enabling the use of more complex (e.g. combined continuous and discrete) cost functions in future works.

Chapter 6 - Autonomous planning and machinery management

[42] **S. Blindheim**, B. Rokseth, and T. A. Johansen, "Autonomous Machinery Management for Supervisory Risk Control Using Particle Swarm Optimization," *Journal of Marine Science and Engineering*, vol. 11, no. 2, p. 327, 2023. DOI: https://doi.org/10.3390/jmse11020327

In this chapter, a novel methodology combining risk-based optimal control and path following with AMM for MASS navigation and SRC is presented. Specifically, a risk-aware PSO scheme utilizes "time-to-grounding" predictions based on weather data and ENC to simultaneously control both the ship's motion as well as the machinery system operation (MSO) mode during transit. The proposed ANS is comprised of an online receding horizon control strategy that uses a PSO approach from previous works, which produces a dynamic risk-aware path with respect to grounding obstacles from a pre-planned path, subsequently given as the input to a line-of-sight guidance controller for path following. Moreover, the MSO mode of the AMMS is simultaneously selected and assigned to explicit segments along the risk-aware path throughout the receding horizon, which effectively introduces an additional safety layer as well as another dimension for risk or resource minimization into the optimization scheme. The performance of the resulting ANS is demonstrated and verified through simulations of a challenging scenario and human assessment of the generated paths.

Chapter 6 contributes towards RQ 5 by proposing an approach for combining and utilizing a model-based autonomous machinery management approach with the sampling-based and risk-aware PSO voyage re-planner method from Chapter 5 in the same cost function, which increases the ability of the resulting ANS to consider and optimize resource consumption and grounding risks through both MSO selection and route re-planning or selection.

Appendix A - Risk-based control system for autonomous ships

[43] **T. Johansen**, S. Blindheim, T. R. Torben, I. B. Utne, T. A. Johansen, and A. J. Sørensen, "Development and testing of a risk-based control system for autonomous ships," *Reliability Engineering & System Safety*, vol. 234, p. 109 195, 2023

The first appendix presents the development of a SRC with decision-making capabilities for autonomous ships, in which an online risk model (Bayesian belief network (BBN)) based on the results of STPA is used along with ENC data to evaluate dynamic operational risks. Specifically, it is a relevant demonstration of how the SeaCharts API may be utilized to process and visualize ENC data for risk-aware route planning and path following for autonomous ships.

Appendix B - Ship collision avoidance and anti-grounding

[29] **T. Tengesdal**, T. A. Johansen, T. D. Grande, and S. Blindheim, "Ship collision avoidance and anti grounding using parallelized cost evaluation in probabilistic scenario-based model predictive control," *IEEE Access*, vol. 10, pp. 111 650–111 664, 2022

The second appendix considers the implementation of a model predictive controller based on probabilistic scenarios, using parallelized cost evaluations for autonomous ship collision avoidance and anti-grounding. This chapter also demonstrates how the SeaCharts API is utilized by the MPC to consider the ship's environment based on both ENC data and structurally analogous dynamic obstacles defined as two-dimensional (2D) general polygons, using the same base framework. It is considered another apt testimony to the capabilities and adaptability of the SeaCharts package.

Part I

Electronic navigational charts

Chapter 2

Application programming interface

This chapter is based on the publication

[37] **S. Blindheim** and T. A. Johansen, "Electronic Navigational Charts for Visualization, Simulation, and Autonomous Ship Control," *IEEE Access*, vol. 10, pp. 3716–3737, 2022. DOI: https://doi.org/10.1109/access.2021.3139767

The software, methods, algorithms, and simulations were developed and implemented by S. Blindheim, under the supervision of T. A. Johansen. The first draft was written by S. Blindheim, and was revised by T. A. Johansen.

2.1 Introduction

ENC have today become the digital standard to replace printed navigational charts. Such formats allow for a range of possibilities with regards to data handling. However, the task of efficiently showing and manipulating ENC data has arguably become more challenging with increased data sizes and degrees of freedom, and solutions are largely developed on a case-to-case basis. More specifically, open API for public use are scarce. Given the limited resources for polygon-based maritime environments available today, it is apparent that there is a need for an open-source ENC API for future research and software development efforts.

2.1.1 Literature review

Active development of increasingly more advanced ENC has been steadily moving forward, since the emergence of digitally stored bathymetry data and specifications of international exchange standards for hydrographic data were formulated [44]. There are several objectives associated with the use of ENC; applications related to pure visualization for navigation purposes, geometric calculations and spatial data operations, automatic control and decision support for manned or unmanned operations such as anti-grounding and obstacle avoidance, safety or risk analysis, and route or path planning. However, there seems to be a lack of open API which sufficiently provide necessary and/or convenient tools for research and development of such systems. Though recent works have been carried out to specify and solidify forms of systematization, standardization and classification of ENC [45]–[48], API solutions for both visualization and direct spatial ENC data manipulation are still limited. The following sections present an analysis of some of the relevant resources available in the literature today.

Visualization

Norms and standards for visualization of 2D bathymetry data has been consistent with practices used for printed navigational charts, and numerous applications have emerged for various areas, such as pure visualization of ENC data [49]–[51], radar display imaging [52], [53], three-dimensional (3D) visualization of bathymetry data [54]–[56], and even endeavors on virtual or augmented reality [57], [58]. However, visualization of spatial data is merely a tool for evaluation and affirmation within research and development on applications for autonomous decision-making or decision support. Thus, the development of such stand-alone user applications or API is not sufficient for research and development within this area, and variations of such solutions are consequently not considered in this work.

Data extraction

In order to be able to utilize environment data for autonomous navigation, it is necessary to make the intrinsic spatial data contained within the ENC accessible for external applications through a programmable software interface, in addition to simultaneous environment visualization and information display. ENC may as such be utilized as direct data inputs for active decision support systems [20], and may be formulated as a vector-based architecture [21] given a spatial point- or polygon-based database from e.g. surveys [22], [59].

Path planning

Decision support systems using operational parameters related to mission objectives and spatial hydrographic ENC data are mainly concerned with path or trajectory planning in a maritime environment. Path planning in accordance with mission objectives is a known problem, and as such there exists a rich collection of research on efficient path planning for unmanned surface vessel (USV), such as long-distance mission planning [25], dynamic or adaptive re-planning [23], [24], control based on objective optimization and vector fields [60]–[64], and optimal paths using the established A* algorithm [65]–[67].

It is noted that path planning in the context of maritime operations in itself is only a tool commonly used to achieve higher levels of autonomy. As such, path planning for this purpose is combined with (sub-)systems for situational awareness (i.e. structured spatial data and prediction or simulation capabilities) based on ENC as well as sensor data, to be used in schemes for decision-making and risk analysis. However, the results of this research indicate some of the functionality required of an adequate ENC API. It is important that objective optimization is facilitated by making relevant ENC data available to the algorithm, such as depth values and distance calculations. Moreover, the prevalent prominence of methods based on vector fields suggests that it is desirable to include a sufficient range of vector-based methods in the ENC API, as well as to preserve the ability for scalable resolutions of the data sets loaded by the system. If methods based on raster images or spatial grids is used for decision-making, such grids may be directly constructed by interpolation between coordinate points through sampling along the spatial axes.

Autonomous navigation, risk and safety

The overarching main objective of ANS or decision support systems is to increase maritime safety [68], [69] as well as efficiency, with respect to risks and/or expected cost analyses. Additionally, effective visualization of important risk factors for humans involved both during the design, planning and operational phases of maritime navigation is an integral part in achieving this goal, further highlighting the need for practical and flexible ENC API for the purpose of additional information overlay visualization. Moreover, risk analysis and risk management of (semi-) autonomous vessels are of increased importance with the development of unmanned operations [2], [70]. With higher levels of autonomy, the requirements for data quality and performance demanded by the involved decision-making systems increase significantly, and as such must be provided by the underlying ENC API.

Autonomous obstacle and collision avoidance

Path planning and autonomous obstacle or collision avoidance for complex maritime environments remains a challenging problem [71]. In order to further advance ANS algorithms for autonomous surface vessels (ASV), decision-making should include considerations related to environmental disturbances, identification and dynamic or adaptive planning with respect to static and dynamic obstacles, COLREGs compliance, and utilization of vessel safety domains [72]. These systems must be highly flexible and robust, and be able to handle both long-term and short-term (reactive) planning for obstacle avoidance based on reliable information, i.e. sensors and detailed bathymetric ENC data constructed as polygons [26], [73].

Recent example applications include traffic monitoring with respect to collision detection and risk assessment [74], obstacle tracking and reactive collision avoidance based on sensor fusion [75], and the development of adaptive safety domains for identification of grounding risks [28] and collision avoidance during vessel interactions (COLREGs) [76]. Thus, it is clear that the demand for open and accessible ENC software and resources for real-time simultaneous visualization and efficient computation of hydrographic data is evermore increasing.

Current available API solutions

In general, web searches for ENC or hydrographical API yield few relevant results, and there currently exists only a handful of ENC API today. Moreover, most of these are proprietary or closed-source, making it exceedingly difficult to identify individual resources suitable for feature-by-feature comparison between these, general desired API capabilities, and the solution presented in this work. Due to this fundamental lack of accessibility and transparency, such alternatives are inherently considered inadequate for open research and development. The available open-source software for ocean mapping is currently varying in quality and functionality, and no single API is identified as an all-in-one solution [77].

Pydro [78] is identified as the most promising candidate solution supporting the International Hydrographic Oranization S-57 standard [79]. However, Pydro is not strictly a pure API for Python programming, but a suite of software tools built for hydrography and cartography. The software is focused on enhancing and automating the hydrographic workflow from data acquisition to hydrographic survey review and nautical chart compilation [80] which leaves the work of visualization and high-level data abstraction to the researcher. Additionally, the application is currently only supported in Windows, and may not be repackaged or redistributed due to license restrictions. These are significant constraints with respect to open research and development, and Pydro is as such considered unsuitable for this purpose.

The conclusion of the literature review is thus that no truly open-source API packages or ENC solutions for open research and development currently exists.

2.1.2 Problem and contribution

Unambiguously structured and visualized spatial data are needed for clear interpretation and efficient computation in autonomous ENC-based maritime navigation. This work addresses the research question: How can the utilization and visualization of ENC data be made fast and user-friendly in the research and development process for autonomous ship control and decision-making? It is proposed that an open-source Python-based API may serve as a framework and/or a valuable support tool for further development of hydrographic information systems (HIS), simulation or control algorithms for ANS based on ENC.

Thus, the purpose of this work is to provide an open and user-friendly API package intended for fast and easy prototyping during software development such that researchers more quickly may get to work on development on (autonomous) navigation systems, rather than spending time on writing basic geographic information systems (GIS) functionality such as polygon manipulation or simply displaying a maritime environment. The package is focused on ease of use and fast prototyping capabilities through high-level spatial computation of polygon data and flexible visualization methods, and may play a part in facilitating more productive and targeted research on the topic of autonomous ships, by allowing the researcher to concentrate more on other aspects than application programming of a baseline simulation environment for testing purposes. Intelligent design of the ENC package may also lead to significant improvements to computation speeds for dynamic path planning and simulation.

An overview of the contributions included in the proposed API is listed in Section 2.2, and shown as implemented class instance methods in Table 2.1.

2.1.3 Scope

Only 2D visualization is considered, in accordance with the goal of a simple and comprehensible interface. As marine environments inherently are concerned with depth-related data and representations, the core objective is thus to visualize objects and ocean depth through the use of distinctly colored 2D shapes. Moreover, all given coordinates are projected onto a locally flat plane via the Universal Transverse Mercator (UTM) coordinate system, subsequently allowing all polygon or other shape-based operations performed by the application to assume a flat 2D plane. Significant efforts in this work are targeted toward the development of clear and uncomplicated methods for straightforward geometric polygon manipulation and operations for spatial depth data sets, such as polygon simplification, intersections, unions, dilution, erosion, convex hulls, and distance and bounding box calculations. Finally, several example algorithms are defined and discussed in order to showcase usage of the implemented methods of the ENC API package, within the fields of autonomous path planning, collision avoidance, and online risk management.

2.1.4 Installation and usage

Python is chosen as the preferred programming language for the development of an open-source API for elegant visualization and straightforward manipulation of spatial data, given its design philosophy with regard to its readability, object oriented structure and readily available libraries for management of HIS resources.

The package is given the name *SeaCharts*, and may be installed through the Python Package Index (PyPI) [81] by executing the command **pip install seacharts**, making it readily available through the standard **import** statements in Python files within an environment. Interested readers may find maintained links to the SeaCharts homepage, source code repository, documentation and usage instructions at pypi.org/project/seacharts.

2.1.5 Design and structure

Table 2.1 presents a class diagram of the main class **ENC** to be instantiated by the user, from which all other processes are initiated. An instance of this main class serves as a single-point interface, which is initialized by arranging and encapsulating polygonal features extracted from external Esri File Geodatabase (FGDB) files as local shapefiles [82]. These files are subsequently read and fed into a private **environment** variable containing all available data computation methods, accessed by the user through the top-level bathymetry attributes (**land**, **shore**, **seabed**) and direct utilization of geometric operations provided by the *Shapely* library [83].

ENC
- environment: Environment - display: Display + supported_crs: **str** + supported_layers: **str** + land: Land + shore: Shore + seabed: **dict**
+ ENC(size: **tuple**, origin: **tuple**, center: **tuple**, buffer: **int**, tolerance: **int**, layers: **list**, depths: **list**, files: **list**, new_data: **bool**, raw_data: **bool**, border: **bool**, verbose: **bool**, multiprocessing: **bool**): ENC + show_display(duration: **float**): **None** + refresh_display( ): **None** + close_display( ): **None** + save_image(name: **str**, scale: **float**, extension: **str**): **None** + colorbar(arg: **bool**): **None** + dark_mode(arg: **bool**): **None** + fullscreen_mode(arg: **bool**): **None** + add_vessels(args: **list**): **None** + clear_vessels( ): **None** + add_ownship(easting: **int**, northing: **int**, heading: **float**, hull_scale: **float**, lon_scale: **float**, lat_scale: **float**): **None** + remove_ownship( ): **None** + add_hazards(depth: **int**, buffer: **int**): **None** + draw_arrow(start: **tuple**, end: **tuple**, color: **str**, width: **float**, head_size: **float**, thickness: **float**, edge_style: **str**): **None** + draw_circle(center: **tuple**, radius: **float**, color: **str**, fill: **bool**, thickness: **float**, edge_style: **str**): **None** + draw_line(points: **list**, color: **str**, width: **float**, thickness: **float**, edge_style: **str**): **None** + draw_polygon(geometry: **Any**, color: **str**, interiors: **list**, fill: **bool**, thickness: **float**, edge_style: **str**): **None** + draw_rectangle(center: **tuple**, size: **tuple**, color: **str**, rotation: **float**, fill: **float**, thickness: **float**, edge_style: **str**): **None**

Table 2.1: Class diagram of the main SeaCharts ENC module in Python.

The **display** variable may if desired be utilized through the provided class methods to display user-selected spatial features produced by the environment variable, serially or in parallel with data computations performed by a possible third-party navigation or optimization program. Users of the SeaCharts package are advised to refer to its *Readme* file in the maintained repository for detailed usage instructions. The following sections nevertheless provide an overview of the currently available API methods and functionality, while leaving most of the programming-related details to the code documentation.

2.2 Methods

This section presents the main contributions of this work. The initialization and creation of an ENC interface instance is firstly described by its data extraction, polygon handling and feature extraction processes, followed by demonstrations of most of the user methods available in the API package as seen in Table 2.1, at the time of writing.

The constructor of the **ENC** class is denoted at the top of the third compartment. Its input arguments are used during initialization, and will be thoroughly discussed in Sections 2.2.1 to 2.2.3. Built-in Python types are denoted in bold.

The methods related to the display are summarized as follows: The display utility methods, namely the **show_display**, **refresh_display**, **close_display** and **save_image**, are used to show and save images from the interactive display during runtime. Additional visual configuration settings are available through the **colorbar**, **dark_mode** and **fullscreen_mode** methods.

Next, methods for adding (drawing and maintaining references to) vessels and a controllable interactive ownship with toggleable hazardous areas within a given horizon are included through the methods **add_vessels**, **clear_vessels**, **add_ownship**, **remove_ownship** and **add_hazards**.

Lastly, opaque or transparent geometric shape overlays may be drawn onto the displayed environment through the draw-methods at the bottom of Table 2.1. Note that the **draw_polygon** method may in general be used to draw *any* polygon-based shape, the rest are provided for convenience.

2.2.1 Data extraction

The API is initialized through the creation of the single ENC instance, which reads and stores geometric shapes for both future offline re-reading and dynamic shape handling during runtime. This section illustrates how the API handles the data during this initialization process, based on any combination of the class constructor arguments of Table 2.1. For all remaining figures in this chapter, the **border** argument is set to **True** in order to produce an encompassing black border around the image edges, and the **multiprocessing** variable is set to **False** such that environment plotting is dependent on the running demonstration applications. The

verbose argument may if desired be toggled on for the purpose of information printing in the terminal during runtime.

The SeaCharts package supports loading and reading spatial data structured in the FGDB 10.0 format. For demonstration purposes, the open-source ENC data used in this work is downloaded from the Norwegian Mapping Authority, which contains 2D ENC polygons of the coast of Norway projected in *EUREF89* UTM *zone 33N*. However, any region of the world may be read and extracted as an ENC, provided a properly structured FGDB for the area in question is available.

At the time of writing, the **depth data** files for Norway may be downloaded as a whole or divided into specific county areas. These files are to be placed in a specific folder relative to the working directory of the Python application such that the Data module may reach it, as detailed in the SeaCharts package Readme. Processing of the downloaded FGDB files is performed during startup when the application is first run, or if the user has explicitly requested a manual data extraction during a subsequent run by setting the **new_data** argument value equal to **True**. During this process, the polygonal or point data files specified by the **files** argument (a list of file names) are loaded into memory and spatially filtered given a user-specified bounding box calculated from the **size** and **origin** (or **center**) arguments, as well as the depth bin groups defined by the **depth** argument (see Section 2.2.2). The specified data is subsequently written to shapefiles and stored in memory for direct access during runtime.

Figure 2.1 presents a visualization of the intended data extraction window hierarchy of the SeaCharts package. The figure shows an example view of the Norwegian counties *Møre og Romsdal* and *Trøndelag* merged together. The black border is the abstraction of any (digital) ENC downloaded or constructed for any given region of the world, and may contain one or several specific regions of spatial data such as e.g. countries or municipalities. Next, the yellow rectangle represents an arbitrary region of *extracted depth data* from the downloaded ENC in black, stored as shapefiles on the system's hard drive, as well as being available through the corresponding top-level bathymetry layers of the **ENC** class instance. This region may be considerably smaller than the initial raw data sets, and contains only the optimized features (Section 2.2.2) selected by the user during the initial FGDB file extraction phase. The orange rectangle shows the *shape handling* subregion or area for which specific polygon- or point-based computations may be performed during runtime, e.g. to extract and merge all depth layers deeper than ten meters in order to construct a (binary) sea-faring domain feasible for any particular ship. Lastly, the red rectangle represents a *dynamic horizon* subregion of the orange area, which may be used by some external path planning, collision avoidance, or simulation algorithm. This area is intended to be dynamic during external algorithm cycles and e.g. follow the ship(s) in question during a voyage, such that appropriate domains may be extracted and/or scaled for feasible optimization or planning given some real-time requirements and mission objectives. Note that the shape handling and dynamic horizon regions may be freely constructed as any arbitrary polygon based on the application for which the API is used, and may if desired be identical.

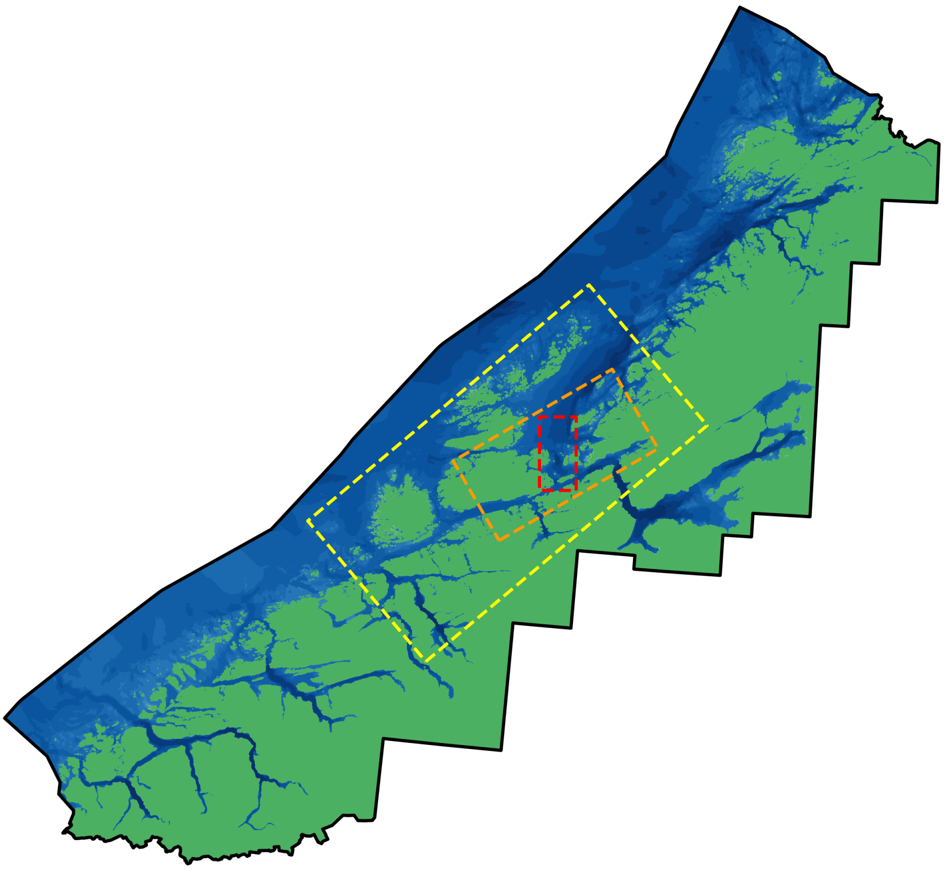


Figure 2.1: Data extraction window hierarchy of the SeaCharts package.

In addition to selecting subregions of spatial data (even across data set boundaries such as e.g. county borders), the package explicitly specifies and selects each desired spatial features or layers to be extracted from the data sets during initialization through the **layers** argument containing a list of valid layer names. If the user of the API is e.g. only concerned specifically with the seabed depth polygons included in the depth data set, all other features like docks or other marine structures are disregarded and filtered out from the constructed shapefiles. The resulting shapefile data set read by the application after the initial startup thus only contains requested data, which allows for faster performance for dynamic filtering and spatial computing during runtime. Additional polygonal data optimization or simplifications are further described in the following sections.

2.2.2 Polygon optimization

Downloaded depth data sets are generally large, and regularly contain suboptimal numbers of additional polygons due to region splits produced by some mapping algorithm or e.g. county boundaries. Thus, the SeaCharts package performs various standard GIS simplification procedures on the raw data in order to speed up runtime calculations and visualization, unless the **raw_data** argument is set to **True**. This process is included and described in this work for completeness, such that users of the package may become familiar with common GIS techniques in the context of the provided API functionality for research and development.

Figure 2.2 shows a SeaCharts visualization of the raw polygon data extracted from the downloaded FGDB files surrounding the Norwegian city of *Ålesund* (with **raw_data** = **True**), in which polygon edges are white for demonstration purposes. It is clear that the depth data polygon regions are divided into orthogonally adjacent subrectangles, creating situations in which there exists more than one polygon for any single disjoint body of land, shore, or seabed depth. In addition to the increased computational complexity introduced by this data structure, such redundant polygon definitions may also produce unexpected, undesirable or potentially incorrect results when performing numerical spatial operations, e.g. irregularities where only half an island is included for path planning purposes due to excluded polygon areas which are not properly intersected with a dynamic extraction window. Calculation performance during runtime may thus be increased significantly by merging or calculating the unions of all orthogonally adjacent polygons with depth data within the same depth bins present in the extracted data sets.

Polygon merging

Figure 2.3 presents the result after performing the *Shapely* operation **unary_union** on the region from Figure 2.2. Notice how the polygons of large areas across the white rectangular divisions have been merged together, such that no continuous regions of the same depths are split. Each seabed polygon is constructed such that it fully envelops any and all polygons with depths *deeper* than its depth value, that it might contain. Thus, the total area of all polygons of depths between e.g. 5 m and 10 m will consequently be larger than the total area of all seabed layers deeper than 10 m. Note that this behavior is specific to the SeaCharts package, and is chosen as such with the purpose of intuitive alternative views during visualization - e.g. to toggle off deeper depth layers such that the remaining encompassing and more shallow layers are still complete. All land polygons are however entirely contained within the complete set of shore polygons, for the analogous inverse reason. This layer structure facilitates an intuitive and effortless lookup interface with respect to the fundamental question of identifying which areas are safe, hazardous or simply impassable for any given ship to navigate within. If e.g. the maximum draft of a ship is 5 m, it is thus straightforward to simply extract the full seabed layer of e.g. 7 m including an additional safety depth margin.

In addition to the merged regional shapes, there are fewer different values of depth polygons shown in the resulting environment in Figure 2.3 compared to the original downloaded data in Figure 2.2 due to the user-specified depth bins, i.e. ranges for which each depth measurement is grouped and separated by, through the **depths** argument list of integers. This feature may serve as another flexible layer

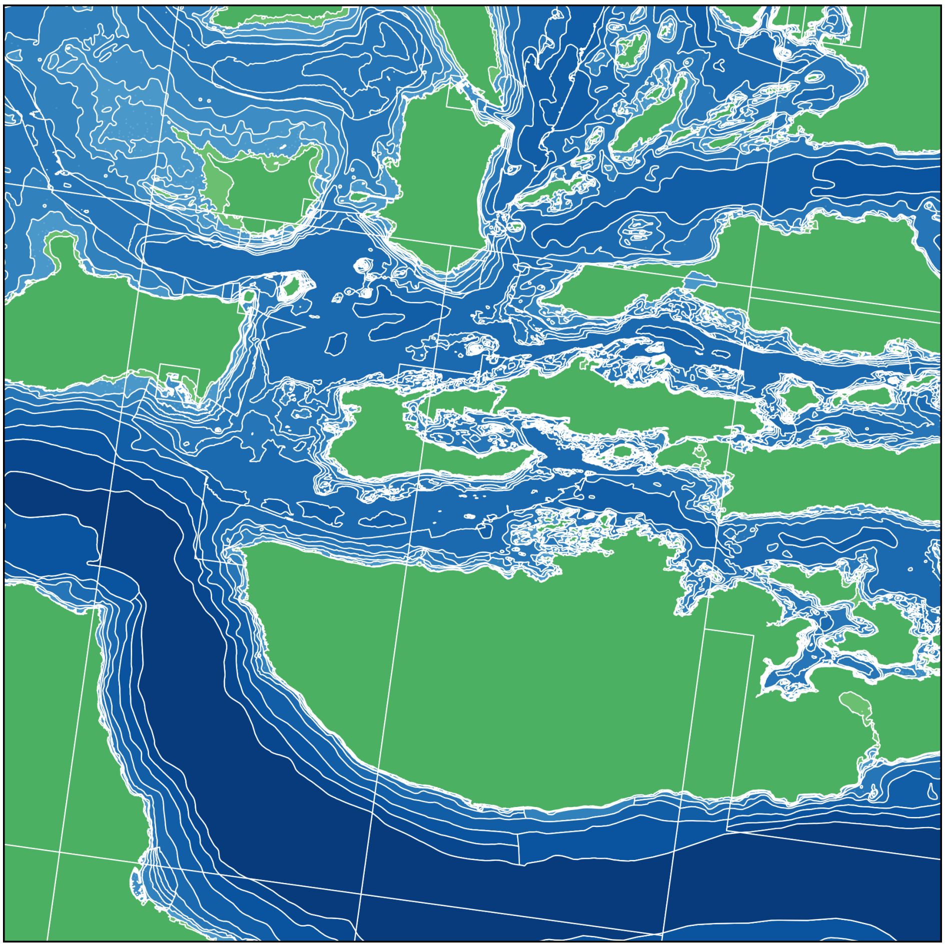


Figure 2.2: SeaCharts visualization of raw downloaded polygon data with depth bins of 5, 10, 20, 50, 100, 200, 350 and 500 m.

of data management for further computation optimization, e.g. by disregarding depth range resolutions outside the scope of a path planning problem. In Figure 2.3, depths of 5 m, 20 m and deeper than 300 m were disregarded, such that the deeper *Sulafjorden* regions shown in dark blue in the bottom are consolidated into a region of a larger range of depth values. Notice however how the different resolutions present in the original raw data may produce polygon artifacts at the boundaries between the resolution partitions, e.g. along the southern coast of the *Sula* island/peninsula, marked by a red rectangle: The gradually deeper depth contours in Figure 2.2 reveals a noticeable resolution discrepancy edge where two depth bins are merged into one. These unavoidable artifacts emerge from the use of several different spatial resolutions in the downloaded data set, and should be treated with care.

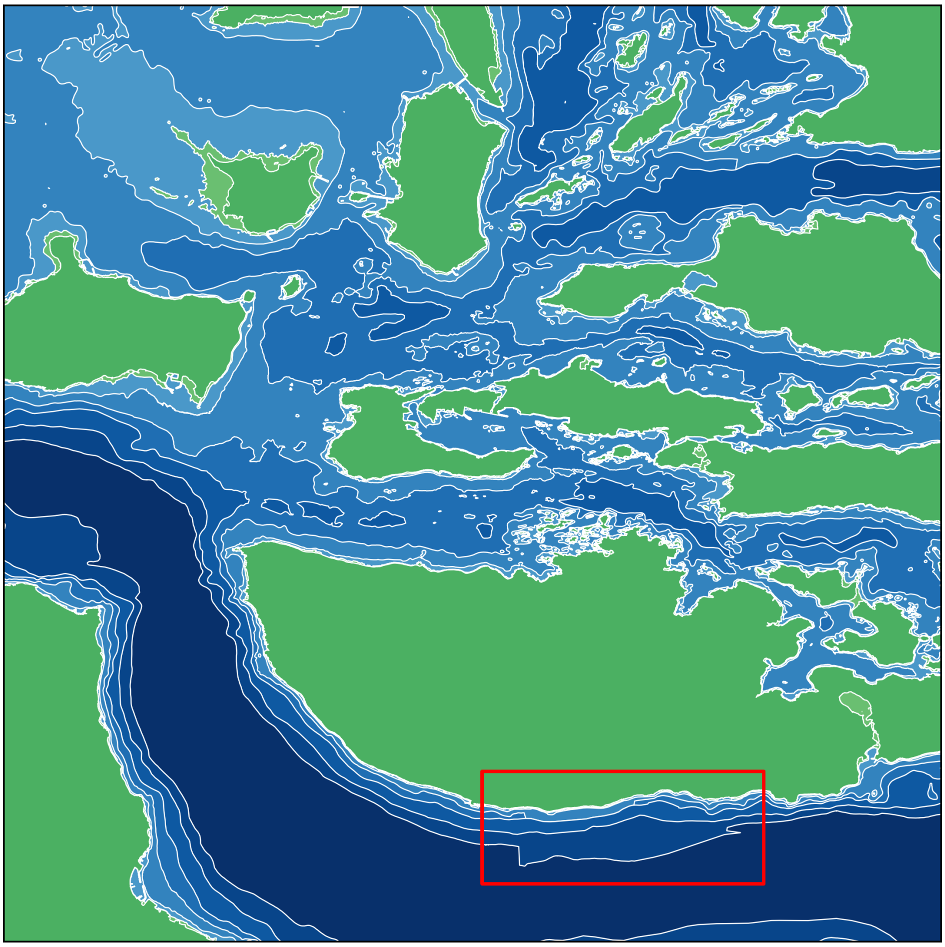


Figure 2.3: Polygon merging with depth bins of 10, 50, 100, 200, 300 m.

Polygon simplification

The next step of the polygon optimization process is to simplify the topology of the polygon edges and vertices. This operation is provided by the Shapely method **simplify**, which removes vertices and edges from the polygon that are within the distance defined by the optional **tolerance** initialization argument. The resulting polygon may have a significantly reduced number of vertices, and consequently may reduce the time complexity of the spatial algorithms performed on the polygon data significantly. Moreover, the geometric shapes may be further simplified by buffering (dilating or eroding) all polygons by providing the optional **buffer** argument.

Figure 2.4 presents an oversimplification example of the same Ålesund fjord area from the previous section, showcasing why the simplification of the polygons must be performed with care. If the tolerance distance is too large, the polygons may

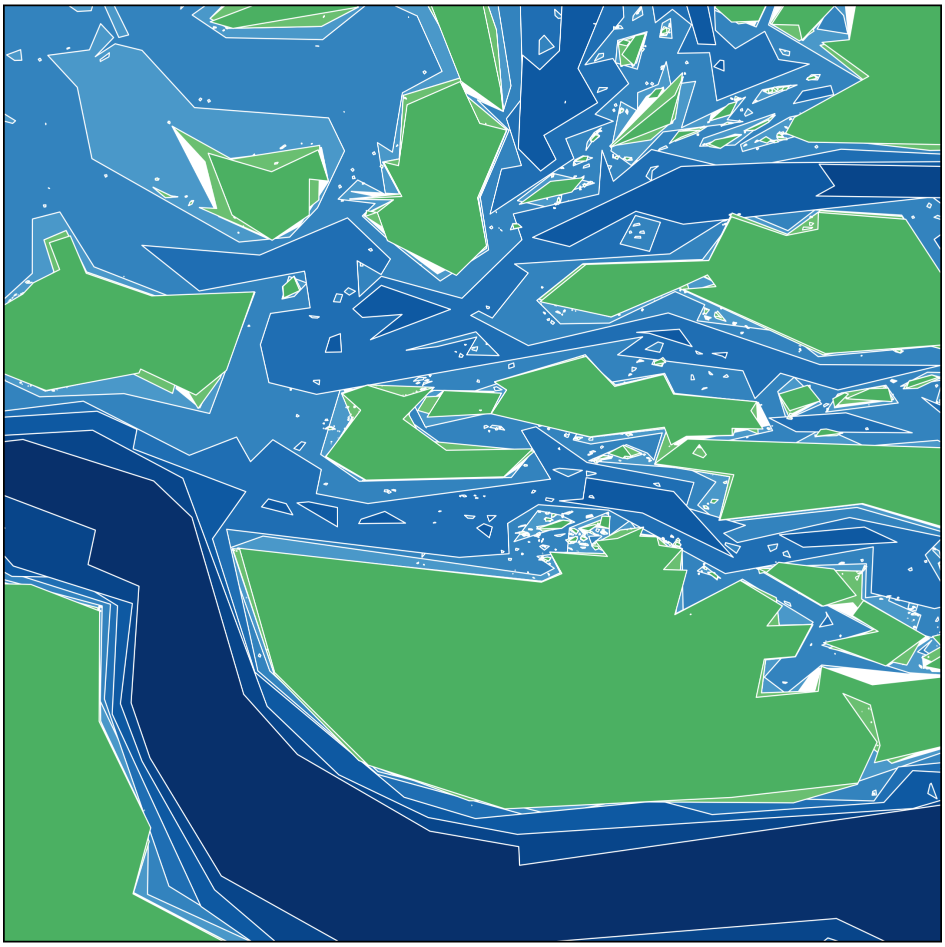


Figure 2.4: Polygon simplification example with a tolerance of 300 m.

become significantly distorted and thus no longer sufficiently represent the real-life obstacles in the environment.

Additionally, oversimplification may lead to some artifact regions not being covered by a depth polygon (as seen in white). These areas inherently have no ocean depth data associated with them, introducing irregularities and holes in the spatial data which are not easily fixed. Thus, the tolerance distance given to the simplification method is during the rest of this work set to be smaller than the width of the ship used for demonstration purposes, such that the topology of any polygon is guaranteed to be of the same or a higher resolution than the ship navigating the environment. This facilitates faster computation on shapes with lower counts of vertices and edges, while simultaneously keeping the resolution within reasonable bounds for the example applications demonstrated later in this chapter. Lastly, the Shapely method **simplify** also provides the option **preserve_topology**, and is in this work set to true in order to avoid polygonal deformation.

2.2.3 Feature selection

In this section, the data extraction hierarchy mentioned in Section 2.2.1 will be discussed in more detail. By selecting specific regions for the extraction of depth data (the yellow rectangle of Figure 2.1), the package may filter out any unnecessary regions from the full ENC data set, such that only the points and polygons of the area of interest are constructed and stored as local shapefiles. An example extraction view of such an area is presented in Figure 2.5, showing a square region of Ålesund smaller than the region presented in Figures 2.2 to 2.4.

Next, one may isolate an even smaller shape handling subregion (in Figure 2.5 shown as the white square defined by some coordinates provided by the user) after

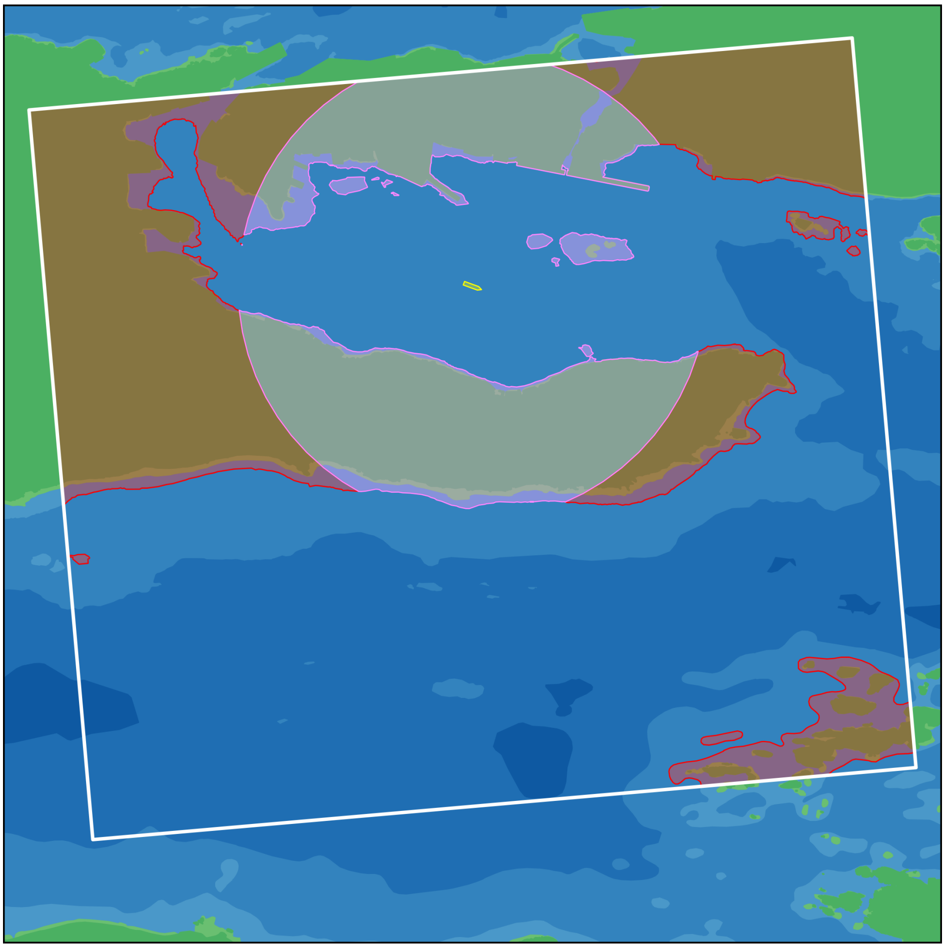


Figure 2.5: Bounded shape handling and dynamic horizon example.

reading the shapefiles to memory during runtime, and perform spatial operations on these points or polygons. Furthermore, if the package is used by e.g. an ANS, it may construct another artificial extraction window or dynamic horizon polygon defined by e.g. the pink disk as in Figure 2.5, or any other shapes like rectangles or general polygons. This dynamic region may subsequently be utilized to isolate all features of interest within the horizon such as e.g. grounding obstacles or nearby vessels for collision avoidance efforts, using methods provided by the SeaCharts package or the Shapely library directly.

The white square is intersected with all of the data points located within it, effectively filtering out all features not inside the rotated square. Here, the polygon merging and seabed range consolidation methods (i.e. the user-specified depth ranges from Section 2.2.2) may be used as a filtering technique to extract "safe" or feasible sea-faring regions for any ship, given user-specified parameters such as a minimum allowed seabed depth e.g. based on the maximum draft of the ship in question. For this purpose, the seabed polygon layer of e.g. 10 m contains any region of the processed data sets that are *deeper* than 10 m, by construction.

Inversely, all areas of insufficient depths may be calculated by taking the spatial **difference** (Shapely) between the white square polygon and the seabed layer, yielding the inverted shallower areas in red within which all depths are *less* than 10 m. Thus, both the dark green and lighter green land and shore polygons, as well as all seabed depths down to 10 m, are merged together and shown as red polygons in this example. The exterior boundary of the horizon disk around the ship of Figure 2.5 has an excessively small radius of 1 km, for demonstration purposes only. The interior of the disk is calculated by performing the Shapely operation **intersection** on the horizon disk polygon and the red polygons, resulting in the isolated pink overlay separated from the red.

An important detail to note in this example is how the selection of the 10 m seabed effectively closes off both of the narrow straits between the individual islands of Ålesund, northwest and northeast of the vessel. This is indeed the expected and desired result, given that both passages are not continuously deeper than the selected seabed layer. As such, appropriate depth bins or depth filters must be selected with care and thorough consideration, based on the possible consequences for any given navigation application.

2.2.4 Bounded depth regions

Extracted regions of arbitrary shape may be transformed into non-convex or convex regions for path or trajectory planning algorithms, by intersecting the polygon of a bounded region – like e.g. a rectangular view window – with any polygon of interest within it.

Figure 2.6 demonstrates a variation of such an application, based on the white square region from Figure 2.5. A new single polygon is constructed by the traversable open water area around the ship within the square with depths deeper

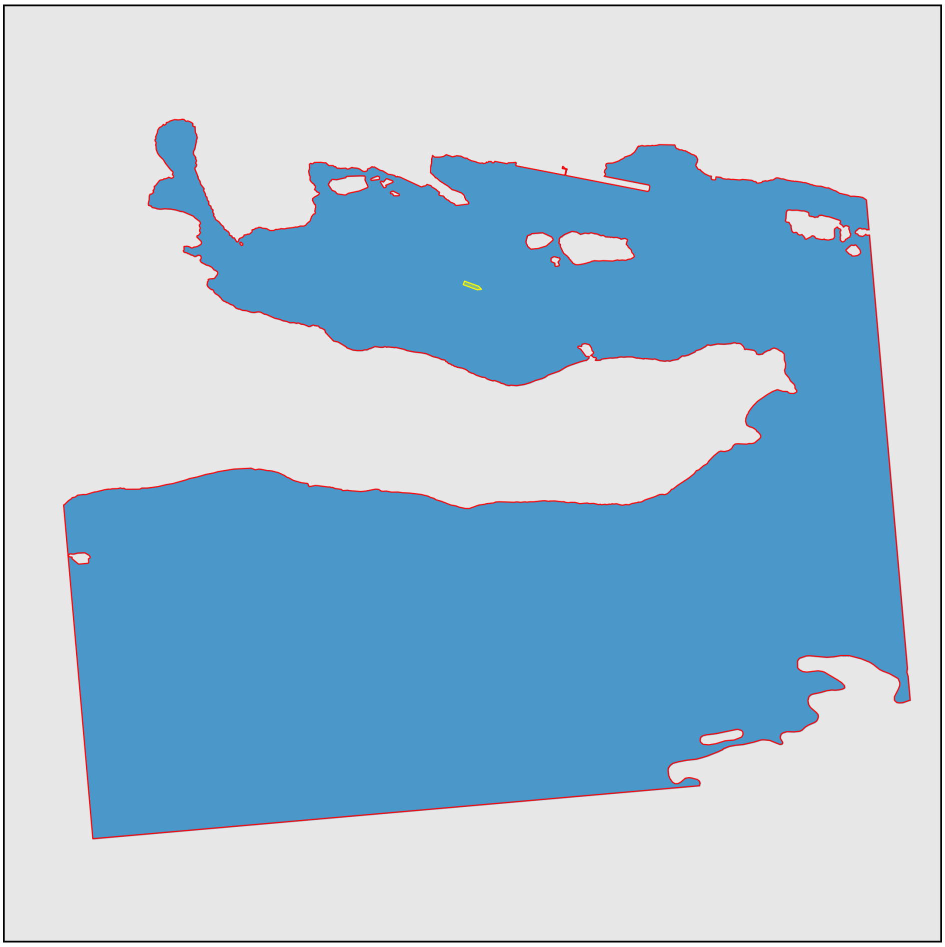


Figure 2.6: Inverted bounded minimum depth region transformation.

than 10 m, using the Shapely methods mentioned in Section 2.2.3. The problem is thus inverted from being based on an open environment with many land polygons, into utilizing a closed environment consisting of a single ocean polygon with islands represented as its inner holes. This transformed topology is inherently finite and bounded by its construction, and may be more feasible for use in autonomous navigation or path planning algorithms. Several overlapping or joined sets of these smaller bounded areas may subsequently be used to calculate subpaths from intermediate points along a larger route, possibly yielding significantly faster solver performances than global techniques.

This inverted topology may furthermore be used to facilitate methods for local path planning or risk analysis through distance-based artificial potential fields (APF) [60], [84], in which e.g. the distance from the ship to the nearest shore may be used to indicate risk levels during transit.

An example plot of a rasterized and distance-based risk topology is presented in Figure 2.7, based on the traversable polygon in Figure 2.6. Notice how the resolution of the point samples providing the base for the risk contours is lowered and exaggerated in this example for demonstration purposes. This lowering of spatial resolution may alongside with other techniques be used as a tool to facilitate faster performance. Points of higher risk closer to the shoreline or the interior and exterior boundaries of the traversable polygon are shown in red, and points far away from any boundary are given shades of blue. The gray area outside of the polygon is non-traversable land or seabed with insufficient depth. This type of plot may be useful for visualization of distance-based navigation algorithms, or to define constraints and cost functions in optimization techniques performed by external programs. Such rasterized data points may e.g. additionally be used directly as an alternative to the vectorized polygon data stored in the shapefiles.

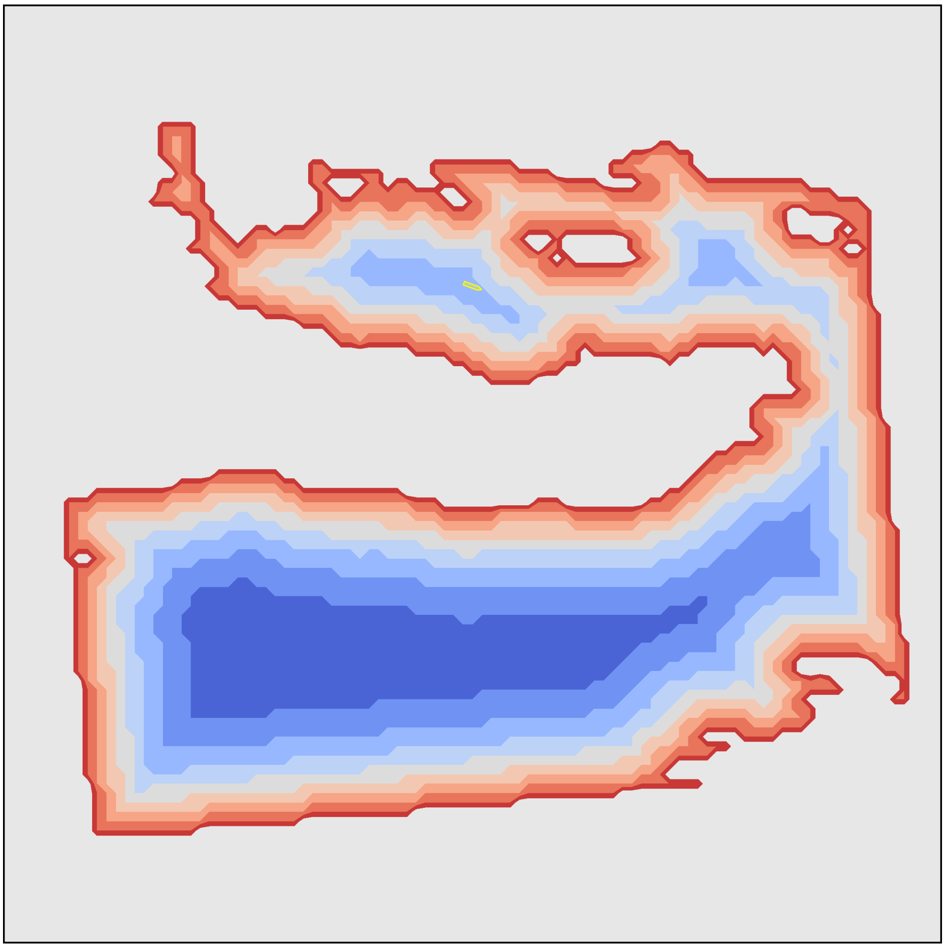


Figure 2.7: Distance-based grounding risk topology example.

2.2.5 Features visualization

This section presents an introductory overview of the visualization methods currently supported by the SeaCharts package. All figures are produced by utilizing the **show_display**, **refresh_display**, **close_display** during testing, and saving the resulting images through the **save_image** method in an appropriate image format.

Vessels plotting

To visualize the environment, the user firstly creates an instance of the SeaCharts main class. This class must be instantiated by defining an **origin** or **center** coordinate pair (easting and northing) as well as the bounding box **size** in meters, specifying any region of interest. After the environment polygon data produced from depth measurements are displayed, other features such as vessels and other physical structures along with abstract entities such as path references, pointer arrows, enclosing circles or area overlays of interest may be visualized on top.

A magnified view of the Ålesund region from Figures 2.2 to 2.7 is presented in Figure 2.8, showing a simplistic scene produced and displayed by merely selecting a desired ENC region and plotting a collection of various vessels in different colors within the environment using the **add_vessels** and **clear_vessels** methods. This is considered the main feature of the package.

Furthermore, the procedure for plotting each ship position is designed such that the *Display* may be created and run in parallel with the main calling process by setting the **multiprocessing** initialization argument equal to **True**, in order to update the visualization plot in real time. This separate process repeatedly reads the comma-separated values (CSV) file containing all coordinate pairs, heading angles and color names as well as other options for each single ship plot to be displayed in the environment. Thus, the normally convoluted operation of visualizing ships in a maritime environment is reduced to writing (or removing) a collection of values to a plain file during runtime.

Information presentation

In addition to simply showing vessels in the environment, supplementary information overlays may be added to the plot in order to present various aspects of e.g. vessel(s) risks, intent, planning, predictions, or other relevant factors like weather conditions or mission objectives. For this purpose, several basic shape-based visualization methods have been added to the package. Specifically, it is possible to add lines, arrows, paths, circles, rectangles and any other general polygon shape on top of the environment plot through the collection of **draw** methods of Table 2.1. Various adjustment options are available to these methods. These will however not be noted in detail in this work, and the user is referred to the maintained repository Readme and code documentation for updated versions of all available input arguments for each function. An example demonstration of these are shown in Figure 2.9.

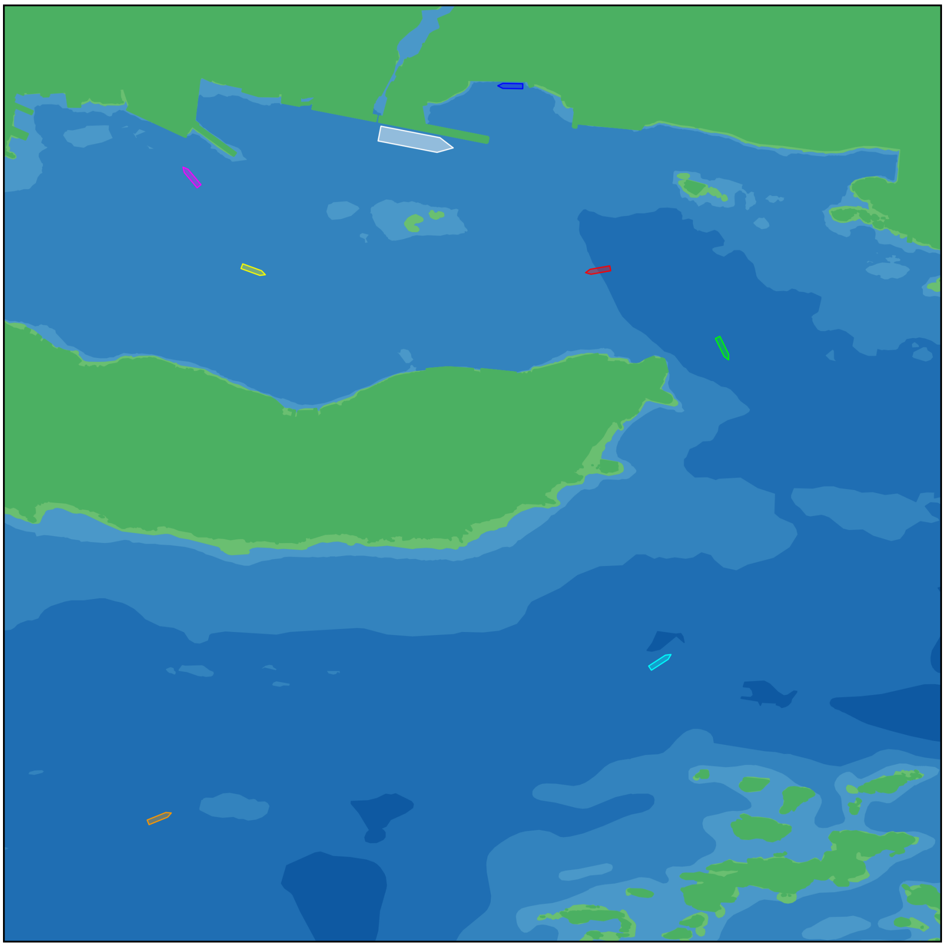


Figure 2.8: Example plots of various vessels in different colors.

Centered in the middle, a clock-like structure of various elements is plotted for the purpose of demonstration. Behind the rotated and semi-transparent rectangle in blue, a yellow disk is drawn with a dashed edge, which again encompasses another smaller green circle. The difference in border style and the non-filled interior color of the smaller circle highlights the flexibility of the available plotting variations, and is possible to adjust similarly for all shapes. Likewise, the pair of two-part straight lines in white and magenta are showing the different variations of straight line segments. The thickness of all bordering edges of shapes as well as for lines may be chosen by the user, as is readily apparent from the large plotted arrow in orange. For this shape, the size of the arrow head may be adjusted as well.

Next, there are additionally shown three groups of polygonal overlays in Figure 2.9. The lighter land and shore polygons of a single island due north in Ålesund is simply a polygon overlay of white transparent color to highlight a region of interest. In

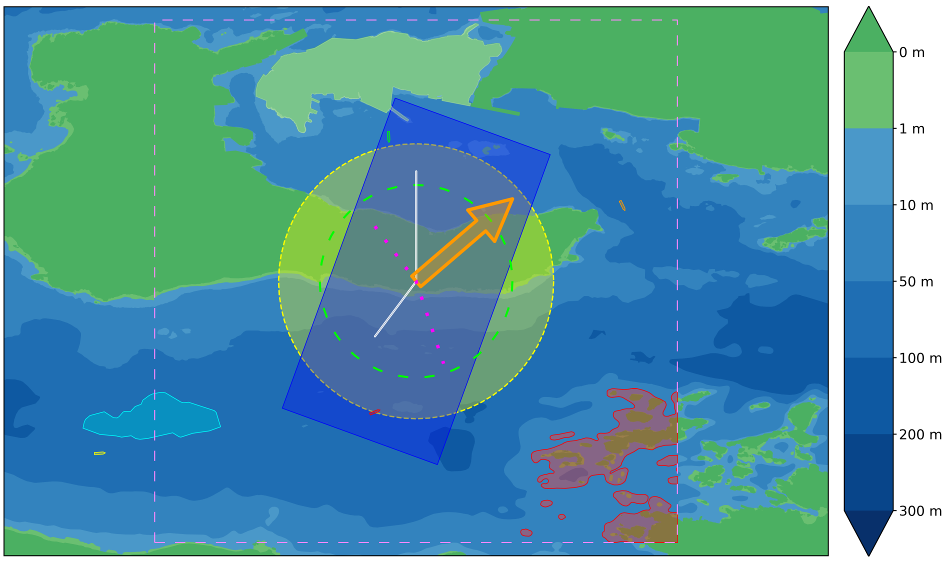


Figure 2.9: Example demonstration of various available shapes and overlays, including a depth bar legend.

the middle of the fjord to the southwest, a cyan shape has the contours of a seabed polygon with a minimum depth of 100 m and a maximum depth of 200 m, given the user-specified depth bins for this example. Thus, this shape outlines an area for which the depth measurements of the ocean are within 100 and 200 meters (assuming the collected data are valid). More interestingly, the group of red isles to the southeast are intersected with the large square with pink dashed edges, once more using the **intersection** method of the Shapely library. These shapes contain areas for which the depth is more *shallow* than 10 m, and are shown with red solid boundaries in addition to a semi-transparent interior color. Note how the demonstration shows that the user may select any subset of shapes resulting from an intersection (or any other spatial operation), as not all of the areas with the same depth within the pink square are highlighted in red.

The optional colorbar to the right of the environment plot is enabled by calling the **colorbar** method, and shows how the colors of the land, shore, and seabed polygons correspond to the depth measurements as grouped by the defined depth bins. The dark green is simply all *land* polygons with a height of 0 meters and above relative to the mean sea level, as indicated by the upwards triangle shape. The lighter green is however chosen to represent all data polygons labeled as *shore* in the downloaded data sets, and is denoted as a value range between 0 m and 1 m on the depth bar legend. The seabed depths in blue are intuitively increasing in color intensity in a downward direction, down to the darkest depth at the bottom. As indicated by the legend shape, the bottom layer (here at 300 m) also contains all depth measurements larger, or deeper, than its value. Note however that the depths of the color legend are not required to be present in the drawn environment, and is only dependent on the user-specified depth bins, for consistency.

Lastly, Figure 2.10 shows the same environment plot with a bounding box window equal to the pink square of Figure 2.9, with no colorbar added and dark mode toggled on by calling the **dark_mode** method. This optional darker view may aid in providing starker contrasts for the color of the informational shapes and vessels, as well as complying with the established industry standard for e.g. vessel and environment plotting on commercial ship bridges during the night. Thus, this mode remains activated for all remaining figures in this work for increased readability.

Planned paths

Path plotting is an essential tool for visualizing navigation and ship routes within an environment. Figure 2.11 presents an example demonstration of a coarse planned

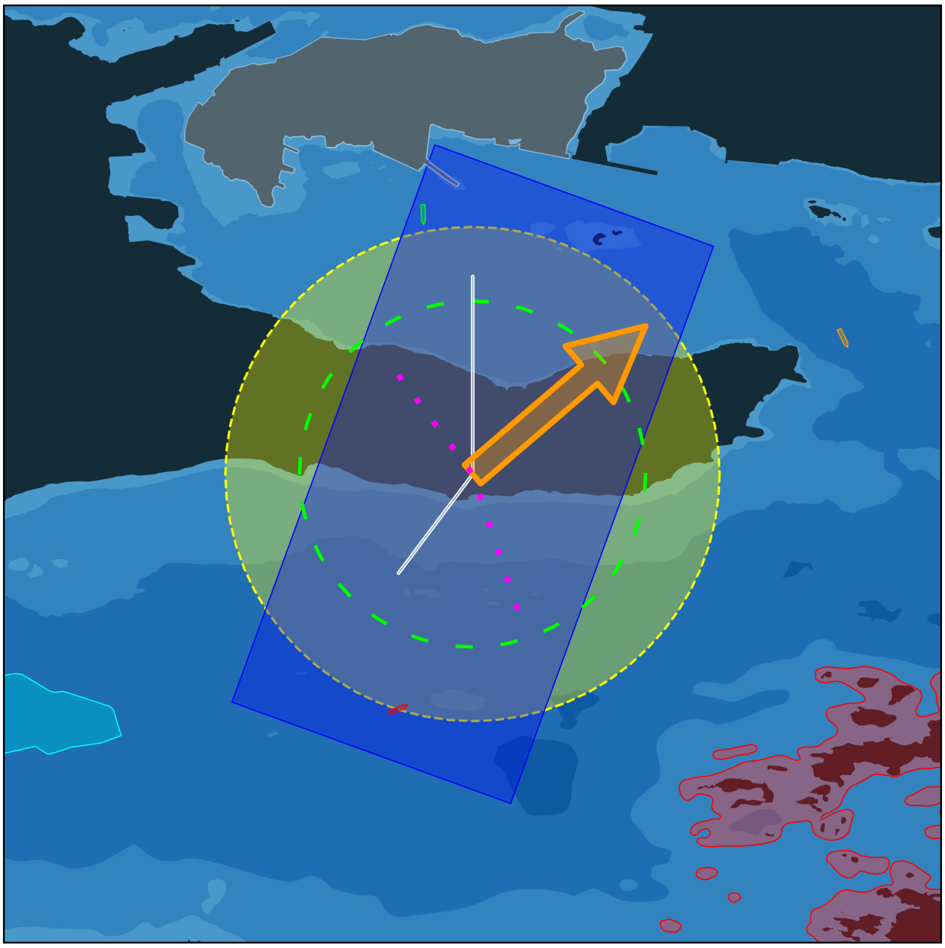


Figure 2.10: Example demonstration of a square dark mode view.

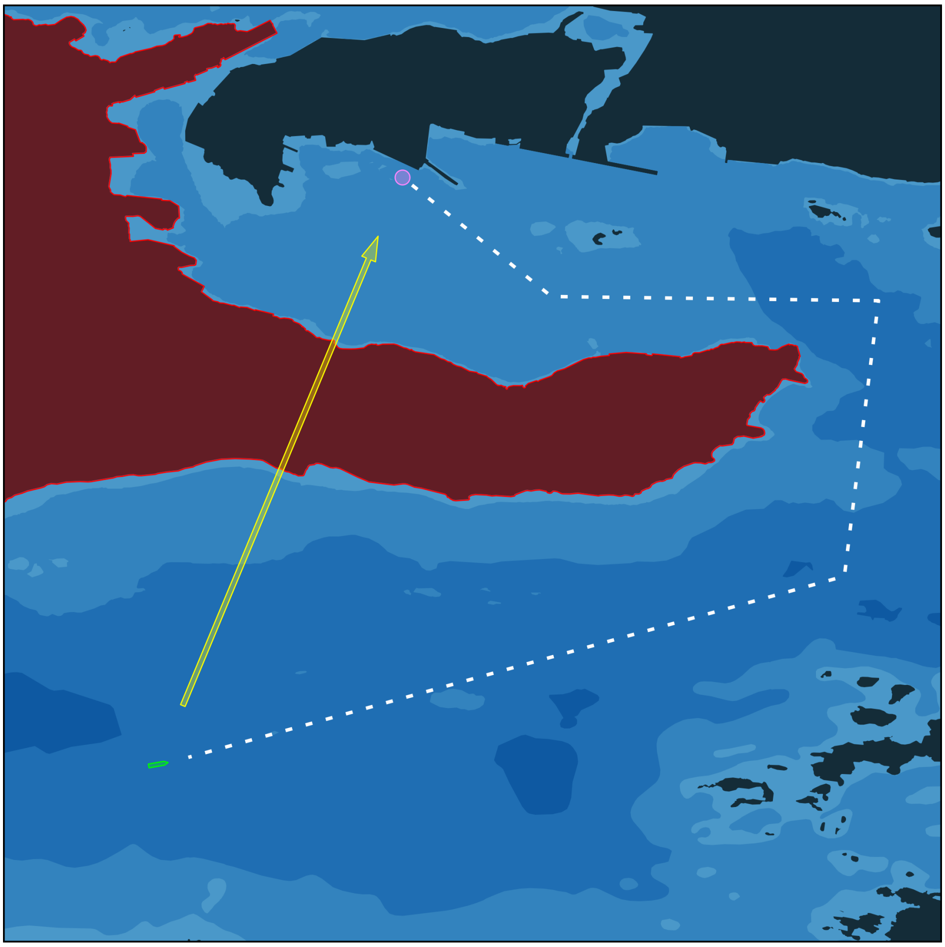


Figure 2.11: Example demonstration of path plotting and additional overlays.

path shown in a dashed white line. The start position of a single vessel is displayed in green, and the target destination (before docking) is added as a pink disk. These straight line segments and supplementary shapes may readily visualize a vessel route e.g. given by defined mission objectives or other subsequent and automated path planning schemes. Moreover, a simple arrow pointing from the vessel's start position and to its target crosses a land mass or island located in between, highlighted in red. Additional visualizations such as these may be used to communicate to the viewer e.g. how a path planning algorithm (Section 2.3.5) performs or utilizes available data resources during execution, like in this example.

2.3 Example usage

The following sections present example demonstrations of various API usage within contexts for which the SeaCharts package is intended. The example usages include plotting of trajectories (trails) and simulations of target vessels, visualization of points of interest, intent, and safety domains, utilizing the interactive capabilities of the API for fast ad hoc prototyping, line of sight calculation, and example applications within the fields of collision avoidance, path planning, online risk analysis, and optimal control.

2.3.1 Trajectories and trails

For the rest of this chapter, a particular area along the coast of Norway is used for demonstration purposes, in order to highlight the capabilities of the SeaCharts

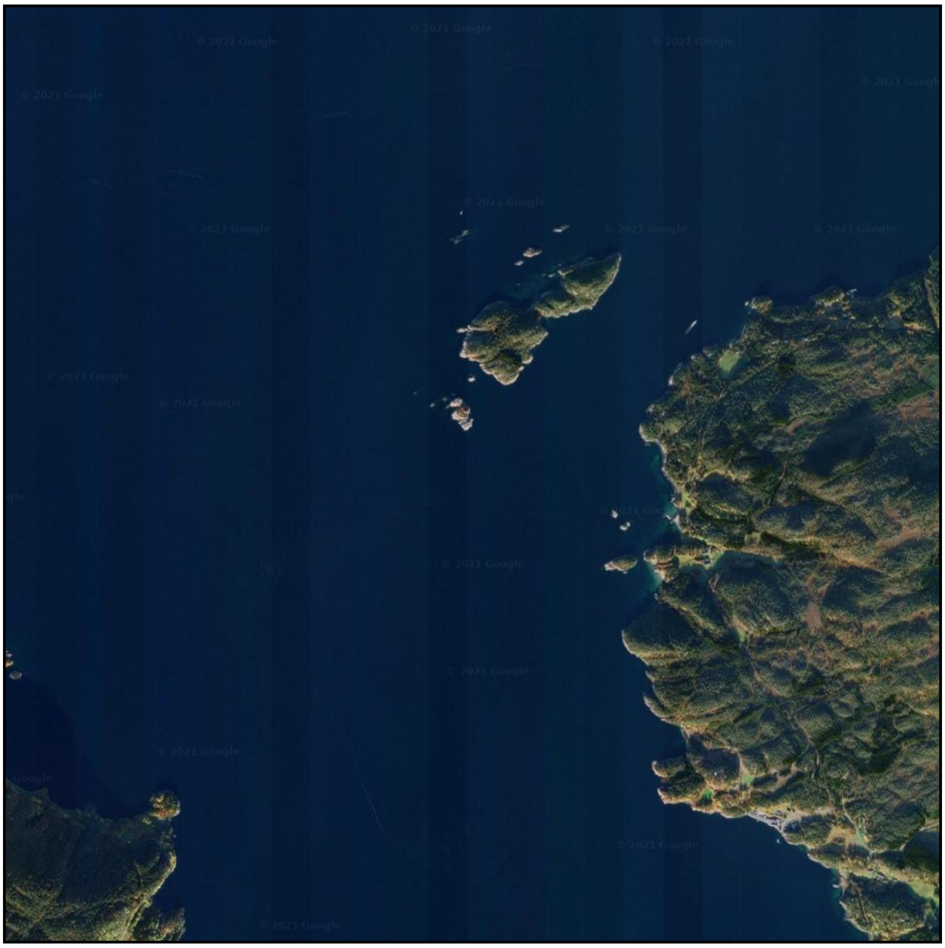


Figure 2.12: Vessel mission area example (Google Maps - satellite view).

package. Figure 2.12 shows an example view of a fjord area northwest of the ferry dock of *Halsa*, at the border between the Norwegian counties *Møre og Romsdal* and *Trøndelag*. For example, an interesting scenario is path or trajectory planning on each side of the small isle group shown in the middle of the image, with regard to risk analysis and energy consumption optimization purposes.

Figure 2.13 shows an example visualization of three different ship trajectories with the same origin, in the environment presented in Figure 2.12. Each color represents a separate trajectory, and are as the other shapes made semi-transparent in order to more easily distinguish between overlapping trajectories, and to make the temporal aspects of each ship path more discernible if vessel polygons overlap. The original planned path for all vessels is represented as green lines with circular waypoints through the isles strait. This straightforward usage provides the user with a basis

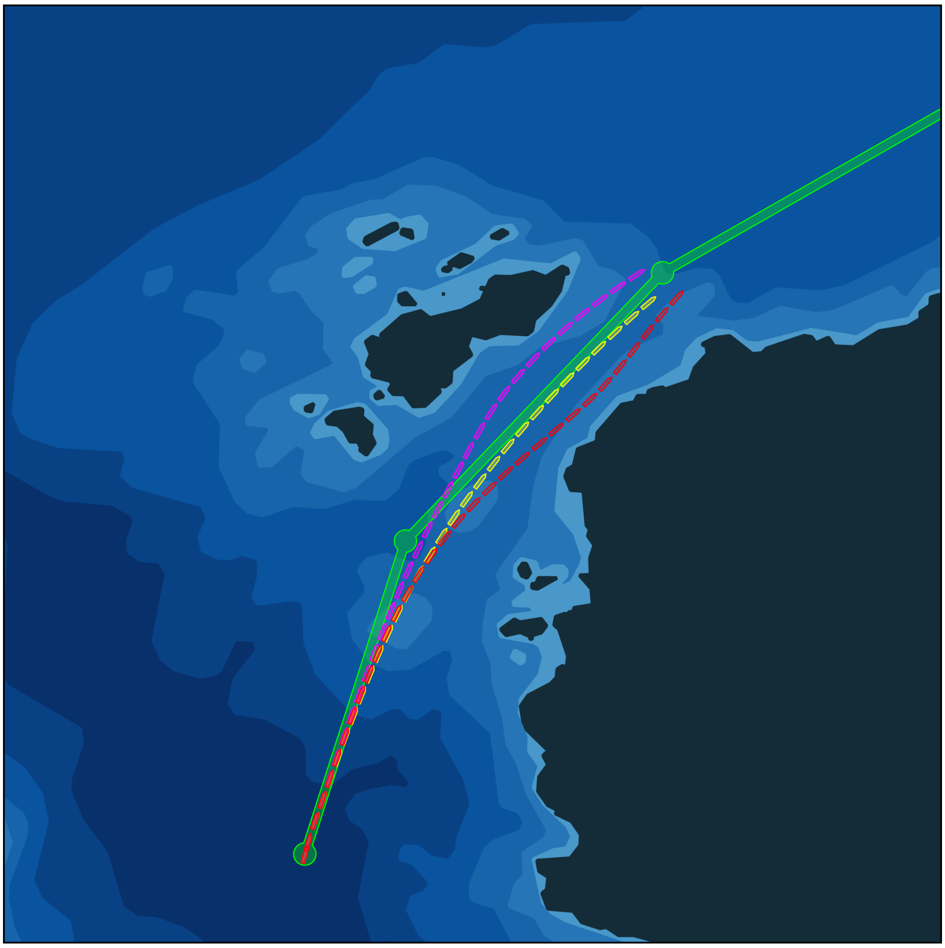


Figure 2.13: Example plot of three overlapping vessel trajectories.

for plotting of vessel trajectories, which may be useful for risk analysis or more advanced optimization visualization.

Another natural application of the package is to show several different types of alternative paths or trajectories for a single ship, such as future predictions or past trails of ship poses. Figure 2.14 presents a visualization of a planned ship path with power blackout simulations at regular intervals. In the environment plot, yellow ship poses are shown along a green pre-planned path with waypoints given as discs. Note that in contrast to the ship trajectories of Figure 2.13, these ship poses are not produced by simulation of a ship dynamics model, and are generated simply by discretizing the planned path and calculating the appropriate ship heading between each intermediate interval along the path using trigonometry. It is thus apparent that the package is not dependent on any past or future (accumulating) inputs

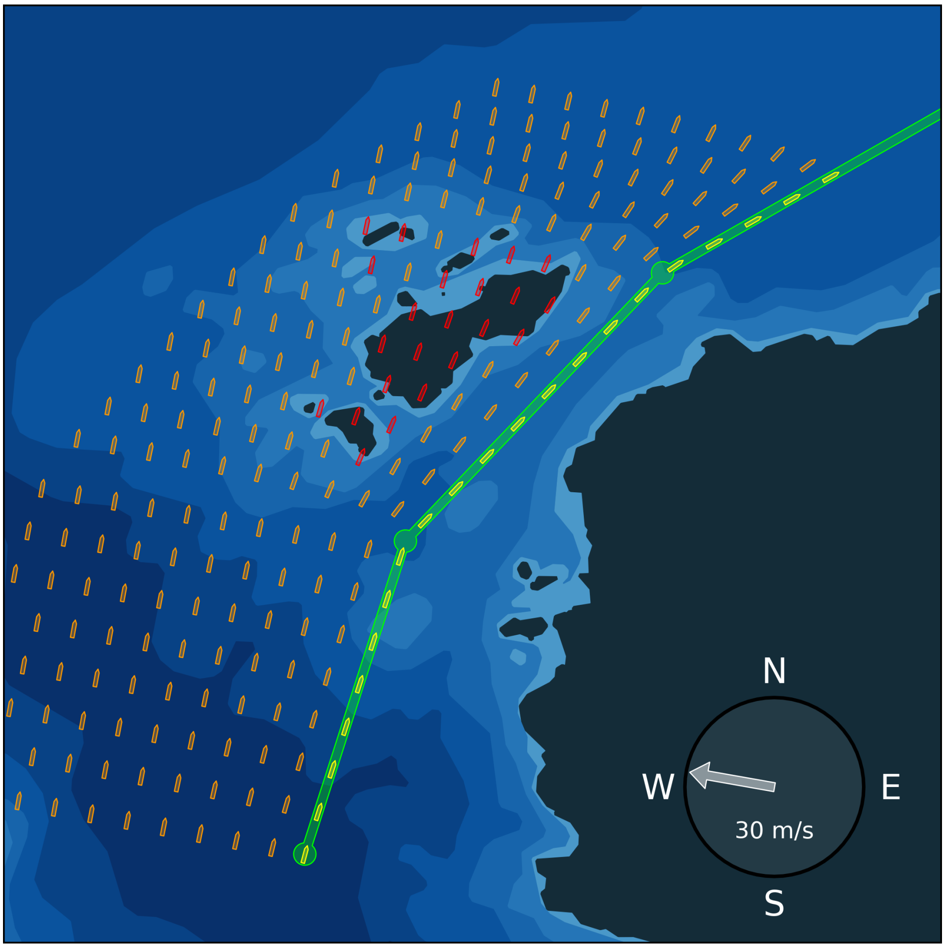


Figure 2.14: Pre-planned path example with wind disturbance simulations.

for plotting, and may be used quite flexibly for visualization of a large selection of various objectives or information of interest at any time instant during the program cycle.

The ship trajectories in orange are however in this example computed by setting the ship velocity to zero at each intermediate waypoint, and iteratively calculating the next ship pose given a simple ship dynamics model, including a wind disturbance force driving the ship westward e.g. as a result of power blackout or machinery failure. The wind direction and velocity are given in the bottom right corner. All ship poses intersecting with seabed polygons of depths less than 5 m are in this example given a red color, for illustrative purposes. This combined visualization of both pre-planned and simulated paths are shown in a single demonstrative display, to further showcase the capabilities of the package.

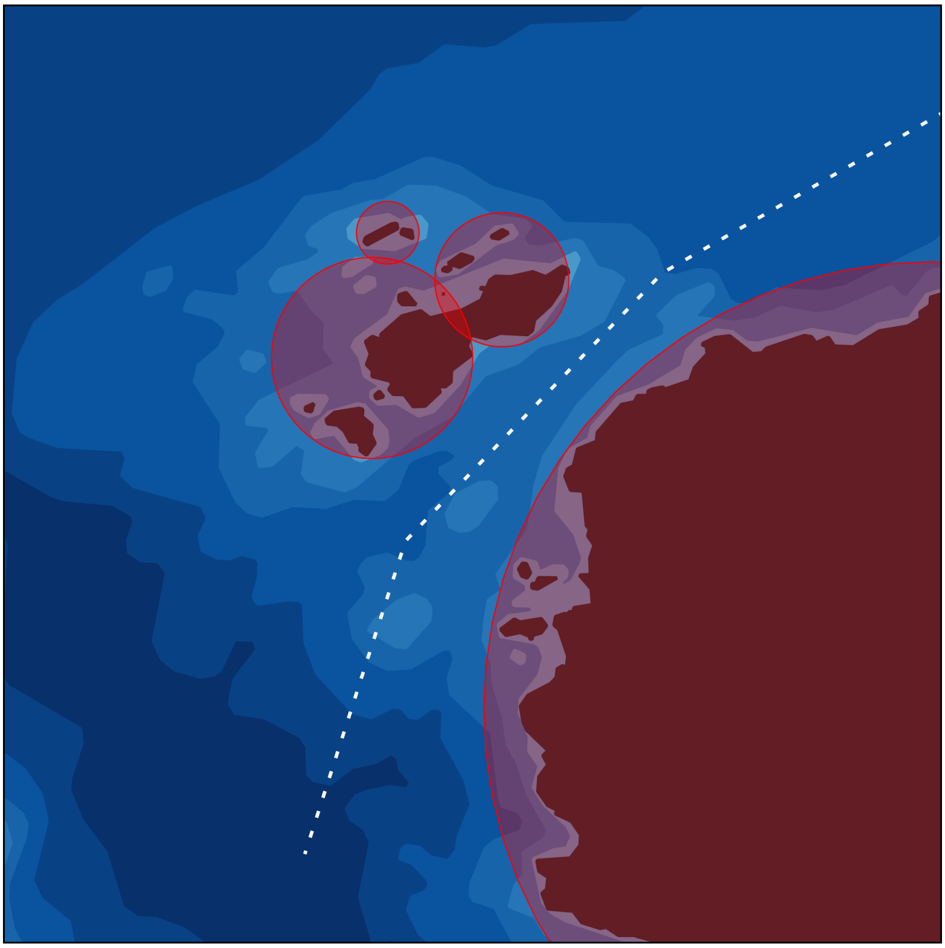


Figure 2.15: Simple vessel path example with danger area disks.

2.3.2 Control and simulation

For planning and/or control applications that require reduced complexity, the user may wish to display information accordingly. Figure 2.15 presents a simple environment in which ship poses are replaced by a single dashed line denoting a planned path, along with overlapping red disc overlays with various diameters. Such environment plots may be useful for fast prototyping or simple problem formulations, and are readily available by the provided package methods. The red overlays may e.g. represent abstract circular grounding obstacles for proof-of-concept planning algorithm research [38], allowing for simple radius-based proximity calculations and faster algorithm computations.

For more advanced usage, Figure 2.16 presents a snapshot of a larger temporal

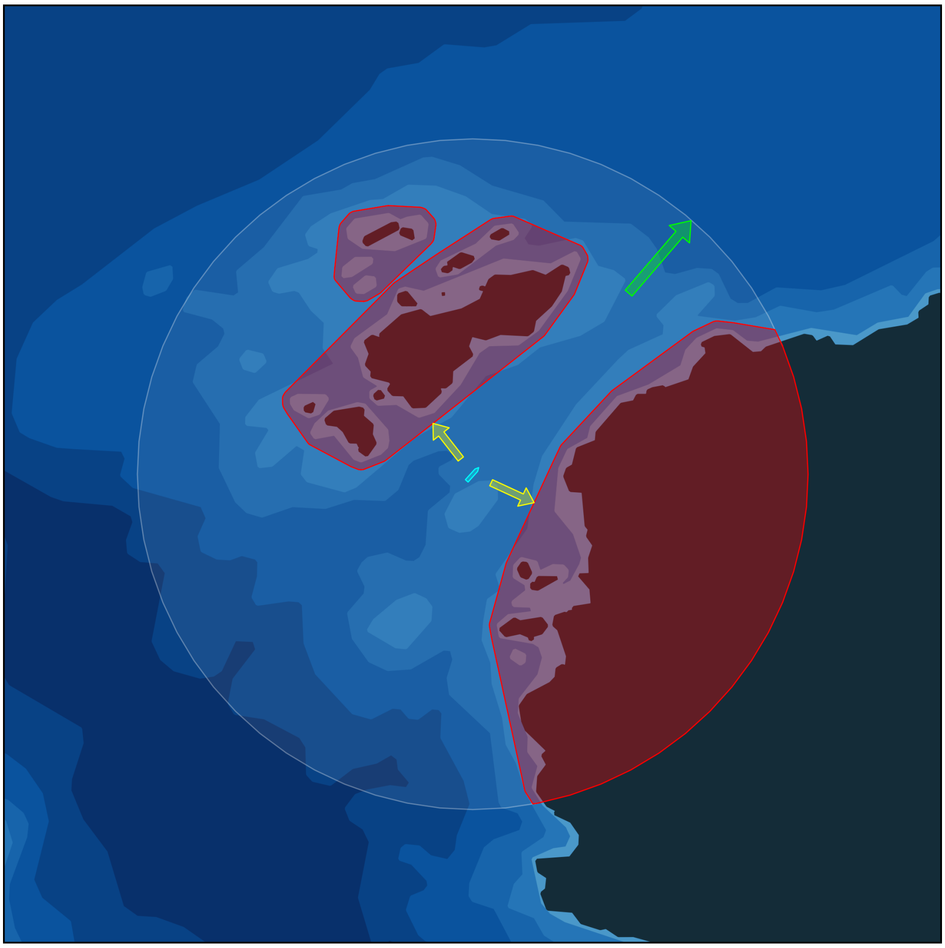


Figure 2.16: Visualization example with red local danger areas and yellow arrows of minimum distances to observable grounding obstacles.

simulation run exported as the graphics interchange format (GIF), showcasing how the spatial operations provided by the Shapely package may be used to visualize dangerous or hazardous areas calculated based on a dynamic horizon radius around the ship. The horizon is shown as a white disk with a radius of 1.5 km around the cyan pose of the ship, and represents the red dynamic extraction window from Figure 2.1. Similarly to the pink disk of Figure 2.5, the horizon is also here made excessively small for demonstration purposes only.

The red regions shown inside the disk are generated by first taking the intersection between the circular horizon and all areas with depths less than the maximum ship draft of 5 m, adding a spatial safety buffer of 30 m, and extracting the *convex hull* from each resulting polygon. This is done to demonstrate that the principle of which areas are considered hazardous or impassable is entirely decided by the user.

A green arrow is shown pointing to a target location along some route through the strait. Moreover, the yellow arrows pointing from the ship center to the closest point on each grounding obstacle are calculated by the Shapely method **nearest_points**. Notice however that there is no yellow arrow pointing toward the smallest red shape, as that polygon is not labeled "observable" by the ship as a result of the larger red polygon between them. This may be verified in a user-defined algorithm by e.g. utilizing the Shapely method **intersects** on each present obstacle shape with respect to the straight line of sight between the ship and any other obstacle.

2.3.3 Collision avoidance

In addition to the concepts related to anti-grounding described in previous sections, a user may also visualize how an autonomous or controlled ship perceives and/or interacts with other vessels within its vicinity. Figure 2.17 demonstrates how the positions and headings of nearby vessels are used to construct safety domains [28] for each corresponding vessel pose based on given proportional parameters and an ownship horizon. Here, the red overlays highlight only physical landmasses disregarding seabed depths, and the yellow arrows are pointing toward all safety domains of any nearby vessel within the horizon. Notice how in this example the safety domains of each vessel shown in orange are scaled with its own velocity vector by some factor. Thus, the distant ship in pink has its safety domain polygon visible within the ownship horizon due to a significantly high velocity.

Another example demonstration of vessel safety domains with respect to collision avoidance is presented in Figure 2.18, serving as a base of discussion for the following section. The user is referred to the SeaCharts Readme for further usage details.

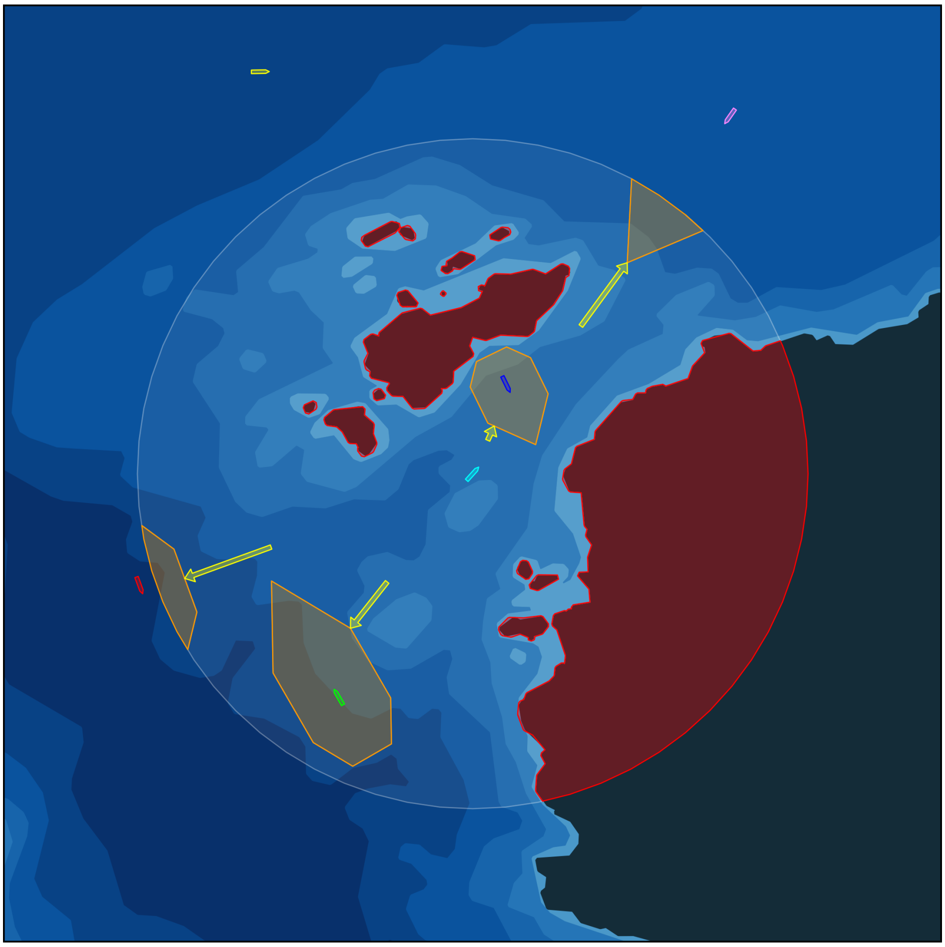


Figure 2.17: Vessel safety domains visualization within ownship horizon in orange, land obstacles in red and yellow distance arrows.

2.3.4 Interactive mode

In addition to the shape plotting methods described in the previous section, the SeaCharts package also includes two interactive programs; ownship and path plotting. Figures 2.18 and 2.19 show example plots during such interactive sessions.

Controllable ownship

In order to easily plan around or estimate the outcome of various scenarios of interest, the user may activate a controllable ownship to move around in the environment using keyboard keystrokes. Figure 2.18 exchanges the circular ownship horizon of Figure 2.17 for a larger ownship domain split into sectors, based on the concept of navigational lights. Here, the safety domain polygons of nearby vessels

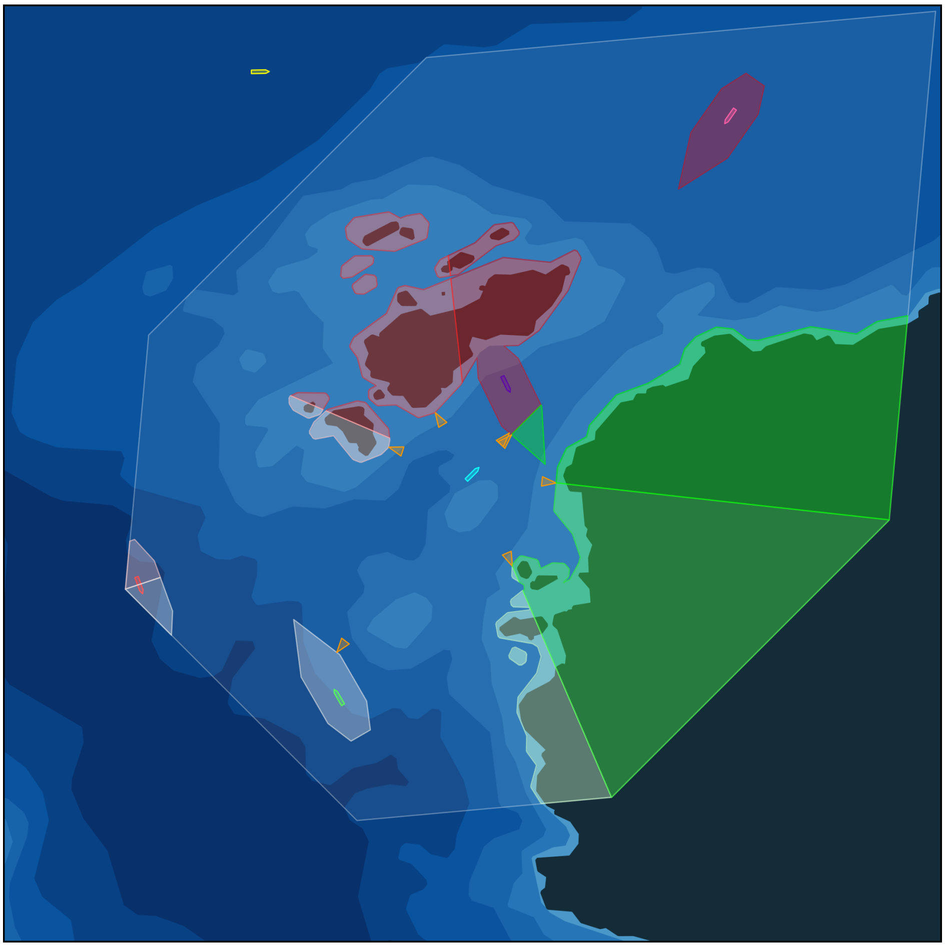


Figure 2.18: Interactive ownship visualization with alternative safety domains for vessels within a sector-based horizon.

are of constant size, and are colored according to their orientation with respect to the cyan ownship.

The diamond-like ship horizon polygon is constructed in a fashion similar to the principles of a vessel's navigation lights, split into seven subregions according to the orientation of the navigation lights on the vessel: The starboard side has two regions in green and lighter green respectively from the forward axis of the ship and to the 112.5° mark, and the red regions are similarly mirrored on the opposite port side. In white, the aft direction is split into three such that one may differentiate between objects located within the different subregions relative to the heading of the ship.

The orange triangles denote the closest point of any selected type of polygon located in each subregion, if it exists. In this example, the user has chosen the seabed layer with 10 m depths. All seabed, land, or shore polygons with depths less than 10 m within the horizon diamond of transparent white are consequently highlighted in tints of green, red and white according to their corresponding subregion color. Thus, there are exactly seven arrows (two overlapping) pointing toward each of the identified regions of different navigation light colors, calculated and stored in a simple CSV file during runtime. The application or user in question is at liberty to freely decide how these closest polygon points are to be utilized for further interpretation. The ownship, arrow triangles, nearby vessels and horizon hazards may all be toggled off and hidden from the environment, if desired. Additionally, the size and proportions of the vessel horizon and hazardous depth filter may be dynamically adjusted during the interactive session by using keyboard keys.

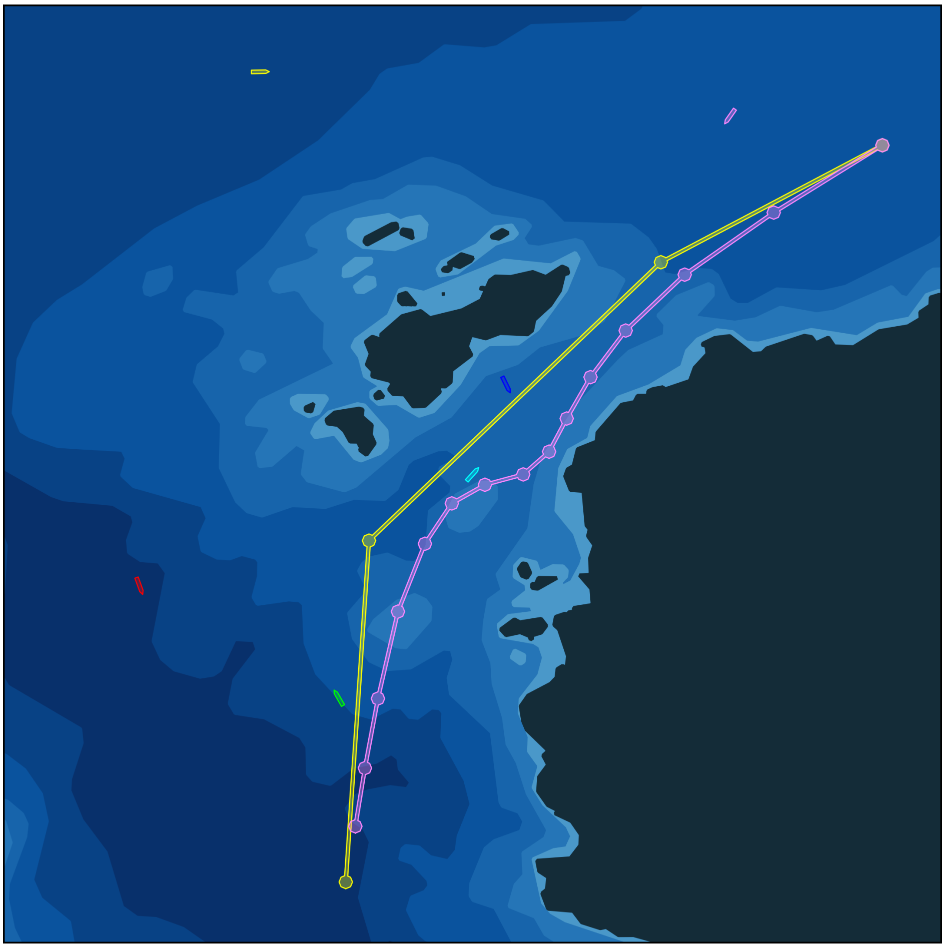


Figure 2.19: Interactive ownship and paths visualization, with original waypoints in yellow and a less crude path shown in pink.

Path drawing and manipulation

Figure 2.19 presents another snapshot of an interactive session. Here, two independent examples of planned paths are drawn between the same ownship and nearby vessels from Figure 2.18. One may view the larger yellow path as the coarse ship path planned for the ship through the isle strait, for which four separate waypoints are given. These waypoints may be decided or planned as a subset of a larger mission objective from port to port, expanding further into the environment at either side. The denoted waypoints demonstrated in Figure 2.19 are part of a subplanning problem constrained within the environment shown, with the ownship horizon and hazards toggled off for clarity.

The smoother path in pink may furthermore show how mission control or the autonomous planner aims to follow the main yellow path, in which the time intervals are shorter and the resulting trajectory has a higher resolution. Based on some given operational costs or thresholds, an algorithm may e.g. produce a more detailed and smooth path compared to the coarse main path for a given part of a larger route. Notice how the path in pink attempts to avoid the safety domain of the blue vessel (as shown in Figure 2.18), creating another significant deviation from the intended path.

The path waypoints for both colors created by the user during the interactive mode are stored in CSV files, and the package may conversely plot paths given by an external program through simple reading of these files. Additionally, the user is able to both move and delete existing path points at any location by appropriate mouse and keyboard commands, allowing for flexible makeshift planning during testing or programming of e.g. ANS.

2.3.5 Path planning

In addition to makeshift planning or prototyping during interactive sessions, the user may also want to display or showcase autonomous path planning e.g. produced by an ANS during or after simulations. Thus, an example path planning algorithm as well as examples of information visualization is presented in this section.

For demonstration purposes, a simple path planning algorithm for constructing a tree of possible route alternatives between two waypoints is presented in Algorithm 1. Note that it is not intended to be a complete path planning algorithm, but is merely included in this work to showcase example usage of a selection of SeaCharts methods. The algorithm is given a set of grounding obstacle polygons G, a safety distance Δd_s, an initial starting waypoint σ, and a single end target waypoint χ to which a path with several potential route alternatives is to be planned. The grounding obstacles G may be any two-dimensional polygons of arbitrary shape with any optional depth(s) of interest, and is defined by the user as an input to the algorithm. Similarly, the start point σ may be any point e.g. along a route or the current position of a vessel, the end point χ may be any user-selected target point, and Δd_s is defined as any desired buffer distance.

Algorithm 1 PlanRoutes

```
Input: grounding obstacles $G$, safety distance $\Delta d_s$,
       start point $\sigma$, end point $\chi$
Output: binary tree $R$ of alternative routes from $\sigma$ to $\chi$
  procedure PlanRoutes($G, \sigma, \chi$)
      $H \leftarrow$ convex hulls of all polygons in $G$
      $I \leftarrow$ dilate $H$ by $\Delta d_s$
      $J \leftarrow$ spatial unions of all polygons in $I$
      $K \leftarrow$ convex hulls of unions $J$
      $\rho \leftarrow$ straight line segment from $\sigma$ to $\chi$
      $R \leftarrow$ new tree of line nodes with root $\rho$
      while $\exists P \in K$ intersects $\exists \rho \in R$ do
          $P \leftarrow$ largest intersecting polygon
          $\rho \leftarrow$ remove intersecting line from $R$
          $V \leftarrow$ visible vertices of $P$
          $\Lambda$, $\Gamma \leftarrow$ group $V$ into left and right wrt. $\rho$
          $\lambda$, $\gamma \leftarrow$ vertices of $\Lambda$ and $\Gamma$ farthest from $\rho$
          $\delta \leftarrow$ start point of $\rho$
          $\alpha_{1,2} \leftarrow$ linear line segments from $\delta$ to $\chi$ via $\lambda$
          $\beta_{1,2} \leftarrow$ linear line segments from $\delta$ to $\chi$ via $\gamma$
          $R \leftarrow$ add $\alpha_{1,2}$ and $\beta_{1,2}$ as new line nodes
      end while[0]
  end procedure
```

Figure 2.20 shows an example in which a vessel intends to navigate around a collection of smaller isles, i.e. the set of grounding obstacles G. The start point σ is represented by the vessel hull in cyan, and the end point χ is denoted by the yellow disk. The initial route path ρ intersecting G is shown as a yellow line from σ to χ. In this example, G is defined by extracting all nearby areas of seabed depths $< 10\,\text{m}$, which consists of the union of all dark gray island masses as well as the light blue ocean polygons sharing at least one edge with (and fully encompassing) the land masses.

An initialization phase of six steps sets up the algorithm before the main loop is initiated, and consists of the following. The convex hulls H of all polygons in G are computed by accessing the Shapely property **convex_hull**, and the resulting new set of polygons H are subsequently dilated by the safety distance Δd_s (here defined as 50 m), using the Shapely method **buffer** to produce the polygon set of I. In Figure 2.20, the pink overlay is isolated by selecting all (in this case two) polygons of I that intersect the initial route line segment ρ, for the purpose of visualization only.

The next step calculates the spatial unions J of all polygons in I using the Shapely method **unary_union**, such that any overlapping polygons are merged. The convex hulls K of J are lastly computed similarly to the first step, yielding the final

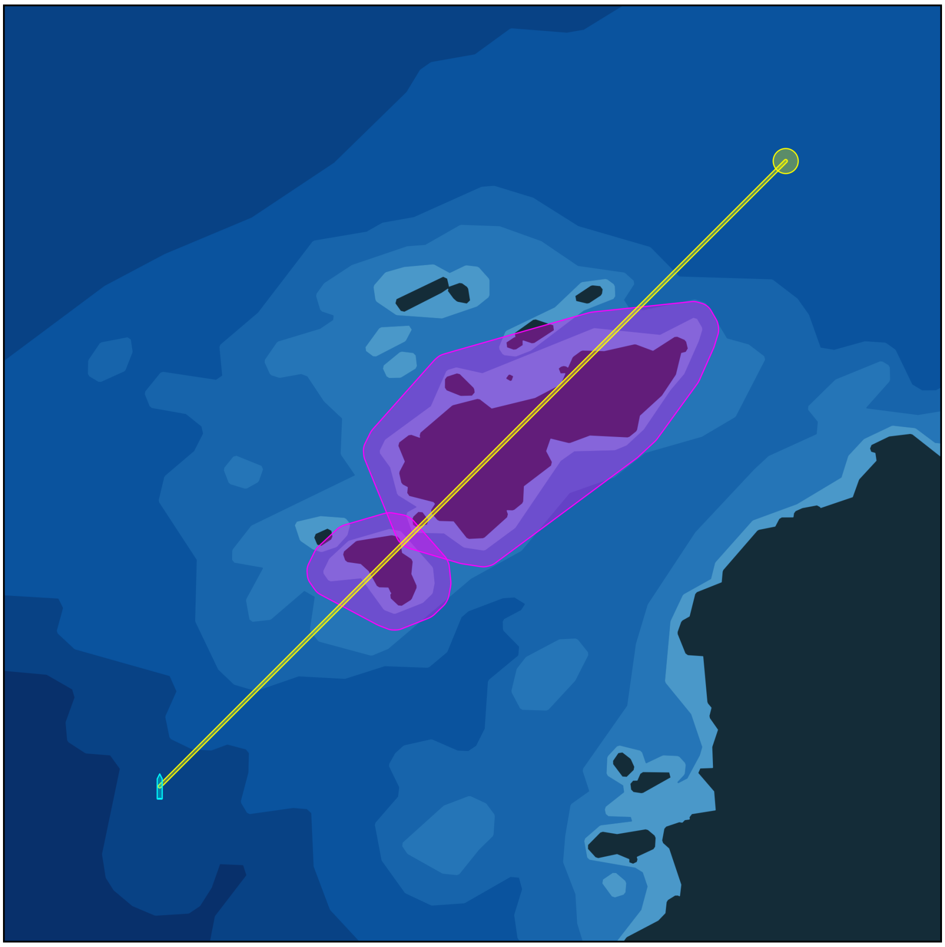


Figure 2.20: Two overlapping polygons from the set of polygons I, which overlap the initial straight line route ρ between start point σ and end point χ.

set of polygons to be used in the main loop. This is done to potentially save a significant number of subsequent algorithm iterations, by reducing the number of considered polygons and disregarding all non-convex areas contained within or between the dilated (larger) obstacle polygons of I. The only such polygon in K intersecting ρ is shown as the pink region in Figure 2.21, demonstrating how the group of isles is reduced to a single convex polygon. Lastly, the initial straight line segment ρ (shown as the yellow line in Figures 2.20 and 2.21) is defined by the start point σ and the end point χ, and a new binary tree R with ρ as its root node is created.

After initialization, the main loop of the algorithm identifies the largest (if any) polygon $P \in K$ that intersects with any line segment $\rho \in R$ and extracts all *visible* vertices V of P, filtered as described in Section 2.3.6. Next, these vertices are split

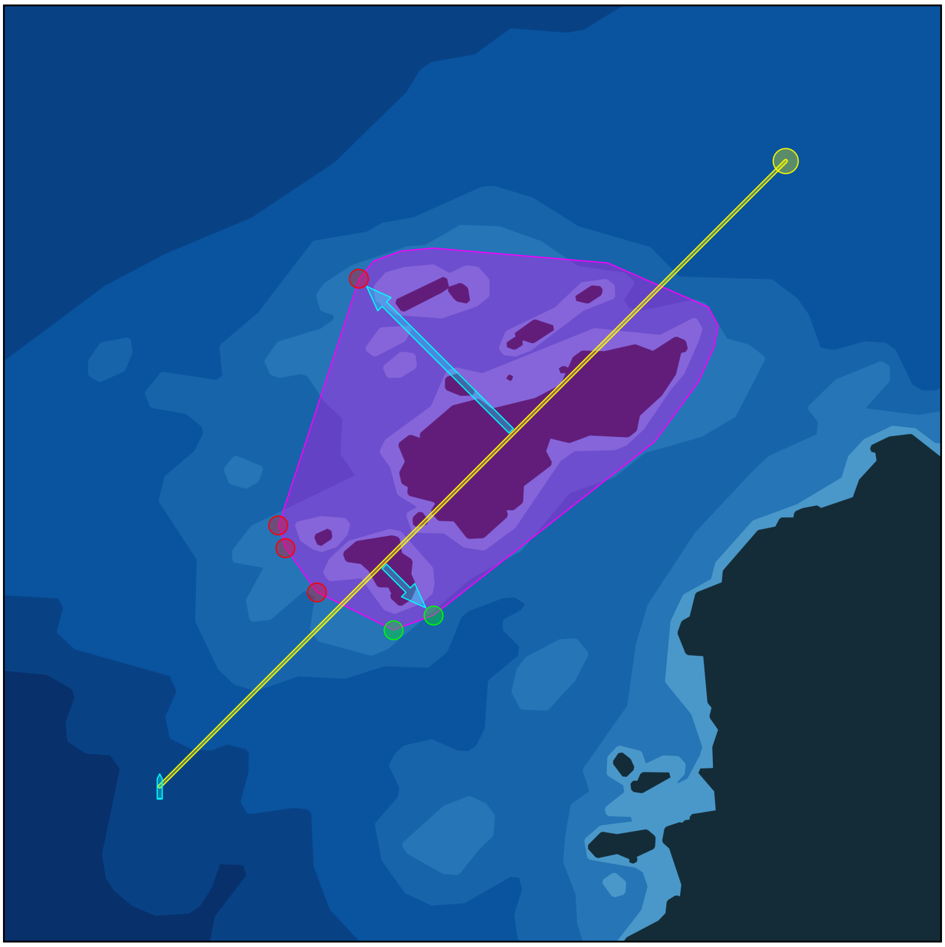


Figure 2.21: Main loop step visualization of the path planning algorithm.

into two sets of *left* and *right* (Λ and Γ, respectively) based on their positions with respect to the line segment ρ. These are shown in Figure 2.21, given the colors red (port) and green (starboard), respectively. This grouping is computed by constructing a triangular polygon between σ, χ and each vertex, in that order. If the resulting polygon is counter-clockwise oriented (asserted using the Shapely method **object.is_ccw**), the vertex is located on the left side of ρ, and vice versa.

The vertices with the maximum distance from ρ in each group (shown in Figure 2.21 as cyan perpendicular arrows from ρ to each respective vertex) are selected as the new intermediate route waypoints λ and γ, i.e. the minimum distance required to circumnavigate the visible part of the obstacle P at each iteration. These waypoints are used to construct two separate splines of straight lines α and β consisting of two linear line segments each, from σ to χ via λ and γ.

These new line segments are subsequently added to the root node of the R tree, leaving two new leaf nodes of line segments sharing the same end target point at χ. If any of the line segments in the resulting tree intersects with any polygon P of K, this process is repeated for that particular line segment, potentially creating more branching nodes along its respective route alternative.

Note that redundant further branching along one side of the obstacle is prevented by the fact that all vertex candidates in that special case are sorted into the same Λ or Γ set. The end result of the algorithm for the demonstrative example case is presented in Figure 2.22, in which two route alternatives in red and green have been constructed with several intermediate waypoints, generated by repeated iterations of the main loop of the algorithm. These path alternatives may subsequently be used by other navigational optimization schemes, e.g. to select the optimal path

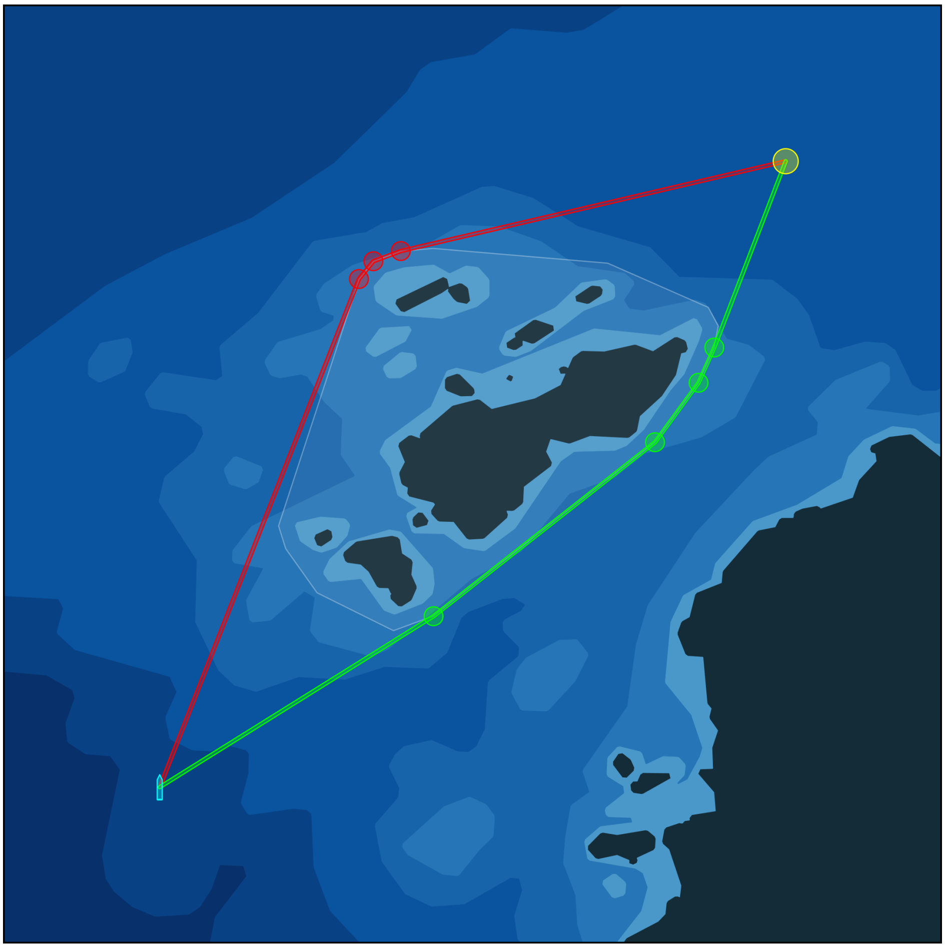


Figure 2.22: Path planning end result visualization of two alternative routes.

with respect to resource consumption or time. Thus, several relevant methods and visualization capabilities of the SeaCharts package is demonstrated.

2.3.6 Visibility-based enclosing circles

Throughout development of various applications such as ANS, simplifications may be implemented in order to facilitate faster computation. Advanced shapes such as polygons with many vertices or irregular forms can be transformed into circular approximations, such that only a single point in space along with a radius can be used for rapid spatial calculations in an otherwise complex environment. This may be useful for formulating convex constraints and optimization costs.

Extending this further, any complicated environment may be closely approximated through disjoint or overlapping sets of circles or disks of various desired resolutions. Consequently, it may be useful to demonstrate an approximation technique for simplifying polygons into circles, using available methods of the SeaChart package.

Figure 2.23 presents the results of the example algorithm **EnclosingCircles**, which calculates *local* polygon approximation circles analogous to the concept of minimal enclosing circles, based only on the currently visible shoreline from any given ship position.

Algorithm 2 outlines an overview of the construction of the visibility-based enclosing circles. Similarly to solving the smallest-circle or minimal enclosing circle problem, the algorithm attempts to construct an enclosing circle that spans all

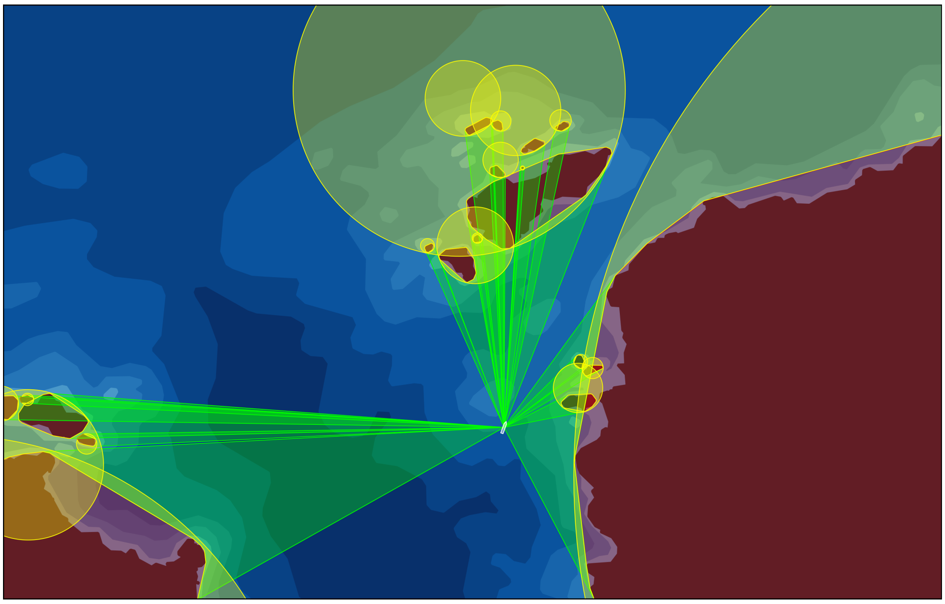


Figure 2.23: End result demonstration of overlapping visibility-based enclosing circles.

Algorithm 2 EnclosingCircles

```
Input: grounding obstacles $G$, horizon disk $D$,
       ship center $s$, distance buffer $\Delta d$
Output: enclosing circles $C$ of obstacles as seen from ship
  procedure EnclosingCircles($G, D, s, \Delta d$)
      $C \leftarrow \varnothing$
      $I \leftarrow$ spatial intersection of $G$ and $D$
      for all $P \in I$ do
          $K \leftarrow \varnothing$
          $H \leftarrow$ convex hull of $P$
          $V \leftarrow$ remove nonvisible vertices from $H$
          for all combinations of $v_1$, $v_2$, $v_3 \in V$ do
              $b_1 \leftarrow$ perpendicular bisector of line $v_1$-$v_2$
              $b_2 \leftarrow$ perpendicular bisector of line $v_2$-$v_3$
              $p \leftarrow$ intersection point of $b_1$ and $b_2$
              $r \leftarrow$ distance between $p$ and $v_2 + \Delta d$
              $c \leftarrow$ circle with center point $p$ and radius $r$
              $K \leftarrow$ candidate circle $c$
          end for
          $C \leftarrow$ max($K$) w.r.t. open water*
      end for
  end procedure
```

**area of unobstructed water between ship and circle*

of the *visible* vertices of a grounding obstacle (i.e. island or land) polygon within a relatively small horizon circle around the ship, such that the overlap between the area of the open water seen from the center of the ship and the constructed circle is minimized. Inversely, the open water seen between the ship and the constructed circle is maximized, such that there is minimal discrepancy between the constructed circle and its enveloped polygon. This is based on the assumption that the enclosing circles should map to its original polygon most accurately along the shoreline closest to the ship, as viewed by the perspective of the onboard navigator. Thus, for short term obstacle avoidance purposes, only the immediate surrounding grounding obstacles are acknowledged, disregarding unnecessary considerations of land masses hidden behind obstacles the ship might potentially hit if moving in any straight line from its current position.

Figure 2.24 presents a diagram showing some of the variables from Algorithm 2 for an example of a single non-convex grounding obstacle $\in G$. The algorithm is firstly initialized by intersecting all grounding obstacles (land and shore polygons in this example) with the dynamic horizon disk D, producing a new set of polygons I. In Figure 2.24, the resulting example polygon is shown in red color, in which the original grounding obstacle with eight vertices is intentionally clipped by the horizon disk D (here shown as a quadrant) in dashed lines. In Figure 2.23, however, the entire environment is chosen as the horizon given to the algorithm. In the main

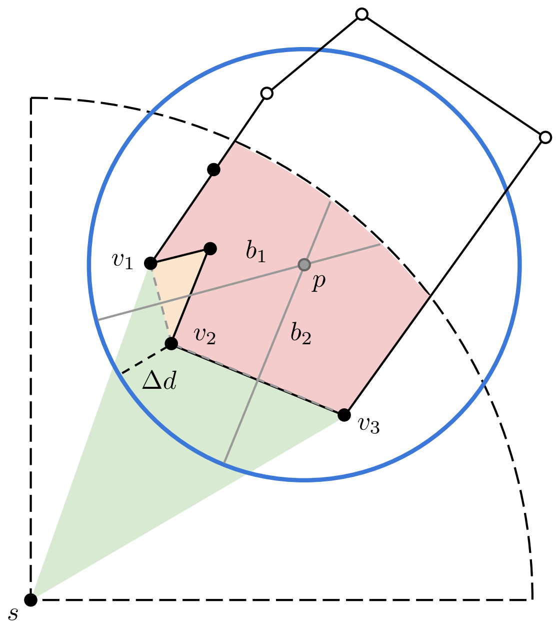


Figure 2.24: Diagram of the construction of perpendicular bisectors in Algorithm 2, with open water* shown in green, I in red, H as the union of red and orange, and the resulting enclosing circle c in blue.

loop, each polygon P is used as a basis to compute its individual enclosing circle.

In order to reduce the number of vertices for further computations, the convex hull H of P is calculated – concave crevices or pockets of any polygon are ignored for high-level navigation purposes. The resulting polygon may be identified as the union of the red and orange regions in Figure 2.24. Thus, it is immediately clear why all polygons are trimmed along the horizon boundary. In a situation in which there exists e.g. a land mass significantly encompassing the current ship position such that the vessel is located within a non-convex crevice, the convex hull of this polygon would remove the feasible navigation area of interest in its entirety.

Moving forward, only *visible* vertices V of H are considered, in accordance with the principle of disregarding all obstacle topology hidden behind the closest shoreline in any direction from the ship center s. The visibility of each vertex is readily examined by asserting that the line constructed from s to the vertex in question does not intersect the interior of H, not including its exterior (boundary). In the simple example in Figure 2.24, the only visible vertices are v_1, v_2, and v_3 exactly. The secondary inner loop repeats the final steps of the procedure for *any* combination of three vertices from V, i.e. all possible ways to construct two lines from three visible obstacle vertices.

Next, perpendicular bisectors $b_{1,2}$ for both visible vertex lines between v_1 and v_2, and v_2 and v_3, respectively, are calculated. By definition, $b_{1,2}$ are perpendicular lines passing through the midpoint of the pair-wise vertex lines, and as such are extended in each direction with respect to the original environment scope. Thus, the intersection point p between b_1 and b_2 is in general computable, unless the vertex lines are parallel. If this occurs, the candidate is silently disregarded.

Lastly, a candidate circle c is constructed from the intersection point p as its center point, and the radius r of c is set equal to the distance from p to v_2 plus the given input distance buffer Δd. All of the enclosing circles constructed by the inner loop are stored in a set of circle candidates K, ultimately sorted by the area of open water left between the ship and the obstacle by each c (shown in green). The more open water is still remaining between the ship and the constructed circle, the more accurate the circle approximates the obstacle boundary shape given the perspective of the ship at the current time instant. The circle with the maximum area of unobstructed water between the ship and itself is added to the set of enclosing circles C.

The results of Algorithm 2 may be verified in the example demonstrations displayed in Figures 2.23 and 2.25, with different reference points as ship centers. In Figure 2.23, the convex hulls of each polygon considered in Algorithm 2 are shown in red, and the end result enclosing C are shown in yellow. Green polygons highlight the visible "open water" between each red convex hull and the ship center,

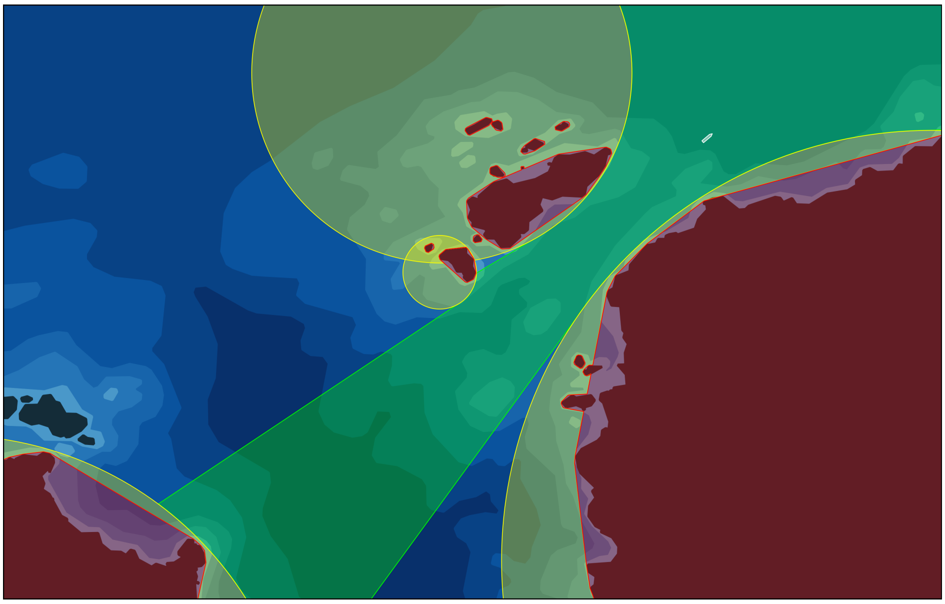


Figure 2.25: End result demonstration of filtered visibility-based enclosing circles and field of view visualization.

i.e. the water surface between all visible shorelines as seen from the ship. It is clear that every vertex of *visible* land or obstructed water is strictly contained within the horizon disk D, and that each obstruction polygon is assigned exactly one enclosing circle. Note however that the entirety of the convex hull of each grounding obstacle need not be fully enclosed by the resulting circle, as is slightly discernible on the second small isle from the west boundary of the environment. Using this method, any radial sector around the ship not covered in green or yellow is in effect considered completely open water, given the specific horizon and ship center.

This intuitive interpretation and visibility classification may also be useful for field of view procedures, e.g. simulating radar images of a ship's surrounding environment. By identifying all visible shoreline edges within the horizon, one may construct and apply radar-based techniques to a separate layer of the simulated environment for additional situational awareness and decision-making algorithms. Figure 2.25 shows an example visualization in which a different reference ship center is used to produce a *filtered* view of the resulting enclosing circles and open waters according to the principle of sight lines, further demonstrating the capabilities of the SeaCharts package.

Here, all minor circles fully encompassed in or located behind other circles are disregarded, and the polygons of visible open waters in green are merged and adjusted appropriately, producing the ship's field of view with respect to nearby obstacles. Notice how the circle of the large land mass to the east has changed considerably given the relocated reference point, and that the small group of isles to the west are hidden behind the yellow circles to the north of the plot.

Furthermore, the green open water polygons once again serve as the optimization objective for the end results of Algorithm 2. In Figure 2.25, only the resulting green region is considered unobstructed or navigable waters. This metric is in this work selected on the premise that only the visible exterior of any nearby polygons is considered e.g. with respect to reactive anti-grounding or collision avoidance, and that any circular boundary completely covering an irregular shoreline should minimize its overlap with otherwise unobstructed open water. The effects of this area maximization are considered adequately sufficient, by comparison of the red polygons against the constructed enclosing circles within the horizon.

2.3.7 Dynamic risk optimization

Research on autonomous ships involve (online) risk analysis with respect to anti-grounding and collision avoidance. In this section, the package methods established in the previous sections are further demonstrated by a numerical gradient-based ANS. The example application utilizes functionality and attributes of the *SeaCharts* package to construct an optimal control problem (OCP), transform it into a NLP and repeatedly solve it during runtime. Algorithm 3 presents a simplified overview of the main ANS procedure for demonstration purposes, based on MPC.

Algorithm 3 ModelPredictiveControl

```
Input: ownship state $x_0$, grounding obstacles $G$
Output: simulated ship trajectory along planned path
  procedure ModelPredictiveControl($x_0, G$)
    $d(x, g) \leftarrow$ Shapely distance-to-polygons method
    $f(x) \leftarrow$ formulate risk cost function using $d(x, g)$
    $x_s \leftarrow x_0$
    while not arrived and $risk < threshold$ do
      $\Pi \leftarrow$ construct new NLP using $f(x_s)$
      $s \leftarrow$ optimal solution of solved $\Pi$
      $u \leftarrow$ first control step of $s$
      $x_s \leftarrow$ apply $u$ to simulate next ownship state
    end while
  end procedure
```

A grounding risk cost function (recall Figure 2.7) is formulated mathematically as $f(x)$ where x is a state vector, based on the Shapely **distance** method applied to all nearby grounding obstacles G. The distance function is denoted as $d(x, g)$, where $g \in G$ is each individual grounding obstacle polygon within the horizon. Additional mission constraints and risk thresholds for emergency management [38] is considered outside the scope of this discussion.

In this work, $f(x) = \sum_g \gamma \cdot \exp(-\frac{d(x,g)}{\lambda}) + w(x, g)$ where γ and λ are tuning parameters. This form is chosen to scale an abstract measure of grounding risk cost by the distance to all grounding obstacles such that the risk gradient is exponentially larger closer to land. $w(x, g)$ is an additional wind disturbance cost to be discussed later. In the environment plot of Figure 2.26, subsequent ship poses of a simulated ship trajectory are shown in yellow.

The colored arrows attached to each ship pose are visual representations of the risk gradients produced by each respective obstacle polygon on each side of the ship path. The direction of each arrow at every time interval is equal to the direction of the unit vector from the closest point of an obstacle polygon and to the center of the ship. The magnitude or length of each arrow increases closer to land due to the inverse exponential scaling, and may as such aid in demonstrating the effects of the risk-based anti-grounding costs for autonomous control.

Figure 2.27 shows an alternate view of the same simulation plot, this time visualizing the wind-related risk gradients produced by the $w(x, g)$ term of $f(x)$ for a wind disturbance with velocity equal to 10 m/s and direction equal to 30° relative to the North axis. Here, the risk magnitude is proportional to the (positive only) scalar product between the wind disturbance vector and the vector from the ship to each grounding obstacle g, and are similarly to previously displayed as risk gradients directed away from the grounding obstacles [38].

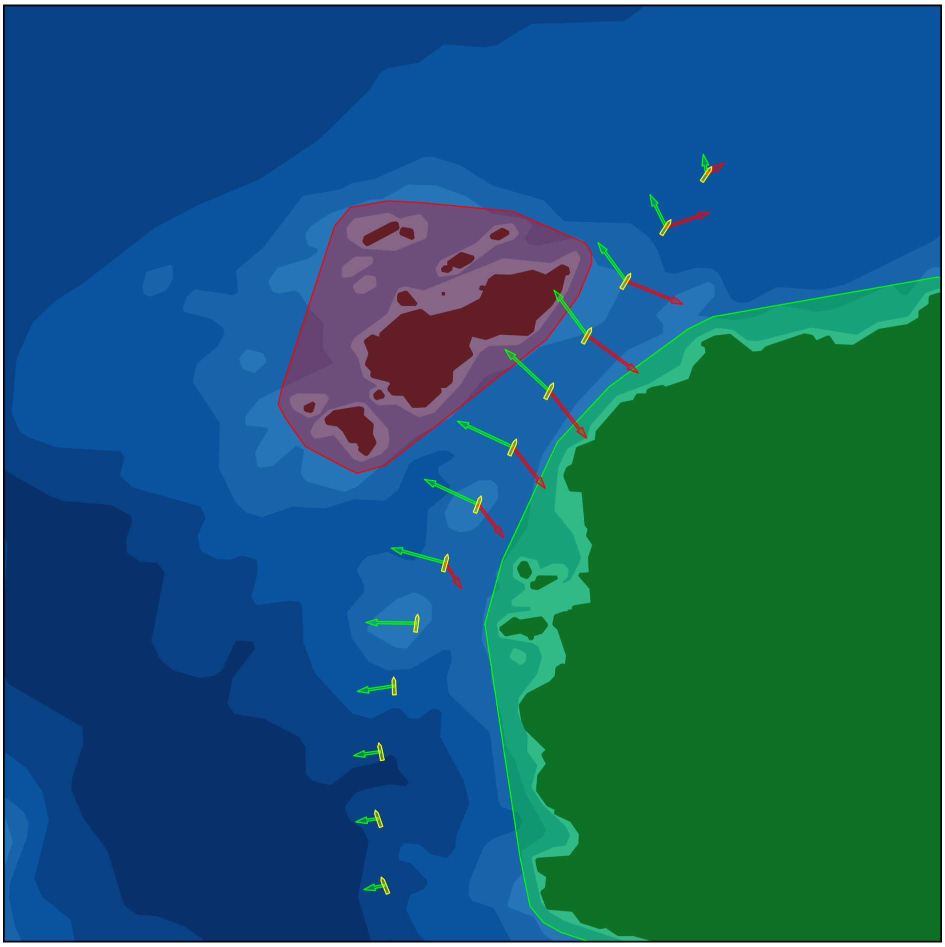


Figure 2.26: Distance-based risk gradient vectors visualization.

Note how the length of the vectors are only significant when the obstacles are located in an onshore wind direction relative to the ship position, given the scaling based on the scalar product between the wind vector and the vector to the nearest point of an obstacle. Thus, the diminishing vector arrows defined by the green (starboard) obstacle shown for the earlier time intervals of the simulation are negligible and consequently not visible during the later intervals. The scaling factors used between Figures 2.26 and 2.27 are not proportional to the terms of the cost function utilized by the ANS , and are adjusted for visual clarity in the example demonstrations.

2.3.8 Path following

Figure 2.28 considers a more complex path following example in order to further demonstrate potential usage of the visualization tools in the SeaCharts package.

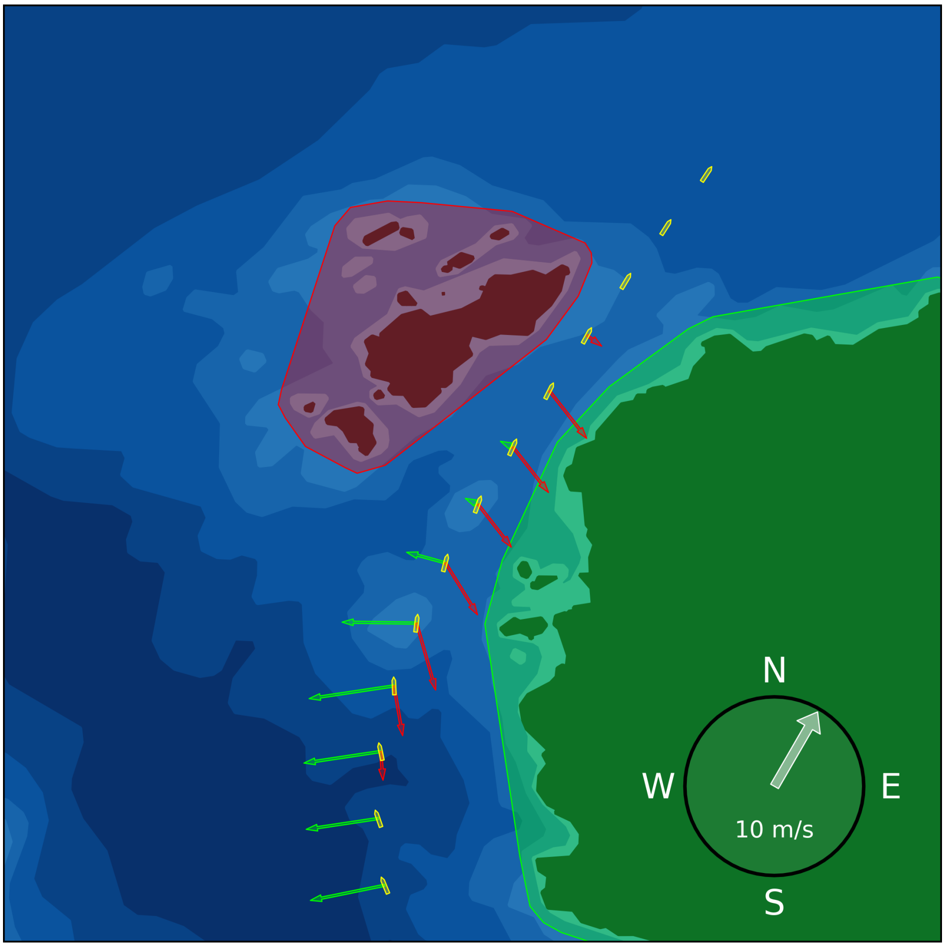


Figure 2.27: Wind disturbance scalar products visualization.

Here, the red ship pose is the initial ship state and the yellow ship poses are part of a discretized pre-planned path (i.e. before optimization or simulations) calculated by the ANS given four route waypoints in green (the last one off-screen), written to the ship pose file for some time horizon and an appropriate sampling interval. Similarly to Figure 2.14 in Section 2.2.5, the planned ship poses are calculated only by simple trigonometry as an initialization step (warm start) of the MPC algorithm, and may as well be valuable to visualize during algorithm demonstrations.

Another example view of the same environment and simulation run is shown in Figure 2.29, in which the past ship pose trail in white is also shown behind the red ship pose for the current simulation time step. Notice how the yellow future predictions in this example have been computed to comply with the dynamics of the ship, as a result of the trajectory optimization performed by the external ANS. This simple two-part example highlights the flexibility resulting from the

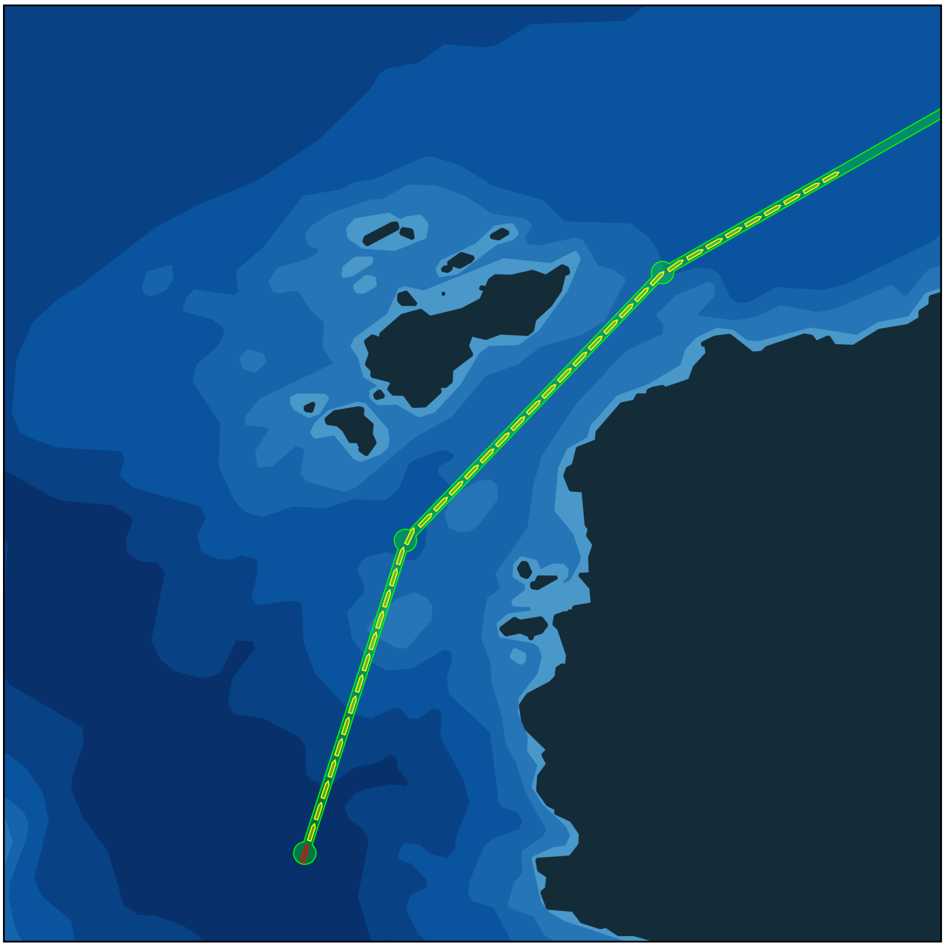


Figure 2.28: Vessel trajectory initialization along the pre-planned path.

visualization module of the SeaCharts package not being dependent on any past or future inputs for plotting of temporal information, and is considered one of the main contributions of this work.

The methods discussed above are focused on online analysis of distance-based risk related to grounding obstacles. However, it is proposed that the anti-grounding approaches described in this section may similarly be applied to collision avoidance e.g. based on vessel safety domains [28], given the general spatial formulations presented and the methods available to the SeaCharts package [85]. If external procedures for predicting or measuring velocities and/or intent of other vessels are included in an ANS, polygon-based risk analysis techniques for grounding obstacles may in general be fused with reactive collision avoidance methods to further enhance the potential of autonomous path and trajectory planning for autonomous ships.

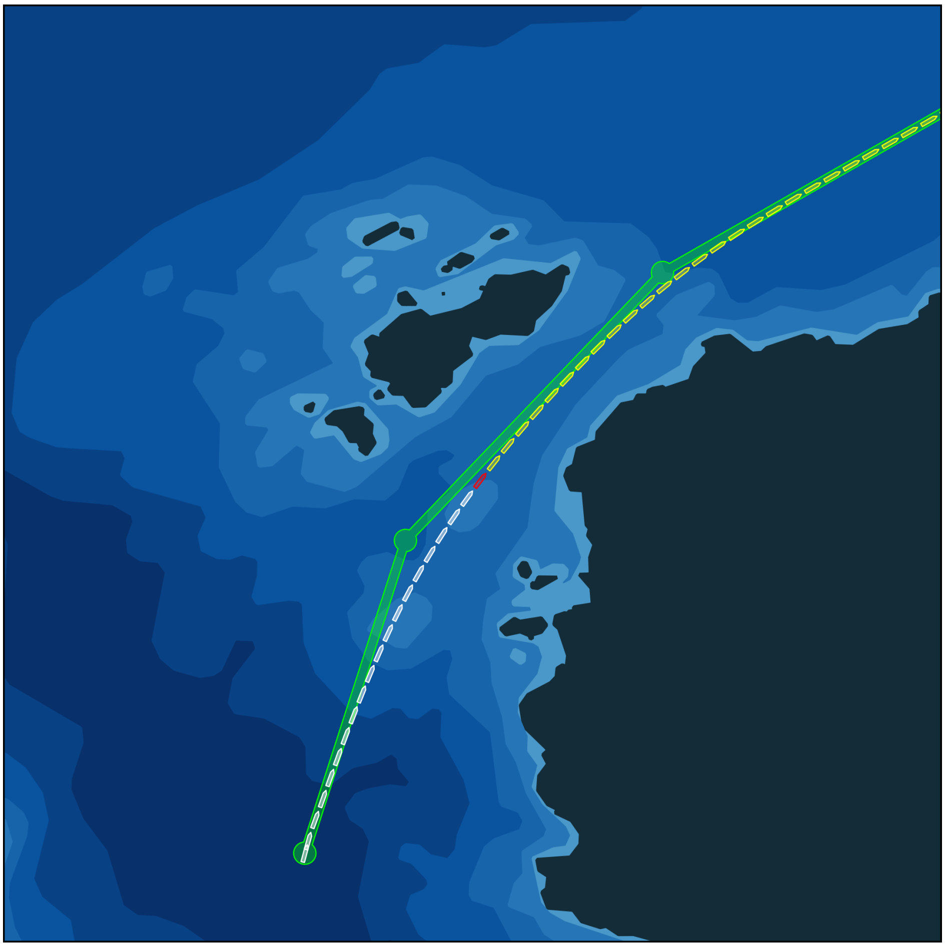


Figure 2.29: Vessel future predictions and past trail during MPC simulation.

2.4 Discussion

This section sums up some of the limitations and areas of improvement for the developed API, as well as points that may provide the basis for future work.

The SeaCharts API is a Python-based package for spatial visualization and computation, targeted at providing methods for fast prototyping and efficient research. As this work presents the very first version of this package, there is vast potential for improvements. Specifically, the API currently only supports the UTM coordinate system / map projection and the FGDB format for spatial data. Moreover, the features that are inherently loaded by the package are currently defined to comply with the feature labels defined by the Norwegian Mapping Authority. Note however that these labels if desired may be added or replaced directly in the source code after installation, for any feature names provided in the FGDB format. Lastly, the installation process may for some be cumbersome if users attempt to install and use the package in non-empty (virtual) environments, due to possible package version mismatches or support conflicts. The package currently uses the *Intel® oneAPI Math Kernel Library* [86] for Numpy/Scipy, which must be properly supported by the environment using the API.

Given the current status of the SeaCharts package and the notes above, it is recommended to continue the development of the API by improving upon its limitations, as well as adding new features and interface methods. For instance, support for the most commonly used coordinate systems or map projections (such as the equirectangular or plate carrée projection) may be useful to integrate into the package. Similarly, support for other file formats for spatial databases may prove beneficial to many potential users. Another suggestion for future work is to attempt to streamline the installation process, e.g. by utilizing the Intel® Distribution for Python. Several additional interface methods may also be added, such as e.g. convenience methods for depth sampling at any location in the plane, and the possibility to change the coordinates and size of the bounding box of the ENC main class during runtime. These are just few of the potential changes and additions that may be made to the SeaCharts package, in order to make the API useful and more practical for researchers and developers within maritime path planning, optimal control, and obstacle avoidance.

2.5 Conclusion

A shortage of versatile and open-source API with simple and user-friendly methods for spatial visualization of maritime environments for research and development has been observed in the literature. In an attempt to fill this need, the open-source Python package *SeaCharts* was implemented and presented in this work. The package includes demonstrated methods for reading and parsing depth data of known formats, spatial operations for polygon merging and simplification, user-specified features filtering and extraction, visualization of environments, vessels and mission objectives, as well as interface methods for use by external programs such as autonomous systems using spatial data for navigation. Additionally, algorithms for enclosing circle approximations and simplified procedures for path planning and optimization was presented in order to demonstrate potential usage of the package. Ultimately, this API may prove useful for high-level autonomous path planning, control, obstacle avoidance and simulation in maritime environments, by facilitating combined usage of convenient spatial computation and visualization methods for autonomous navigation.

Part II

Model predictive control

Chapter 3

Autonomous ship emergency management

This chapter is based on the publication

[38] **S. Blindheim**, S. Gros, and T. A. Johansen, “Risk-Based Model Predictive Control for Autonomous Ship Emergency Management,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 524–14 531, 2020, 21st IFAC World Congress. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1456

The method and simulations were developed by S. Blindheim, under the supervision of S. Gros and T. A. Johansen. The first draft was written by S. Blindheim, and was revised by S. Gros and T. A. Johansen.

3.1 Introduction

This chapter focuses on determining sequences of control actions to be taken in maritime emergency situations, in which it is not deemed appropriate to – or the ship is not able to – operate normally. The motivation behind this work is the ever-increasing desire to further reduce both operational costs and risks during shipping operations, particularly by moving personnel normally on board the vessel to onshore control centers. To achieve this, ships need increased autonomy and onboard decision-making capabilities. MPC has shown great results for autonomous vehicle steering [87], ship heading control [30], path following [31] and collision avoidance [32], [33]. However, these systems usually have strictly defined operational constraints or limited available decision spaces in which they are explicitly allowed to make autonomous decisions. Conditions such as these are normally the default operational stages, like the transit phase of a fjord-crossing autonomous ferry. In order to reach higher levels of autonomy, a more high-level supervisory system for risk or threat assessment and decision-making [88]–[90] for maritime operations is needed. Thus, the main purpose of this chapter is to investigate the use of MPC for handling emergency situations that are normally taken care of by human operators, through the use of some risk model and optimization-based decision-making. The approach is summarized as follows. The ship control is performed using a receding horizon approach, based on a dynamic ship model, a cost function and operational constraints. Each term in the constructed cost function targets different aspects of trajectory planning during normal operations and various emergency scenarios. Specifically, the developed algorithm handles a selection of abnormal or hazardous operational situations in which some degree of uncertainty is involved. As such, the novel contribution of this work is to include a separate *risk* term as an additional optimization cost, which makes it possible to address the uncertainty inherent in emergency scenarios directly. This term combined with other costs for resource management and mission objectives may collectively form a total emergency management algorithm, capable of handling all of the presented scenarios simultaneously. The resulting risk-based decision-making method may in turn serve as a foundation for a decision support system for human operators and as an autonomous navigation system for fully autonomous vessels.

3.2 Problem description

3.2.1 Scope and simplifications

The main objective of this work is to control the trajectory of a ship along a preplanned path in a challenging maritime environment, demonstrated by simulating a crossing through a strait with grounding obstacles on both sides. A simplified dynamic environment is used in this work, in which only variables related to the horizontal movement of the ship position are considered. The ship model is equipped with two freely rotating azimuth thrusters (one at the bow and one at the stern) with given maximum power specifications, and wind and currents velocities are assumed to be constant. No collision avoidance or sensor data quality handling is considered in this work, as these concepts are assumed to be added as natural extensions in a more exhaustive system [10]. Docking is also disregarded here, as it may be viewed as a separate control mode. Lastly, the approach presented in this work assumes that the ship and/or the operator is able to stop or react quickly when the risk is too large. However, it is considered a trivial task to appropriately increase the related risk coefficients to account for the stopping dynamics of the anchor drop or other significant delays as a consequence of higher velocities. These simplifications and approximations are used to develop a simple model serving as a proof of concept.

3.2.2 Failure modes and emergency scenarios

A collection of scenarios are presented in Table 3.1 to showcase the proposed method in this work:

1. *Impaired thrusters*
In this failure mode, the propulsion system has reduced maneuvering capabilities. No wind disturbances are assumed. **a)** Both thrusters lose the ability to rotate for a period of time, leaving the ship with constant thruster azimuth angles. Steering along the path is achieved by changing thrust magnitude only, until azimuth rotation capabilities are restored. **b)** The bow thruster goes offline. Thus the MPC scheme must use reduced degrees of freedom, i.e. the stern thruster only, to complete its mission.
2. *Total blackout*
The ship experiences a complete loss of propulsion due to a *temporary* power blackout, until the crew is able to restart the engines. Moderate wind disturbances lead to drifting. If and when to drop the anchor is continuously assessed by the algorithm. **a)** If the ship recovers its propulsion capabilities before the grounding hazard is too large, an alternative trajectory is calculated after drifting away from the original path. **b)** An anchor drop is triggered if the maximum grounding risk threshold becomes violated.
3. *Strong winds*
The increased grounding risks due to exceedingly strong winds are assessed in order to perform sufficiently safe control actions. **a)** A reference scenario demonstrates how the added risk term contributes to adjustments in the ship trajectory close to grounding obstacles. **b)** Crossing the strait is deemed too dangerous due to strong winds. As a result, the ship holds its position and waits for improved weather conditions for some time. **c)** The ship avoids the narrow strait in its entirety and opts to navigate around the nearby smaller isles, as a result of an alternative risk cost tuning approach.

Table 3.1: Demonstration scenarios

3.3 Mathematical modeling

3.3.1 Variables and reference frame definitions

First, the locally flat North-East (NE) coordinate frame $\{n\}$ and the body coordinate frame $\{b\}$ are defined as presented in Figure 3.1. The variables are defined as follows: x and y denote the position of the ship along the North and East axes, u and v are the surge and sway velocities of the ship, X and Y are the surge and

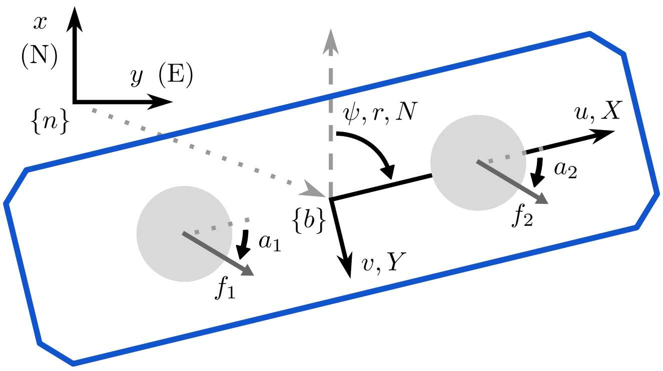


Figure 3.1: The model variables and coordinate frames used in this work.

sway forces of the ship, ψ, r and N are the yaw angle, velocity and moment of the ship, and $\boldsymbol{a}_t = \begin{bmatrix} a_1 & a_2 \end{bmatrix}^\top$ and $\boldsymbol{f}_t = \begin{bmatrix} f_1 & f_2 \end{bmatrix}^\top$ are the azimuth angles and propulsion forces of the ship's stern and bow thrusters, respectively.

3.3.2 Ship model and dynamics

The model variables are given in Table 3.2. From (2.1) and (2.2) in [91], the reduced three-dimensional ship kinematic and kinetics equations in the horizontal NE-plane (disregarding Coriolis, wave, ballast, buoyancy or gravitational forces) are given as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.1}$$

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}(\boldsymbol{\tau} + \boldsymbol{d} - \boldsymbol{D}\boldsymbol{\nu}) \tag{3.2}$$

where $\boldsymbol{d} = \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{current}}$ is the system disturbance vector. The ocean currents forces $\boldsymbol{\tau}_{\text{current}} \triangleq 0$ in the example simulations presented in this work, for simplicity. The wind forces are defined as

$$\boldsymbol{\tau}_{\text{wind}} = \begin{bmatrix} -c_x \cos(\gamma_w) A_{Fw} \\ c_y \sin(\gamma_w) A_{Lw} \\ c_n \sin(2\gamma_w) A_{Lw} L_{oa} \end{bmatrix} \frac{1}{2} \rho_a V_w^2 \tag{3.3}$$

from [91], where V_w is the wind velocity relative to the ship's velocity, $\gamma_w = \psi - \psi_w - \pi$, and ψ_w is the clockwise wind angle relative to the North axis. The wind coefficients c_x, c_y and c_n are in this work set to 0.7, 0.8 and 0.1, respectively. See Section 3.5.3 for all remaining model parameter definitions and their given values, and see [91] for generalizations to other propulsion and steering configurations.

Entity	Symbol	Elements
North-East ship position	$\boldsymbol{p}^n_{b/n}$	$\begin{bmatrix} x \\ y \end{bmatrix}$
North-East ship attitude	$\boldsymbol{\Theta}_{nb}$	$\begin{bmatrix} \psi \end{bmatrix}$
Ship position and orientation	$\boldsymbol{\eta}$	$\begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix}$
Body-fixed linear velocity	$\boldsymbol{v}^b_{b/n}$	$\begin{bmatrix} u \\ v \end{bmatrix}$
Body-fixed angular velocity	$\boldsymbol{\omega}^b_{b/n}$	$\begin{bmatrix} r \end{bmatrix}$
Linear and angular ship velocities	$\boldsymbol{\nu}$	$\begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix}$
System state vector	$\boldsymbol{x}$	$\begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \end{bmatrix}$
Principal rotation matrix	$\boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})$	$\begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}$
Ship pose Jacobian	$\boldsymbol{J}_\Theta(\boldsymbol{\eta})$	$\begin{bmatrix} \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix}$
North-East linear velocity	$\dot{\boldsymbol{p}}^n_{b/n}$	$\boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})\boldsymbol{v}^b_{b/n}$
North-East angular velocity	$\dot{\boldsymbol{\Theta}}_{nb}$	$\boldsymbol{\omega}^b_{b/n}$
Thrusters transformation matrix $s_* := \sin(a_*) \quad c_* := \cos(a_*)$	$\boldsymbol{T}(\boldsymbol{a}_t)$	$\begin{bmatrix} c_1 & c_2 \\ s_1 & s_2 \\ -l_x s_1 & l_x s_2 \end{bmatrix}$
Body-fixed propulsion forces	$\boldsymbol{f}^b_b$	$\begin{bmatrix} X \\ Y \end{bmatrix}$
Body-fixed moment (torque)	$\boldsymbol{m}^b_b$	$\begin{bmatrix} N \end{bmatrix}$
Ship forces and moments	$\boldsymbol{\tau}$	$\boldsymbol{T}(\boldsymbol{a}_t)\boldsymbol{f}_t = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix}$
Control input vector	$\boldsymbol{u}$	$\begin{bmatrix} \boldsymbol{f}_t \\ \dot{\boldsymbol{a}}_t \end{bmatrix}$
Constant damping matrix	$\boldsymbol{D}$	$\text{diag}(X_u, Y_v, N_r)$
Hydrodynamic added mass	$\boldsymbol{M}_A$	$\text{diag}(X_{\dot{u}}, Y_{\dot{v}}, N_{\dot{r}})$
Rigid-body ship mass	$\boldsymbol{M}_{RB}$	$\text{diag}(m, m, I_z)$
Total model mass	$\boldsymbol{M}$	$\boldsymbol{M}_{RB} + \boldsymbol{M}_A$

Table 3.2: Model terminology and definitions

3.3.3 Optimal control problem formulation

An OCP is defined as follows:

$$\begin{aligned} \min_{\boldsymbol{x}(.),\boldsymbol{u}(.)} \quad & \int_{t=0}^{T} \tilde{\phi}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\theta}(t))\ dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{d}(t)) \\ & \boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \leq \mathbf{0} \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad 0 \leq t \leq T \end{aligned} \tag{3.4}$$

where $\tilde{\phi}$ is a scalar stage cost function, $\boldsymbol{\theta}$ is a parameter vector, $\boldsymbol{x}_0$ is the initial state, T is the prediction horizon, and $\dot{\boldsymbol{x}}$ is given by the system dynamics (3.1) and (3.2). The constraints $\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t))$ are given as:

$$\begin{array}{ccccc} -f_{\max} & \leq & u_1 & \leq & f_{\max} \\ -f_{\max} & \leq & u_2 & \leq & f_{\max} \\ -\omega_{\max} & \leq & u_3 & \leq & \omega_{\max} \\ -\omega_{\max} & \leq & u_4 & \leq & \omega_{\max} \end{array} \tag{3.5}$$

where $\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\top$ from Table 3.2, and $f_{\max}$ and $\omega_{\max}$ are the maximum propulsion force and rotational turning rate of the azimuth thrusters, respectively. The solution to problem (3.4) will be deployed in a receding horizon fashion, yielding an MPC scheme.

3.3.4 Nonlinear programming

Next, the model is discretized in order to solve the problem numerically. The continuous time variable t is divided into a time grid of N intervals, defined by discrete time instants $t_k \in \{t_0, t_1, ..., t_N\}$. The system inputs are discretized as piecewise constant over that time grid, i.e. $\boldsymbol{u}_k = \boldsymbol{u}([t_k, t_{k+1}])$. The system state is discretized using a numerical integration function $\boldsymbol{x}_{k+1} = \boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k)$, based on the widely used *Runge-Kutta 4th order method.* The discretization allows one to treat (3.4) as an NLP by defining a vector of decision variables

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{x}_0^\top & \boldsymbol{q}_0^\top & \boldsymbol{u}_0^\top & \dots & \boldsymbol{x}_{N-1}^\top & \boldsymbol{q}_{N-1}^\top & \boldsymbol{u}_{N-1}^\top & \boldsymbol{x}_N^\top & \boldsymbol{q}_N^\top \end{bmatrix}^\top \tag{3.6}$$

where $\boldsymbol{q}_k$ is a vector of additional decision variables related to mission objectives to be defined in Section 3.4.1. Additionally, a parameter vector comprised of various control settings, desired states and coefficients through time is denoted as $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_0^\top & \dots & \boldsymbol{\theta}_N^\top \end{bmatrix}^\top$. The only parameters considered in this work are

$$\boldsymbol{\theta}_k = \begin{bmatrix} s_{\text{ref}} \\ \alpha_{\text{step}} \\ \boldsymbol{\sigma} \end{bmatrix} \in \mathbb{R}^{2+3J} \tag{3.7}$$

for each t_k where s_{ref} is a constant reference transit speed and α_{step} is a path progression parameter (see Section 3.4) across all N control intervals. The grounding obstacles are modeled as a union of J circles. Thus, $\boldsymbol{\sigma}$ is the collection of all grounding hazard vectors of the form

$$\boldsymbol{\sigma}_j = \begin{bmatrix} \boldsymbol{c}_j \\ r_j \end{bmatrix}, \qquad \boldsymbol{c}_j = \begin{bmatrix} x_j \\ y_j \end{bmatrix}, \qquad j = 1, ..., J \tag{3.8}$$

where $\boldsymbol{c}_j$ and r_j are the center point and radius of each obstacle, respectively. The resulting NLP is defined as

$$\begin{aligned} C(\boldsymbol{\theta}, \boldsymbol{x}_0) = \min_{\boldsymbol{w}} \quad & \phi(\boldsymbol{w}, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{w}) = \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{w}) \leq \mathbf{0} \end{aligned} \tag{3.9}$$

where $C(\boldsymbol{\theta}, \boldsymbol{x}_0) \in \mathbb{R}$ is the minimum cost generated by a given set of parameter values and initial conditions $\boldsymbol{x}_0$. The inequality constraints $\boldsymbol{h}(\boldsymbol{w})$ are given by (3.5), and the equality constraints $\boldsymbol{g}(\boldsymbol{w})$ hold the system dynamics:

$$\boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k) - \boldsymbol{x}_{k+1} = \mathbf{0}, \qquad k = 0, 1, ..., N-1 \tag{3.10}$$

The cost function ϕ is defined and discussed in Section 3.4.2. Note that the discretization chosen here is based on the *direct multiple-shooting* approach [92]. Because of the nonlinear dynamics and since the obstacles yield a non-convex feasible set, the NLP (3.9) is non-convex. As a result, the goal is to compute a feasible and *local* optimal solution for a given control horizon N and initial conditions. The preplanned path and ship speed reference parameters are used to calculate a reasonable initial guess for the ship trajectory. Rather than using hard constraints in addition to the ship dynamics and the natural input constraints, only costs balancing is utilized to achieve the desired control behavior. This ensures feasibility of the NLP solutions.

3.4 Planning and decision-making

The decision-making of the MPC algorithm consists of two main tasks: the planning of the ship trajectory achieved by propulsion and steering control, and the decision to drop the anchor in emergency situations. If the planning algorithm is unable to produce a trajectory that does not violate the given grounding risk thresholds, emergency procedures are triggered in order to minimize damages and costs – such as dropping the anchor. It may be noted that this can also be achieved by a formulation based on mixed-integer programming, allowing for more complex decisions to be made during emergency situations. The following subsections describe the chosen path-following method, as well as how different mission objectives are evaluated and weighted in order to produce desired ship trajectories.

3.4.1 The path following method

There are many different methods readily available for following a preplanned path, e.g. by using a predefined stride each time interval along the path, or a line-of-sight method [93]. A time-invariant method is chosen in this work to generate consistent and robust solutions at any time interval. First, a preplanned path is chosen by designing a piecewise linear (spline) function given an initial position, discrete intermediate points and a destination. It is assumed that the path is designed such that fuel/resource consumption, time spent and distance traveled is considered (close to) optimal for the given mission. Next, the reference path is parameterized, giving the two-dimensional reference function

$$\boldsymbol{r}(\alpha) = \begin{bmatrix} x(\alpha) \\ y(\alpha) \end{bmatrix} \tag{3.11}$$

for calculating path points where $\alpha \geq 0$ is an advancement parameter acting as a decision variable along the preplanned path, and $x(\alpha)$ and $y(\alpha)$ are piecewise linear functions. As such, advancing along the path is a simple matter of increasing α. The desired ship speed along the path is furthermore established by penalizing ship transit velocities larger than the given reference speed s_{ref}. This is achieved by minimizing a speed penalty decision variable β, where

$$u^2 + v^2 \leq s_{\text{ref}}^2 + \beta, \qquad 0 \leq \beta \tag{3.12}$$

Collecting the additional decision variables into a vector for each time step through the control horizon N yields

$$\boldsymbol{q}_k = \begin{bmatrix} \boldsymbol{a}_t \\ \alpha_k \\ \beta_k \end{bmatrix}, \qquad k = 0, 1, ..., N-1 \tag{3.13}$$

and the NLP decision variable vector $\boldsymbol{w}$ is well defined.

3.4.2 Objectives and cost function definitions

In order to complete the NLP, a cost function to be minimized is constructed. In this research, the cost function is heuristically defined with the purpose of producing a safe ship trajectory that fulfills the mission objectives. The primary cost function is denoted as

$$\phi(\boldsymbol{w}, \boldsymbol{\theta}) = \sum_{k=1}^{N} \xi(\boldsymbol{x}_k, \boldsymbol{q}_k, \boldsymbol{q}_{k\text{-}1}) + \epsilon(\boldsymbol{u}_k, \boldsymbol{u}_{k\text{-}1}) + \rho(\boldsymbol{x}_k, \boldsymbol{\theta}_k) \tag{3.14}$$

where the individual cost terms are defined as follows:

i. The path progression cost function

$$\xi(\boldsymbol{x}_k, \boldsymbol{q}_k, \boldsymbol{q}_{k-1}) = \boldsymbol{\kappa}^\top \begin{bmatrix} ||\boldsymbol{r}(\alpha_k) - \boldsymbol{p}_k||^2 \\ ||\alpha_k - \alpha_{k-1} - \alpha_{\text{step}}||^2 \\ \beta_k \end{bmatrix} \tag{3.15}$$

where $\boldsymbol{\kappa} > \mathbf{0}$. These terms are responsible for driving the ship position $\boldsymbol{p}_k$ (trajectory) along the precomputed feasible path, through the constant path step parameter α_{step} and the reference function $\boldsymbol{r}(\alpha_k)$. The β_k term penalizes violations of the transit speed reference as detailed in Section 3.4.1. It is recommended that α_{step} is chosen such that $s_{\text{ref}} \approx \alpha_{\text{step}}/t_\Delta$, where t_Δ is the sampling period of the NLP.

ii. Next, the control input cost function is defined as

$$\epsilon(\boldsymbol{u}_k, \boldsymbol{u}_{k-1}) = \boldsymbol{u}_k^\top \boldsymbol{\Lambda} \boldsymbol{u}_k + (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})^\top \boldsymbol{\Delta} (\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \tag{3.16}$$

where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda}) > \mathbf{0}$ and $\boldsymbol{\Delta} = \text{diag}(\boldsymbol{\delta}) > \mathbf{0}$ are tuning matrices. These terms collectively help conserve power and reduce the input variations, consequently lowering environmental and operational costs.

iii. Finally, an ad hoc risk cost function is introduced to keep the risk levels present in the system acceptable:

$$\rho(\boldsymbol{x}_k, \boldsymbol{\theta}_k) = \sum_{j=1}^{J} (\mu_1 + \mu_2 \chi_j V_w) e^{-\zeta(||\boldsymbol{c}_j - \boldsymbol{p}_k|| - r_j)} \tag{3.17}$$

with $\boldsymbol{\mu} > \mathbf{0}$. Moreover, $\chi_j = \max(0, \hat{\boldsymbol{\iota}}_j \cdot \hat{\boldsymbol{\omega}})$ where $\hat{\boldsymbol{\iota}}_j$ is the unit vector from the ship to each obstacle center and $\hat{\boldsymbol{\omega}}$ is the unit wind direction vector. Note that the risk costs are not formulated as explicit constraints to ensure safe distances between the ship and obstacles. Rather, this formulation utilizes violatable risk costs in order to acknowledge that grounding risks may still be evaluated even if they are very high. Using exponential terms for the obstacle or grounding risk costs serves to strongly dominate the other objectives in the cost function, heavily favoring staying safe from grounding obstacles. The grounding risk sensitivity constant ζ may for this purpose be tuned for optimal behavior. Lastly, the dot product scales the wind force contribution toward the land obstacles in any orientation around the ship, i.e. increasing the risk close to an obstacle to the east of the ship if the wind is coming from the west, etc. Negative dot products are however set to zero, disregarding favorable winds with respect to perceived risks.

3.5 Implementation and settings

3.5.1 MPC scheme overview

Pseudo-code for the complete MPC algorithm is presented in Algorithm 4, and a more detailed rundown is as follows: Vectors, matrices, cost functions and custom functions for numerical integration are set up using the CasADi symbolic

Algorithm 4 MPC algorithm

```
Ensure: optimal control input
  initialize empty solution
  while not arrived at destination do
      if any thrusters online then
          construct NLP ← last state
          optimal states and inputs ← solve NLP
          current state ← extract first optimal step
      else
          current state ← simulate drifting ship
      end if
      if risk > accepted maximum threshold then
          activate emergency protocol
      else
          solution ← current state and control inputs
          last state ← current state
      end if
  end while
```

framework [39]. The implemented MPC module then solves one NLP for each control interval, using the nonlinear optimization package Interior Point OPTimizer (IPOPT) and the linear solver MUltifrontal Massively Parallel sparse direct Solver (MUMPS). However, if all inputs are constrained to zero, i.e. during the drifting ship or blackout scenarios, an open loop simulation is used instead. Finally, the total risk at each time step is evaluated through the risk cost model from (3.17). If the risk level rises above a given maximum acceptable threshold, the algorithm is terminated with the assumption that emergency procedures such as remote control or automatic drop of anchor would be applied.

3.5.2 Initialization and performance factors

Due to the non-convex nature of the control problem, the initial values given to each NLP for solving have a significant impact on its solutions. In this work, the current state and input vectors, the previous solution as well as the internal solver parameters are given to the next NLP to be solved as initial guesses (warm start), as described in Algorithm 4. However, for the first NLP solve, initial guesses are generated by assuming that the ship will follow the path with a velocity equal to the given reference transit speed. The sampling period of $t_\Delta = 30\,\mathrm{s}$ was chosen to have a reasonable balance between the slowest time constant of the ship dynamics and a desired long prediction horizon. Longer sampling periods give faster MPC tuning and testing, but for certain time-sensitive scenarios however, this may be inadequate. It is recommended that this interval should be reduced if used in applications. Similarly, a control horizon of $N = 40$ was chosen, resulting in a total of 570 decision variables for each NLP. This was set by assessing the solving time of the algorithm with respect to the quality of the computed solutions. Increasing the control horizon past this point seemed not to improve the NLP solutions noticeably, and the average solving time of 2 s across all scenarios was deemed appropriate compared to the total prediction horizon of 20 min. If the NLP solver does not converge to a solution within the maximum limit of 300 iterations, the last solution calculated is used. This maximum iteration limit was set such that the maximum solve time during any emergency scenario would be $27\,\mathrm{s} < t_{\Delta}$.

3.5.3 Model parameters and values

The model parameters and MPC settings are given in Table 3.3, based on the dimensions and onboard systems of a cargo vessel with a maximum surge velocity of ~15 knots. The rest of the initial and final values are all set to zero. The remaining cost coefficients of $\boldsymbol{\kappa}$, $\boldsymbol{\lambda}$, $\boldsymbol{\delta}$, $\boldsymbol{\mu}$ and ζ are set constant throughout all scenarios, and were empirically determined in order to produce desired trajectories: The path following and speed-related costs were first established to encourage path progress. Next, the input force costs were tweaked to smooth out gradients and reduce power consumption. Finally, appropriate risk costs were estimated by assessing the resulting trajectories during various wind angle and velocity configurations. This ad hoc evaluation approach should however be replaced by more systematic and robust methods for approximating or learning risk coefficient values in future works.

3.5.4 Visualization

Figure 3.2 illustrates how the slope of the separate risk cost term influences path planning and decision-making. The risk values along the z-axis are given as percentages of the maximum risk cost present at a selected time instant, and the x and y axes are given in meters. The surface plot indicates how the gradient of the obstruction- and disturbance-related terms naturally push the trajectory of the

Parameter	Symbol	Value	Unit
Path step size	α_{step}	75	m
Overall ship length	L_{oa}	75	m
Thruster arm lengths	l_x	33	m
Transit reference speed	s_{ref}	2.5	m/s
Frontal projected area	A_{Fw}	110	m^2
Lateral projected area	A_{Lw}	624	m^2
Max thruster force	$f_{\max}$	200	kN
Max thruster azimuth rate	$\omega_{\max}$	2.0	rpm
Viscous damping force surge	X_u	5.0×10^1	kN s/m
Viscous damping force sway	Y_v	2.0×10^2	kN s/m
Viscous damping force yaw	N_r	3.0×10^4	kN s/rad
Hydrodynamic added mass surge	$X_{\dot{u}}$	4.4×10^4	kg
Hydrodynamic added mass sway	$Y_{\dot{v}}$	8.6×10^5	kg
Hydrodynamic added mass yaw	$N_{\dot{r}}$	4.9×10^7	kgm^2
Rotational inertia yaw	I_z	9.8×10^8	kgm^2
Rigid-body ship mass	m	1.5×10^6	kg

Table 3.3: Model parameters

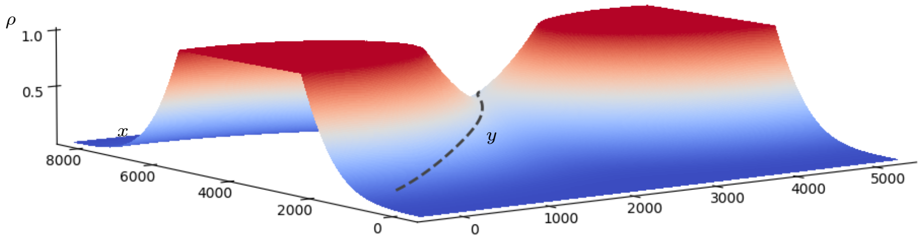


Figure 3.2: Surface plot of MPC risk costs at some time instant.

ship (the dashed line) down into the "valley" of lower costs between two grounding obstacle examples. This topography is similar to that of artificial potential fields and is unique for each time step, as well as for each *predicted* time step into the control horizon N. The risk term surface thus evolves dynamically based on the current ship position and the disturbance circumstances influencing the ship. That is, if the ship moves closer to an obstacle with wind disturbances directed toward it, the slope of the risk cost around that obstacle steepens and rises accordingly. Figure 3.3 presents the usage of an early version of the ENC API module SeaCharts [37] for visualization of the scenario simulations, which displays polygon data of a Norwegian fjord area shown using the Mercator projection. A color bar indicates the ocean depths for the surrounding maritime environment, here used for display

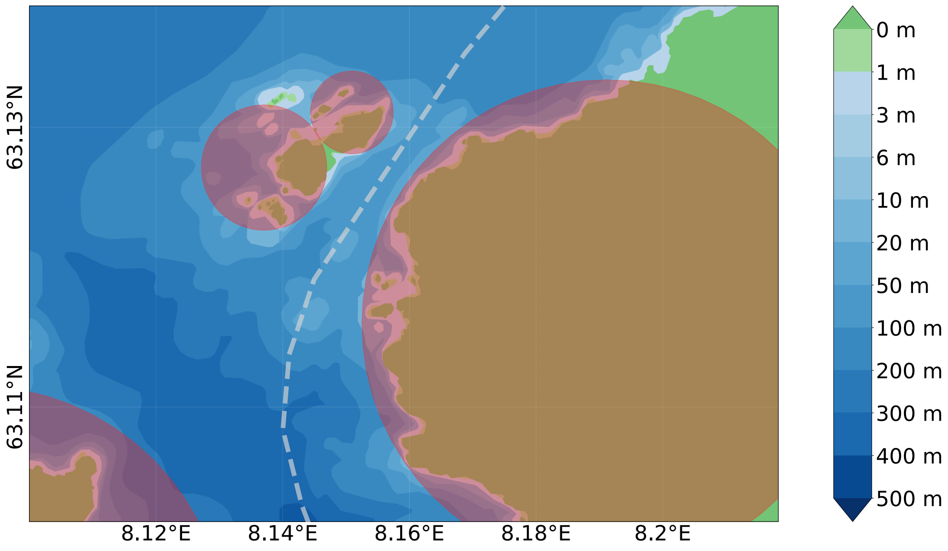


Figure 3.3: Visualization of the circular obstacles used by the MPC algorithm during cost function calculations.

purposes only. The preplanned reference path is shown as a dashed gray line. Additionally, red circles mark the grounding obstacles $\boldsymbol{\sigma}$ used by the MPC for cost function calculations. In this work, these obstacle circles are static, and are deliberately made simplistic for proof of concept. Methods for dynamic calculation of grounding areas more shallow than the maximum draft of the ship should however be used in applications. This is left to be further expanded upon in future works.

3.6 Results

Ceasing normal operations mid-transit is considered a costly action. The main goal of the decision-making algorithm is thus to assess and optimize the risk levels against other operational costs at each time interval, and determine to what degree the ship should follow its initial route within some safety threshold. This section presents the MPC performance of seven scenarios in a simulated maritime environment, to demonstrate the capabilities of the algorithm during extreme circumstances. System stability follows in general from optimality of solutions and by utilizing an adequately long horizon [94], as well as through low-level controllers used by the ship's propulsion subsystems. Figures 3.4 to 3.6 and 3.8 show the generated ship trajectories for the investigated scenarios, plotted every 2 min for visibility. Figure 3.7 shows time series plots of the azimuth thrusters during runtime.

3.6.1 Impaired thrusters

The defective propulsion system failure mode is split into two separate faults, with no wind disturbances for analysis purposes. Figure 3.4 shows both scenario simulations in red and yellow, as well as a trajectory during normal operations in cyan, for reference. It may be noted how the normal operations trajectory deviates slightly from the planned path, due to imposed thruster input costs and the internal prediction horizon each time step. This is a result of the preplanned piecewise reference path being linear and non-optimal. The red ship trajectory depicts the scenario (**1.a**) in which the azimuth thruster rotators are disabled after 5 min e.g. due to an auxiliary system fault, effectively leaving the ship with limited steering capabilities for a time period. However, the algorithm is capable of adjusting the thruster propulsion forces appropriately to continue along an almost identical trajectory until the thruster rotation capabilities are restored, by utilizing the current azimuth angles and varying their propulsion output appropriately. Straying away some distance from the path and closer to the shoreline is considered sufficiently safe, and the ship thus continues ahead despite the imposed system faults. In the second scenario (**1.b**) shown in yellow, the bow thruster shuts down after 3 min, and stays offline until the ship reaches its destination. It is nevertheless apparent that the ship is still able to continue its voyage safely and almost as quickly as the nominal trajectory, even with only the stern thruster available. Note that the algorithm has no access to information about future faults, and consequently is forced to operate with only its current knowledge each control interval. It is considered trivial to investigate less extreme scenarios than the total steering and propulsion loss cases presented here.

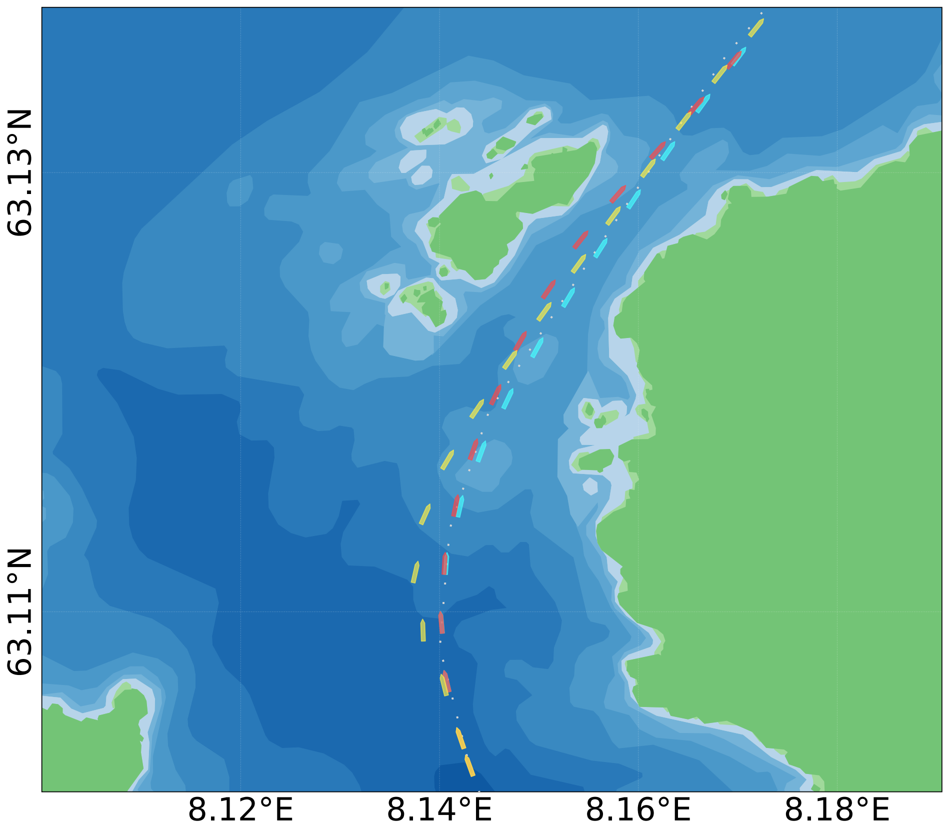


Figure 3.4: Simulations during normal operations (cyan), a temporary loss of steering power scenario (red), and a single offline thruster scenario (yellow).

3.6.2 Total blackout

In this failure mode, the ship experiences a power blackout and drifts due to moderate disturbances. An open-loop simulation produces new states each time step during this time period, as there are no available degrees of freedom through the zero-constrained system inputs. If and when to drop the anchor is consequently the only decision left to make in these circumstances. The act of dropping the anchor is assumed to be an exceptionally costly action, and is thus postponed as long as reasonably possible in order to give the crew a chance to restart the engines before doing so. Both scenarios have $V_w = 10\,\text{m/s}$ set constant for consistency and proof of concept. Color-coded risk gradient vectors are added to the ship trajectory plots every time step. These vectors show how the perceived risk cost penalizes staying close to the shoreline during difficult weather conditions. The arrows have lengths inversely proportional to $||\boldsymbol{c}_j - \boldsymbol{p}_k|| - r_j$, and are parallel to $-\hat{\boldsymbol{\iota}}_j$. Yellow to red arrow colors represent a medium to large scalar product χ_j, respectively. Figure 3.5 shows the first scenario (**2.a**) with $\psi_w = 90°$ (east), demonstrating how the ship may still salvage the situation and complete its mission if a blackout is recovered

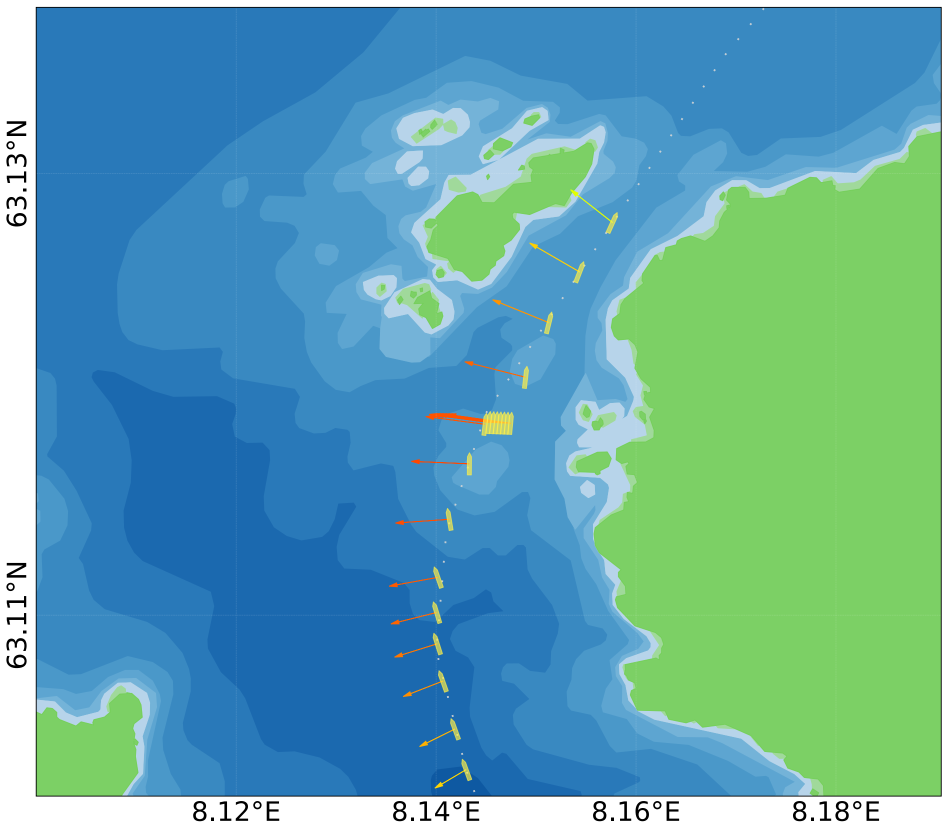


Figure 3.5: A temporary blackout and recovery scenario.

from quickly. Though the ship deviates from the planned path, the solutions of the NLP solver are able to swiftly find their way back to resume the mission as normal. The second scenario (**2.b**) with $\psi_w = -90°$ (west) is shown in Figure 3.6, in which the ship gets too close to the shoreline after a blackout. Thus, the risk rises above the given threshold, and the anchor is ultimately dropped. Figure 3.7 contains the time series plots of the azimuth thrusters for each blackout scenario. The time axis is given in minutes, and azimuth angles $\boldsymbol{a}_t$ for the stern and bow thrusters are shown in green and yellow, respectively. The propulsion forces $\boldsymbol{f}_t$ applied by the stern and bow thrusters are respectively shown in blue and red, given as percentages of the maximum thruster force $f_{\max}$.

3.6.3 Strong winds and risk cost tuning

The choice of specific cost tuning strategies play a critical part in both the trajectory planning algorithm and the anchor drop decision-making. In these hazardous scenarios with strong winds where $V_w = 20\,\mathrm{m/s}$ and $\psi_w = 45°$ (north-east), it is demonstrated in Figure 3.8 how particular risk cost tuning strategies may lead to different desired behaviors. First, the cyan trajectory (**3.a**) shows a fully completed

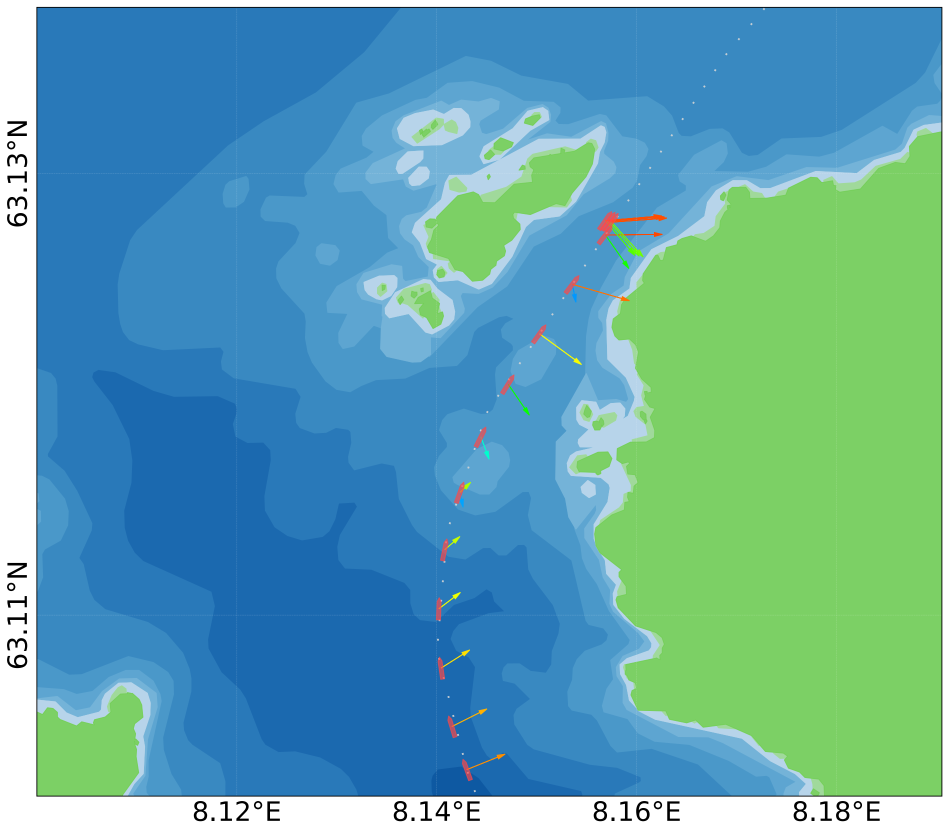


Figure 3.6: A blackout too close to the shoreline, leading to an anchor drop.

transit. The risk level does not exceed the maximum threshold, and the algorithm is allowed to continue as normal despite the increased perceived risk levels due to greater wind disturbances. The second trajectory in red, however, (**3.b**) shows how the ship alternatively may hold a position for some time and await improved weather conditions. This tuning approach is based on a significantly increased emphasis on safety. Here, the ship is prohibited to pass the narrow strait due to too high predicted risk levels introduced by the strong winds, until the wind velocities recedes after 25 min. This temporary dynamic positioning (DP)-like behavior demonstrates some of the inherent versatility of the MPC scheme, due to the combined steep gradients of the nominal grounding risk and the wind disturbance risk from (3.17) creating a local minimum (in an undesirable crosswind pose due to initial conditions), blocking the path progression forward. An appropriate system improvement in this situation may be to also consider making an anchor drop if the wait time exceeds some given time period. Alternatively, a second similar approach may be applied, in which the control of the ship is given to a separate DP controller if the opposing risk cost gradients rises above some threshold. The last scenario in yellow (**3.c**) showcases a different cost tuning philosophy, in which the NLP solu-

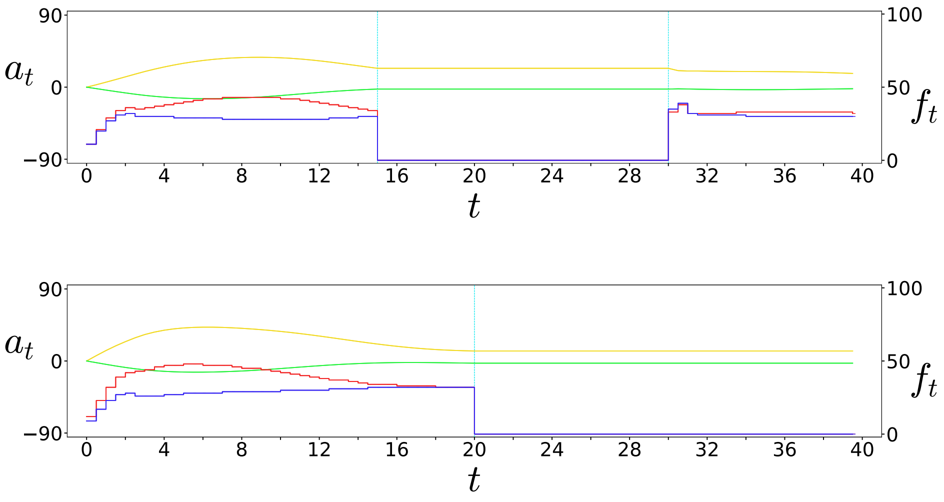


Figure 3.7: Thruster inputs for the temporary blackout and blackout with anchor drop scenarios, respectively.

tions steer the ship around the smaller isles of the narrow strait if the reference path cost coefficient κ_1 is sufficiently relaxed. Which of these approaches is the most suitable for any given scenario may vary greatly depending on specific environmental conditions, ship dimensions and system configurations. These diverging trajectories do however demonstrate another aspect of the flexibility of the MPC algorithm, allowing one to carefully tune cost parameters for several alternative desired behaviors.

3.7 Conclusion

The results from various simulation scenarios show that the MPC algorithm is capable of managing extreme cases of different ship failure modes to a satisfying degree. The implemented system typically solves each nonlinear program with a prediction horizon of 20 min in approximately 2 s, and is thus considered a relatively fast and online decision-making algorithm. There is however much potential for improvement within both the algorithm itself and the complexity of the investigated scenarios, e.g. by combining scenarios to produce more intricate behaviors. Furthermore, it is recommended that the ad hoc risk function implemented for proof of concept should be replaced with probabilistic risk models, and that further advancements of the path progression or risk cost functions may be achieved by adding additional terms related to other operational objectives. This is left to be explored in future works. Ultimately, risk-based MPC is considered an appropriate method for trajectory planning and decision-making for autonomous ships during emergencies.

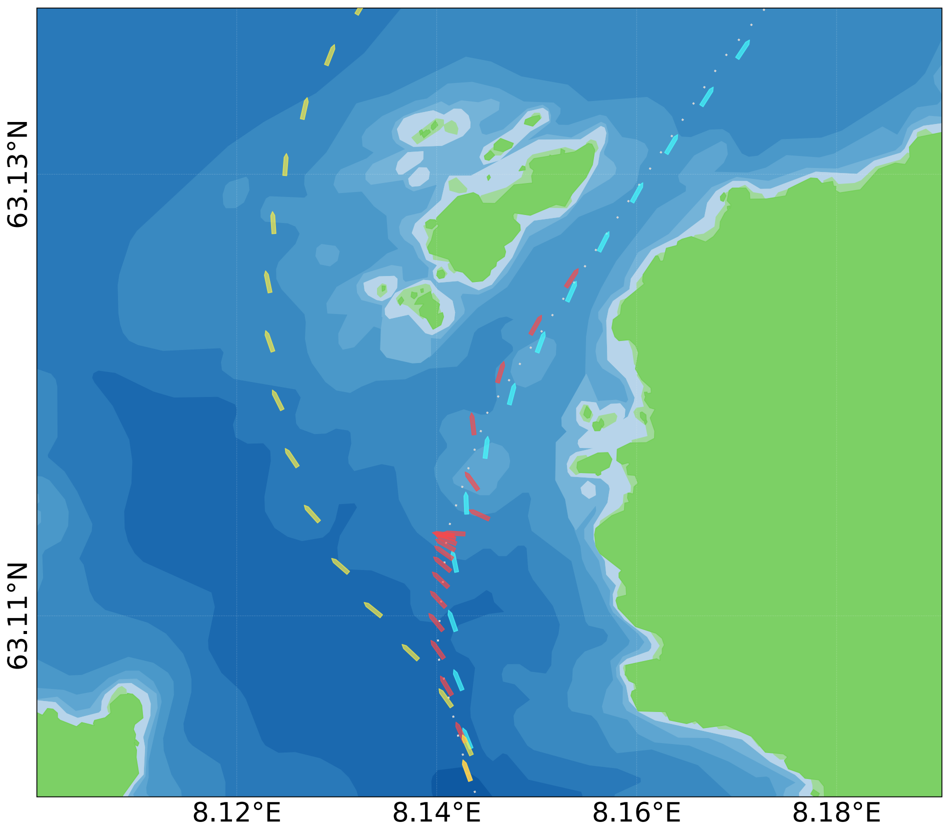


Figure 3.8: Different cost tuning approaches generate diverse behaviors during strong winds.

Chapter 4

Risk-based predictive supervisory control

This chapter is based on the publication

[40] **S. Blindheim**, I. B. Utne, and T. A. Johansen, "Risk-Based Supervisory Control for Autonomous Ship Navigation," *Journal of Marine Science and Technology*, pp. 1–25, 2023. DOI: `10.1007/s00773-023-00945-6`

The methodology and simulations were developed by S. Blindheim, under the supervision of I. B. Utne and T. A. Johansen. The first draft was written by S. Blindheim, and was revised by I. B. Utne and T. A. Johansen.

4.1 Introduction

There is an increasing desire to reduce operational costs and risks during ship operations by improving the intelligence and autonomous decision-making capabilities of maritime vessels [95]. Bridging the gap, however, between qualitative risk analysis and quantitative supervisory optimal control is a challenging task. The aim of this work is to develop a method for applying results from risk analysis to be utilized by an supervisory optimal control algorithm. Results from risk analysis may provide useful input to determining safe and efficient sequences of control actions to be taken in a complex maritime environment.

Higher levels of autonomy is not the main objective in itself, but rather to realize safer and more efficient operations involving human personnel. One way of improving *human safety* may be to move operators to an ROC, which may increase the productivity and efficiency of operations by giving, e.g., ship captains the opportunity to focus their abilities on monitoring a larger fleet of vessels simultaneously, supported by increased analysis of human factors or interference related to the altered supervision and (semi-)autonomous control hierarchy [96], [97].

Caution should nonetheless be exercised when making changes to a large operational infrastructure such as cargo or passenger transport. Communication and cooperation with conventional ships and compliance with international regulations, such as COLREGs are decisive. The safety and well-being of smaller vessels also need to be taken into account, e.g., smaller sailboats sailing in the vicinity of the larger (semi-)autonomous vessels.

In general, accidents occur due to unpredictable conditions, erroneous decision-making, or unexpected emergent system failures [98]–[100]. Risk assessment is therefore required to identify and analyze hazardous events, and determine the need for potential risk mitigation measures. One potential measure for autonomous ships is to implement onboard online consequence analysis-based optimization algorithms with some prediction horizon, weighting different operational objectives in light of the risks associated with each action considered for execution. MPC is such a method, and has shown promising results for development of operational constraints [101], DP [102], path following [31] and collision avoidance [32], [33].

These systems, however, usually have strictly defined operational areas or limited available decision spaces in which they are explicitly allowed to make autonomous decisions. The conditions are normally the default operational stages, like the crossing transit phase of an autonomous ferry. Nor do typical applications of MPC include risk models or results from risk analysis as input to the optimization algorithm. In order to reach even higher levels of autonomy, a high-level supervisory control system for risk assessment and safety-aware decision-making is needed [7].

STPA considers safety as a control problem, which makes it feasible for revealing hazards related to autonomous systems. Such hazards should be considered in the design of control algorithms and when optimizing decisions during operations to improve safety. One of the challenges with STPA, however, is that it only brings forward qualitative results, which are impossible to use directly for MPC.

In this chapter, a step-by-step approach for the design of supervisory risk control and risk-aware MPC is proposed as follows. (i) Risk analysis or hazard identification in terms of a STPA is performed in order to identify how inadequate control of a maritime vessel may occur. Next, (ii) the qualitative results of the STPA are transformed into an OCP, subsequently (iii) solved numerically by nonlinear programming through an existing MPC-based decision-making algorithm for path planning with anti-grounding [38]. The ship control is performed using a receding horizon approach, based on the chosen dynamic ship model and a combination of cost functions and operational limitations, each targeting different aspects of online path planning and risk management. The performance of the developed risk terms in the MPC cost function is furthermore demonstrated by example simulations.

Ultimately, the novel contribution of this work is a method for transforming the results of qualitative risk analysis into a tractable optimization problem to be solved by an online decision-making algorithm. It provides a systematic approach to the design of nonlinear MPC cost, constraints, and solution strategy, with systematic considerations for hazards and risks, which is a highly challenging task. The resulting framework may serve as a foundation for future autonomous decision-making or online consequence analysis techniques for both accident prevention and online risk control. To the authors knowledge, this is the first time qualitative risk analysis and MPC are explicitly coupled.

4.2 Background

4.2.1 Supervisory risk control

Risk analysis in general is concerned with identifying what may go wrong, and determining the likelihoods and consequences of those events [103]. Risk modeling represents risk qualitatively or quantitatively, and risk control is the use and integration of such models to support situation awareness and decision-making, e.g., by autonomous systems [7]. Separating the performed risk control into two equally important and dependent "modes", i.e. by human supervision of an automated or autonomous system, and by an autonomous system itself; supervisory risk control focuses on the latter of the two [2].

The control of an autonomous system may be divided into the offline mission planner layer, the online guidance and optimization level, and the control execution level [104]. The work in this chapter is mainly focused on control related to guidance and optimization, for which the high-level mission is given and preplanned, and the lower-level control is taken care of by corresponding subsystems.

Previous work within supervisory risk control proposes to use BBN for *online risk control* [7], in which the BBN serves as an underlying risk model in order to update and assess the current risk levels during operations. This is in contrast to the approach presented in this work, wherein the main objective is to use the results of the risk analysis in the design process of the online control algorithm. Though both methods involve updating risk levels during runtime, no underlying risk model is used in this work. Instead, a risk-based cost function is constructed and used to solve the resulting numeric OCP.

4.2.2 STAMP and STPA

The Systems-Theoretic Accident Model and Processes (STAMP) [11] is an accident causation model for analyzing and explaining how accidents may occur, attempting to handle the ever-increasing complexity of systems. Complex systems may have emergent properties only surfacing through the interconnection of the various parts of the system, which is difficult to predict by component failure analysis alone. STAMP regards safety as a control problem, preventing accidents by controlling the system or process according to appropriate safety constraints (SC). Accidents may thus occur if these constraints are broken, i.e., inadequate control. STPA [11] is a hazard analysis method based on STAMP, which attempts to identify how hazards or inadequate control may take place. This method is feasible for complex and automated maritime control systems [12] and as a basis for safety verification [105], and is generally applied through the following steps:

1. Define losses, system-level hazards and system-level safety constraints.
2. Define a system representation by analyzing and modeling the system as a hierarchical control structure.
3. Identify unsafe control actions.
4. Identify loss scenarios in which the unsafe control actions may occur.

The first step identifies or defines what types of losses one wants to prevent. In the case of autonomous ships, one may specify what the focus of the analysis is aimed at, e.g., fires, grounding, collision, or security threats such as piracy. Thus, the scope and purpose of the analysis is clarified by defining how the system-level hazardous states may lead to such losses. Next, the system is modeled and represented as a hierarchical control structure, i.e., a set of feedback control loops, and all relevant process model variables related to the internal belief of the controllers are identified. The third step is to identify unsafe control actions, i.e., supervisory control actions (SCA) that in any way may lead to one of the system-level hazards.

Note that supervisory control actions in the context of this risk analysis are defined as any type of decision or program flow between system modules which allows for hazardous states or outcomes to develop or occur, in contrast to automatic control actions typically performed by machinery systems (i.e. propulsion or steering control). STPA uses four general categories of unsafe control actions, which are presented in Table 4.1 [106].

Scenarios in which unsafe control actions may occur and their causes are identified by inspecting relevant parts of the control loops in the control hierarchy in specific contexts, e.g., incorrect feedback, lack of feedback, decision-making flaws, time-delay, (lack of) situation awareness, component failures, process disturbances, communication errors, or other risk influencing factors related to the control loops.

The identified scenarios from the STPA may subsequently be considered for testing and simulation, and the results used to construct a suitable risk-based cost function for the MPC-based decision-making algorithm.

Table 4.1: Categories of unsafe control actions

A	A SCA required for safety is not provided or not followed.
B	A SCA that causes a hazard is provided.
C	A SCA is provided too early, too late or in the wrong sequence.
D	A SCA is applied for a too long or too short period of time.

4.2.3 Scope and simplifications

An important prerequisite for risk analysis is to define its scope. The main objective is to develop an optimization-based algorithm controlling the ship, given a preplanned path in a maritime environment. To ensure safe optimization and that hazards and risks are considered in the optimization, STPA is performed and results are transformed into mathematical constraints, logic and objectives, implemented in MPC. The results from STPA are used as input to construct an OCP, subsequently discretized into a nonlinear program and solved by an MPC algorithm, and assessing the resulting trajectories. The goal is to improve the autonomy of the control system, and enhance safe operation of autonomous ships.

The cost function of the resulting MPC consists of risk-based cost terms. Unlike the established nomenclature of shipping economics concerned with e.g. capital and operational expenditures [107], the concept of cost in this context is specifically applied solely to weigh and balance opposing interests or penalties as part of the standard terminology used within the field of optimization.

A simplified overview of the system scope of this work is shown in Figure 4.1. The implemented software used for simulations is highlighted in blue, and represents a core element of a simulated ANS. The MPC algorithm utilizes a mathematical *Ship Model* (the dynamical model in Section 4.10) to predict future states given the current system state. This structure assumes communications between systems such as a SAS and an AMMS, controlling the ship's machinery. These system modules shown in gray color are, however, simplified in this work to only contain and relay their intended inputs and outputs between appropriate software modules.

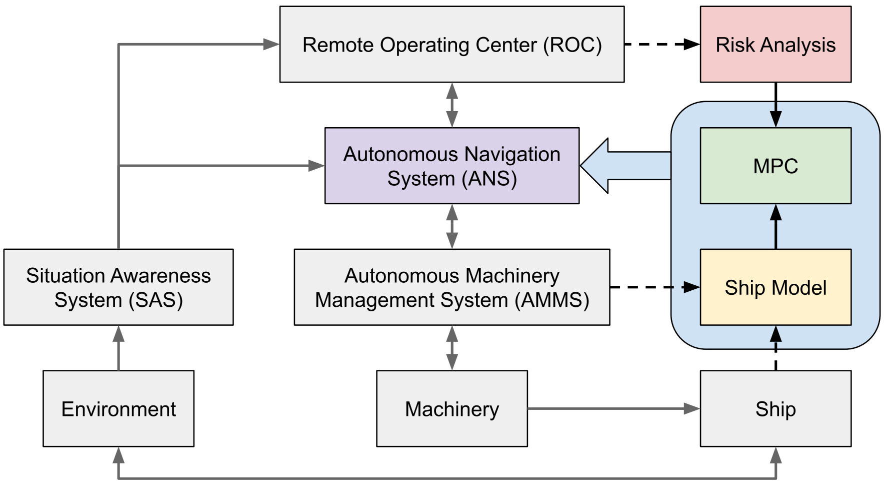

Remote Operating Center (ROC)
Risk Analysis
Autonomous Navigation System (ANS)
MPC
Situation Awareness System (SAS)
Autonomous Machinery Management System (AMMS)
Ship Model
Environment
Machinery
Ship

Figure 4.1: Scope of the system considered in this work.

This work is mainly concerned with grounding hazards, allision, and anti-grounding functionality, for proof of concept. Thus, collision avoidance with respect to dynamic obstacles, e.g., other maritime vessels, is not considered. Disturbances applied to the system are simplified such that only wind direction and wind velocity is considered, i.e., no current or wave disturbances are included. Furthermore, COLREGs compliance (or violation) is disregarded. As the act of docking (or berthing) the autonomous ship may be viewed as a separate control mode, docking is not considered in this work. The approach assumes that the autonomous ship is able to execute appropriate emergency measures or otherwise surrender control of the ship to human operators if the supervisory risk control algorithm exceeds a certain risk threshold or enters a specific hazardous situation. Additionally, consequences related to or occurring *as a result* of grounding, such as environmental pollution or loss of human lives, are not included in the risk analysis. This is considered appropriate for a proof-of-concept study.

Even though approximations are used, it is proposed that extensions for more complex analysis and risk modeling, such as collision avoidance and COLREGs compliance, may be equivalently added using the presented method in future works, without loss of generality.

4.3 Modeling

The STPA in this chapter is based on an analysis performed in workshops together with industry participants during Spring 2019. The main objectives of the control process is to: **a)** avoid grounding and allision with mapped obstacles, and **b)** complete the given mission of the ship operations with optimized resource allocation weighted against risk considerations from **a**, within defined limits.

4.3.1 STPA Step 1 - Purpose

Step 1 of the STPA is presented in Tables 4.2 to 4.5, to be used as a basis for the following steps. The hazards were specified with respect to which motion control objectives that may lead to violation of the safety constraints, defined in Table 4.5.

Table 4.2: Accidents

A1	Allision with a stationary mapped obstacle.
A2	The ship grounds or makes contact with the seafloor.

Table 4.3: System-level hazards

H1	The ship violates the minimum separation distance to a stationary obstacle (**A1**).
H2	The ship violates the minimum separation distance to the shore (**A2**).
H3	The ship sails in too shallow water (**A2**).

Table 4.4: System-level safety constraints

SSC1	The minimum separation distance to obstacles must be maintained.
SSC2	The minimum separation distance to shore must be maintained.
SSC3	The ship must not sail in too shallow water.

Table 4.5: Specified hazards

H1a	Motion control objectives that result in violation of the minimum distance of separation to an obstacle are formulated, i.e., the cost function of the trajectory planning algorithm is inappropriately designed with respect to avoiding obstacles during assigned time intervals.
H1b	Motion control objectives that do not result in violation of the minimum distance of separation to an obstacle are not followed, i.e., the subsystems of the ANS are unable to apply the required motion control to avoid obstacles.
H2a	Motion control objectives that result in violation of the minimum distance of separation to the shore are formulated, i.e., the cost function of the trajectory planning algorithm is inappropriately designed with respect to avoiding the shoreline during assigned time intervals.
H2b	Motion control objectives that do not result in violation of the minimum distance of separation to the shore are not followed, i.e., the subsystems of the ANS are unable to apply the required motion control to avoid the shoreline.
H3a	Motion control objectives that result in sailing in too shallow water are formulated, i.e., the cost function of the trajectory planning algorithm is inappropriately designed with respect to keeping the vessel in deep enough waters during assigned time intervals.
H3b	Motion control objectives that do not result in sailing in too shallow waters are not followed, i.e., the subsystems of the ANS are unable to apply the required motion control to keep the vessel within deep enough waters.

4.3.2 STPA Step 2 - System representation

The system control flow and feedback hierarchy of this work is shown in Figure 4.2, and consists of all implemented software modules and physical entities present in the system, including peripheral interfaces between the software system and the physical environment. Directed connections between the various modules show the program flow.

Ship denotes the physical ship, which also encompasses all software modules communicating with the ANS, as well as the presence of a physical ship hull and machinery interacting with the physical environment. Numerical values for physical *Environment* forces and states are provided to the Ship by the *SAS* module through forecasts and sensor measurements, which are in turn interpreted by the MPC through its *Scenario* module. Additionally, the *Dynamics* module contains the ship's equations of motion, which are used to generate simulated (expected) movement within the physical environment.

These system states along with bathymetry data from the ENC module [37] is utilized by the Supervisory Risk Control module to predict future states into a discrete time horizon using its internal MPC, NLP, and *Solver* modules, based on the cost function given by the *Objectives* constructed from the preplanned route and an internal online risk estimation (ORE) module (not shown), and applied control from the AMMS. The output of real-time computations from the ORE module as defined by the operators in a ROC is used both as a direct input to the MPC module during runtime, as well as serving as a measurement or monitoring tool for the ROC. Additionally, the AMMS is updated with events occurring in the environment detected by sensor measurements generated in the SAS, such as machinery failures.

The ANS module is in turn given a mission by the ROC, i.e., to follow a pre-planned route or path within certain time and risk limits, and uses the predicted trajectories to make autonomous control decisions and give commands to the AMMS. The resulting decisions, including the trajectory, risk cost calculations, and current internal states of the autonomous ship, are communicated to the ROC through information reports and ENC visualization, for supervisory human assessment and potentially human intervention.

Note that the mission given by the ROC to the ANS may be to follow a pre-planned route or mitigate damages through various emergency protocols. Moreover, the risk levels calculated by the ANS is of integral importance to the supervisory risk controller, through scaling coefficients provided to the MPC objective or cost function for trajectory predictions. The risk analysis, as well as transforming the results of a qualitative analysis into a quantifiable optimization problem, thus provide the very important basis for the supervisory risk control and must be performed with care.

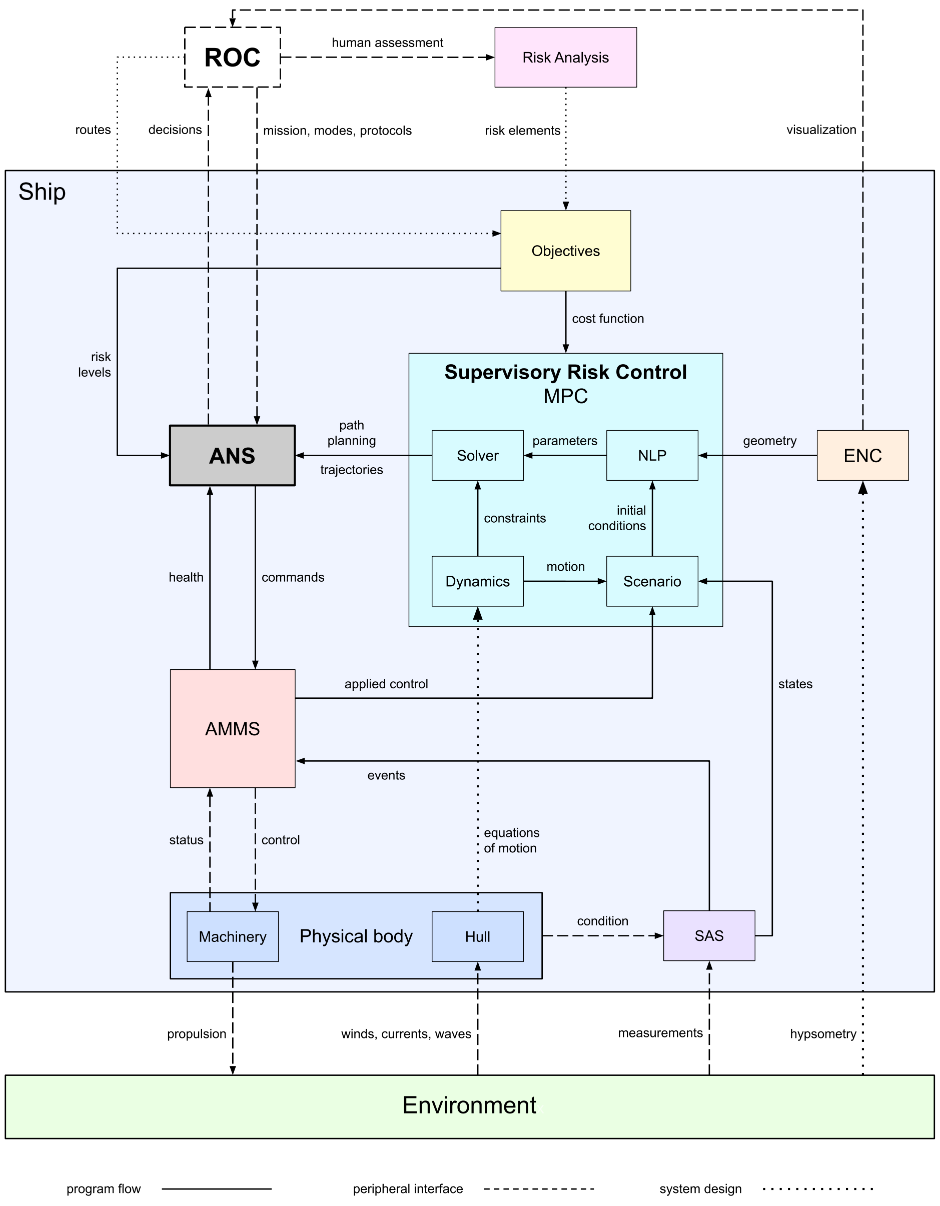


Figure 4.2: Overview of the system control architecture. The supervisory risk control module is the primary scope of the design.

4.3.3 STPA Step 3 - Unsafe supervisory control actions

Given the hazards presented in Table 4.5, hazardous or unsafe supervisory control actions (UCA) are to be identified with respect to the control flow in the system control architecture, as shown in Figure 4.2. The supervisory control actions are given in Table 4.6, and the process variables considered during the identification of the control actions are given in Table 4.7. No command or SCA given by the ROC to the ANS is considered, as the ROC handles higher levels of decision-making than the ANS, and as such, is generally considered outside of the control bounds or complexity which the autonomous ship is expected to be able to control on its own.

It is assumed that the mission to be carried out by the ship is static throughout the decision process, and that no changes to objectives are made, i.e., the risk-aware cost function based on the results from the risk analysis and the specified mission objectives is unchanged during the entire process of autonomous navigation. This is assumed both in an effort to limit the scope of the system, and the fact that changing the mission to the ANS also is considered a higher level form of decision-making than the autonomous system should be able to perform by itself. Consequently, any change in the general mission objectives is thus considered to either be the initiation of an emergency protocol, or simply the start of another mission. For dynamic missions, the structure of the constructed solution would be unchanged, as the proposed analysis is generic. The same methodology may thus either be repeated in its entirety if a mission changes considerably, or one may alternatively change and re-evaluate the cost coefficients only (steps 12 and 13). It is suggested that an explicit methodology for online evaluation and tuning of the resulting cost function en route (for e.g. dynamic missions) should be investigated in future works.

The main SCA of the control hierarchy originates from commands given by the ANS to its two main modules: the MPC and the AMMS. Human operators in the ROC set the operational mode of the ANS to either autonomous mode or remote

Table 4.6: Considered supervisory control actions

Controller	Process	Supervisory control action
MPC	Solver	Calculate trajectory over time horizon
Solver	NLP	Solve NLP
NLP	ENC	Compute obstacle distances
Scenario	SAS	Construct scenario from states
MPC	Objectives	Compute risk costs
ANS	ORE	Assess risk levels
ANS	AMMS	Command first control step
AMMS	Machinery	Apply first control step

Table 4.7: Process variables

Process variable	**Available states**
Trajectory on horizon	Distinct future ship states
Current risk level	Continuous interval
Current wind disturbance	Constant velocity and angle
First MPC control step	Next optimal control input
Machinery health / status	Online / Offline / Fixed

control, or initiate an emergency protocol during emergencies. If the ANS is in an autonomous mode, the MPC is run to predict the next optimal trajectory (i.e., the next set of future ship states within the sampled time horizon) by solving an NLP. During this solve, the equations defining the ship dynamics are used to compute costs, based on calculated distances to spatial polygons of obstacles or grounding areas. The result is returned to the ANS as an optimal ship trajectory given the current ship and environmental states. It may be noted that trajectories are dependent on time, whereas paths (or routes) are not. Thus, the time aspect is in this context considered as one of the three main evaluation criteria during optimization, along with resource allocation and risk levels.

If the risk levels after the first control interval (or along the trajectory) are within accepted thresholds, the first control step of the trajectory is applied to the steering, power and propulsion machinery by the AMMS. Additionally, the current status and health of the physical machinery is estimated and reported back to the ROC for potential human intervention. The *Online* state means that both the steering and propulsion are available with ordinary capabilities, *Offline* indicates no propeller propulsion or steering available, and *Fixed* represents no available steering.

All identified UCA are listed in Tables 4.8 and 4.9, evaluated with respect to the ability of the ANS to keep the ship sufficiently safe during a specified time interval (i.e., the time required to apply appropriate SCA or counter-measures due to physical limitations or safety constraints). Note that complex computations, such as calculating future trajectories in this context are also considered supervisory control actions, due to the fact that they may *lead* to later (autonomous) decisions-making which can cause hazardous events. Each SCA is given a unique identifier and a short description, in addition to being assigned a relevant control action category label (mode) from Section 4.2.2.

Together, these UCA are used to identify the controller constraints (CC) presented in Table 4.10. From Figure 4.2 and Table 4.8, it is clear that **UCA4-UCA12** are all part of subsystems related to computations for constructing scenarios and nonlinear problems to be solved by the MPC module. These UCA are thus closely related and connected, and are all found to potentially lead to the first three UCA identified between the ANS and the MPC. However, these underlying UCA also

Table 4.8: Unsafe supervisory control actions (part 1)

ID	**Controller**	**Process**	**Supervisory control action**	**Mode**	**Description**
UCA1	MPC	Solver	Calculate trajectory on horizon	A	The calculated risk levels along the predicted trajectory is unacceptable, i.e., exceeds the risk threshold as defined by the ROC.
UCA2	MPC	Solver	Calculate trajectory on horizon	B	The predicted trajectory returned by the MPC directly causes an obstacle allision or grounding.
UCA3	MPC	Solver	Calculate trajectory on horizon	C / D	The MPC does not return a calculated trajectory within the required time interval, i.e. the result was provided too late or the computation was performed for too long.
UCA4	Solver	NLP	Solve NLP	A	The computed solution trajectory is infeasible, i.e., a solution satisfying all physical constraints as well as risk constraints was not found, leading to **UCA1** or **UCA2**.
UCA5	Solver	NLP	Solve NLP	B	The Solver produces a feasible trajectory which contains obstacle allisions or grounding events, leading to **UCA2**.
UCA6	Solver	NLP	Solve NLP	C / D	A solution is not calculated within the required time interval e.g. due to a divergent or infeasible nonlinear problem, leading to **UCA3**.
UCA7	NLP	ENC	Compute obstacle distances	A / B	The returned distances to obstacles are incorrect, producing an inaccurate or unsafe basis for the MPC trajectory calculations, leading to **UCA1** or **UCA2**.
UCA8	NLP	ENC	Compute obstacle distances	C / D	The geometric operations or distance calculations applied to polygons are too computationally expensive, leading to **UCA3**.
UCA9	Scenario	SAS	Construct scenario from states	A / B	The scenario classifications defined by the SAS are improperly formulated (e.g. if calm winds are classified as adverse weather), leading to **UCA1** or **UCA2**.
UCA10	Scenario	SAS	Construct scenario from states	A / B	The scenarios produced are erroneous due to incorrect environment measurements or state estimations, leading to **UCA1** or **UCA2**.
UCA11	MPC	Objectives	Compute risk costs	A / B	The calculated risk costs defined by the scaling coefficients from the ORE module are improperly formulated by the ROC, leading to **UCA1** or **UCA2**.
UCA12	MPC	Objectives	Compute risk costs	C / D	The risk cost function is too computationally expensive, leading to **UCA3**.

Table 4.9: Unsafe supervisory control actions (part 2)

ID	Controller	Process	SCA	Mode	Description
UCA13	ANS	ORE	Assess risk levels	A	The calculated risk level of the currently estimated ship position during a time interval exceeds the maximum risk threshold set by the ROC, leading to the activation of an emergency protocol, a change of control mode or adjustments made to mission objectives by the ROC.
UCA14	ANS	AMMS	Command control step	A	The next ship position resulting from applying the first optimal control step calculated by the MPC exceeds the maximum risk threshold defined by the ROC, leading to **UCA13**.
UCA15	ANS	AMMS	Command control step	B	The ship position resulting from applying the first optimal control step calculated by the MPC causes an obstacle allision or grounding.
UCA16	AMMS	Machinery	Apply 1st control step	A / D	The control input as commanded by the ANS is not followed by the AMMS (i.e. the control is not applied to the machinery long enough or not at all), causing the maximum risk level threshold to be violated and leading to **UCA13**.
UCA17	AMMS	Machinery	Apply 1st control step	B	The control input as commanded by the ANS is not followed by the AMMS, causing an obstacle allision or grounding.
UCA18	AMMS	Machinery	Apply 1st control step	C / D	The AMMS or the ship machinery does not carry out the control commanded by the ANS within the required time interval e.g. due to physical system constraints or machinery faults.

Table 4.10: Controller constraints

ID	UCA	Constraint description
CC1	1, 2, 3	The calculated trajectories on the horizon must be computed within the required time interval, cannot violate the maximum risk threshold, and shall not lead to any obstacle allision or grounding.
CC2	4, 5, 6	A feasible NLP must be constructed, and the computed solution must converge to an optimal solution within the require time interval.
CC3	7, 8	The calculations for obstacle distances must be sufficiently accurate as well as not too computationally expensive.
CC4	9	All scenario definitions must be properly formulated, such that the behavior resulting from the calculated trajectories matches the expected behavior in any given scenario.
CC5	10	The generated environment states from sensor measurements or simulations must be sufficiently accurate.
CC6	11	The risk-based cost function and associated risk scaling coefficients must be correctly and sufficiently defined, such that obstacle allision or grounding does not occur due to logical or mathematical inconsistencies or assumptions that do not hold.
CC7	13	Dependent on **CC6**, the maximum risk level threshold given by the ROC must be set with respect to the risk elements of the ORE module such that the threshold is violated only when the system should appropriately engage automatic emergency protocols or human intervention.
CC8	14, 15	The first optimal control step of the MPC trajectory must not result in an obstacle allision or grounding.
CC9	16, 17, 18	The physical machinery must carry out the control as commanded by the ANS through the AMMS within the required time interval.

Table 4.11: Loss scenarios and safety constraints

ID	Hazards	CC	Loss scenario	Safety constraint
SC1	**H1a, H2a, H3a**	**CC1, CC8**	The first control step or later intervals of the predicted future trajectory violates the minimum separation distance to an obstacle, the shore or too shallow water.	The predicted ship trajectory must be spatially constrained to avoid crossing the minimum separation distance to obstacles, the shoreline or too shallow waters at all times.
SC2	**H1b, H2b, H3b**	**CC1, CC2**	The MPC module is not able to compute a feasible ship trajectory within the required time interval, leading to a violation of the minimum separation distance to an obstacle, the shore or too shallow water due to inappropriate control during the computation period.	Given some defined available computation power, the constraints of the NLP must be well-formulated and feasible to enable proper solver performance and satisfactory convergence rates within the required time interval, i.e. the ship model must sufficiently account for the ship dynamics, and the initial conditions and state constraints must be physically and logically consistent.
SC3	**H1a, H2a, H3a**	**CC4**	The parameters or complexity of an estimated scenario for a given time interval are incorrect or insufficient, leading to unsuitable decision-making or MPC trajectory solutions which cause obstacle allision or grounding.	The mathematical model of environmental variables and ship system states included in the risk-based cost function must be adequately formulated as to properly simulate the physical behavior of the ship in a designated scenario.
SC4	**H1b, H2b, H3b**	**CC6**	The risk cost function of the NLP is too computationally expensive, which during computation leads to a violation of the minimum separation distance to an obstacle, the shore or too shallow water due to absent or sub-optimal decision-making by the ANS (inappropriate control).	The cost function must be computationally feasible with respect to the given computation power capabilities, within the specified time interval or calculation frequency, based on the dynamics of the ship. For instance, the cost function should be sufficiently smooth, locally convex, and computationally simple, such that the risk costs are easily calculated by the NLP solver.
SC5	**H1b, H2b, H3b**	**CC9**	The machinery of the ship is not able to control the ship as required during the designated control interval, leading to drift-off or drive-off and a violation of the minimum separation distance to an obstacle, the shore or too shallow waters due to current ship velocities or external disturbances.	The risk-based cost function must be designed and tuned such that the ship is sufficiently far enough away from grounding obstacles during normal operations as well as with limited propulsion allocation during unexpected failures or emergency scenarios, in order to avoid obstacle allision or grounding during drive-off or drift-off due to external disturbances. Namely, the risk levels of trajectories closer to minimum separation distances plus a safety distance margin must be weighted sufficiently high.

introduce separate system design concerns, which in this context are treated as supervisory control actions and must be considered during formulation of the individual controller constraints. The UCA presented in Table 4.9 are related to the ship in its current state during operations, i.e., either with respect to the maximum risk level threshold being violated or inadequate AMM or machinery control.

4.3.4 STPA Step 4 - Safety constraints (SC) and loss scenarios

In the final step of the STPA, the system-level safety constraints of Table 4.4 and specified hazards of Table 4.5 are combined with the identified CC of Table 4.10 to identify loss scenarios and UCA-level SC, ultimately presented in Table 4.11.

Note that **CC3**, **CC5** and **CC7** from Table 4.10 are disregarded, due to the following. The ENC and SAS modules are considered separate subsystems, which are assumed to independently perform adequately and within their given requirements. Similarly, setting the maximum risk threshold is in this work considered outside the scope of the ANS, and a process that must be performed by human operators on the ROC. However, potential machinery faults (**CC9**) are included as part of the core of the supervisory risk control. The resulting safety constraints **SC1-5** are used as the fundamental basis for further decisions related to the design of relevant supervisory risk control components.

4.4 Methodology

The proposed steps for transforming the qualitative results from the STPA into a quantifiable optimal control problem for supervisory risk control are defined below. Details of the methodology are provided in the following subsections.

State variables identification

1. Define explicit mathematical state variables for all relevant measurable or quantifiable nouns or variables related to the identified loss scenarios and safety constraints of the risk-based supervisory control problem.
2. Represent the relationships between related variables as mathematical equations, and introduce intermediate variables or remove redundant variables if applicable.
3. Structure the identified variables into explicit system state and parameter vectors of the OCP.
4. Add all physical and logical equalities to the system dynamic equations of the OCP.

Safety inequalities construction

5. Formulate a risk-related inequality for all parts of a safety constraint that contains quantifiable variables.
6. Rank the safety inequalities based on the concept of risk priority numbers (RPN) [108].
7. Merge and/or remove any redundant safety inequalities by evaluation of assigned RPN and mathematical inspection.

Cost function formulation

8. Define a slack variable s for each inequality of the form $g(x) \geq 0$ such that $g(x) + s = 0$ holds, where x is a state variable.
9. Define an exponential cost term of the form $\mu e^{-\zeta s}$ for each slack variable s, where $\mu > 0$ and $\zeta > 0$ are tuning parameters approximately weighted according to the RPN assigned to its respective safety inequality.
10. Define the risk-related part of the cost function ρ as the sum of the exponential slack variable terms.
11. Formulate the total cost function as a sum of the resulting risk terms and other terms, e.g., related to resource consumption and mission objectives.

Evaluation and performance verification

12. Tune the coefficients of the cost function until the desired solver performance and control behavior is achieved.
13. Verify through inspection and test simulations that the performance satisfies all safety constraints.

This methodology leads to the development of a risk-based OCP and numerical solution. Note how the resulting safety inequalities and risk parts of the cost function are only related to the specific risk analysis performed through STPA, producing the mathematical *risk elements* of Figure 4.2. Thus, all additional physical system constraints and resource consumption costs (i.e., fuel and/or time), as well as mission objective costs (e.g., path following) are in the steps 11 and 12 combined and tuned in tandem such that the total system is complete, meaning that the resulting cost function is appropriately weighted between the various aspects of autonomous navigation. This comparative tuning or weighting process through extensive testing and simulation, denoted as steps 12 and 13 of the methodology, must be performed in a case-by-case basis, and may be difficult to generalize. Even though risk analysis provides inputs to such a tuning, this procedure must be performed so that the formulation of the cost complete function is appropriate.

The following subsections apply each step of the proposed method for a case study presented in this work, and the performance of the implemented system is ultimately simulated and assessed in order to evaluate the quality of the mathematical formulation and risk quantification formulated by the procedure.

4.4.1 State variables identification

The physical state variables of the ship relevant to the control problem and their relationship equations are defined as given in Section 4.10.

Next, the content of the SC description sentences are dissected and interpreted into additional mathematical variables by language analysis:

- **SC1**: Identified quantifiable nouns include the predicted ship trajectory, grounding obstacles, and the minimum separation distances to obstacles, the shoreline and too shallow waters.
- **SC2**: No quantifiable nouns or variables related to system states are identified in this safety constraint aside from the initial conditions of the NLP solver.
- **SC3**: No specific quantifiable nouns are identified aside from general mentions of physical states/behavior and (risk) cost function formulation.
- **SC4**: The only identified quantifiable noun is "risk cost".
- **SC5**: The identified quantifiable or measurable nouns are propulsion (allocation), risk level, (ship) trajectories, external disturbances, minimum separation distances, and a safety distance margin. Unexpected failures, drift-off, and drive-off are terms for special events or scenarios during extraordinary circumstances, and are consequently not measurable system states.

Some of the identified quantifiable nouns concern the same physical quantities: The predicted ship trajectory contains multiple ship states, which include the positions, orientations and velocities of the ship at each discrete time step within the given time horizon. These ship states, as well as propulsion forces, are defined in Section 4.10. Moreover, the term *grounding obstacles* will throughout the remainder of this text encompass all possible allision obstacles, shorelines and/or too shallow waters. This simplification assumes that there are no consequence or outcome differences between ship grounding and obstacle or shoreline allision, as per the scope defined in Section 4.2.3.

The additional system state and environmental variables identified during Step 1 are listed collectively as follows:

- ☐ $\boldsymbol{X}_N = \begin{bmatrix}\boldsymbol{x}_0^\top & \dots & \boldsymbol{x}_N^\top\end{bmatrix}^\top$, the vector of N ship states $\boldsymbol{x}_k$ (including positions, orientations and velocities), i.e. the discretized ship trajectory throughout the predicted horizon of N control intervals
- ☐ $\sigma_j \in \Theta_J$ = one of J grounding obstacle polygons provided by the ENC (see Section 4.5.4)

- ☐ d_{sep} = the minimum allowed separation distance between the ship position $\boldsymbol{p}$ and any grounding obstacle σ_j
- ☐ d_{safe} = the safety distance margin which is added to d_{sep} in order to have the ship positioned sufficiently far away from obstacle boundaries such that temporary loss of propulsion does not result in grounding
- ☐ $\rho(\boldsymbol{x})$ = numerical value denoting the current risk cost of any ship state $\boldsymbol{x}_k$
- ☐ ρ_{max} = numerical threshold value denoting the maximum accepted risk (cost) level of the autonomous system at any point in the defined two-dimensional space, selected by the ROC with respect to the definition of ρ in the cost function
- ☐ v_d = velocity of the generalized disturbance forces acting on the ship during transit (i.e. wind velocity)
- ☐ ψ_d = angle of attack of the generalized disturbance forces acting on the ship during transit

All remaining system state and parameter vectors of physical constraints (such as propulsion or steering limitations) not directly related to the risk analysis of the optimal control problem are given in Section 4.5.1.

4.4.2 Safety inequalities construction

The above mathematical definitions are subsequently used to construct safety inequalities (SI) for all sub-parts of each SC, given some appropriate interpretation of the SC formulations with respect to key logical, quantitative, qualitative and comparative statements.

In this work, only **SC1** and **SC5** are identified as containing safety inequalities. The remaining safety constraints are discussed in Section 4.4.4.

SI1a: $min\Big(d(\boldsymbol{x}, \sigma_j)\ \forall \sigma_j \in \Theta_J\Big) > d_{sep}$

SI5a: $min\Big(d(\boldsymbol{x}, \sigma_j)\ \forall \sigma_j \in \Theta_J\Big) > d_{safe} + d_{sep}$

SI5b: $min\Big(d(\boldsymbol{x}, \sigma_j)\ \forall \sigma_j \in \Theta_J\Big) \cdot f(v_d, \psi_d) > d_{sep}$

where $d(\boldsymbol{x}, \sigma_j)$ is a distance function which returns the distance between its two arguments. The scaling function $f(.)$, dependent on the disturbance velocity v_d and disturbance angle of attack ψ_d, is given as

$$f(v_d, \psi_d) = \max(0, \hat{\boldsymbol{\iota}}_j \cdot \hat{\boldsymbol{\omega}}) \cdot v_d \tag{4.1}$$

where $\hat{\boldsymbol{\iota}}_j$ is the unit vector from the ship to the obstacle j with the minimum distance to the ship position $\boldsymbol{x}$, and $\hat{\boldsymbol{\omega}}$ is the unit direction vector of the disturbance [38].

Next, each inequality is assigned an RPN [108], based on severity (result or consequence of failure/loss), occurrence (failure probability) and detection (failure identification difficulty). For the purpose of demonstration, three RPN are in this work defined to serve as simple categories classifying the three identified safety inequalities: Recall that the loss scenario of **SI1a** is related to grounding without considering external disturbances, see Table 4.11. Table 4.12 shows the resulting RPN for each safety inequality, based on 1 to 10 rankings of its severity factor (S), probability of occurrence (O), and ease of detection (D) [109]. Note that the RPN of Table 4.12 are assigned with respect to how the *absence* of each cost term would affect the autonomous navigation behavior of the ship, e.g. a moderate probability of grounding is assumed if the short-term re-planning navigation algorithm does not include *any* term directly related to anti-grounding based on ENC.

In this example, an RPN of 108 is assigned to **SI1a** due to high severity with respect to grounding (9), a moderate probability of occurrence (4), and high ease of detection (3). This is considered appropriate due to the fact that the ship should preemptively follow a pre-planned feasible path within well-defined safety boundaries and parameters, with only dynamic and unplanned obstacles providing the main uncertainty aspect of the equation. If the MPC planner has to significantly re-plan the trajectory due to some unforeseen circumstances such as a crossing ship, the probability of grounding may consequently rise accordingly. Moreover, the onboard and remote sensors with respect to spatial movement in the environment are assumed to be relatively robust with sufficient levels of e.g. accuracy and redundancy shared across multiple different types of technologies, and should thus provide a reasonably high probability of detecting internal failures.

Next, **SI5a** is given an RPN of 30 due to low severity in which only the safety distance margin between the ship and the minimum distance of separation to obstacles is violated (2), moderate probability just slightly more likely than direct grounding (5), and similar failure detection capabilities (3). However, **SI5b** is given an RPN of 216 due to the following. This SI specifically is concerned with taking into account how external disturbances such as winds affect the ship trajectory, and how it relates to avoiding grounding events with respect to the physical propulsion and steering limits. Thus, the severity of grounding is high as in **SI1a** (9), the occurrence is somewhat higher due to expected disturbances (6), and the difficulty of detecting failures in the equipment for measuring disturbance forces is moderately low (4).

Table 4.12: RPN assignment to each safety inequality, based on its severity factor (S), probability of occurrence (O), and ease of detection (D)

Safety inequality	**S**	**O**	**D**	**RPN**
SI1a	9	4	3	108
SI5a	2	5	3	30
SI5b	9	6	4	216

In general, the definition of RPN is closely related to risk acceptance of the system and its operation, which are usually determined based on stakeholders´ perspectives, current risk levels for similar activities, rules, and regulations. Since risk acceptance is outside the scope of this work, reasonable RPN are derived for illustration. From this example, and with these assigned RPN, it is clear that there is a distinct disparity between the highly ranked **SI1a** and **SI5b** compared to **SI5a**. This will be addressed in the following steps.

4.4.3 Cost function formulation

All safety inequalities are subsequently transformed into risk cost terms which contribute to the accumulated system risk levels present at any point in time, and are used as additional guidance objectives during the autonomous decision-making and ship trajectory optimization.

In Step 8, the slack variables (SV) of each safety inequality are consequently defined as:

$$\begin{aligned} \textbf{SV1a}: &\quad s_1 = d_{min}(\boldsymbol{x}) - d_{sep} \\ \textbf{SV5a}: &\quad s_2 = d_{min}(\boldsymbol{x}) - d_{safe} - d_{sep} \\ \textbf{SV5b}: &\quad s_3 = d_{min}(\boldsymbol{x}) \cdot f(v_d, \psi_d) - d_{sep} \end{aligned} \tag{4.2}$$

where $d_{min}(\boldsymbol{x}) = min\big(d(\boldsymbol{x}, \sigma_j)\big)\ \forall \sigma_j \in \Theta_J$. In general, $d_{min}(\boldsymbol{x})$ is a non-smooth function, in which the J distances $d(\boldsymbol{x}, \sigma_j)$ to each grounding obstacle σ_j are themselves minimum distances between a singular geometric point $p(x, y) \in \boldsymbol{x}_k$ and the boundary of the obstacle polygon as provided by the ENC.

Note that the resulting SV, when feasible (greater than or equal to zero), indicate "increased" compliance with the SI constraint for larger values, directly analogous to the slack variables used by nonlinear numeric solvers to satisfy constraints through *barrier functions* [110].

The risk cost terms are during Step 9 constructed as monotonic and strictly increasing exponential functions with the (negatively) weighted SV as the exponents. Thus, the resulting risk cost function ρ for a single time interval k is by Steps 8 and 9 of the procedure defined as

$$\rho(\boldsymbol{x}_k, \sigma_j) = \mu_1 e^{-\zeta_1 s_1} + \mu_2 e^{-\zeta_2 s_2} + \mu_3 e^{-\zeta_3 s_3} \tag{4.3}$$

Note that the form of (4.3) directly follows from Step 9, and that the slack variables from Step 8 by definition serve as the only variables to be weighted or scaled through their respective coefficients as strictly positive cost terms. Moreover, (4.3) is only defined for a ship state during a single time interval ($\boldsymbol{x}_k$), with respect to an individual grounding obstacle polygon σ_j. It is proposed that this weighted cost may simply be summed for all grounding obstacles $\boldsymbol{\Theta}_J$ within the spatial horizon (see Section 4.5.3). Due to the exponential form of (4.3), far away obstacles are evaluated as negligible, making only nearby obstacles significant with respect to the total cost value at any point. This is indeed in accordance with the desired behavior, i.e., to first and foremost avoid nearby shorelines or shallow waters – due to the exponential function, any land mass consequently yields insignificant costs compared to, e.g., a small reef closer to the ship, if located behind it.

Determining the values of the risk coefficients $\mu_{1,2,3}$ and $\zeta_{1,2,3}$ is achieved approximately through the RPN of the SI associated with each resulting exponential term, i.e., larger coefficient values for higher RPN, and lower values for lower RPN. Consequently, smaller minimum distances between the location of the ship and grounding obstacles lead to (exponentially) larger costs, as expected. Moreover, each individual cost is weighted so that the terms with larger RPN have larger costs closer to their constraint boundary, with respect to the other terms. Note, however, that as the RPNs are semi-qualitatively defined, the correspondence between the RPN and the resulting cost scaling coefficients may after tuning be diminished.

More detailed discussion of the tuned risk cost terms with respect to optimal control and desired behavior for autonomous navigation is presented in Section 4.7.

4.4.4 Evaluation and performance verification

Evaluating and verifying the performance of the cost function and resulting NLP formulation is challenging. The coefficients of the cost function are tuned, e.g., incrementally through repeated simulations, and the performance is evaluated by comparing the resulting behavior with all safety constraints. Section 4.4.2 denotes how **SC1** and **SC5** are exercised through **SI1a**, **SI5a** and **SI5b**. However, **SC2**, **SC3** and **SC4** are directly related to the solver performance, i.e. that the resulting solution convergence and behavior is appropriate and achieved within the required time intervals. Compliance with these constraints are thus verified through the following discussion and simulation results presented in Section 4.6, both validating the proposed methodology in this work as well as the autonomous behavior resulting from the application of the method to the considered use case. A special case worth noting is nevertheless that the resulting cost function, as defined by following the proposed methodology, by construction satisfies **SC4** with respect to computational feasibility.

The initial solution given to the solver (i.e., the pre-planned trajectory) during the first solve will greatly affect the produced trajectories due to the problem being non-convex, and must be pre-computed appropriately. The solver used to calculate optimal trajectories at each time interval may also have a significant impact on the generated results. In this work, a gradient-based solver is used – non-smooth functions such as the minimum function used for polygon distances are handled by the solver through automatic differentiation (differentiable programming). However, one may consider using solvers employing e.g. evolutionary algorithms, PSO or similar techniques, in order to utilize discontinuous and non-differentiable cost functions.

Additionally, the non-convexity of the constructed NLP makes no guarantees with respect to optimality of its local solutions, and the generated ship trajectories must consequently be assessed during development, with respect to the current conceptual view of acceptable risks, desired operations behavior and defined mission objectives. As such, the implementation and performance of a MPC algorithm and a case study with example simulations are presented in the following sections for evaluation and verification purposes.

4.5 NLP and MPC formulation

This section presents the resulting MPC and final cost function to be solved by the NLP algorithm, extended and improved from previous works [38]. Additionally, simulation results of the various test scenarios are summarized to demonstrate the performance of the formulated risk-based supervisory control problem.

4.5.1 Optimal control problem (OCP) formulation

An OCP is in general defined as follows:

$$\begin{aligned} \min_{\boldsymbol{x}(.),\boldsymbol{u}(.)} \quad & \int_{t=0}^{T} \tilde{\phi}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{\theta}(t))\ dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{d}(t)) \\ & \boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \leq \boldsymbol{0} \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad 0 \leq t \leq T \end{aligned} \tag{4.4}$$

where $\tilde{\phi}$ is a scalar stage cost function that will be defined in Section 4.5.3, $\boldsymbol{\theta}$ is a parameter vector, $\boldsymbol{x}_0$ is the initial state, T is the prediction horizon, and $\dot{\boldsymbol{x}}$ is given by the ship dynamics as presented in Section 4.10. The hard constraints $\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t))$ are given as:

$$\begin{array}{ccccc} -f_{\max} & \leq & u_1 & \leq & f_{\max} \\ -\omega_{\max} & \leq & u_2 & \leq & \omega_{\max} \end{array} \tag{4.5}$$

where $\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top$ is the control input vector where u_1 is the propulsion force of the rudder, u_2 is the rotational turning rate of the rudder, and $f_{\max}$ and $\omega_{\max}$ are the maximum propulsion force and rotational turning rate of the rudder, respectively. The solution to problem (4.4) will be deployed in a receding horizon fashion, yielding an MPC scheme.

4.5.2 The reference path

A preplanned reference path is defined as a piecewise linear (spline) function given an initial position, discrete intermediate points and a destination. Next, the reference path is parameterized, giving the two-dimensional reference function

$$\boldsymbol{r}(\alpha) = \begin{bmatrix} x(\alpha) \\ y(\alpha) \end{bmatrix} \tag{4.6}$$

for calculating path points where $\alpha \geq 0$ is a scalar advancement parameter (i.e. the traveled distance during a control interval) acting as a decision variable along the preplanned path, and $x(\alpha)$ and $y(\alpha)$ are piecewise linear functions.

4.5.3 Objectives and cost function definitions

In order to construct the NLP, a cost function to be minimized is defined. Advancing along the path is a simple matter of increasing α, and the desired ship speed along the path is furthermore established by penalizing ship transit velocities larger than the given reference speed s_{ref}. This is achieved by minimizing a speed penalty decision variable β, where

$$u^2 + v^2 \leq s_{\text{ref}}^2 + \beta, \qquad 0 \leq \beta \tag{4.7}$$

in which u is the forward surge velocity, and v is the sideways sway velocity.

Collecting the additional decision variables into a vector for each time step through the control horizon N yields

$$\boldsymbol{q}_k = \begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix}, \qquad k = 0, 1, ..., N-1 \tag{4.8}$$

which in Section 4.5.4 is used to define the NLP decision variable vector $\boldsymbol{w}$.

In this work, the cost function is defined with the purpose of producing a safe ship trajectory that fulfills the mission objectives:

$$\phi(\boldsymbol{w}, \boldsymbol{\theta}) = \sum_{k=1}^{N} \xi(\boldsymbol{x}_k, \boldsymbol{q}_k, \boldsymbol{q}_{k\text{-}1}) + \epsilon(\boldsymbol{u}_k, \boldsymbol{u}_{k\text{-}1}) + \rho(\boldsymbol{x}_k, \boldsymbol{\theta}_k) \tag{4.9}$$

The cost terms are defined as follows:

i. The path progression cost function

$$\xi(\boldsymbol{x}_k, \boldsymbol{q}_k, \boldsymbol{q}_{k-1}) = \boldsymbol{\kappa}^\top \begin{bmatrix} ||\boldsymbol{r}(\alpha_k) - \boldsymbol{p}_k||^2 \\ ||\alpha_k - \alpha_{k-1} - \alpha_{\text{trav}}||^2 \\ \beta_k \end{bmatrix} \tag{4.10}$$

where $\boldsymbol{\kappa} > \boldsymbol{0}$ is a vector of tuning parameters. These terms are responsible for advancing the ship position $\boldsymbol{p}_k$ (trajectory) along the precomputed feasible path, through the constant path step parameter α_{trav} and the reference path $\boldsymbol{r}(\alpha_k)$. The β_k term penalizes violations of the transit speed reference as detailed in Section 4.5.3. It is recommended that α_{trav} is chosen such that $s_{\text{ref}} \approx \alpha_{\text{trav}}/t_\Delta$, where t_Δ is the sampling period of the NLP.

ii. Next, the control input cost function is defined as

$$\epsilon(\boldsymbol{u}_k, \boldsymbol{u}_{k-1}) = \boldsymbol{u}_k^\top \boldsymbol{\Lambda} \boldsymbol{u}_k + (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})^\top \boldsymbol{\Delta} (\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \tag{4.11}$$

where $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda}) > \boldsymbol{0}$ and $\boldsymbol{\Delta} = \text{diag}(\boldsymbol{\delta}) > \boldsymbol{0}$ are tuning matrices. These terms collectively help conserve power and reduce the input variations, consequently lowering environmental and operational costs.

iii. Finally, the constructed risk cost function is used to keep the grounding risk levels low:

$$\rho(\boldsymbol{x}_k, \boldsymbol{\theta}_k) = \sum_{j=1}^{J} \mu_1 e^{-\zeta_1 s_{1,j}} + \mu_2 e^{-\zeta_2 s_{2,j}} + \mu_3 e^{-\zeta_3 s_{3,j}} \tag{4.12}$$

where $s_{1,j}$, $s_{2,j}$ and $s_{3,j}$ are defined as in (4.2), with respect to each grounding obstacle $\sigma_j \in \Theta_J$. Here, the dot product within $s_{3,j}$ scales the disturbance contribution toward the grounding obstacles in any orientation around the ship, i.e. increasing the risk close to an obstacle to the east of the ship if the wind, waves or currents are coming from the west, etc. Negative dot products are however set to zero, disregarding "favorable" disturbances with respect to perceived risks. The remaining variables were defined in Section 4.4.3.

It may be noted that all of the initial safety inequalities are transformed into risk costs, in favor of being formulated as explicit constraints to ensure safe distances between the ship and obstacles. Thus, this "soft constraint" formulation utilizes violatable risk costs in order to acknowledge that grounding risks may still be evaluated even if high, and guaranteeing NLP feasibility. Using exponential terms for the obstacle or grounding risk costs serves to strongly dominate the other objectives in the cost function, heavily favoring staying safe from grounding obstacles. The grounding risk sensitivity constant ζ may for this purpose be tuned for optimal behavior.

4.5.4 Nonlinear programming

The dynamic ship model is discretized in order to solve the problem numerically. The continuous time variable t is divided into a time grid of N intervals on the horizon T, defined by discrete time instants $t_k \in \{t_0, t_1, ..., t_N\}$. The system inputs are discretized as piecewise constant over that time grid, i.e. $\boldsymbol{u}_k = \boldsymbol{u}([t_k, t_{k+1}])$. The system state is discretized using a numerical integration function based on the widely used *Runge-Kutta 4th order method*: $\boldsymbol{x}_{k+1} = \boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k)$. The discretization allows one to treat (4.4) as a nonlinear program by defining a vector of decision variables

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{x}_0^\top & \boldsymbol{q}_0^\top & \boldsymbol{u}_0^\top & \dots & \boldsymbol{x}_{N-1}^\top & \boldsymbol{q}_{N-1}^\top & \boldsymbol{u}_{N-1}^\top & \boldsymbol{x}_N^\top & \boldsymbol{q}_N^\top \end{bmatrix}^\top \tag{4.13}$$

where $\boldsymbol{q}_k$ is the vector of additional decision variables related to mission objectives defined in Section 4.5.3. Additionally, a parameter vector comprised of various control settings, desired states and coefficients through time is denoted as $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_0^\top & \dots & \boldsymbol{\theta}_N^\top \end{bmatrix}^\top$. The parameters considered in this example are

$$\boldsymbol{\theta}_k = \begin{bmatrix} s_{\text{ref}} \\ \alpha_{\text{trav}} \\ \boldsymbol{\Theta_J} \end{bmatrix} \tag{4.14}$$

for all t_k where s_{ref} is a constant reference transit speed and α_{trav} is a path progression parameter (see Section 4.5.3) across all N control intervals. The grounding obstacles are in this work modeled as a union of convex polygons $\sigma_j \in \boldsymbol{\Theta}_J$. Thus, $\boldsymbol{\Theta}_J$ is the collection of all grounding obstacles, enumerated by $j = 1, ..., J$ in (4.12).

In other words, the entirety of all grounding or allision polygons present in some specified data (sub)set provided by the ENC module is included for distance calculations in each time step, yielding J distance computations for each subsequent NLP solver iteration, for each time interval k throughout the time horizon of N sampling intervals. Moreover, the distance to each polygon is computed based on its inherent complexity, given by its number of vertices and edges. Thus, it is important to limit the resolution of the grounding polygons such that the computation time is fast enough (**SC4**), e.g., by simplifying and reducing the number of vertices of each ENC polygon to some extent relative to the size of the ship or the size of the considered environment during application runtime. Note, however, that simplifying the complexity, which consequently also lowers the resolution of the grounding obstacles, directly affects the spatial error margins needed for safe navigation near the obstacle polygons. As a result, both d_{sep} and d_{safe} from (4.2) must be defined with respect to the resolution and inherent complexity of the grounding obstacles as provided by the ENC module (**SC3**).

The resulting NLP is defined as

$$\begin{aligned} C(\boldsymbol{\theta}, \boldsymbol{X}_0) = \min_{\boldsymbol{w}} \quad & \phi(\boldsymbol{w}, \boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{0} \\ & \boldsymbol{h}(\boldsymbol{w}) \leq \boldsymbol{0} \end{aligned} \tag{4.15}$$

where $C(\boldsymbol{\theta}, \boldsymbol{X}_0) \in \mathbb{R}$ is the minimum cost generated by a given set of parameter values and initial conditions $\boldsymbol{X}_0$, i.e., a full initial trajectory of N ship state vectors $\boldsymbol{x}$ given by the path $\boldsymbol{r}$: The preplanned path and ship speed reference parameters are used to calculate a reasonable initial guess for the ship trajectory $\boldsymbol{X}_N$. For all subsequent NLP solve calls, the last solution (forward shifted one time step) is used as the initial guess, i.e. warm start. The inequality constraints $\boldsymbol{h}(\boldsymbol{w})$ are given by (4.5), and the equality constraints $\boldsymbol{g}(\boldsymbol{w})$ hold the system dynamics:

$$\boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k) - \boldsymbol{x}_{k+1} = \boldsymbol{0}, \qquad k = 0, 1, ..., N-1 \tag{4.16}$$

The cost function ϕ was defined and discussed in Section 4.5.3, based on the risk cost formulation of Section 4.4.3. Note that the discretization chosen here is based on the *direct multiple-shooting* approach [92]. Because of the nonlinear dynamics and since the obstacles yield a non-convex feasible set, the NLP (4.15) is non-convex. As a result, the goal is to compute a feasible and *local* optimal solution for a given control horizon N and initial conditions. Moreover, only costs balancing is as noted utilized to achieve the desired control behavior, rather than using hard constraints in addition to the ship dynamics and the natural input constraints. This ensures feasibility of the NLP solutions.

Table 4.13: Model parameters

Parameter	Symbol	Value	Unit
Path step size	α_{trav}	75	m
Overall ship length	L_{oa}	75	m
Transit reference speed	s_{ref}	2.5	m/s
Frontal projected wind area	A_{Fw}	110	m^2
Lateral projected wind area	A_{Lw}	624	m^2
Max propulsion force	$f_{\max}$	400	kN
Max rudder turn rate	$\omega_{\max}$	2.0	rpm
Viscous damping force surge	X_u	5.0×10^1	kN s/m
Viscous damping force sway	Y_v	2.0×10^2	kN s/m
Viscous damping force yaw	N_r	3.0×10^4	kN s/rad
Hydrodynamic added mass surge	$X_{\dot{u}}$	4.4×10^4	kg
Hydrodynamic added mass sway	$Y_{\dot{v}}$	8.6×10^5	kg
Hydrodynamic added mass yaw	$N_{\dot{r}}$	4.9×10^7	kgm^2
Rotational inertia yaw	I_z	9.8×10^8	kgm^2
Rigid-body ship mass	m	1.5×10^6	kg

4.6 Results

4.6.1 Simulation verification

The purpose of this section is to present simulation results which showcases how the constructed risk cost terms affect the performance and resulting trajectory of the MPC scheme, to serve as the verification method for this work. Thus, the goal is to verify that the safety constraints of Table 4.11 are satisfied during the autonomous navigation shown in the simulations, with respect to suitable compliance relative to the expected behavior. The model parameters used for simulation verification are presented in Table 4.13.

Figure 4.3 presents a reference trajectory resulting from a simulation using the

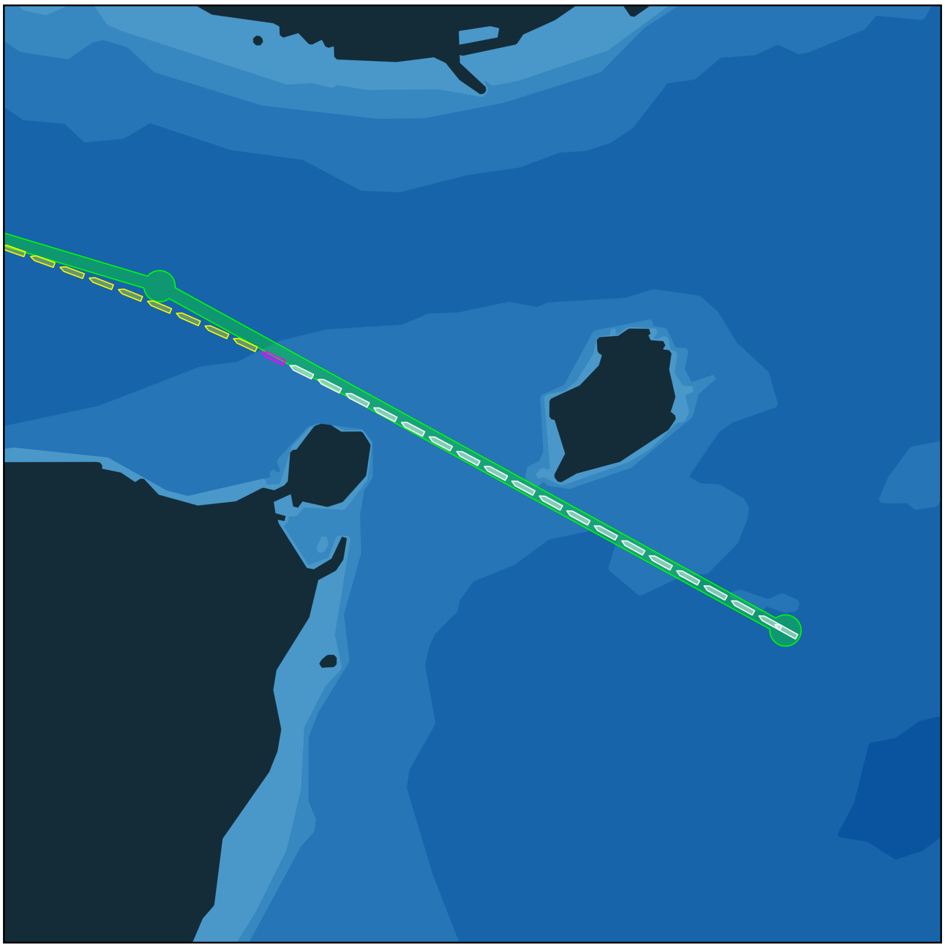


Figure 4.3: A demonstrative path following simulation using MPC, not including any risk cost terms in the cost function.

MPC scheme discussed in this chapter, for the purpose of comparison to later simulation trajectories. A square example region with an area of $9\,\text{km}^2$ just south of Giske island in Norway is to demonstrate various aspects and discussion points related to the construction of the risk cost terms. Using the visualization capabilities of the ENC package [37], several information overlays are added. Three waypoints are given as a route reference (the western-most off-screen), and a green path is drawn between the waypoints. The ship trajectory is shown as a white trail of ship poses (i.e., the position and orientation of the ship at each time interval), and the future trajectory solution computed by the MPC is shown in yellow. The current ship pose is shown in magenta, recorded as a snapshot during a simulation run. It may be noted that the path following and resource consumption costs are tuned such that the trajectory is allowed to deviate slightly from the planned path, in order to save time, rudder actuation, or fuel into the time horizon.

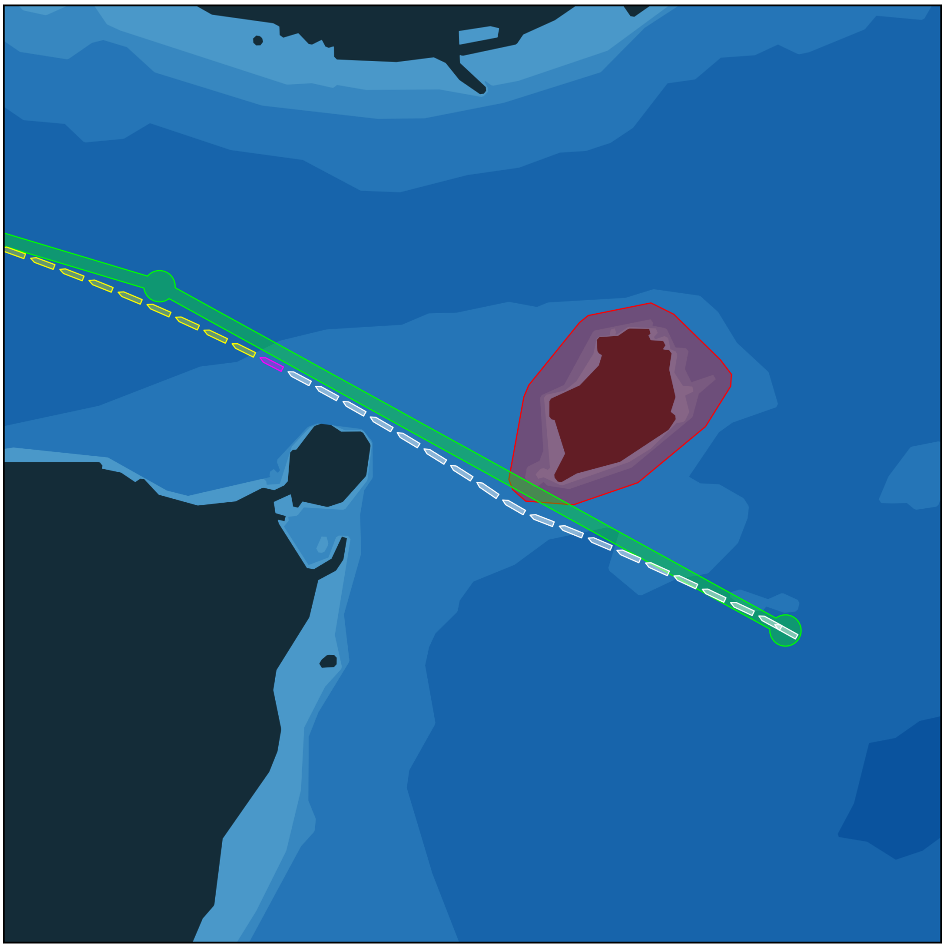


Figure 4.4: MPC simulation showcasing the effect of the first risk cost term (**SV1a**) for only one grounding obstacle shown in red.

Figure 4.4 shows a simplified example in which only the first risk cost term for **SV1a** is included in the complete cost function during the MPC run. Moreover, only one single island is considered as a grounding obstacle in this example, for clarity. Shown in red, the single grounding obstacle is constructed as the convex hull of all ocean depths more *shallow* than 10 m, closest to the initial ship position. The first waypoint serves as the starting point of the simulation, and is shown as a green disk. The convex hull of the original grounding obstacle is chosen, based on the assumption that concave crevices along the boundary of any obstacle polygon are not considered for purposeful navigation along a planned path. Furthermore, though the island itself (i.e., land mass above sea level) is not intersected by the planned path, the convex grounding obstacle intersects slightly. This is as a result of an additional safety margin added to the grounding obstacle in all directions, in Figure 4.4 set to 50 m. This static margin is added at the ENC level in order to ensure that inaccuracies related to the numerical charts data and/or e.g. tides are accounted for, and is considered a separate discussion than the cost term added related to the d_{safe} variable of **SV5a**. This static margin should be kept small, and is exaggerated in this work for the purpose of demonstration. See the discussion related to Figure 4.9 on this topic.

The consequence of the combination of these factors is that the resulting optimal trajectory computed by the MPC module deviates from the planned path close to the grounding obstacle, as expected. Thus, it is apparent that the first risk cost term is sufficient to produce the necessary behavior in order to avoid grounding obstacles, even when the grounding obstacle is intersecting with the original path.

The next example is shown in Figure 4.5, which includes the south-western island group as a grounding obstacle. Notice the small perturbation of the ship trajectory close to the northern tip of the obstacle, indicating that both islands have an effect on the risk cost term, as expected. Moreover, it is shown that the risk cost is properly defined also for grounding obstacles located on opposite sides of the ship. The northern island is, however, located far away, and is through the inverse exponential weighting and resulting value of the risk cost term negligible compared to the closer islands. This is considered appropriate for the specific environment domain shown in the example demonstrations in this work.

Figure 4.6 demonstrates how the addition of the second risk cost term related to **SV5a** appropriately adds an extra virtual safety buffer or safeguard with respect to nearby grounding obstacles, indeed in accordance with the wording of **SC5**: The risk levels of trajectories closer to the minimum separation distance plus a safety distance margin must be weighted sufficiently high, such that the ship to a larger degree is able to avoid grounding if unexpected failures or disturbances are introduced – which effectively increases the time available to, e.g., restart the ship engines after a power blackout in order to regain ship control. The trajectory shown in Figure 4.6 is ultimately considered an appropriate trajectory for sufficiently safe navigation through the strait as presented here.

It may be noted, however, that this behavior also can be achieved in this example

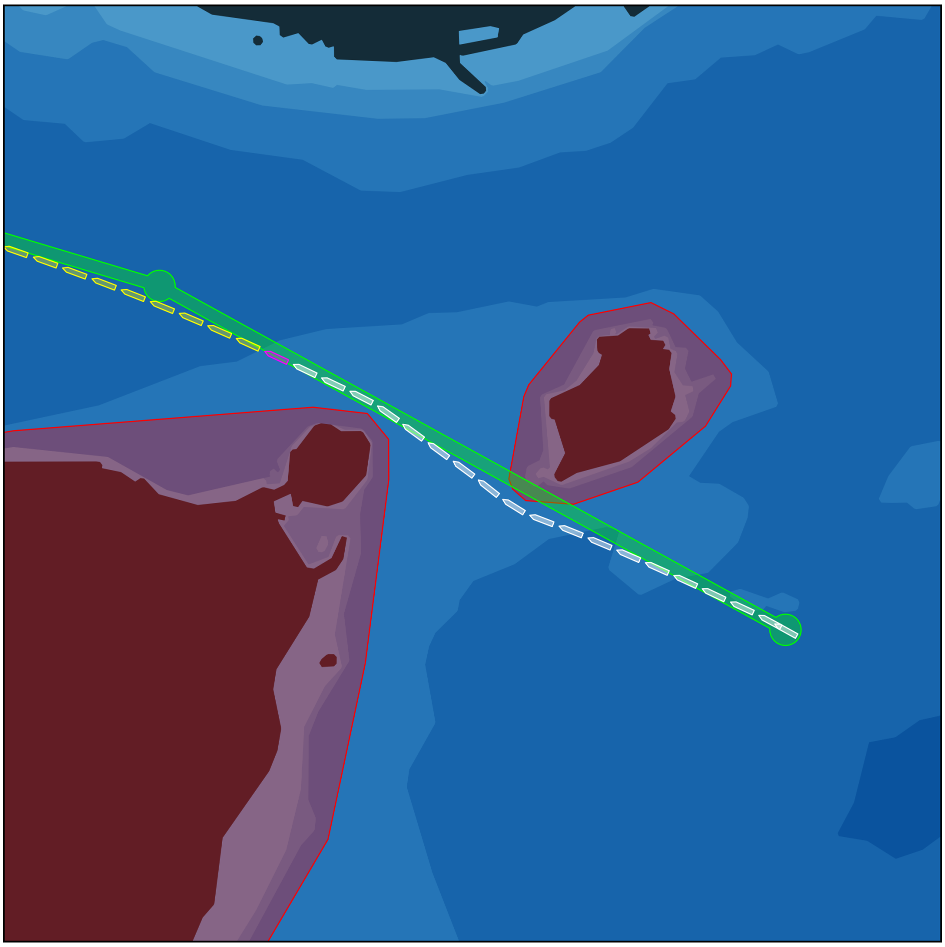


Figure 4.5: MPC simulation utilizing the first risk cost term (**SV1a**) to avoid two opposing islands at each side of the ship.

simply by adjusting the sensitivity coefficient ζ_1 of the first risk cost term, as may be inferred through the mathematical similarities of both terms. The second risk cost term is nevertheless included in this work, for completeness.

In contrast to the previous example demonstrations, Figure 4.7 shows a simulation in which the non-convex forms of each grounding obstacle are used in place of the previous convex hulls. Additionally, the minimum distances between the ship and both nearby grounding obstacles are shown as orange lines, in order to clearly visualize the exact coordinates of each distance calculation provided to the exponential risk cost terms during MPC optimization.

Note, however, that the resulting trajectory of Figure 4.7 is identical to that of Figure 4.6. This is a consequence of the fact that only the minimum distances

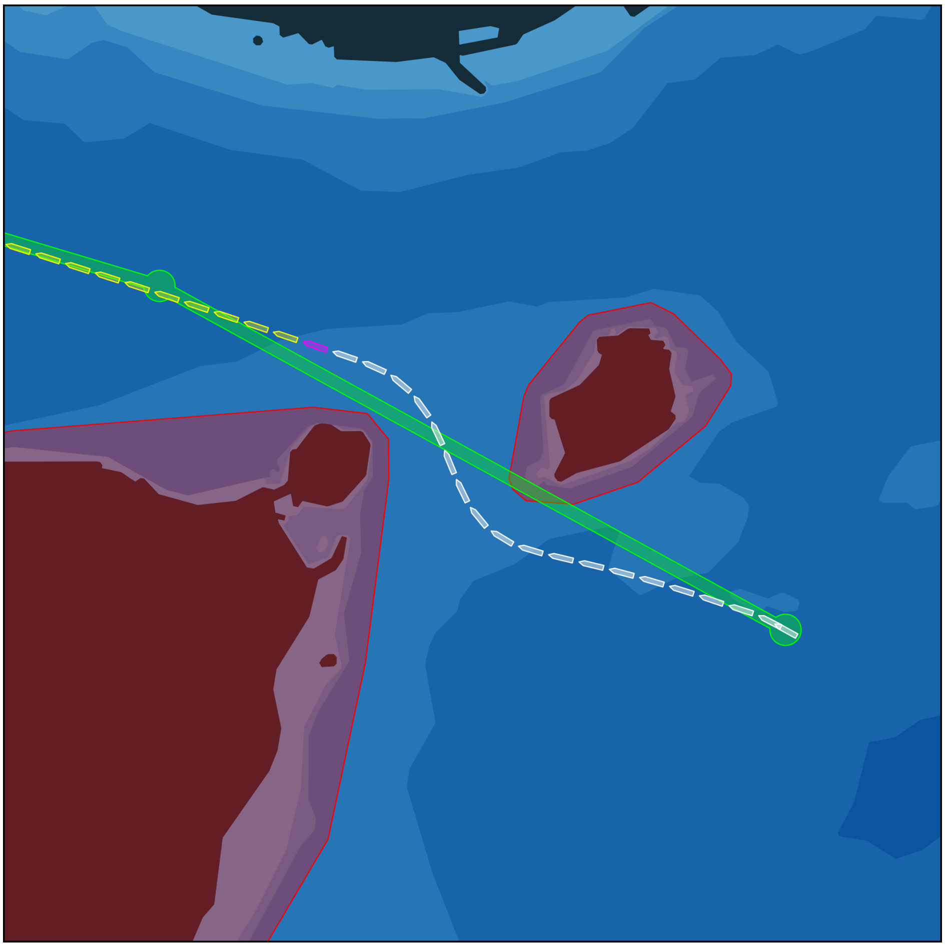


Figure 4.6: MPC simulation with increased grounding sensitivity through the second risk cost term (**SV5a**), or alternatively by tuning of the ζ coefficient.

between the ship and all obstacles are provided to the risk cost terms. By definition, no point not located exactly on the boundary of the convex hull of an obstacle may exist as a minimum distance to the ship. Thus, the advantage of calculating the convex hulls of all obstacles during initialization of the algorithm through the functionality provided by the ENC module is clear. This results in fewer polygon vertices for distance computations, which leads to faster solver performance in accordance with the objective described in **SC4** related to computational feasibility. The approach may nevertheless also be used on non-convex obstacles if necessary, at a price of reduced computational efficiency.

Figure 4.8 presents the same orange distance visualizations as those of Figure 4.7, applied to the convex grounding obstacles in Figure 4.6. In addition to the previous discussion with respect to convexity, it is apparent that these line segments are more

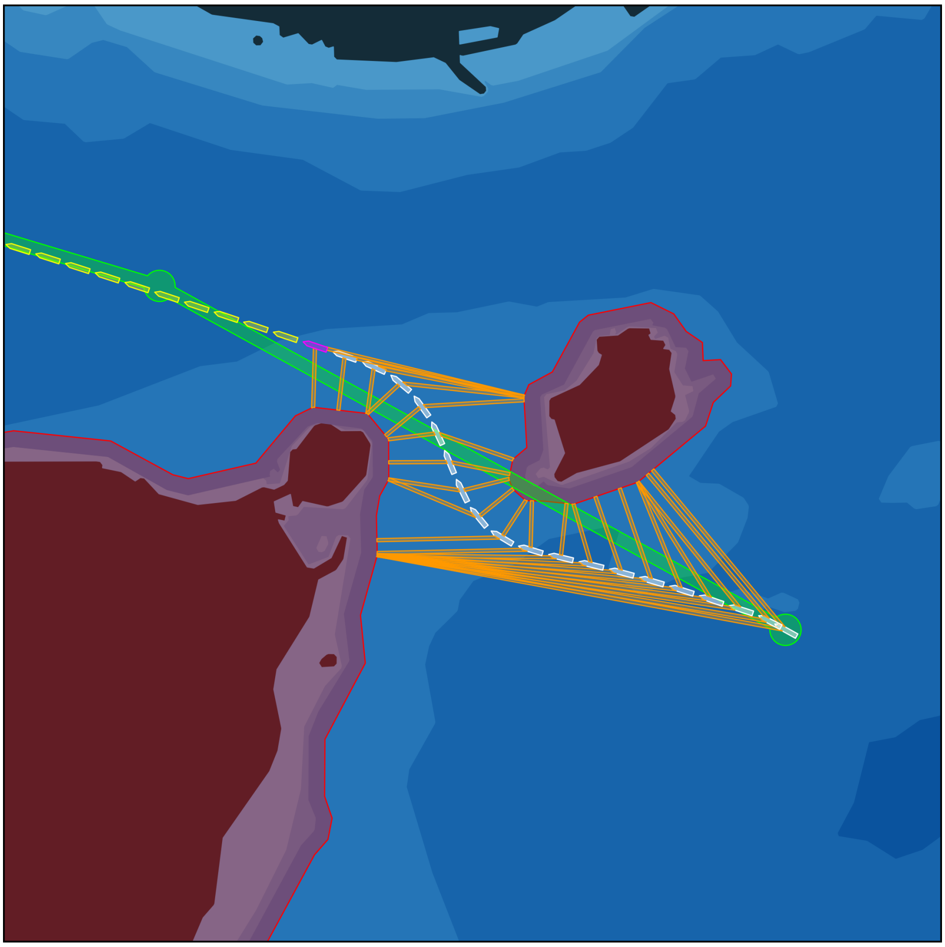


Figure 4.7: MPC simulation with non-convex grounding obstacles, and minimum distances shown as orange lines between each past vessel position and the obstacle polygons.

well-behaved and the variation between each consecutive line is smaller, which in turn improves the smoothness of the gradients as present in the MPC during the NLP solve. This effect is seen directly by faster solver timings.

The last example related to the first two risk cost terms and the structure of the grounding obstacles provided by the ENC is presented in Figure 4.9, in which the safety buffer added to the obstacles are increased from 50 m to 200 m. This is shown in order to demonstrate the effects of this static approach compared to the addition of the increased safety risk cost applied through the second risk cost term.

One may notice how the resulting trajectory in this case is significantly more restricted through the consequently narrower isle strait. It may be argued that the approach of adding static safety margins in this manner – i.e., through the initial

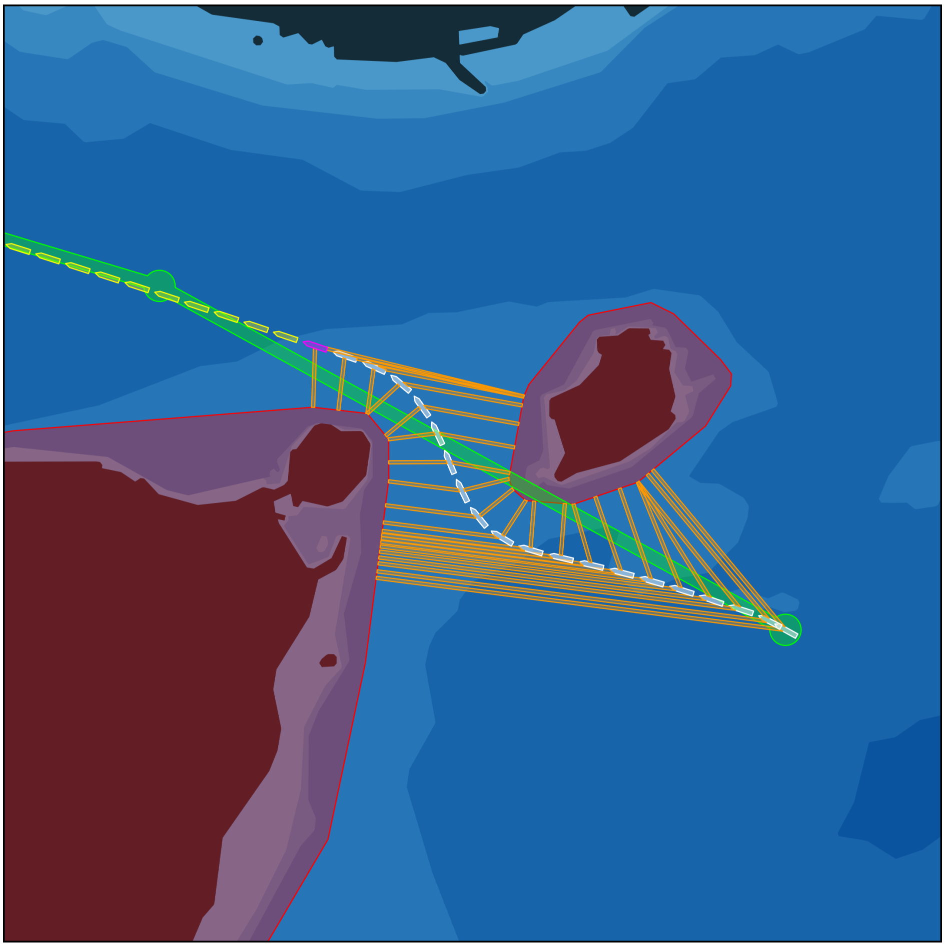


Figure 4.8: MPC simulation showing the orange minimum polygon distances for convex grounding obstacles. Less variance between each evaluated minimum distance along the trajectory yields improved solver performance.

creation process and structure of grounding obstacles, as constructed by the ENC module – reduces the flexibility of the MPC algorithm, and should be avoided in favor of the added virtual safety margin cost exemplified through the second risk cost term in this work. Moreover, it is clear that if the safety buffer is too large (e.g., 400 m), the strait would be entirely closed, which would lead to an entirely new behavior in which the ship must sail around the resulting merged obstacle of both islands. Though potentially appropriate in some cases, this approach is considered unstable and prone to produce irregular solutions given any particular environment.

The effects of the third and last risk cost term of the constructed risk cost function produced through the methodology presented in this chapter is visualized in

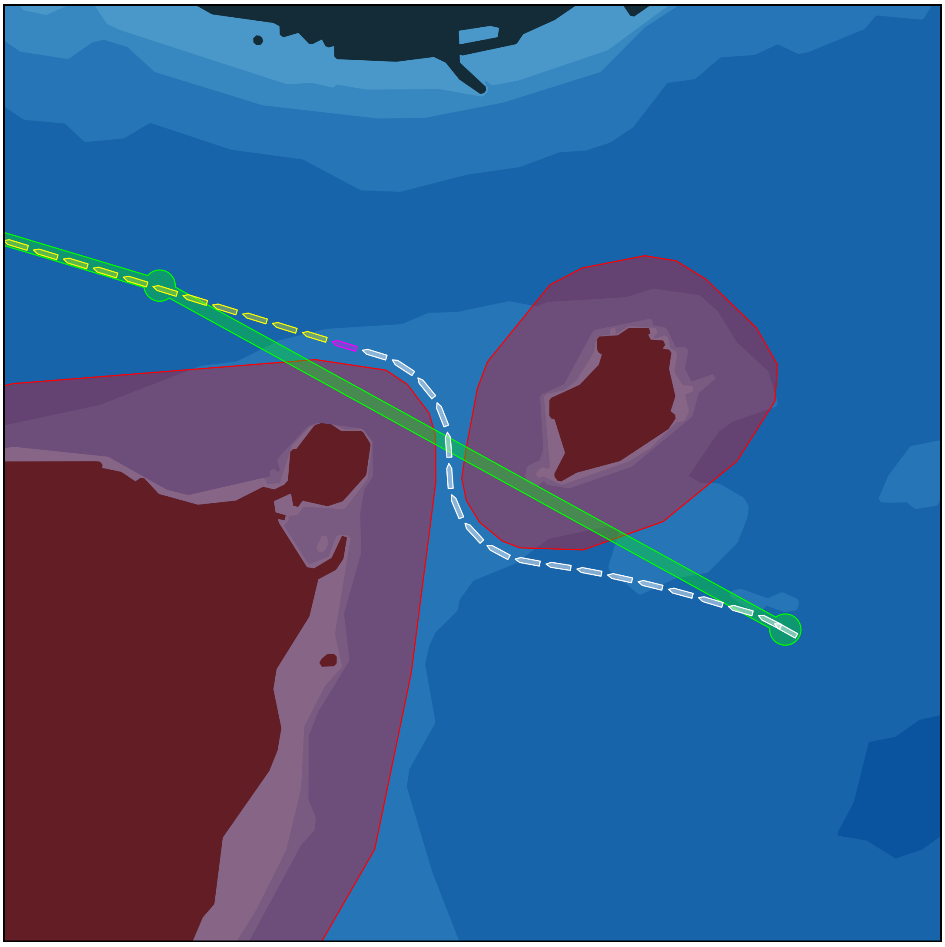


Figure 4.9: MPC simulation with an increased safety buffer added to the convex obstacle polygons, resulting in less flexible performance and lower degrees of solver feasibility.

Figure 4.10. Though the effects of the cost term is limited with respect to changes in the trajectory due to the direction of the wind compared to the vessel heading, the example illustrates how both isles each contribute separately to the risk cost scaling. Here, the two grounding obstacles located at the port and starboard side of the ship are given the colors red and orange, respectively. Furthermore, the value of s_3 at each time step is multiplied by the unit vectors with directions equal to the *opposite* of the direction of the vectors between the ship center and the minimum distance point on each obstacle. The wind disturbance direction and wind velocity is shown in the compass in the top-right corner of the environment plot.

Intuitively, the red and orange arrows point away from their respective grounding obstacles at each side of the ship, due to the risks increasing *toward* the obstacles – which the MPC attempts to minimize, effectively directing the ship *away* from

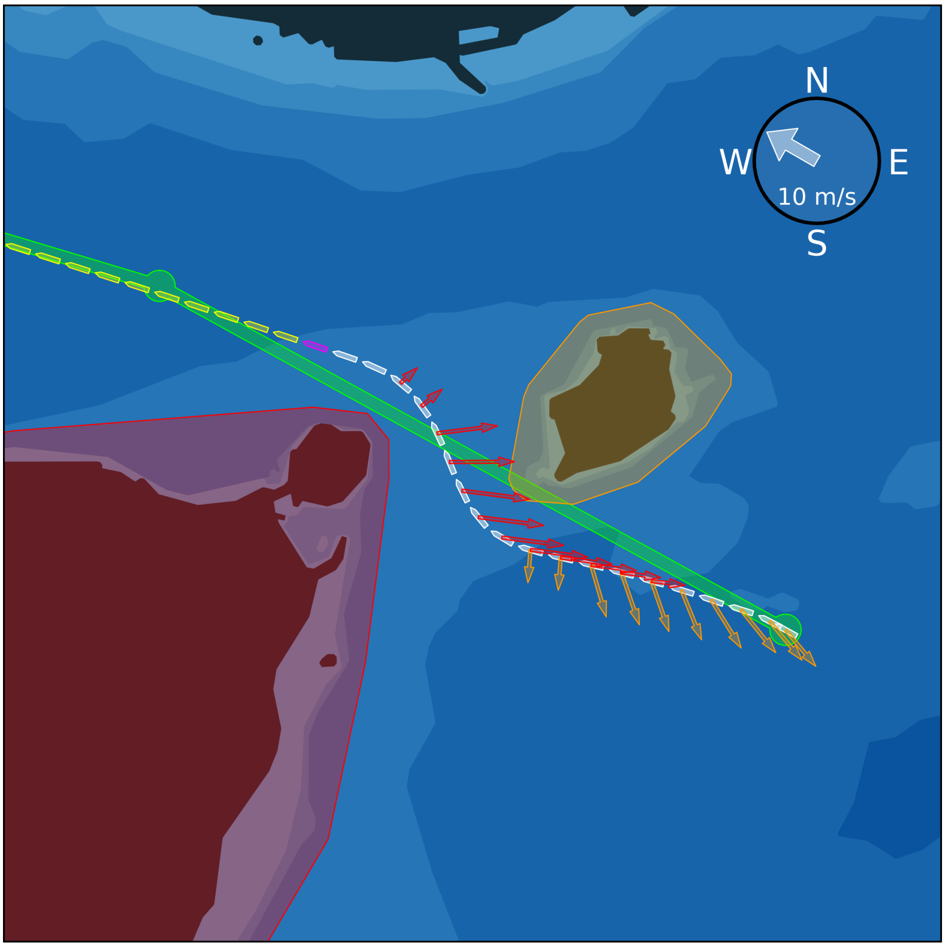


Figure 4.10: MPC simulation including visualization of the third risk cost term (**SV5b**), demonstrating the effects of a wind disturbance with respect to nearby grounding obstacles.

the obstacles in accordance with the arrow directions. Moreover, the length of the arrows are proportional to the gradient of the risk cost term at each point, i.e., the magnitude of how much the perceived risks promote evasive maneuvers with respect to each grounding obstacle.

This illustration aims to visualize how the external disturbances (here limited to wind disturbances only) affect the risk levels through the third risk cost term. In this case, the wind direction is in an on-land direction towards both grounding obstacles with respect to the initial location of the ship, yielding positive scalar products. Thus, if the wind force is driving the ship towards the shoreline or other grounding obstacles, the risk level increases as expected.

Figure 4.11 presents a situation in which the wind direction is directed towards the south-west grounding obstacle shown in red. Notice how the trajectory in this case intersects the orange grounding obstacle, due to the scalar products between this obstacle and the ship location being non-positive and consequently disregarded. This is indeed the expected behavior, as this risk cost term is only concerned with the risk levels related to external disturbance forces, and must be combined with the first (two) risk cost term(s) in order to produce appropriate trajectories during autonomous navigation. It is argued that no disturbance forces should be included as a positive or favorable driving force toward safe autonomous control, and as such is factored out in this context.

Lastly, the quality of the solutions are considered appropriate: All figures are

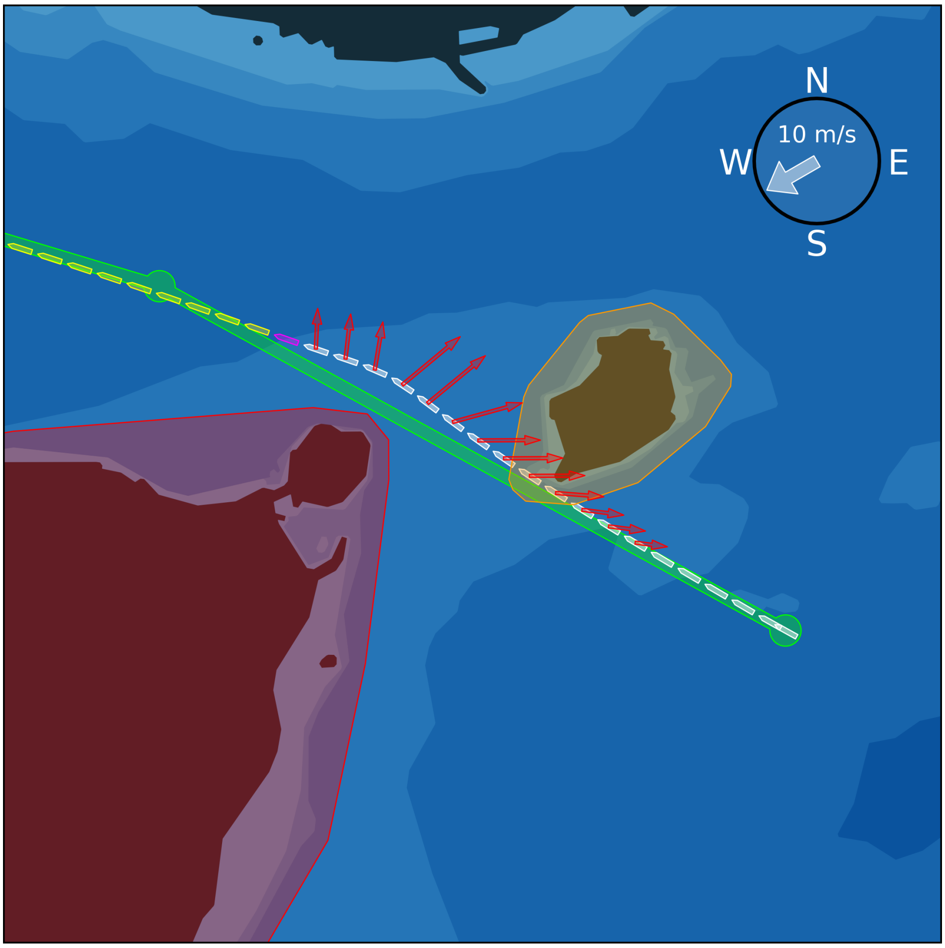


Figure 4.11: MPC simulation visualizing how the third risk cost term (**SV5b**) is only significant for on-land wind directions.

generated within 6.5 s to 12.4 s on an Intel® Core™ i7-9700K (3.6 GHz), with a 20 min future horizon using a sampling time of 30 s. This means that the algorithm is able to predict, optimize and plot the future ship states along the route for 40 time intervals into the future, repeated 20 times (one for each trailing ship pose shown in gray), within a maximum of 12.4 s on a desktop computer. The average optimization timing for a single run is thus well below one second. Consequently, the proposed algorithm may be repeatedly utilized online to predict the real-time effects of the latest measured or predicted weather conditions on a voyage 20 min into the future, recalculated every second using this setup. It is however noted that this performance is strongly dependent on the trade-off between the chosen data resolution and the resulting solution quality. Moreover, the MPC solutions consistently convergence to similar sets of trajectories across various tested simulations of different initial conditions, further assuring the validity of the approach. This concludes the verification of the constructed risk cost terms, by human assessment and review of the presented simulation demonstrations.

4.7 Discussion

4.7.1 Choice of methods: STPA and MPC

The risk analysis part of this work is based on STPA, mainly due to its feasibility for large and complex control system structures such as autonomous navigation, control and awareness systems of a ship presented in this work. By focusing on potential unsafe control actions, loss scenarios, and associated safety constraints, the integrity and safety of the system is thus considered through the emergent behavior of the interconnected system as a whole.

For the quantifiable and optimization side of the problem, an MPC scheme was chosen due to the flexibility and robustness of the method. As presented and discussed in this work, the MPC approach is largely capable of solving quite complex optimization problems if given appropriate and well-defined system dynamics and cost formulations. In this regard, the difficult part of the method is rather to provide the MPC with a feasible and satisfactory cost function, as well as an initial guess that produces the desired results when used for autonomous navigation.

4.7.2 Risk analysis versus optimal control

There is a definite distinction between the fields of qualitative risk analysis and numerical optimal control. As such, it is challenging to standardize a transformation between STPA and MPC. Firstly, STPA is considered an effective method for identification of hazardous events for a range of applications. However, the results of the analysis often yield extensive collections of possible failures or unsafe control actions. A challenge with STPA is that it only considers negative losses, which means that any rewards and trade-offs between risks and system performances are not analyzed or supported in decision-making. Furthermore, STPA is a qualitative approach, which means that several additional steps are needed to translate the results into meaningful representations in MPC. A methodology for such a "translation" is provided in this chapter.

One of the main strengths of the traditional applications of MPC is that it usually has relatively straightforward and well-tested costs, similar to feedback controllers, such as the linear-quadratic regulator (LQR). If instead the cost terms of the MPC are extensively non-smooth or nonlinear, feasibility and solvability problems for a given real-time constraint may arise. It is also apparent that operational optimization is not the same as emergency management. During extreme conditions, the focus is arguably only to handle or contain the situation to a satisfactory degree, to avoid further loss of control within strict and short time periods. Hence, there is usually limited benefits to be gained from optimizing the best possible solution during these scenarios. The method is nonetheless chosen due to the parallels between MPC and human decision-making, in the sense that humans inherently weigh costs (negative consequences) and rewards against each other when making most logical decisions on a day to day basis. The challenge, as previously mentioned, lies in the uncertainty and intricacies that arise when quantifying the decision variables for optimization, and as such, this must be performed with care.

4.7.3 Risk cost verification

There is no conclusive way to verify if the constructed costs of the OCP and NLP are sufficient to satisfy the given safety constraints for *any* possible scenario. Generally, some form of evaluation method has to exist for any given method for which the performance is to be examined. However, for the performance to be evaluated and classified, real meaning and/or actual values need to be assigned to abstract concepts for the purpose of assessment, as it is not possible to know the "true" or objective risk. This has been shown to be exceedingly difficult in the case of risk quantification or cost estimation, due to the inherent ethical and computational challenges of evaluating human lives and environmental damages [111], [112], and the discrepancy between the meaning of different interpretations of core terminology, such as *risk*. Even for humans, risk aversion in itself is a highly subjective concept, and it is difficult to conclude upon a universal perception of the term. For example, one may relate (minimum) distances to the size of the vessel, such that larger vessels require larger minimum separation and safe distances. Another suggestion can be to utilize common law rulings to inform what constitutes a safe distance. Until a global consensus is reached and explicit definitions are obtained for these notions, however, the evaluation of risk "costs" is still somewhat abstract, and is consequently treated as such for the time being.

Thus, verification of the resulting risk cost function is approximated through visual presentation and human interpretation of the simulation results presented in this work. More testing and research is recommended to achieve higher technology readiness levels.

4.7.4 Safety inequalities and hard constraints

Even though, for example, the safety inequality of **SC1** may be implemented as an explicit (hard) constraint in the constructed control problem, it is argued that hard constraints simultaneously reduce the feasibility and may raise the computational complexity of the NLP to be solved [113]. Thus, the decision was made to relax this constraint and allow it to be violated if the situation calls for it. Note that this subsequently allows for handling of hazardous situations in more complex implementations, i.e., scenarios in which the combined decision-making algorithm deems the cost of purposely grounding the ship favorable to other alternatives (e.g., to avoid a complete loss of the vessel, reduce the probability of an oil spill and/or potential loss of human lives).

Consequently, all explicit inequality constraints were formulated as weighted risk costs, in accordance with Steps 5 to 10 of the proposed method. However, inspection of the other risk cost terms reveals that the act of weighting distances close to grounding obstacles with high values of risk is already achieved by the second (and to some degree the third) term of the constructed risk cost function (4.12). As such, one may if desired merge the terms if the corresponding cost coefficients are appropriately adjusted such that the cost term strongly and sufficiently discourages violation of the minimum separation distance to grounding obstacles.

This is left to be further explored in future works.

4.7.5 The structure of the risk cost terms

Making the cost terms monotonic and convex greatly simplifies the complexity of the NLP, leading to faster solving times as well as more predictability with respect to solution quality or expected trajectories (performance). Using various polynomials in place of exponentials was in this context considered, for different levels or ranges of assigned risk priority numbers. However, compared to e.g. x^{-2} or x^{-4}, the exponential function does not contain singularities for inverse proportional relationships such as e^{-x}, making it a more suitable candidate for continuous risk scaling when approaching the safety boundary. Additionally, it is argued that due to limited function domains (e.g., 0 - 10 km horizon ranges around a ship in a specific environment), there is little to no practical difference between terms of e.g. the form x^{-2} compared to ae^{-bx} for a given domain range and appropriate tuning of the a and b parameters. It is thus assumed that any approximately equivalent behavior may be obtained through the exponential terms alone.

In summary, the first risk term related to **SV1a** is introduced to enforce minimal distances to grounding obstacles by a natural inverse exponential relationship. For **SV5a**, it is assumed that the objective of not grounding is unchanged during loss scenarios with reduced propulsion capabilities, regardless of environmental disturbance forces potentially being dominant. However, **SV5a** serves to encourage even more conservative avoidance distances to grounding obstacles if such scenarios occurs. The magnitude of this effect may be tweaked by adjusting μ_2 and ζ_2 relative to μ_1 and ζ_1. This is due to the fact that by definition, the safety buffer variable d_{safe} of **SI5a** is merely a shifted or more conservative formulation of d_{sep} from **SI1a**. Lastly, the third term of **SV5b** is modified to increase only with positive scalar products between the wind disturbance vector and the vector between the ship and any grounding obstacle σ_j, scaled by the disturbance velocity v_d. This effectively adds an additional safety margin toward down-stream obstacles, which may improve the initial system state if loss scenarios such as machinery faults occurs.

4.7.6 Simulation performance and parameter tuning

In general, the performance of the MPC is both expected and verified as appropriate. However, the simulation results are heavily dependent on the specific tuning of parameters as applied in the example demonstrations. Recent research indicate that there exists methods for data-driven or automatic tuning of simple problems [114], [115], in which the latter software is available as an open-source Python package. This is nevertheless a limitation of this study, which means that further evaluation and verification of the simulation performance is necessary.

Similarly, the connection between the assigned RPN and the resulting coefficient values of the exponential risk cost terms presented in the methodology is somewhat abstract. The RPN only provide initial values for parameter tuning, and as such, the cost coefficients must be fine-tuned to each application on a case-to-case basis. Thus, detailed parameter tuning was mostly left out of the scope of the main contribution of this work.

4.8 Future work and extensions

4.8.1 Forward velocity and risks ahead of the ship

In autonomous navigation for surface vessels, the forward velocity and the uncertainties and related risks in the front of the ship are considered as significant contributions to the risk of a given situation, potentially dominating other risk factors, such as the uncertainty and grounding risk related to lateral on-land wind disturbances. It is recommended that this is addressed in future implementations.

4.8.2 Machinery system additions

Additional risk cost terms for wear and tear or component failure in the machinery system related to high-intensity operation periods or over-use may be included in future versions of the implementation, including certain thresholds or dynamically weighted costs for machinery utilization. With varying AMM modes, the machinery system may experience changes in its inherent uncertainty and probability estimations for, e.g., a blackout scenario based on various available system modes or power configurations. The margins could be smaller with safer machinery modes, and the ship may in such situations consequently sail closer to land.

Fuel consumption modeled in the machinery model may similarly also yield a more thorough understanding of the actual costs related to various control actions.

4.8.3 NLP solver considerations

Due to the complexity of the environment (i.e., sea depth polygon obstacles provided by an ENC module) in the constructed NLP, feasible solutions that successfully carries out the given mission are not guaranteed. Moreover, if the solver is not able to converge to a solution within the given maximum time limit, the returned solution may be dangerous or even physically impossible with respect to the defined ship dynamics. In this case, the MPC could fail to produce a suitable trajectory, and this is regarded as a drawback to this method. A potential remedy to this challenge may be to, e.g., employ the use of an additional backup controller and a performance monitor to assume emergency control or make the human intervene in the ROC if failures or problems are detected in the MPC, as suggested by recent research [9].

Moreover, the complexity or resolution of the mapped grounding obstacles constructed during initialization of the ENC, as well as the discretization step or resolution of the ship trajectory (i.e., sampling time) are of significant importance with respect to the performance of the NLP solver. Lower spatial and temporal resolutions reduce the time complexity of the ENC minimum distance computations, but also decrease the accuracy and confidence of the computed optimal solution. An appropriate balance between these essential factors is in general difficult to determine, and must in addition to the cost function parameter tuning be established and verified on a case by case basis.

As a result of the non-convexity of the ENC grounding obstacles and their respective risk costs, no global minimum solution is guaranteed. Hence, providing a suitable initial guess to the NLP solver during setup is critical in order to both achieve adequate solver performance and to ensure feasibility of the optimal NLP solutions. It is thus necessary that a conservative initial guess is properly constructed such that the solution converges to an appropriate local minimum, with respect to the expected trajectory of the navigational mission given to the ship. Due to warm start, subsequent initial guesses are provided to the solver as the forward shifted solution of the previous solve, and are consequently also largely dependent on the solution guess of the very first solve. Furthermore, this initial guess should not diverge from the optimal solution to such a degree that the solver is not able to calculate and return the solution within the required time interval. It is proposed that evolutionary or genetic algorithms such as particle swarm optimization may serve as a possible alternative approach for this problem, in order to obtain global convergence less dependent on the initial conditions of the NLP.

4.8.4 Safety framework and risk model utilization

An additional consideration may be to transform or model tuning variables or cost coefficients into a safety framework, or to employ the use of an appropriate risk model. Recent research has shown that scenario-based MPC may utilize a probabilistic uncertainty model to achieve safe path traversal for e.g. inspection drones [116]. This may prove useful in order further structure the considered problem, to speed up the tuning process, and to enable use of models for resource limited embedded and real-time computing.

4.8.5 Other improvements

No sensor uncertainties were considered in this study, and may be implemented in future works. Additionally, parallel scenario simulations may be utilized during runtime to predict more complex risk pictures for any given time instant, beyond the current system state and environment conditions considered in this work. Collision avoidance and COLREGs handling are considered natural features of autonomous navigation systems, and should be included in future works. It is lastly recommended that more scenarios are investigated for analysis and simulation purposes in order to further increase the robustness and reliability of the MPC scheme.

4.9 Conclusion

A systematic and novel method has been proposed that enables the use the results of risk analysis to formulate an optimizable supervisory risk control problem through MPC, taking safety constraints and risk factors systematically into account. The risk analysis was performed by using STPA with a focus on anti-grounding for an autonomous ship. A method providing appropriate system state variables and equations and a risk-based cost function for an optimal control problem, based on the STPA results has been proposed. The optimal control problem was subsequently transformed into a nonlinear program and solved using an MPC scheme with a receding horizon approach. Several demonstrated control scenarios for an autonomous ship, simulated by an MPC scheme, show that the proposed method for construction of quantitative and optimizable risk-based costs based on safety constraints from STPA produces adequate and safe control trajectories. Additionally, the analysis has identified some vulnerabilities that should be addressed in future works. Ultimately, this work shows that constructing the MPC objective function based on the results from STPA produces ANS behavior appropriate for safe navigation of ships, thus supporting the hypothesis that increased levels of safety may be achieved by the MPC-based ANS through systematic analysis of unsafe control actions and hazards when designing the MPC cost function. This approach is consequently considered a reasonable bridge between the realms of qualitative risk analysis and numerical optimal control.

4.10 Appendix: Ship model and dynamics

To model the risk and enable optimization and control of related physical processes, the mathematical and physical relationships between the autonomous ship and its environment are formulated. This section defines the ship model used in this work, adapted from the ship model and the terminology as presented in [91].

The horizontal plane NE and BODY coordinate frames are defined as given in Figure 4.12. The NE coordinate system assumes a locally flat ocean surface plane and is oriented with its X- and Y-axes toward the true North and East, respectively. The BODY reference frame is positioned with its origin located in the centroid of the ship.

Given the previous reference frame definitions, the model variables are defined according to Figure 4.12:

$$\begin{aligned}
&\text{NE position} && \boldsymbol{p}^n_{b/n} = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2 \\
&\text{Body-fixed linear velocity} && \boldsymbol{v}^b_{b/n} = \begin{bmatrix} u \\ v \end{bmatrix} \in \mathbb{R}^2 \\
&\text{Body-fixed propulsion forces} && \boldsymbol{f}^b_b \;\; = \begin{bmatrix} X \\ Y \end{bmatrix} \in \mathbb{R}^2 \\
&\text{Attitude (yaw angle)} && \boldsymbol{\Theta}_{nb} \;\; = \begin{bmatrix} \psi \end{bmatrix} \in \mathbb{R} \\
&\text{Body-fixed angular velocity} && \boldsymbol{\omega}^b_{b/n} = \begin{bmatrix} r \end{bmatrix} \in \mathbb{R} \\
&\text{Body-fixed rotational moment} && \boldsymbol{m}^b_b \;\; = \begin{bmatrix} N \end{bmatrix} \in \mathbb{R}
\end{aligned}$$

where $\{n\}$ is the NE reference frame and $\{b\}$ is the BODY reference frame. The ship states, forces and moments are defined by the variables

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^n_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix}, \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix} \tag{4.17}$$

where $\boldsymbol{\eta}$, $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$ denotes the position, velocity and forces or moments vectors in the horizontal plane, i.e. surge, sway and yaw, respectively. Moreover, the principal rotation matrix in the XY-plane is defined as

$$\boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \in \mathbb{R}^{2\times 2} \tag{4.18}$$

Due to the roll and pitch angles being neglected, the body-fixed velocity vectors can be expressed in $\{n\}$ as

$$\dot{\boldsymbol{p}}^n_{b/n} = \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})\boldsymbol{v}^b_{b/n}, \quad \dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{\omega}^b_{b/n} \tag{4.19}$$

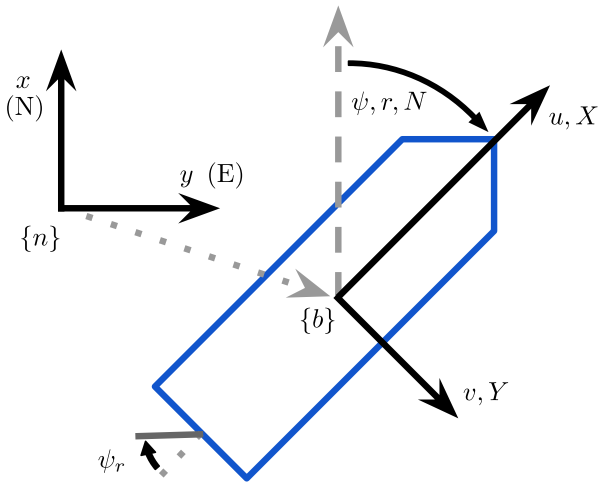


Figure 4.12: The model variables and coordinate frames of the autonomous ship used in this work.

The Jacobian $\boldsymbol{J}_\Theta(\boldsymbol{\eta})$ is further given by

$$\boldsymbol{J}_\Theta(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{4.20}$$

and the resulting kinematic equations are formulated as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{4.21}$$

The reduced three-dimensional ship kinetics equations in the horizontal XY-plane (with no Coriolis, wave, ballast, buoyancy or gravitational forces included) are given by

$$(\boldsymbol{M}_{RB} + \boldsymbol{M}_A)\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{currents}} \tag{4.22}$$

where $\boldsymbol{M}_{RB}$ is the rigid body mass matrix, $\boldsymbol{M}_A$ is the hydrodynamic added mass matrix, $\boldsymbol{\tau}_{\text{wind}}$ and $\boldsymbol{\tau}_{\text{currents}}$ are the wind and currents forces, and $\boldsymbol{D}$ is a constant damping matrix. In the example simulations presented in this work, $\boldsymbol{\tau}_{\text{currents}} = 0$ for simplicity, and the wind forces $\boldsymbol{\tau}_{\text{wind}}$ are defined according to [91] as

$$\boldsymbol{\tau}_{\text{wind}} = \begin{bmatrix} X_{\text{wind}} \\ Y_{\text{wind}} \\ N_{\text{wind}} \end{bmatrix} = \begin{bmatrix} -c_x \cos(\Psi_w) A_{Fw} \\ c_y \sin(\Psi_w) A_{Lw} \\ c_n \sin(2\Psi_w) A_{Lw} L_{oa} \end{bmatrix} \frac{1}{2} \rho_a V_w^2 \tag{4.23}$$

where V_w is the relative wind velocity with respect to the ship's velocity, $\Psi_w = \psi - \psi_w - \pi$, ψ_w is the clockwise wind angle relative to the North axis, and the wind coefficients c_* are generated by polynomial approximations of a wind coefficient table for a given ship.

Lastly, the propulsion and steering forces vector is given as

$$\boldsymbol{\tau}(\psi_r) = \begin{bmatrix} u_1 \\ 0 \\ -f_{rudder}(.) u^2 \sin(\psi_r) \end{bmatrix} \tag{4.24}$$

given the definitions from Figure 4.12, where $f_{rudder}(.)$ is a rudder coefficient function, u is the forward surge velocity, and ψ_r is the rudder angle.

See [91] for generalizations to other propulsion and steering configurations.

Part III

Particle swarm optimization

Chapter 5

Dynamic risk-aware path planning

This chapter is based on the publication

[41] **S. Blindheim** and T. A. Johansen, "Particle Swarm Optimization for Dynamic Risk-Aware Path Following for Autonomous Ships," *IFAC-PapersOnLine*, 2022, IFAC CAMS. DOI: https://doi.org/10.1016/j.ifacol.2022.10.411

The method and simulations were developed by S. Blindheim, under the supervision of T. A. Johansen. The first draft was written by S. Blindheim, and was revised by T. A. Johansen.

5.1 Introduction

In this work, a standard path following problem for MASS is formulated, given a pre-planned route in a maritime environment with several isles or islands nearby. This is considered a common MASS control problem of moderate difficulty, with high safety requirements. Moreover, due to increasing complexity of various additional operating modes and emergent behaviors in larger composite systems, the act of formulating, handling and optimizing more complex cost functions accommodating the safety requirements of MASS should be advanced accordingly. Thus, this work presents an approach which allows future works to mix and incorporate discrete and non-smooth cost objectives together with smooth penalties or optimization variables to be provided to the solver, which may make ANS more risk-aware during online guidance or control in uncertain or changing environments.

5.1.1 Literature review

Path planning and obstacle avoidance for MASS remains a challenging problem [71], [72]. There exists a wide range of methods for long-distance (global) autonomous path planning such as the A*-algorithm [65], [67], Voronoi diagrams [23] or hybrid trajectory planning [34], [117]. Moreover, research on adaptive or dynamic path planning for ASV or MASS including environmental disturbances (i.e. wind, waves and currents) have accelerated in recent years, such as energy optimization and path smoothing with respect to sea currents [64], [118], mixed-integer linear programming [73], A* and repulsive vector fields [62], harmonic potential fields [60] for obstacle avoidance and increased safety [66], as well as grounding safety domains [28] and COLREGs-compliant collision avoidance for safe autonomous navigation [76], [119].

However, the notion of *safe navigation* noted in the literature is largely dependent on the definition of inherent system *risks*, and is still a somewhat ambiguous concept with respect to quantitative path planning and optimal control in an environment with grounding hazards. It is suggested that risk perception should largely be based on system uncertainty or lack of knowledge, rather than probabilities [2]. Thus, several recent works utilize risk-based or *risk-aware* approaches for autonomous navigation, such as risk-based MPC for anti-grounding [38], rapidly-exploring random trees (RRT) [120] for grounding-aware collision avoidance, and verification of COLREGs-compliant vessel behavior for collision avoidance [33], [121]. There is however much potential left to be explored within the domain of safe navigation, with respect to the utilization of risk within (dynamic) path planning and optimal control.

Significant research efforts on PSO for safe autonomous maritime navigation has emerged in recent years. In this work, however, the scope of the risk-aware PSO approach is limited to grounding or allision avoidance only, for the purpose of a proof-of-concept study. Nevertheless, it is argued that the approach discussed with respect to risk-based anti-grounding in general may be extended to additionally consider collision avoidance of dynamic obstacles (e.g. nearby vessels in motion) in future research.

PSO has since its initial introduction in 1995 loosely based on the social flock behavior of birds, evolved into a wide array of algorithm variations and methodologies [14]. Due to the computational nature of the PSO method, the cost or fitness function provided to the algorithm may in general take any form. This is most notably in contrast to the gradient-based MPC method which – while computationally fast for feasible problems – requires a sufficiently smooth cost function to achieve satisfactory convergence rates and robust performance. Thus, non-smooth and mixed-integer cost or fitness functions may be utilized in the PSO algorithms, enabling a larger and more complex range of optimization objectives both related to resource consumption and formulation of risk or safety concerns.

Within this particular scope, similar recent works on standard or extended PSO methods for robot path planning or MASS navigation with respect to static obstacles include the following:

a) Adaptive PSO for distance-based path planning toward an end target while avoiding static obstacles [15]

b) PSO path planning using distances between the start and target location as well as intermediate waypoints, including static obstacle avoidance [16]

c) Energy-efficient PSO path planning in which the energy consumption with respect to environmental forces is weighted against the path length [17]

d) PSO combined with the sine cosine algorithm for safe MASS navigation using a cost function which includes the terms path length, grounding risk, bathymetry (depths), and path smoothness [18]

e) Two-part global path planning and multi-objective path following using a cost function also including path length, path smoothness and grounding risk, as well as economic cost and safety risk [19]

5.1.2 Novel contribution

This chapter presents a dynamic waypoint planning algorithm for path following based on PSO, using a fitness function with risk-related objectives. The approach may be summarized as follows. With heavy focus on modularity, simplicity and verifiability within each control module in the hierarchy of a larger ANS, autonomous navigation is presumed split into several independent algorithms or control loops. Hence, it is assumed that a coarse (global) pre-planned path or route is already defined and provided to the algorithm. However, no assumptions related to the efficiency, safety or optimality of the provided route is made.

Moreover, the output of the method is a new risk-aware sequence of waypoints, distributed along the pre-planned path (in this work distinctly denoted as the original *route vertices*) defined by the provided route of consecutive line segments. The distribution of the resulting waypoints is optimized with respect to a fitness function including explicit risk-based terms. These waypoints are subsequently used by a standard guidance algorithm, i.e. path following using a LoS PID controller. This PSO-based structure may allow for both computation parallelization for performance increases, and more complex and dynamic risk-aware behavior utilizing discrete parameters or measurements of e.g. weather conditions or similar environment variables such as non-smooth operational modes, compared to solvers which require continuous or differentiable cost functions.

In order to further demonstrate the novel contribution of this work with respect to the current literature, the method is compared to those of the listed similar works: Though based on the same principle, the simple PSO path planning approaches of **a)**, **b)** and **c)** do not consider risks nearby grounding obstacles, which may lead to unacceptable (i.e. dangerous) solutions with respect to safe navigation. In contrast, the approaches of **d)** and **e)** consider aspects of risk and safety through the use of linear distance weighting, with respect to static obstacles as well as bathymetry data in the PSO fitness function. However, several improvements may be made to the structure of both the fitness function and the utilization of ENC data, which will be presented and discussed throughout this work. Thus, the main goal of this work is to introduce a problem formulation which improves upon the approaches presented in [18] and [19], for increased performance and safety within MASS navigation.

5.2 Method

Particle swarm optimization is an iterative computational method based on weighted individual and social behavior among a swarm (collection) of candidate solution particles, given a set of defined learning parameters. The two main groups of PSO alternatives consist of variations using a *local* neighborhood (local best), and those using a *global* neighborhood (global best). Though there is no conclusive research declaring one PSO algorithm superior to the other for constrained problems [122], the global approach is chosen in this work due to the problem structure of the presented case study (see Section 5.5.2).

5.2.1 Algorithm initialization

The variables of the PSO algorithm are denoted as follows:

- ρ = the ship route (path) to be followed, i.e. a collection of J line segments L_j defined by static vertices (pre-planned route vertices), $j \in \{1, ..., J\}$.
- K = the number of new waypoints along each path segment $L_j \in \rho$, which partitions each line segment into K line subsegments l_{jk}, $k \in \{1, ..., K\}$.
- w_{jk} = a dynamic waypoint to be optimized, corresponding to each line (sub)segment l_{jk}.
- $\phi_{jk}(w_{jk})$ = a specific objective function for each waypoint w_{jk}, defined $J \cdot K$ times.
- G = a collection of convex grounding obstacle polygons, provided by an ENC module (Section 5.3.1).
- B = the spatial lower (minimum) and upper (maximum) bounds of all particles in the 2D NE plane, in this work equal to the boundaries of the visual environment constructed by the ENC module.
- S = an integer denoting the particle swarm size, i.e. the number of particles contained within each swarm.
- P_k = the set of S particles p for each waypoint, where each p contains a 2D waypoint particle w_p (point), a 2D particle velocity v_p, the local best particle w_b, and the local best cost c_b of w_b, calculated from $\phi_{jk}(w_{jk})$.
- σ = the set of all $J \cdot K$ particle swarms s_w, where each s_w contains all particles P_k, the cost function $\phi_{jk}(w_{jk})$, the global best swarm cost c_g, and the global best 2D swarm point w_g corresponding to each waypoint.
- N = an integer denoting the number of iterations to be repeated by the main loop of the PSO algorithm.
- $\theta = [w, c_1, c_2]$, a vector of PSO hyper-parameters.

Algorithm 5 PSO-Initialization

```
Input: $\rho$, $\phi_{jk}$, $B$, $G$, $K$, $S$
Output: $\sigma$ particle swarms for $J \cdot K$ waypoints
  procedure PSO-INIT($\rho, \phi_{jk}, S, B, G, K$)
      $\sigma \leftarrow \varnothing$ empty set of particle swarms
      for each line segment $L_j \in \rho$ do
          for each route partition $l_{jk} \in L_j$ do
              $\phi_{jk}(w_{jk}) \leftarrow \phi_{jk}$ wrt. $w_{jk\text{-}1}, w_{jk+1}, L_j, G$
              $P_k \leftarrow \varnothing$ empty set of particles for each $w_{jk}$
              for each $s$ between 1 and $S$ do
                  $w_p \leftarrow$ random waypoint particle wrt. $B$
                  $v_p \leftarrow$ random particle velocity wrt. $B$
                  $c_b \leftarrow \infty$ as minimum known local cost
                  $w_b \leftarrow w_p$ as best known local particle
                  $P_k \leftarrow$ add particle $p(w_p, v_p, c_b, w_b)$
              end for
              $c_g \leftarrow \infty$ initial minimum global swarm cost
              $w_g \leftarrow \varnothing$ best global swarm state with $c_g$
              $s_w \leftarrow$ swarm with $(P_k, \phi_{jk}(w_{jk}), c_g, w_g)$
              $\sigma \leftarrow$ add waypoint swarm $s_w$
          end for
      end for
  end procedure
```

Algorithm 5 and the notation of Figure 5.1 outline the procedure for constructing the particle swarms σ for PSO in the context of the case study presented in this chapter. The resulting collection of optimizable particle swarms σ are along with θ and N provided to the general main loop PSO detailed in Algorithm 6 of Section 5.5.

During initialization, any line segment of a ship route ρ is considered. Next, a

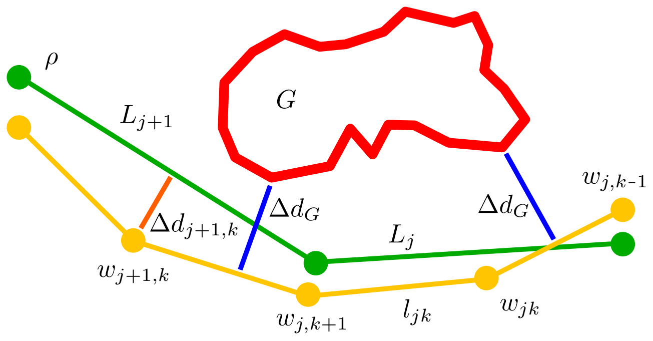


Figure 5.1: An overview of the notation used in Algorithm 5.

distinct objective (fitness) function is defined with respect to the nearby grounding obstacles G and each line segment partition l_{jk}, given as

$$\phi_{jk}(w_{jk}) = \lambda(\Delta d_{k-1} + \Delta d_{k+1}) + \varepsilon \Delta d_{jk} + \mu e^{-\zeta \Delta d_G} \tag{5.1}$$

where Δd_{k-1} and Δd_{k+1} are the absolute distances (minus half of the line segment length l_{jk}) between w_{jk} and the previous and next waypoints $k-1$ and $k+1$, respectively, Δd_{jk} is the distance between w_{jk} and the original route line segment L_j, Δd_G is the minimum distance between the nearest individual polygon in G, and λ, ε, μ and ζ are tuning parameters. Thus, a unique objective function is constructed for each individual waypoint w_{jk} to be optimized along each line segment of the original ship route ρ, as well as the exponential cost of (5.1) for anti-grounding.

An initially empty set to hold the set of particle swarms is defined as σ, and is expanded as follows. For each route partition $j, k \ldots$, S number of candidate particles P_k are constructed, with each particle p containing the attributes w_p, v_p, c_b and w_b. Here, w_p is the initial waypoint location of the particle, v_p is the initial particle velocity, both uniformly randomized within the spatial bounds B defined by the ENC environment. Moreover, c_b is the best known local fitness (i.e. minimum cost) of the particle yet seen and w_b is the corresponding best known local waypoint location w_p which produced the best known fitness, to be updated throughout the PSO iterations.

In addition to the best known local fitness and waypoint seen by each particle p, the best known *global* fitness and waypoint location are denoted as c_g and w_g, respectively. These variables are shared by all candidate particles P_k of each particle swarm s_w, and are also updated during the main loop. Each of the constructed swarms s_w are added to the set of all swarms corresponding to each waypoint w_{jk} to be optimized. Thus, the initial variables of the PSO procedure are defined.

5.2.2 PSO main loop

From Algorithm 5, the collection of $J \cdot K$ particle swarms σ of size S with distinct objective functions ϕ_{jk} and candidate particles P_k, are used to optimize all corresponding waypoints for each line segment partition l_{jk} such that a complete global solution $W_g = \{w_1^g, w_2^g, ..., w_{JK}^g\}$ is returned after N iterations. The general procedure for a standard PSO implementation for waypoint planning is presented as Algorithm 6 in Section 5.5.

The result of the PSO is returned as the set W_g of all best known global states w_g from every swarm s_w in σ, which is the optimal solution after N iterations. See Section 5.5.3 for more detailed discussion related to PSO initialization and tuning with respect to the hyper-parameters of θ.

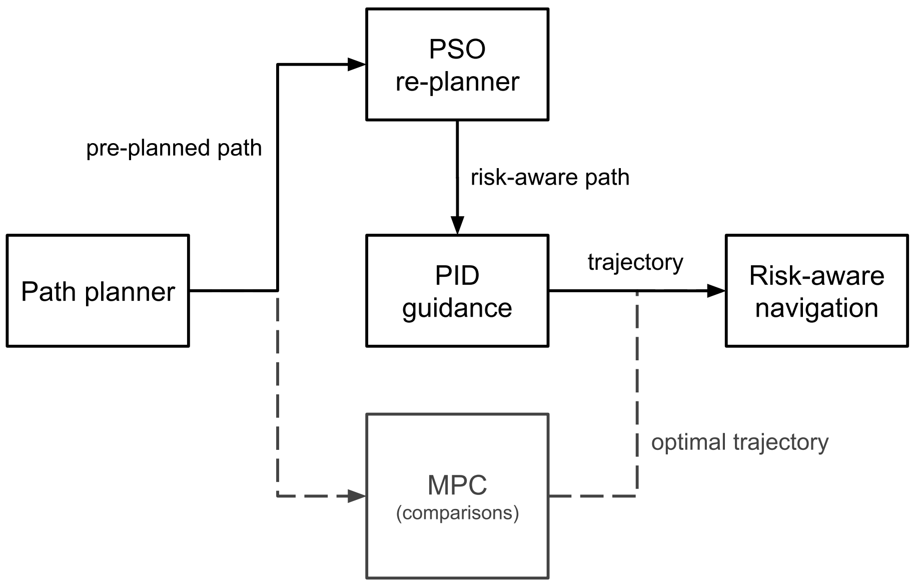


Figure 5.2: Block diagram of the two ANS alternatives.

5.2.3 Guidance and control

The overall ANS structures for risk-aware navigation as discussed in this work are shown in Figure 5.2, and the simulations of Section 5.3 demonstrate and discuss the performance of the system alternatives.

The speed of the ship is held constant, for simplicity. Thus, the control problem at the lower level is course keeping, which is readily controllable using well-established methods such as PID control. Note however that due to the modular structure of this approach, a supervisory ANS may also employ an independent ship speed controller to ensure that the ship is able to sufficiently follow the risk-aware path resulting from the optimized waypoints.

The optimal waypoints of W_g computed by the PSO algorithm are subsequently used as inputs to a simple kinematic line-of-sight PID controller for MASS path following, based on the introductory geometric equations of [93]:

$$\delta(t, w_{jk}) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \dot{e}(t) \tag{5.2}$$

where $e(t) = \chi(t) - \chi_\Theta(w_{jk})$, and $\delta(t, w_{jk})$ is the controlled rudder angle of the ship. $\chi(t)$ is the current course angle of the ship, and $\chi_\Theta(w_{jk}) = arctan(y_w - y_s, x_w - x_s)$ is the current target course calculated from $w_{jk} = P_w(x_w, y_w)$ and the current ship center position $P_s(x_s, y_s)$. K_p, K_i and

K_d are the proportional, integral and derivative tuning coefficients of the controller, tuned such that the ship automatically steers toward the target position w_k along ρ. It may be noted that no assumption in general is made by the PSO method regarding the *selection* of the chosen target waypoint at any given time during simulation and/or operations, with respect to the path following algorithm subsequently applied to the resulting waypoints. Thus, the independent and modular structure assumed within the ANS furthermore promotes the utilization of well-known schemes for verifiable control problems and solutions, such as the ones mentioned in this work.

5.3 Results

Several example simulations of PSO runs produced by Algorithms 1 and 2 implemented in Python are presented in this section, applied to a specific ENC environment located north-west of the Norwegian city of Ålesund, for the purpose of demonstration. The ENC data as well as methods for geometric computations with respect to points, lines and polygons, as well as environment and vessels visualizations are provided by the *SeaCharts* ENC package [37].

5.3.1 ENC environment and visualization

In contrast to the 3D environments used in the similar works of **d)** and **e)** from Section 5.1.1, the ENC environment is in this work structured as a 2D plane for computational efficiency. It is argued that this structure vastly reduces the computational load within the PSO algorithm, as the problem of safe navigation for a *surface* vessel by definition is naturally restricted to a 2D domain. Consequently, there is no practical reason to perform 3D distance calculations on ocean depths deeper than some static safety threshold (with respect to the maximum ship draft and shallows), if the ocean bed gradients or topography is not otherwise considered with respect to e.g. prediction of ocean current dynamics. The computational complexity required for such considerations should not be included within the main loop of the PSO algorithm, and may rather be moved to a separate and independent ANS module for e.g. analysis and/or prediction of environmental forces in future works.

Similarly, it is argued that it is not necessary to differentiate between too shallow waters, the shoreline or other static obstacles with respect to grounding or allision risks. For this purpose, the SeaCharts ENC package provides a significantly more efficient environment in which *any* (polygon) area of insufficient depth for navigation are consolidated into a single set of 2D grounding obstacles without loss of relevant information, constructed in its entirety before the PSO initialization. Thus, the minimum distance Δd_G of (5.1) to any nearby grounding obstacle is readily computed by a simple call of the **distance** function provided through SeaCharts by the dependency library *Shapely* [83] for efficient geometric calculations.

5.3.2 Risk-aware waypoint re-planning

In this section, a visualization of an optimal waypoint distribution produced by the PSO algorithm is presented, as well as a comparison visualization showing the end result trajectories of the PSO algorithm and PID controller versus those of a gradient-based MPC solver. Additional discussions concerning the initial values and convergence rates of the PSO run are presented in Section 5.5.

Figure 5.3 demonstrates a PSO run using the objective function (5.1), i.e. including the risk cost term $\mu \cdot \exp(-\zeta \Delta d_G)$. As noted, the purpose of the risk term is to enable anti-grounding behavior with respect to the set of grounding obstacles G provided by the ENC module. The original ship route path ρ is shown with green disks as the pre-planned vertices, and green straight line segments drawn between them. The optimal re-planned waypoints solution with respect to the grounding obstacles are similarly shown as light yellow disks with straight line segments between them, highlighted by a black border. The grounding obstacles considered in this example are shown in red as two convex polygons surrounding two islands nearby the ship route.

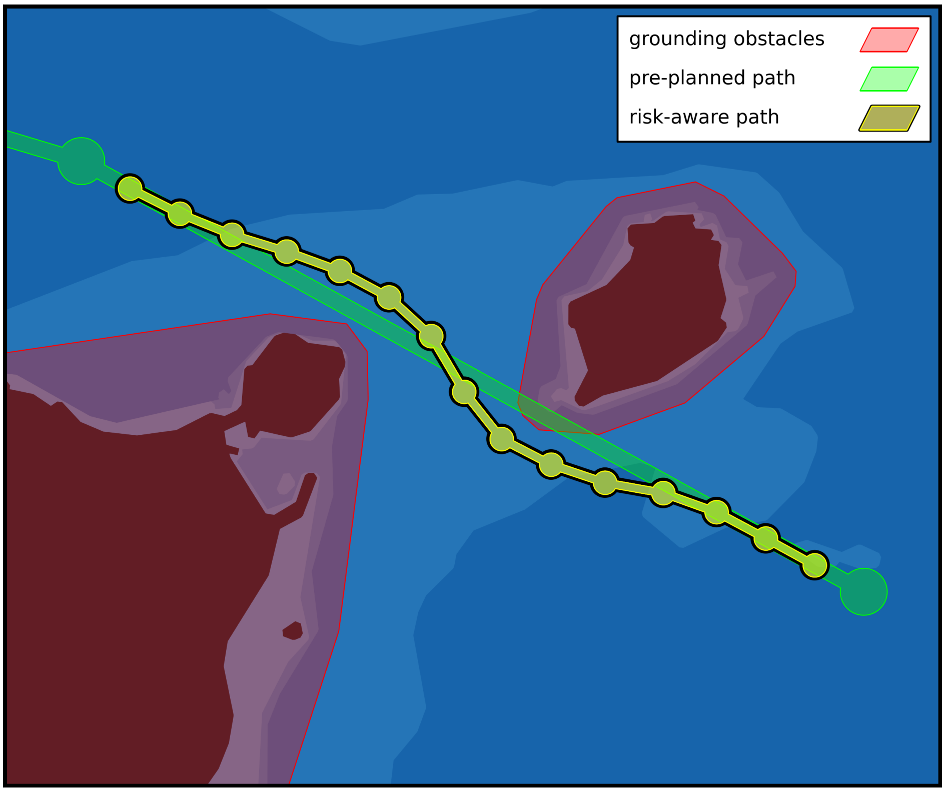


Figure 5.3: Example demonstration of risk-aware waypoint re-planning, in which a risk-based cost term is added to the objective function.

Notice how the weighting of the distances between each waypoint in yellow and the grounding obstacles generates behavior similar to evasive maneuvers, shown by the significant deviations from the original straight line segment. This is considered an appropriate and desired result, ultimately verifying the effect of the risk term.

Next, Figure 5.4 shows a comparison visualization of a yellow trajectory with black borders generated by the path following controller when given the re-planned path shown in Figure 5.3, against two additional ship trajectories shown in white. These are produced by a gradient-based MPC scheme utilizing a similar risk-based cost function for anti-grounding [38]. The two white trajectories are in general generated with the same parameters, with the only difference being the values given to the sensitivity constant ζ of the exponential risk cost term, assessed as edge case demonstrations for sufficiently safe autonomous navigation in this environment.

Moreover, Figure 5.4 shows actual trajectories, i.e. the resulting simulation runs which utilize the LoS PID controller (with constant speed) and a receding horizon MPC for comparison. Thus, in this context, the optimal path generated by the PSO

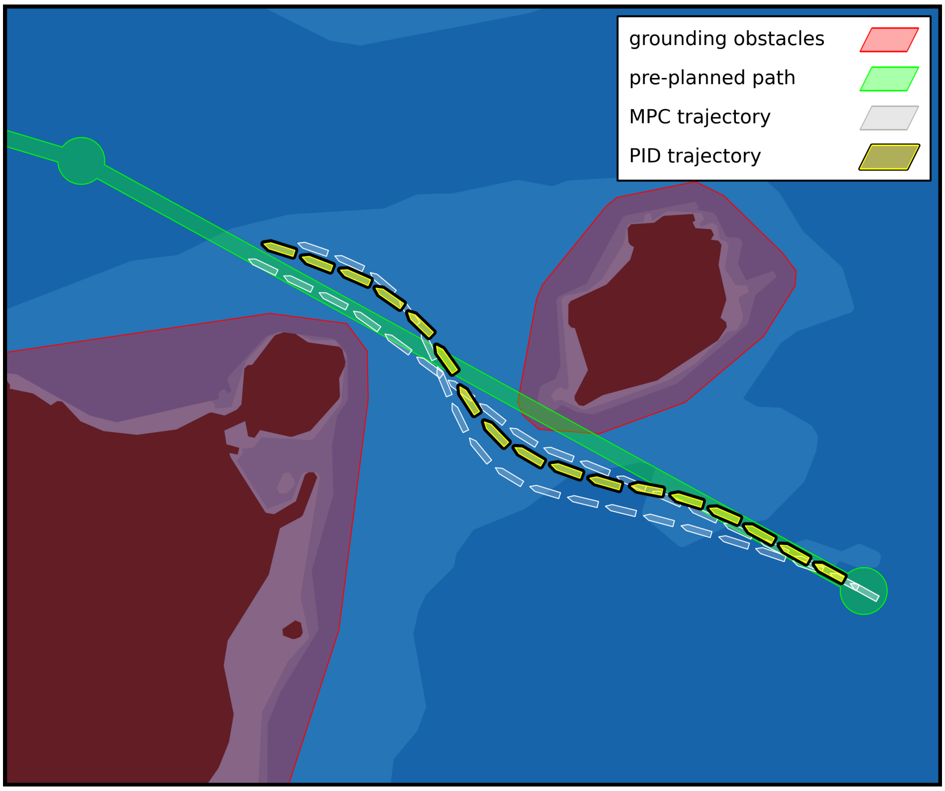


Figure 5.4: A trajectory generated by the risk-aware waypoint planning and PID guidance approach of Figure 5.3, compared with two other trajectories produced by an MPC scheme using a gradient-based solver.

based on the original ship route is provided as the input to the PID controller, which together produce a trajectory output (Figure 5.2).

The MPC, however, calculates an optimal trajectory directly from the original ship route. The first white example demonstration of the MPC trajectory closest to the original pre-planned path is given $\zeta = 0.3$, which generates a trajectory in which the ship only barely avoids the convex polygons in red. The second example is given $\zeta = 0.07$, effectively reducing the rate of decay (weighting) applied to the distance measure between the ship center and the grounding obstacle, which in turn results in higher risk cost evaluations closer to the obstacles. It is apparent that the optimal waypoint placements computed by the PSO method as well as the resulting trajectory produced by the PID guidance controller primarily are located between the two MPC boundary trajectories, providing further confirmation of the validity of the risk-aware PSO approach.

Note, however, that the objective function (5.1) does not contain an explicit term related to path or waypoint smoothness, in contrast to the similar works of **d)** and **e)** of Section 5.1.1. The reason for this is two-fold: The inherent distance weighting between each consecutive waypoint acts as a self-smoothing mechanism, due to the iterative PSO cost updates in which each waypoint are adjusted with respect to its previous and subsequent neighbor. Moreover, it is argued that the significance of path smoothness of an (emergency) evasive maneuver should be negligible compared to the importance of simply avoiding the grounding obstacle in its entirety – particularly if the approach is extended to include collision avoidance of dynamic obstacles such as moving vessels. No distributive priority weighting is thus shared between path smoothness and grounding avoidance. Rather, the parameters of the exponential term should be appropriately tuned with respect to the ship dynamics and environment resolution, such that the trajectory generated by the path following algorithm given the resulting PSO waypoints is sufficiently smooth during normal operations.

5.3.3 Problem parameterization and alternatives

Several additional comments may be made with respect to the choice of parameterization and general performance of the approach compared to available alternatives: The cost function (5.1) essentially formulates the problem to be solved, i.e. safe navigation along a path. Next, the problem solution is parameterized as $K = 20$ two-dimensional waypoints, resulting in 40 parameters to be optimized along the original route. In this work, the structure of these waypoints is also utilized by the PSO solver itself, i.e. 20 independent particles of 2 dimensions which are weighted with individual cost functions with respect to its neighbors. However, the parameterization used by the PSO may also have been structured as a single solution of a *waypoint collection* with 40 dimensions. In this case, the resulting particle swarms would consist of complete path solutions, and may be subject to be solved by other methods such as genetic algorithms. It is however argued that this structure is less intuitive (see Section 5.5), and the utilization of such parameterizations and e.g. other evolutionary algorithms (EA) is left for future works.

The PSO algorithm is not guaranteed to find the optimal solution, and is indeed subject to potential convergence to local minima, similar to nonlinear MPC and artificial potential fields. However, the chosen PSO structure alleviates this issue through a diversified initialization of particles that should cover the expected solution space. The "best known global particle" is shared across all particles in the swarm, which may help steer a larger selection of candidate solutions toward the true global minimum if it is found.

Moreover, the initialization of the PSO particles (the initial positions of the waypoints) has a critical impact on the optimal solution generated by both the PSO and the MPC solvers. However, the consideration of alternative paths e.g. around the island due to different initial conditions are considered outside the scope of this work. See Section 5.5 for more comments on aspects of the PSO performance.

5.4 Conclusion

In this work, a method for dynamic risk-aware waypoint planning based on PSO was presented. The resulting sequence of optimal waypoints with respect to grounding obstacles as computed by the PSO algorithm, was subsequently used as inputs to a simple PID course controller for path following. This two-part modular structure is employed in order to facilitate a standardized and interchangeable hierarchy of potentially parallel, verifiable and robust controllers within a larger supervisory ANS for safe MASS navigation. Moreover, the performance of the PSO path replanner was considered well-suited for the purpose of anti-grounding, compared to that of an analogous implementation using a gradient-based MPC solver including the same exponential risk term from previous works. Ultimately, the approach may be regarded as an adequate alternative to gradient-based controllers for safe MASS navigation, when utilizing more complex (i.e. mixed-integer or non-smooth) cost functions for risk-aware control.

5.5 Appendix: The PSO Algorithm

Algorithm 6 presents the general procedure of a standard PSO, and Algorithm 7 presents the standard subroutine for updating the velocities of each swarm particle within the main loop.

For each iteration $i = 1, ..., N$, the fitness (i.e. objective cost) of every particle $p \in P_k$ within each particle swarm $s_w \in \sigma$ is evaluated with respect to its distinct objective function ϕ_{jk}, denoted as c_p. If c_p is less than the current best known local cost c_b, the new cost replaces it as the best known local cost for that particle. Moreover, if c_p is less than the current best known *global* cost c_g, the new cost value also replaces the global best known cost for any particle across all swarms in σ. Both the local and global best known waypoints w_b and w_g are thus dynamically updated during runtime.

Algorithm 6 PSO-MainLoop

```
Input: $\sigma$, $\theta$, $N$
Output: best global solution $W_g$ after $N$ iterations
  procedure PSO-Main($\sigma, \theta, N$)
    while iteration $< N$ do
      for each waypoint swarm $s_w \in \sigma$ do
        $P_k \leftarrow$ current particles of swarm $s_w$
        $\phi_{jk} \leftarrow$ specific cost function of swarm $s_w$
        for each waypoint particle $p \in P_k$ do
          $w_p \leftarrow$ current waypoint location
          $c_p \leftarrow$ evaluate local cost $\phi_{jk}(w_p)$
          if $c_p < c_b$ of $p$ then
            $c_b \leftarrow c_p$ new best local fitness
            $w_b \leftarrow w_p$ new best local waypoint
            $p \leftarrow$ update particle with $c_b, w_b$
          end if
          if $c_p < c_g$ of $s_w$ then
            $c_g \leftarrow c_p$ new best global fitness
            $w_g \leftarrow w_p$ new best global waypoint
            $s_w \leftarrow$ update swarm with $c_g, w_g$
          end if
        end for
        for each particle $p \in P_k$ do
          $v_p \leftarrow$ update $ParticleVelocity(p, w_g, \theta)$
          $w_p \leftarrow$ add particle velocity $v_p$ to $w_p$
          $p \leftarrow$ update particle with $w_p, v_p$
        end for
      end for
    end while
    $W_g \leftarrow$ complete solution of all $K$ global bests $w_g$
  end procedure
```

After each objective function evaluation, the current velocity and state of each particle p is updated. The velocity is computed through the *ParticleVelocity* procedure presented in Algorithm 7, in which the hyper parameters w, c_1 and c_2 are used as weights to adjust the direction and magnitude of the new particle velocity, with respect to the previous value. The difference between the local and global best known state w_b and w_g is weighted with the *cognitive* constant c_1 and the *social* constant c_2 respectively, as well as two uniformly random variables r_1 and r_2 between 0 and 1, with both terms added to the inertia constant w multiplied by the previous particle velocity v_p to produce the updated velocity v_{p+1}. The updated state of the particle is simply the sum of the previous state w_p and the calculated particle velocity v_p. The global best waypoint particles W_g are lastly returned as the PSO solution.

Algorithm 7 ParticleVelocity

```
Input: $p$, $w_g$, $\theta$
Output: updated weighted particle velocity $v_{p+1}$
  procedure PV($p, w_g, \theta$)
      $w \leftarrow$ particle inertia weight $\in \theta$
      $c_1 \leftarrow$ cognitive weight $\in \theta$
      $c_2 \leftarrow$ social weight $\in \theta$
      $w_p \leftarrow$ state vector of particle $p$
      $w_b \leftarrow$ local best state vector of particle $p$
      $r_1 \leftarrow$ first uniformly random value $\in [0, 1]$
      $r_2 \leftarrow$ second uniformly random value $\in [0, 1]$
      $\Delta b \leftarrow w_b - w_p$ local best difference
      $\Delta g \leftarrow w_g - w_p$ global best difference
      $v_{p+1} \leftarrow wv_p + c_1 r_1 \Delta b + c_2 r_2 \Delta g$
  end procedure
```

5.5.1 Choice of parameterization and algorithm

Several alternative methods for risk-aware path adjustment or re-planning were considered. Though the well-established A* algorithm and RRT* are suitable for fast optimal path planning in large search spaces from a start point toward an end goal, they are considered less appropriate for path adjustment or re-planning in the case where an optimal or near-optimal pre-planned path is given.

The structure and principles of other EA compared to that of PSO are more similar, and are particularly suited for combinatorial problems. However, methods such as e.g. differential evolution and genetic algorithms are generally dependent on a valid and efficient function for recombinations and mutations between candidate solutions in order to explore the search space. Consequently, it is considered more appropriate to utilize the inherent concepts of PSO particle velocities and fitness for this particular application.

5.5.2 PSO performance and convergence

The convergence rate of the computed PSO solutions may be evaluated with respect to the resulting global best solution, in order to further assess the performance of the PSO method. The initial *global* best known waypoints are in this work given completely random start values for demonstration of robustness. It may also be noted that some of these values can be initialized *onto* obstacles, as the cost rises even higher inside obstacles due to negative distances within the exterior boundary of a polygon. Though the cost is non-negligible if grounding polygons are nearby, the increased cost of the risk term is relatively insignificant with respect to the initial best global costs present in the system during PSO initialization. Nevertheless, the algorithm is considered feasible, with respect to computational time during real-time navigation: The PSO optimization for Figures 5.3 and 5.4 with $N = 100$ and $S = 300$ was applied to the considered segment of length 2.3 km, and was computed within 13 seconds – using the random waypoint initialization approach for the purpose of demonstration.

Though the nominal route is assumed optimal or near-optimal, grounding risks are implicitly always present in an environment with inherent uncertainties, such as e.g. weather conditions, tides, winds or currents, and the potential for unexpected events such as e.g. the sudden presence of unmapped grounding hazards such as moored vessels, and failures or faults onboard the vessel. Thus, the main application of the proposed system is to continuously adapt the nominal route with small alterations, in order to adhere to changes that were introduced into the environment after the route was originally planned. Rigorous handling of uncertainty aspects such as e.g. reduced safety margins with respect to estimated time to grounding due to winds or currents will be addressed in future works.

5.5.3 PSO initialization and hyper-parameters

The random initialization of particle candidates of each particle swarm (recall that one swarm corresponds to one waypoint) does in general favor *particle exploration.* However, the PSO consequently has a large initial cost due to being located far away from the low-cost regions around the pre-planned path (as defined through the defined cost function), and the initial values of two subsequent runs will by construction always be random, potentially leading to unstable results between runs and higher convergence rates. The monotonic nature of the cost gradient is nevertheless easily verified during runtime through visual inspection of e.g. a cost graph, and is a direct consequence of the fact that new global best particles by definition always have a lower cost or better fitness compared to the last.

Another approach is to give the initial global best particles e.g. the exact optimal placements along the original straight path segment of the ship route (analogous to the concept of "warm start" of e.g. gradient-based solvers). Assuming no grounding obstacles are present, the initial cost would in this case be vastly reduced, and the convergence rate accordingly improved. Moreover, it is argued that particle exploration is not the primary objective of the PSO re-planner, as the main function of the risk-aware re-planned path is merely to lightly adjust a presumed optimal or near-optimal pre-planned path due to online risk-related influences such as changing weather conditions, nearby vessels, unforeseen events or disturbances. Due to this obvious increase in performance with the above scope in mind, this approach is recommended as the favored initialization method for future works.

Ultimately, one may achieve satisfactory performance for a large selection of optimization problems by careful tweaking of the PSO hyper-parameters of Algorithm 7. Candidate particles are during optimization thus allowed to probe previously not yet evaluated fitness values of unvisited states through semi-random exploration, while simultaneously approaching both previously seen local and global minima. In the simulation of Figure 5.3, $w = 0.75$, $c_1 = 1.0$, and $c_2 = 2.0$.

Chapter 6

Autonomous planning and machinery management

This chapter is based on the publication

[42] **S. Blindheim**, B. Rokseth, and T. A. Johansen, “Autonomous Machinery Management for Supervisory Risk Control Using Particle Swarm Optimization,” *Journal of Marine Science and Engineering*, vol. 11, no. 2, p. 327, 2023. DOI: https://doi.org/10.3390/jmse11020327

The method and simulations were developed by S. Blindheim in collaboration with B. Rokseth, under the supervision of T. A. Johansen. The first draft was written by S. Blindheim, and was revised by B. Rokseth and T. A. Johansen.

6.1 Introduction

An important prerequisite for the realization of autonomous ships is that safe and reliable performance of guidance and navigation tasks is ensured. One possible way of achieving this is to develop risk-based guidance and navigation control systems that uses risk models as part of their decision-making process. Collision avoidance and obstacle avoidance for autonomous guidance and navigation is a topic that recently has received much attention, see for example [123]–[134]. However, reliable obstacle and collision avoidance is not the only concern that should be addressed. Another important aspect of the guidance and navigation task is the grounding risk. Grounding accidents are commonly classified into powered grounding and drifting grounding. Groundings where the ship drifts aground as a consequence of machinery failures are classified as drifting groundings, while groundings that occur due to navigational errors, are referred to as powered groundings [135]. The powered groundings can be seen as part of the obstacle avoidance problem, where the seabed or shore is considered as an obstacle, [136]–[141], while the drifting groundings are not covered by the literature with respect to obstacle avoidance.

Drifting grounding is a problem that, in principle, could be addressed by avoiding to sail close to the shore when there is onshore wind. This increases the probability that grounding (following a mechanical or electrical failure) can be prevented by dropping anchor or restoring sufficient propulsion and steering capability. There is, however, limited work on autonomous ships where the guidance and navigation problem is studied in terms of drifting grounding avoidance or risk reduction. [38] presents a decision-making algorithm to plan suitable trajectories (minimized grounding risk) in situations where the ship unexpectedly experiences reduced maneuverability due to e.g. thruster faults. [36] propose a control system for automatically selecting the most appropriate operating mode for a hybrid machinery system in order to minimize the drifting grounding risk and fuel consumption.

In this work, a supervisory control algorithm is proposed, which integrates the machinery system mode selection problem with the guidance and navigation problem, based on data of ENC from [37]. The main reason why it is of interest to integrate these two control problems is that the controlled states (the ship trajectory and machinery system mode) are important influencing factors for the drifting grounding risk. In general, benefits may be achieved by considering several distinct control problems that share the common feature that they somehow affect the risk associated to the same loss scenario, as this potentially results in an extended or improved set of possible actions for reducing the risk. Thus, the proposed hypothesis is that the potential for reducing the grounding risk at a reasonable operational cost will improve if the two control problems are merged into a single optimization problem weighting both aspects simultaneously, compared to only optimizing for purely spatial and distance-based grounding risks in previous works [41] (see Section 6.3.5). It is argued that this structure may increase the number of ways in which the control algorithm can make safe decisions, and thus a reasonably safe decision may be computed at a lower operational cost, such as fuel consumption and expected costs based on grounding probabilities.

While the proposed control algorithm in [36] successfully identifies the optimal operating modes, the choice of MSO mode has a limited impact on both the grounding risk and the fuel consumption. Here, it is instead proposed to model the grounding risk and explicitly address the trade-off between fuel consumption and grounding risk in an optimization framework.

A more reliable operational mode is generally more costly in terms of fuel consumption. When the ship is sailing such that loss of the propulsion power may cause the ship to drift aground in a short amount of time (i.e. close to land while the environmental forces acting on the ship is directed toward the shore), a reliable mode of operation is considerably safer. In this context, a grounding event occurs if the time it takes to drift aground is shorter than the time it takes to recover propulsion capabilities. An alternative way of achieving equal levels of safety is to change the route e.g. such that the ship is sailing further away from grounding obstacles (i.e. the shore or shallow waters) or such that the environmental forces acting on the ship is not directed toward grounding obstacles, or there is more time to recover from a machinery fault.

6.2 Materials and methods

6.2.1 Problem definition and approach

In the proposed framework, autonomous ships are following routes defined by a sequence of waypoints. Each waypoint (WP) is described with longitude and latitude coordinates. As illustrated in Figure 6.1, it is assumed that a separate global planning process has prepared a pre-planned route for the entire voyage. This global planning is normally performed onshore when the voyage is planned. Next, a tentative pre-planned route is generated and optimized or adapted online. Thus, the ship re-evaluates the part of the tentative route that falls within a given prediction horizon, while sailing.

The re-planning process consists of two tasks: The first one, referred to as "Level 1 route planning" in Figure 6.1, is carried out to generate a number of proposed feasible routes. This is achieved by first using the most recent sensor data and ENC data to check if the part of the tentative route that falls within the prediction horizon passes over objects or too shallow waters that were not identified as obstacles in the planning stage, e.g. if a fish farm has appeared that was not present on the map during voyage planning or the water depth is different due to tides. If not, the tentative route is considered a feasible route. If, on the other hand, there is an obstacle in the way, two alternative routes (one on each side of the obstacle) will be generated (see Section 6.6). In principle, 2^N options exists if there are N obstacles being considered. It may be noted that this level of re-planning or online avoidance maneuvers may also be applied directly to avoid areas with opposite or dense maritime traffic, nearby vessels or other dynamic obstacles in future works. This could build on preliminary results that combines anti-grounding and anti-collision while considering the traffic rules at sea (COLREGs), albeit without considering MSO and failure modes, as presented in [29].

The second task of the tentative route re-evaluation is to optimize the proposed route alternatives (see "Level 2 route planning" in Figure 6.1). This is achieved by first generating a new set of intermediate waypoints, essentially increasing the resolution and smoothness of the original tentative route alternative. Next, the new waypoints are adjusted with respect to resource consumption and grounding risk, in which the latitude, longitude and MSO mode for all waypoints of each proposed feasible route are the decision variables available for optimization (Figure 6.1). Specifically, each adjustment is considered in terms of the total resulting cost, which is a function of the fuel consumption, the grounding risk (using a risk model), and the deviation from the estimated time of arrival (ETA) through calculations of the measured states of the sailing process.

The cost related to each proposed control output (adjustment) is estimated through the simulated states from a ship simulator and the probability of grounding from a risk model, and the selected control output of the Level 2 route planner is applied to a sailing process controller - which in turn yields the next measured and/or initial ship states for the subsequent simulation (see the respective modules in

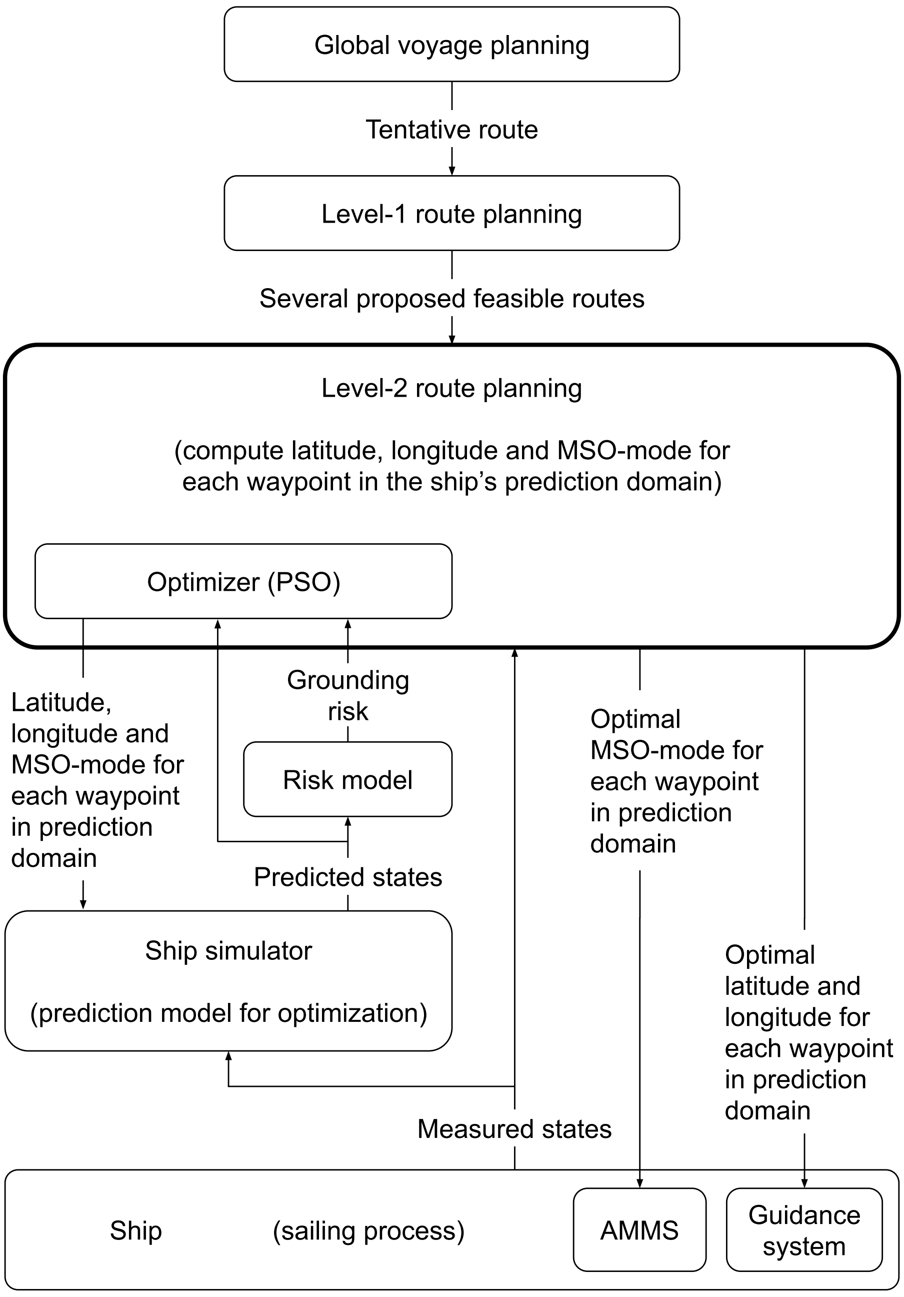

Global voyage planning
Tentative route
Level-1 route planning
Several proposed feasible routes
Level-2 route planning
(compute latitude, longitude and MSO-mode for each waypoint in the ship's prediction domain)
Optimizer (PSO)
Latitude, longitude and MSO-mode for each waypoint in prediction domain
Grounding risk
Risk model
Predicted states
Ship simulator
(prediction model for optimization)
Optimal MSO-mode for each waypoint in prediction domain
Optimal latitude and longitude for each waypoint in prediction domain
Measured states
Ship
(sailing process)
AMMS
Guidance system

Figure 6.1: An overview of the overall control strategy and structure of the implemented system.

Figure 6.1). Thus, the optimization loop of the Level-2 route planning process is to utilize current measured states from the sailing process as inputs, predict future states using a ship simulator, use these predicted states to estimate grounding risks using a risk model, and weigh these risks against other economic or environmental factors using a cost function to produce an optimized series of waypoints. In this work, an optimization algorithm based on PSO is used to search for the set of control outputs that results in the lowest overall cost across the receding horizon. PSO was proven to be effective for solving a simplified version of this problem in [41], and was also chosen for solving similar problems related to unmanned aircraft [142], [143]. Note that in this proof of concept it is not claimed that PSO is the best method to solve the optimization problem, and it is recommended to study alternative methods such as genetic algorithms in future work.

Figure 6.2 illustrates the relationship between various factors affecting the cost, as well as the terms that the cost function is composed of. First, the fundamental factors such as ship speed (and available top speed), environmental forces and infrastructure along the shore affects the ETA, grounding risk and fuel consumption as shown in the figure. The position (longitude and latitude) of each waypoint affects the target ETA, i.e. if a WP is moved such that the distance the ship has to sail to reach the target is changed, the ETA and the fuel consumption may change accordingly. Moreover, the grounding risk may change if a waypoint is moved such

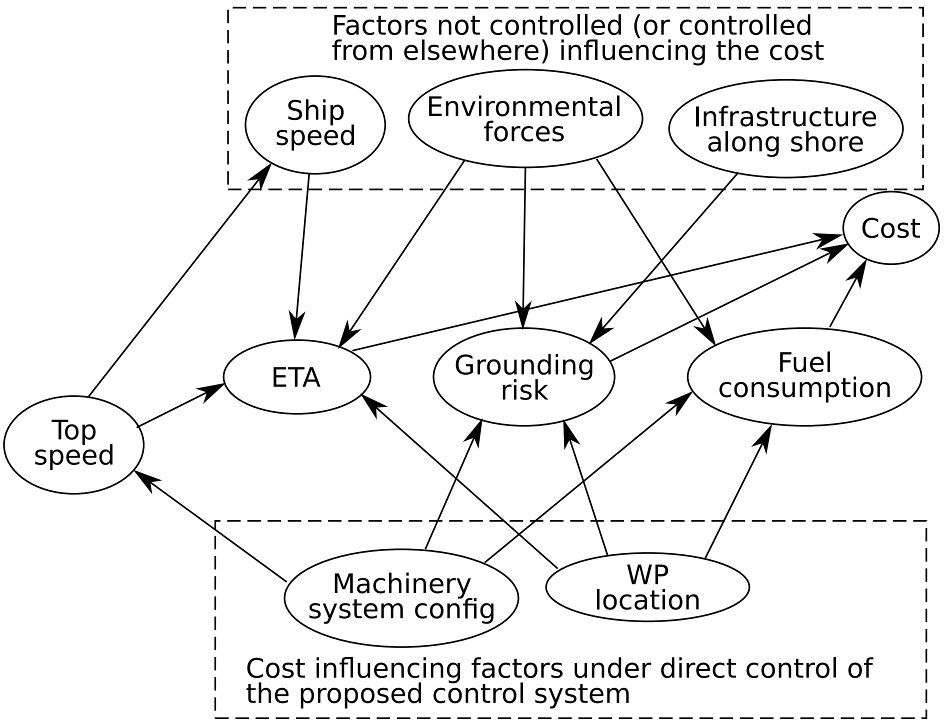

Factors not controlled (or controlled from elsewhere) influencing the cost
Ship speed
Environmental forces
Infrastructure along shore
Cost
ETA
Grounding risk
Fuel consumption
Top speed
Machinery system config
WP location
Cost influencing factors under direct control of the proposed control system

Figure 6.2: An overview of the cost influencing factors as structured in this work.

that the distance between the ship and obstacles is changed, or the duration of exposure to disturbances with respect to e.g. a downwind obstacle changes. Interestingly, the top speed of the ship additionally indirectly affects the ETA. An example may be that longer exposure to increased risks near obstacles of a narrow strait due to a lower available top speed compared to a different ship, can alter the resulting optimal waypoint distribution along a route alternative. The MSO mode (Machinery system config in Figure 6.2) directly affects the fuel consumption, possible top speed, and additionally the grounding risk because the MSO modes are different in terms of robustness against drifting grounding. Ultimately, the ETA, the grounding risk, and the fuel consumption affect the cost to be minimized.

Level 1 route planning

The level 1 planning algorithm from previous works [37] is presented and summarized in Section 6.6. It is used to generate pair-wise alternative routes on each side of static grounding obstacles, if any such obstacle crosses the global pre-planned voyage path. Note, however, that the computed paths are only concerned with purely spatial avoidance of any obstacle boundary in the horizontal plane, and is subsequently evaluated, adjusted and optimized with respect to resource consumption and risks by the level 2 route planner.

Level 2 route planning

Due to the uncontrolled factors shown in Figure 6.2, an estimate of expected costs has to be formulated and computed during sailing, based on probabilities and available online (and offline) data. The cost estimate is from Figure 6.2 given by an ETA, a fuel consumption estimation, and an estimated grounding risk r_G, and ultimately serves as the optimization variable for the level 2 route planner. The computation processes for these estimated terms are presented in the following sections.

Table 6.1: Overview of the system variables used in this work

$\boldsymbol{f}$	propulsion/steering forces	$\boldsymbol{C}$	Coriolis matrix	δ	rudder angle
k	proportional coefficient	$\boldsymbol{D}$	damping matrix	$\boldsymbol{\eta}$	ship pose
$\boldsymbol{m}$	steering moments	E	easting	$\boldsymbol{\nu}$	ship velocity
$\boldsymbol{p}$	ship position	F	propulsion force	$\boldsymbol{\omega}$	rotational velocity
r	yaw rate	$\boldsymbol{J}$	Jacobian matrix	ψ	ship heading
u	surge velocity	$\boldsymbol{M}$	mass matrix	$\boldsymbol{\tau}$	forces & moments
v	sway velocity	N	northing	$\boldsymbol{\Theta}$	ship orientation
		P	power	ζ	tuning variable
		$\boldsymbol{R}$	rotational matrix		

6.2.2 Modeling

The ship simulator

A three degrees of freedom (DoF) ship model is proposed, for the purpose of state predictions within the optimization algorithm. The ship's position $\boldsymbol{p}^n_{b/n}$ is described by N (north) and E (east) coordinates, and ψ is the ship's heading. As seen in (6.1), its time derivative is a rotation transformation of the ship´s forward (surge) velocity, sideways (sway) velocity and yaw rate are given as u, v and r, respectively. Based on [93], the ship dynamics can be modeled by the notation as presented in Table 6.1, and the following relationship definitions and equations:

$$\boldsymbol{p}^n_{b/n} = \begin{bmatrix} N \\ E \end{bmatrix} \qquad \boldsymbol{v}^b_{b/n} = \begin{bmatrix} u \\ v \end{bmatrix} \qquad \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix}$$

$$\boldsymbol{\Theta}_{nb} = [\psi] \qquad \boldsymbol{\omega}^b_{b/n} = [r]$$

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} \qquad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \qquad \boldsymbol{J}_\Theta(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix}$$

$$\dot{\boldsymbol{p}}^n_{b/n} = \boldsymbol{R}^n_b(\boldsymbol{\Theta}_{nb})\boldsymbol{v}^b_{b/n} \qquad \dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{\omega}^b_{b/n} \qquad \dot{\boldsymbol{\eta}} = \boldsymbol{J}_\Theta(\boldsymbol{\eta})\boldsymbol{\nu} \tag{6.1}$$

$$\boldsymbol{f}^b_b = \begin{bmatrix} F \\ -k_{sway}\ \delta\ u \end{bmatrix} \qquad \boldsymbol{m}^b_b = [-k_{yaw}\ \delta\ u] \qquad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix}$$

$$\dot{F} = -\frac{k}{\zeta}F + \frac{1}{\zeta}P \qquad k = \frac{P_{\max}}{F_{\max}} \tag{6.2}$$

where (6.2) represents the propulsion force dynamics. Here, ζ is a tuning parameter, F and $F_{\max}$ are the input and maximum propulsion forces, and P and $P_{\max}$ are the input and maximum power available for propulsion, respectively. This model may be found online at GitHub [144].

Equation (6.3) relates the inertial force given by the 3 by 3 mass matrix $\boldsymbol{M}$ (including hydrodynamic added mass) times the acceleration in surge, sway and yaw, with the other forces acting on the vessel. The Coriolis and centripetal forces $\boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}$ and $\boldsymbol{C}_A(\boldsymbol{\nu}_R)\boldsymbol{\nu}_R$ as described in [145] are included, where $\boldsymbol{\nu}_R$ is the ship's velocity vector relative to a water particle floating with the current, and linear and nonlinear damping terms are described by $\boldsymbol{D}_L\boldsymbol{\nu}_R$, and $\boldsymbol{D}_NL(\boldsymbol{\nu}_R)\boldsymbol{\nu}_R$. This gives

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{C}_A(\boldsymbol{\nu}_R)\boldsymbol{\nu}_R - \boldsymbol{D}_L\boldsymbol{\nu}_R - \boldsymbol{D}_{NL}(\boldsymbol{\nu}_R)\boldsymbol{\nu}_R + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau} \tag{6.3}$$

where τ_{wind} and τ represents the wind forces and control forces acting on the ship.

In the presented algorithm, environmental forces are considered as input, and it is not within the scope of this work to provide algorithms for weather or current forecasting given the terrain and bathymetry. In general, the optimization should include margins when defining the cost function and constraints in order to account for the uncertainty in these forecasts, given the mentioned challenges. In this way, the control system will make robust decisions by taking into account such uncertainty bounds. The example in this work utilizes simple models of the environmental forces which do not include considerations of being close to obstacles or varying depths, for simplicity and clarity when demonstrating the effects of the novel contribution as a proof of concept. The extension to use more complex and accuracy models (if available) is straightforward since the proposed framework is flexible with respect to the format of the provided information (i.e. no requirements have been made for deterministic operations, smoothness or continuity).

The risk model

The purpose of the risk model is to estimate the grounding risk R_G^k given in (6.4) for each simulated future scenario k. In this work, the drifting grounding risk model presented in [36] is defined as follows:

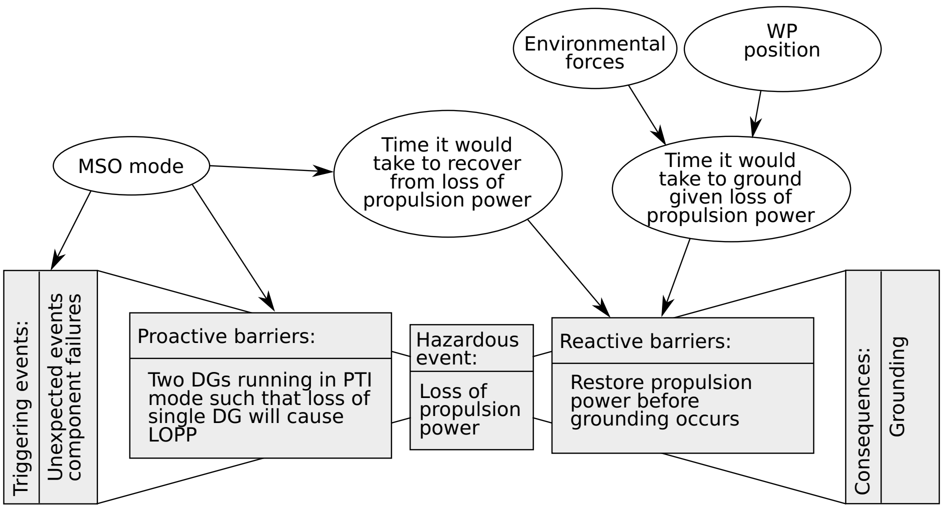
$$R_G^k = P(G_k) \cdot C_G \tag{6.4}$$

It is used as a measure of the grounding risk, where $P(G_k)$ denotes the probability of experiencing a grounding event G_k during a future prediction horizon in scenario k (if the scenario k were to be executed), and C_G is the cost per grounding.

This model uses ENC data and the position, heading, velocity and yaw rate of a ship as well as nearby grounding obstacles at some time instance t to calculate the probability that a grounding scenario may occur, and can be used as an online real-time risk model for a ship. Note that the grounding obstacles (hazards) are constructed according to [37], in which the desired minimum depth may be selected by the operator. Thus, one may include considerations such as ship size and the water depth in the area around the ship by selecting a minimum depth with an added safety margin.

In this work, the ship's instantaneous states and relevant ENC data are fed into the risk model at regular intervals (e.g. every 30 seconds) to produce an estimate of the probability of grounding in the next time interval, applied repeatedly across a receding horizon of e.g. 1 hour to predict future probabilities into an imagined scenario. Thus, the model will be used to evaluate the probability of grounding during a potential future scenario k corresponding to a set of proposed control outputs. In this case, predicted ship states and ENC data corresponding to scenario k is produced by the ship simulator.

The grounding risk model is illustrated in Figure 6.3, and deals specifically with the case where the loss of propulsion power may cause the ship to drift aground if propulsion power is not restored in time to prevent it. It is structured as a bow-tie diagram, with unexpected component failures as the triggering events, loss of propulsion power (LOPP) as the hazardous event, and grounding as the considered consequence. This diagram thus conforms with the scope of this work, i.e. online



Environmental forces
WP position
MSO mode
Time it would take to recover from loss of propulsion power
Time it would take to ground given loss of propulsion power
Triggering events:
Unexpected events component failures
Proactive barriers:
Two DGs running in PTI mode such that loss of single DG will cause LOPP
Hazardous event:
Loss of propulsion power
Reactive barriers:
Restore propulsion power before grounding occurs
Consequences:
Grounding

Figure 6.3: Illustration of the collision risk grounding model.

navigation before a potential loss of propulsion event is considered with respect to grounding risks. Moreover, the MSO modes affect both the engine recovery time during LOPP as well as the potential for unexpected component failures, and may act as proactive barrier if selected appropriately. Lastly, environmental forces (disturbances) and the waypoint positions distributed along the navigated trajectory or path affect the time it would take to ground the ship if an hazardous event occurs.

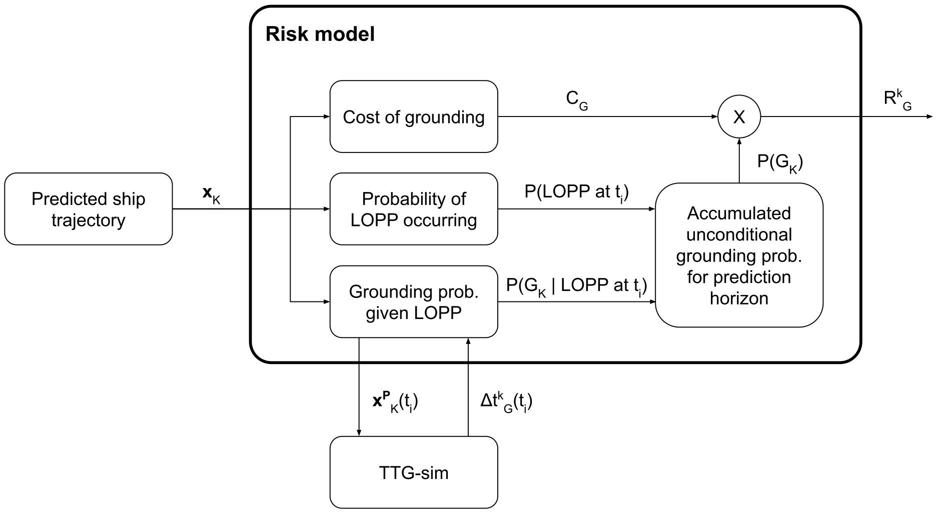


Figure 6.4: Block diagram illustrating the main principles of the risk model. TTG-sim: The simulated "Time-To-Grounding" prediction, in which $\boldsymbol{x}_K^P(t_i)$ are the predicted black-out trajectories at any time instant t_i, and $\Delta t_G^k(t_i)$ are the simulated TTG predictions.

Figure 6.4 shows the implementation and structure of the risk model. The hazardous event of Figure 6.3 serves as the key element in the risk model, in which the probability of a LOPP event occurring is combined with a time-to-grounding (TTG) prediction to calculate the accumulating grounding probability distribution across the prediction horizon, based on the future predicted ship states as inputs. This probability is ultimately multiplied with the cost of grounding to produce the total grounding risk R_G^k.

The probability of grounding

There may be significant transients in R_G^k over the duration of a prediction horizon. To capture the effect of these transients, the prediction horizon is subdivided into n time intervals. $P(G_k)$ can be formulated as

$$P(G_k) = \sum_{i=1}^{n} P(G_{i,k}) \tag{6.5}$$

where $G_{i,k}$ denotes the event of grounding during time interval i in scenario k. It is only possible to ground once during the prediction horizon. Therefore, the probability of grounding during a specific interval i in the prediction horizon must account for the possibility that a grounding already occurred earlier in the prediction horizon.

The notation $P(G_{i,k}|\bar{G}_{1,k}, \bar{G}_{2,k}, ..., \bar{G}_{i-1,k}) := P(G_{i,k}|\bar{G}_{i-,k})$ used in (6.6) is defined as the conditional probability of grounding during the interval i in scenario k, given that it did not occur prior to the i^{th} interval in scenario k, and $\bar{G}_{j,k}$ is the complementary event of $G_{j,k}$. Thus, the probability of grounding during the time interval i in the prediction horizon k can be formulated as

$$P(G_{i,k}) = \begin{cases} P(G_{i,k}|\bar{G}_{i-,k}) \prod_{j=1}^{i-1} (1 - P(G_{j,k})) \, , & \text{if } i > 1 \\ P(G_{1,k}), & \text{if } i = 1 \end{cases} \tag{6.6}$$

Next, each time interval in the prediction horizon k is considered. To simplify the notation, the indices i and k are not included in the following derivation. A potential grounding scenario can be subdivided into a scenario that leads to loss of propulsion power (a LOPP scenario), and a recovery scenario. If a LOPP scenario occurs, a grounding follows if a recovery scenario cannot be successfully executed within the time it takes the ship to drift aground. A LOPP scenario can be described by a set of triggering events, while a recovery scenario is described by a set of startup events. In the model, a set of potential LOPP scenarios is associated to each MSO mode. Moreover, a set of potential recovery scenarios are associated to each LOPP scenario in each MSO mode. That is, the possible ways of recovering the system after LOPP depends on the scenario that caused the LOPP event and the state of the system (MSO mode) when the LOPP event occurred.

Consider the power and propulsion system illustrated in Figure 6.5 as an example. The system can be operated with two propellers, each with its independent power source. It is assumed that one power source and propeller is sufficient to prevent the ship from grounding. As shown in the fault tree in Figure 6.6, a potential LOPP scenario, in this case, is that both power sources are lost (i.e. the set containing the two events "ME1 stops" and "ME2 stops"). The event tree in Figure 6.6 illustrates the two corresponding potential recovery scenarios, namely either the recovery scenario in which ME1 is restarted, or the recovery scenario where ME2 is restarted. If the ship is operated with only one online power source, e.g. ME1, then a LOPP scenario is described by the event "Loss of ME1", while the potential recovery scenarios becomes "Restart ME1" or "Start ME2". In general, the event of starting a component and the event of restarting a component are distinguished from each other. Whether one is restarting a component (after unexpected loss) or starting a component may e.g. affect the probability of success.

The probability of experiencing the particular LOPP scenario S consisting of triggering events e_t is denoted $P(S)$, see (6.7). If the triggering events can reasonably be modeled as independent events, and the ship is operated in a mode where S is a potential scenario, then

$$P(S) = \prod_{e_t \in S} P(e_t). \tag{6.7}$$

In this chapter, the triggering events e_t are modeled as exponentially distributed events with constant frequencies of occurrence λ_t, as defined in (6.8). Thus, the probability of experiencing the event e_t during a time interval Δt is

$$P(e_t) = F_{exp}(\Delta t; \lambda_t) = 1 - e^{-\lambda_t \Delta t}. \tag{6.8}$$

A set E_r of possible recovery scenarios following a particular LOPP scenario in a given machinery system configuration is referred to in a recovery event tree, where each possible recovery scenario r_i is a branch on the event tree (as exemplified in Figure 6.6). The probability of not succeeding in recovering the system, given

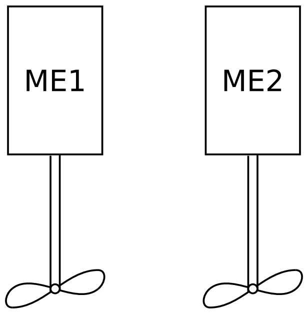


Figure 6.5: Example power and propulsion system with two propellers using independent ME power sources.

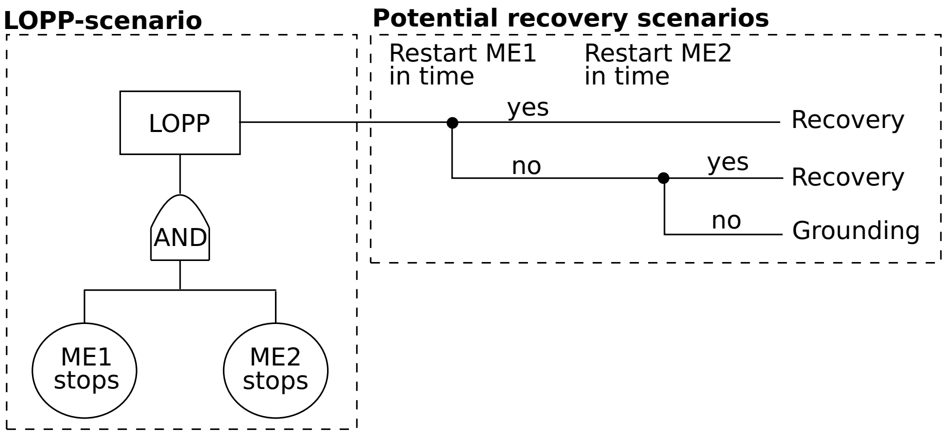


Figure 6.6: A potential set of two grounding scenarios for the example system.

the occurrence of LOPP, is equal to the probability that none of the possible recovery scenarios in E_r succeed within the time it takes the ship to drift aground. The probability $P(G|L) = 1 - P(E_r|L)$ of not succeeding in recovering before grounding can be calculated directly from the event tree according to the standard event tree methodology, (e.g. in Figure 6.6, the probability of grounding given LOPP would be the product of the two probabilities of not restarting main engine (ME) in time and not restarting M2 in time). To achieve this, it is necessary to find the probability $P(e_r)$ of each recovery event e_r occurring before grounding (e.g. restarting ME1 in Figure 6.6). It is noted that the time Δt_r it takes from LOPP occurs to event e_r occurs (e.g. M1 is successfully restarted), is a random variable. Thus, the probability that the event occurs before a grounding occurs (i.e. before the time Δt_G elapses), is given by the cumulative distribution of Δt_G, $P(\Delta t_r < \Delta t_G) = F_{e_r}(\Delta t_G)$ and a nominal probability of success p_r associated with the event (e.g. the probability that it is possible to restart ME1 given infinite time),

$$P(e_r) = F_{e_r}(\Delta t_G) \cdot P(r). \tag{6.9}$$

The time Δt_G it would take to ground given the occurrence of LOPP, is found by simulating that the ship drifts without propulsion (and steering), subject to environmental forces. This is achieved by using the TTG simulator for predictions as illustrated in Figure 6.4. This prediction uses the same model as described in Section 6.2.2, but without propulsion and steering. Lastly, the TTG prediction is initialized by using the predicted ship state each time that LOPP is simulated to occur.

The cost of grounding

The cost of grounding is a function of the ship state vector $\boldsymbol{x}$, as illustrated in Figure 6.4. In general, the cost of grounding can have a large range of contributions. In this research, it is proposed to divide the contributions into costs associated with:

- C_{ship} := damage to the ship. This depends on the system states $\boldsymbol{x}$, such as impact speed and location of grounding (i.e. the type of surface the ship grounds into), as well as the sea state, $\mathcal{S}$ (i.e. large waves may cause a more violent impact) to be used in the cost function.
- $C_{recovery}$:= rescue fee that must be paid to recover the ship. This may depend on the constant parameters of the ship such as the length of the ship, but also the system states $\boldsymbol{x}$, and in particular the location of the ship (e.g. if it is far from civilization and the nationality of the rescue team).
- C_{cargo} := damage to or loss of cargo. This may be set as a fixed parameter according to the value of the cargo, as well as being dependent on the magnitude and nature of the impact.
- $C_{environment}$:= environmental damages such as oil spill in the ocean. This may depend on fixed parameters such as the amount of oil carried by the ship, but also the system states $\boldsymbol{x}$, and in particular the location of the ship (e.g. the sensitivity of the marine area) and weather conditions.
- $C_{infrastructure}$:= damage to infrastructure on the shore such as fish farms, harbors, promenades, and so forth. This cost may depend on the system states $\boldsymbol{x}$, (e.g. location of the impact and whether or not there are infrastructure there to be damaged).
- $C_{reputation}$:= loss of reputation due to loss of or damage to cargo or major delays in delivery. This may be modeled as a fixed quantity.

The total cost of grounding may then be estimated as

$$\begin{aligned} C_G = {} & C_{ship}(\boldsymbol{x}, \mathcal{S}) + C_{recovery}(\boldsymbol{x}) + C_{cargo} \\ & + C_{environment}(\boldsymbol{x}) + C_{infrastructure}(\boldsymbol{x}) + C_{reputation}. \end{aligned} \tag{6.10}$$

This concludes the methodology of this work, and will in the following sections be implemented in a simulation study, serving as the foundation for the results and discussion in which the proposed approach presented in this chapter is validated and assessed.

6.3 Results

In this section, a simulation study is presented. The objective is to test and demonstrate the proposed control algorithm for autonomous ship control. The control system is implemented on a simulation model of a coastal cargo ship with a length overall (LOA) of 81.5 meters and a beam of 16 meters and a displacement of 5335 tons.

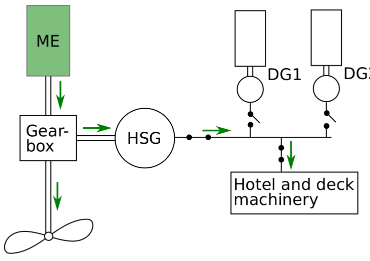
6.3.1 The machinery management system

The ship is equipped with a hybrid-electric propulsion system. There is one propeller that is powered from a gearbox. The gearbox can be powered either from the

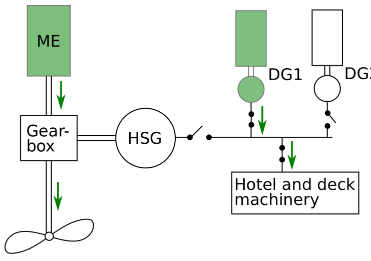
ME, or from a hybrid shaft generator (HSG). The HSG converts electrical power from an electrical bus that can be powered from two identical diesel generators (DGs). There are several ways in which the propulsion system can be operated. In this case study, only the three predefined MSO modes; power take out (PTO), mechanical (MEC) and power take in (PTI) are considered.

As illustrated in Figure 6.7(a), PTO refers to a mode where the ME is responsible for the main propulsion, as well as auxiliary electrical loads. In this case, both DGs are offline and the HSG functions as a generator, transforming mechanical power from the gearbox to electrical power. In MEC mode, (see Figure 6.7(b)) the auxiliary electrical loads are served by one of the DGs instead of the HSG. Thus, the HSG is off, and all the power produced by the ME is used for propulsion. Finally, as seen in Figure 6.7(c), PTI mode uses the DGs to provide power for propulsion. In this case, the HSG is acting as an electrical motor, transforming the electrical power from the DGs into mechanical power on the gearbox.

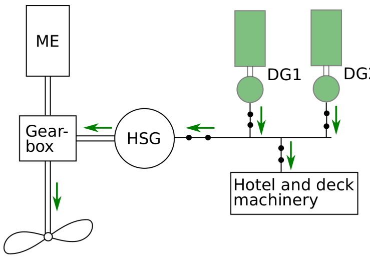
The main engine is a marine diesel engine with a maximum continuous rating (MCR) of 2160kW, while the two diesel generators are rated at 590kW each.



(a) PTO-mode where the ME is responsible for both propulsion and electrical loads.



(b) MEC-mode where the ME is responsible for propulsion and a DG is used for auxiliary electrical loads.



(c) PTI-mode where two DGs are responsible for main propulsion and auxiliary electrical loads.

Figure 6.7: Diagrams of the machinery system's layout in the three operational modes. Green color indicates online components and the arrows indicate the direction of energy flow (power) [36].

6.3.2 Risk model setup

Table 6.2 presents the possible scenarios (consisting of a LOPP scenario and a set of possible restoration scenarios) that can occur in each MSO mode. The LOPP-scenarios and restoration scenarios are described in terms of triggering events and restoration events, respectively.

The expected rate of occurrence for each triggering event is presented in Table 6.3, and the restoration events and their parameters are given in Table 6.4. Here, the nominal probability refers to the probability of success of a recovery event given infinite amounts of time. The mean time, standard deviation and minimum time, are parameters in the restoration events success time, where mean time refers to the mean time given that it will start (i.e. assuming that the nominal probability is one). A lognormal distribution is assumed, where the minimum time parameter refers to a time interval. As an example, "Start ME" takes at least 20 seconds, according to the parameters in Table 6.4.

6.3.3 Environment setup and route planning

For proof of concept, a simple simulation environment is created using the ENC package *SeaCharts* [37] in Python 3.10. An area of approximately 14 square kilometers west-northwest of the Norwegian city of Ålesund is chosen for the simulation study, shown in Figure 6.8. This environment showcases an interesting scenario in

Table 6.2: Description of the possible scenarios in each MSO mode

MSO mode	LOPP-scenarios	Possible restoration scenarios
PTO	"ME stops"	"Restart ME"
		"Start DG1" AND "Start HSG"
		"Start DG2" AND "Start HSG"
MEC	"ME stops"	"Restart ME"
		"Start HSG"
PTI	"DG1 stops" AND "DG2 stops"	"Restart DG1"
		"Restart DG2"
		"Start ME"
	"HSG stops"	"Restart HSG"
		"Start ME"

Table 6.3: Overview of considered triggering events and rates of occurrence

Triggering event	ME stops	DG1 stops	DG2 stops	HSG stops
Failure rate	3e-9	6e-9	6e-9	2e-9

Table 6.4: Overview of restoration events and their statistical parameters

Recovery events	Start ME	Restart ME	Start DG1	Restart DG1	Start DG2	Restart DG2	Start HSG	Restart HSG
Nominal prob.	1	0.4	1	0.5	1	0.5	1	0.8
Mean time	50	50	35	35	35	35	12	12
Std. deviation	1.4	1.4	1	1	1	1	1	1
Minimum time	20	20	14	14	14	14	3	3

which one may choose between two different paths on either side of an island, and is considered well suited for a proof of concept.

The tentative ship route or path to follow is shown in Figure 6.8 as green line segments connected by "links" at each given waypoint, as generated by the global voyage planner of Figure 6.1. Notice however how one of the green line segments are intersecting an island, highlighted by the red color where the island crosses the globally planned line segment. This setup is specifically chosen to demonstrate that if a planned tentative route is somehow inaccurate or incomplete such that grounding obstacles are present along the route, one may utilize e.g. the Level-1 route planner from [37] to generate alternative feasible routes on opposite sides of the obstacle in question. Moreover, one may analogously extend the anti-grounding algorithm to also encompass collision avoidance of dynamic obstacles, through e.g. the concept of (polygonal) adaptive safety domains [28] constructed around e.g. nearby vessels. Thus, it is argued that the approach shows significant flexibility and adaptability. Section 6.6 contains a summary of the planning algorithm, as well as a visual demonstration of each algorithm step.

Figure 6.9 shows the result of the Level-1 path planning performed on the ship route of Figure 6.8, as generated by Algorithm 8 from Section 6.6. First, the green line segments are checked for intersections with any grounding obstacles in the environment, which in this case yields the red streak as shown in Figure 6.8. Second, the convex hull of the intersected grounding obstacle (island) is extracted, and an added buffer of a 50 m safety margin is applied in all directions from the obstacle exterior boundary, in addition to the already added 10 m buffer and vertex simplification process performed during the construction of the polygons of the *SeaCharts*

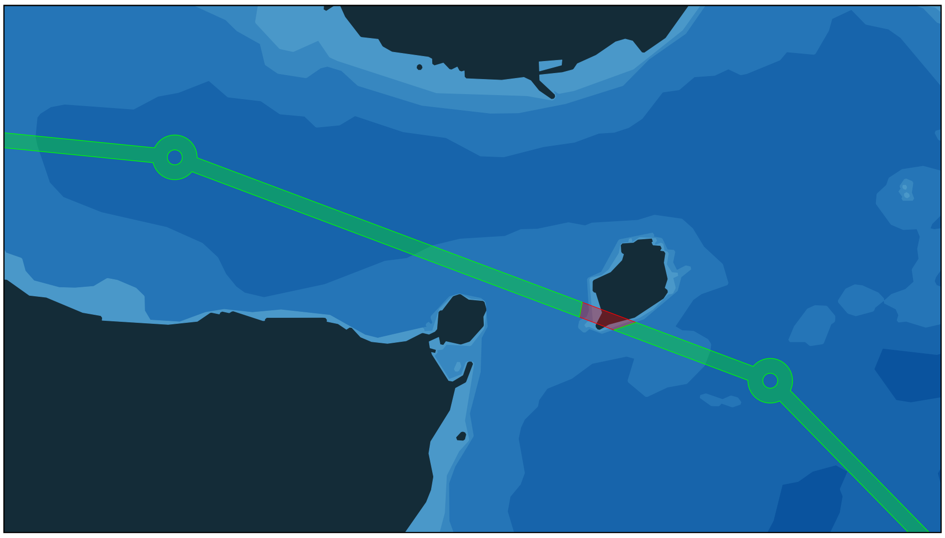


Figure 6.8: Visualization of the simulation study area and ship route using the *SeaCharts* package [37].

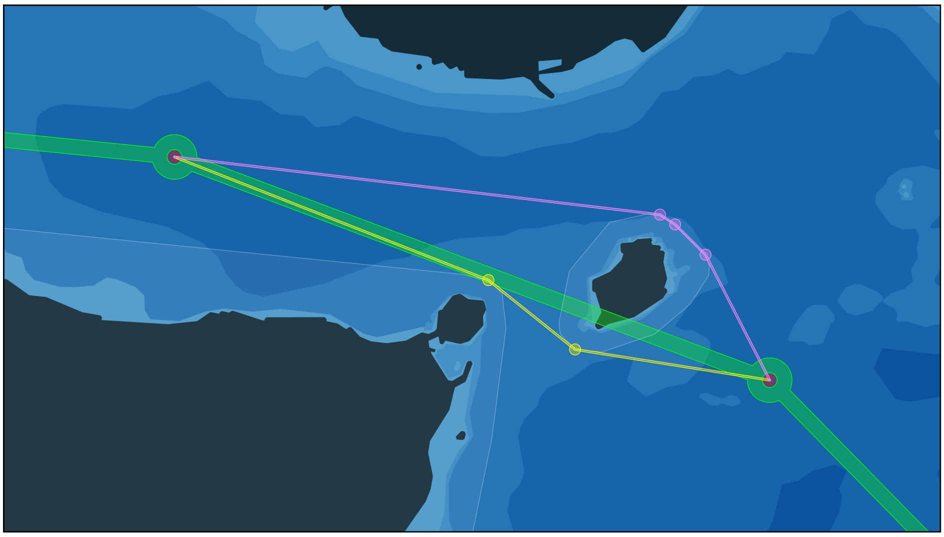


Figure 6.9: End result visualization of the Level-1 route planning algorithm [37].

ENC. This yields the two convex polygons highlighted around each of the islands south in the environment.

It is important to notice that one should be careful with this "hard" static safety margin. If this buffer around each grounding obstacle is too large, the subsequent path following or guidance controller may have trouble with navigation through extremely narrow straits, or one may even risk closing the strait in its entirety, losing the possibility of navigating through it as a route alternative. Thus, it is argued that the buffer should be somewhat conservative, and that the path following algorithm or controller is expected and required to be capable of operating in the interior of the feasible domain, as opposed to at the boundaries of hard constraints such as the grounding obstacle exteriors. Nonetheless, the Level-1 alternative route path planner is indeed a linear optimization algorithm operating on the vertices and line segments of each grounding obstacle polygons, essentially generating an approximate ship path to be used both during initialization and as part of the cost function of the Level-2 route planning optimizer of Figure 6.1.

In Figure 6.9, the red disk within the path waypoint link to the east shows the start point of the simulation study. Conversely, the red disk to the west is the next target path waypoint. Algorithm 8 iterates through each of the grounding obstacle vertices, and checks if the point is visible (i.e. accessible along a straight uninterrupted line) from the reference point. The first reference point is thus the red east-most starting point, and the distances between each vertex visible from the reference location and the green line segment are measured. The visible vertex farthest away from the path is selected as the first alternate waypoint, and the process is repeated with each newly generated waypoint as the visibility reference location. This generates a new collection of line segments on each side of the grounding obstacle, and these alternative paths are in Figure 6.9 shown in yellow and pink. Notice how the generated yellow path originally intersected with the larger island to the south-west, which prompted another sub-run of the algorithm such that the new intersection is considered in the final path alternatives. See Section 6.6 for more details on this procedure.

6.3.4 Particle swarm optimization

The waypoint (route) optimizer used in this simulation study is a risk-aware PSO waypoint planning algorithm which is extended, based on previous works [41]. Compared to other methods such as MPC [38], PSO is not subject to any special cost function construction or feasibility concerns in order to generate solutions (not guaranteed to be optimal). Thus, one may utilize highly discontinuous or discrete cost definitions, allowing for more complex general optimization.

The principle behind PSO is to randomly generate an initial *swarm* of N-dimensional solution *particles*, and repeatedly update the particle positions with respect to semi-random particle velocities based on their performance measured by the cost function. The technique is widely covered in the literature, and the reader is referred to previous works for more in depth background on PSO [41].

A simple demonstration case is shown in Figure 6.10, in which only the 2D XY-coordinates of the path waypoints in the horizontal plane are optimized through purely distance-based and spatial costs from the ad hoc risk-aware implementation discussed in [41]. The same green line segments, start and target in red from Figure 6.9 are considered, as well as the newly generated route alternatives – here, shown in gray on each side of the smaller island.

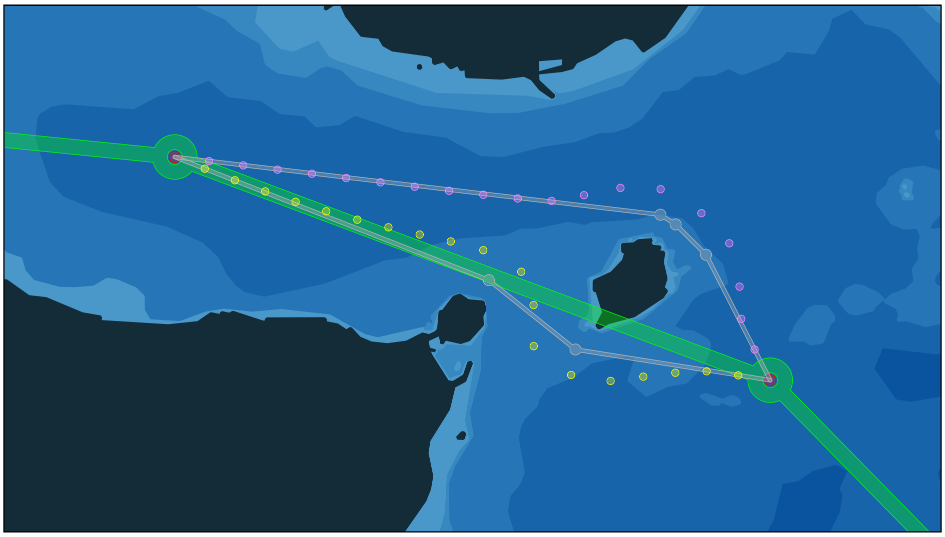


Figure 6.10: PSO demonstration of 2D waypoints along the ship route.

The alternate paths are subsequently split into 20 sub-segments, corresponding to 19 waypoints shown in yellow and pink, respectively. The first line segment corresponds to the given start waypoint. These intermediate waypoints are used directly as the 2D particles to be optimized by the PSO, with respect to any nearby grounding obstacles. The cost function is a sum of simple path-related costs such as total path length and waypoint distances to the original path, as well as a simple risk-aware exponential function applied to the grounding obstacles [41]. The latter term weighs small distances between the ship position and nearby obstacles very highly compared to far-away locations, essentially adding a dynamic "soft constraint" which prevents the optimized path from crossing obstacles.

It is clear how the yellow and pink waypoints simulate increased risk-aware behavior with respect to the nearby islands, if followed by a navigation or guidance controller (see e.g. Figure 6.13). Furthermore, one may also note how the lack of hard constraints keeps the problem well-behaved, even in the more narrow strait between the two islands shown in yellow. If e.g. the safety margins discussed previously had been increased as a substitute for the distance-based "interior" cost inside the feasible region, one could end up with sharp and even infeasible paths between narrow straits such as the one shown. The magnitude of the obstacle avoidance costs are exaggerated for visibility in this proof of concept.

6.3.5 Risk cost formulation

The formulation of the final risk-aware cost function is subject to many considerations. Figure 6.11 presents a visualization in which the same intermediate yellow and pink waypoints from the Figure 6.10 are shown in the colors yellow and cyan (replacing the pink for visibility), respectively. Here, the ends of the green line segments replace the initial route "links", and denote the original red start and target locations. The increased risks simulated by the exponential term in the cost function is readily apparent from the overlapping contour polygons shown around each grounding obstacle, increasing in intensity and color from light yellow to dark red within the obstacle interiors. The optimized waypoints in yellow and cyan are seen traversing over or along the "hills and valleys" of the risk contours around the obstacles, and there is a strong correlation between the risk contour magnitudes and the resulting waypoints arrangement.

Obstacles previously hidden from sight also become apparent in this view, as every land area, shores and/or seabed depths more shallow than 10 m are included as (convex) red obstacle interiors. Thus, these obstacles also contribute to the spatial optimization, but are in this scenario negligible if sufficiently far away from the considered waypoints. This effect can be verified by comparing the colored intermediate waypoints with the previous alternate line path segments of Figure 6.10: There is no considerable discrepancy between the optimized waypoints and the Level-1 planned paths when no grounding obstacles are nearby the original path, as a direct consequence of the exponential nature of the risk cost term.

These distance-based risk awareness contours of Figure 6.11 resembles artificial po-

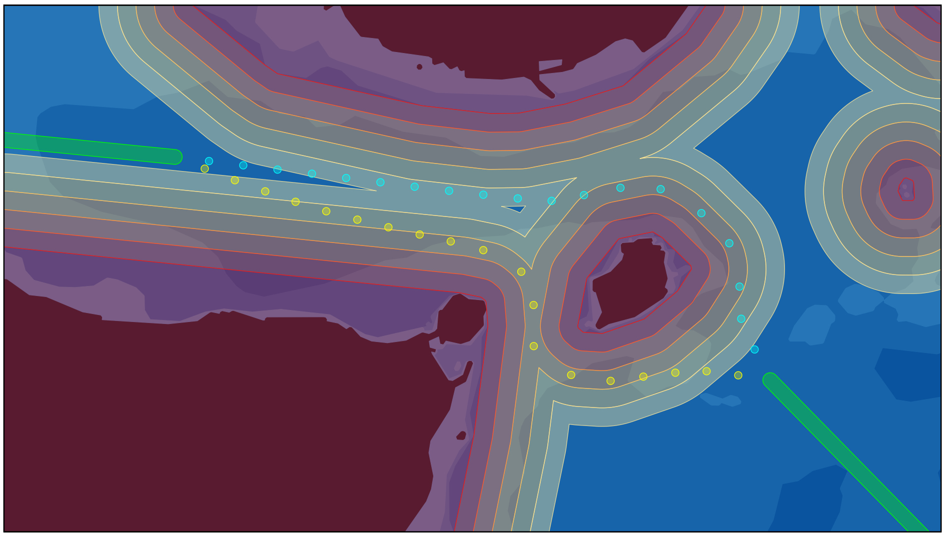


Figure 6.11: Contours visualization of the distance-based grounding risk cost term used in this work.

tential (repulsion) fields, which is another popular approach used for path planning for e.g. unmanned autonomous vehicles. This method is however prone to becoming stuck in local minima and may show poor performance in narrow passages such as the isle strait considered here, and these issues must also be recognized and handled when using PSO. The sum of additional path-related costs are valuable in this regard, strongly related to the previous point with respect to the negligible divergence between the Level-1 routes and the optimized waypoints further away from obstacles: By enforcing large costs associated with straying away from the original path (as well as increasing the total path length), the (near-) optimal placements of each waypoint are semi-forced along the original path. This approach does however place more responsibility onto the Level-1 planner in order to achieve satisfactory solutions, which is considered appropriate following that the global planned path is already assumed to be near-optimal in this study.

In previous works, a scalar cost with respect to environment (wind) disturbances was used in conjunction with the static distance-based grounding obstacle costs to account for the increased risks present when obstacles are located down-wind (or down-stream) of the ship [38], [41]. Figure 6.12 shows a comparison view of the effect this extra cost term has on the waypoint distribution across each route alternative. The yellow and pink waypoint paths of Figure 6.10 are here denoted in orange and magenta, respectively, and the new resulting waypoints of each alternative including the added scalar product cost term are shown in yellow and pink.

For simplicity, only wind disturbances are included in the proof of concept demonstration. In the upper-right corner of Figure 6.12, the wind direction and wind velocity of the disturbance forces are shown as 250° and 10 m/s, respectively. It is

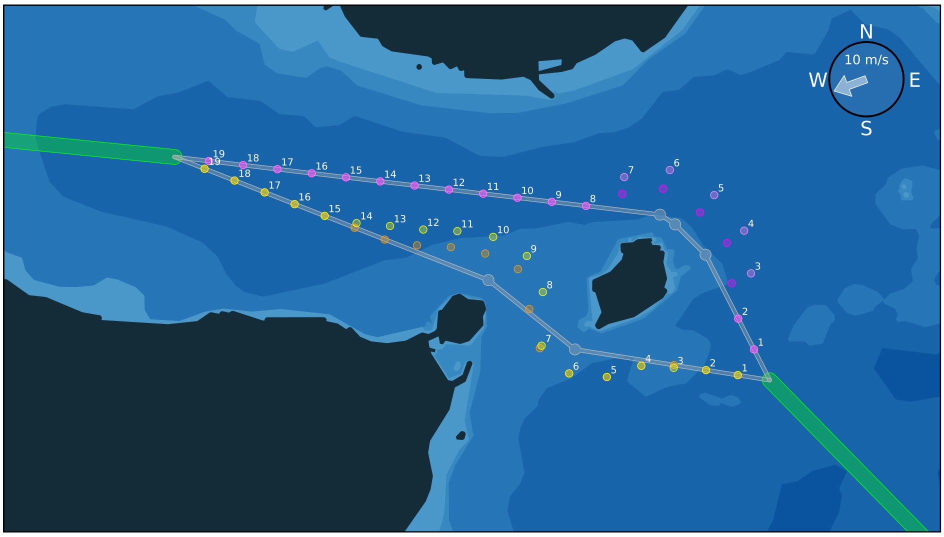


Figure 6.12: Comparison of the scalar product grounding risk cost used in previous works.

clear how the scalar product of the wind direction and the direction to each grounding obstacle weighs more heavily onto the waypoint costs, effectively shifting them in approximately the opposite direction. Though the PSO algorithm is entirely sample-based and not gradient-based, the direction of the extra perturbations of the spatial waypoint locations are very similar, as expected. See the visualizations and discussions presented in the previous works for more details [38], [41].

Some interesting effects are seen on a few waypoints. On the pink path, one can see how WP3 is more eastward, and WP7 is almost completely northward compared to their magenta counterparts, due to the scalar product of the closest point on the nearby grounding obstacle and the wind direction. Most notably, WP3 of the yellow path demonstrates a slightly unintended effect of using this risk cost formulation. Here, the wind direction (in this example) compared to the direction of the nearest potential point of grounding as seen from the ship, is such that the scaled extra cost of the exponential scalar product term is sufficient to noticeably move the waypoint southward unnecessarily. Though the risk cost scaling in these examples are exaggerated greatly for visual clarity, effectively resulting in less efficient routes around the islands, there is evident potential for improvements.

Thus, a new risk cost formulation is presented in this chapter, which utilizes a ship model and the concept of TTG in order to produce more precise and appropriate waypoint planning solutions. It is argued that this cost formulation reflects realistic scenarios to a higher degree, more accurately incorporates the dynamics (i.e. the trajectory) of the ship, and is considered a natural addition to the cost function given the new scope which also includes machinery management considerations. The final cost function is presented in Section 6.3.9.

6.3.6 Path following and trajectory control

The output of the Level-2 route planner of Figure 6.1 is ultimately given as input to the ship's guidance system, which in turn controls the trajectory of the ship toward the resulting waypoints. Figure 6.13 shows an example simulation of trajectories produced by a LoS guidance controller, following the paths generated by both alternative sets of waypoints. Here, the speed of the ship is set constant, for simplicity. Most strikingly, the yellow trajectory is noticeably faster than its counterpart in pink. Its end position is readily seen in the figure being located farther along the path, after the same number of sampled time intervals.

In the example trajectories shown in Figure 6.13, the yellow trajectory is faster than the pink trajectory, but intuitively it does also involve higher levels of grounding risks – as apparent in Figure 6.11. This leads to the very purpose of this chapter, and is indeed the main research question to be considered: How can both the efficiency and risk aspects of ship paths or trajectories be weighted such that the resource consumption is minimized during a successful mission execution, while simultaneously achieving safety?

6.3.7 Time-to-grounding (TTG) predictions

As noted in the discussion related to Figures 6.3 and 6.4, the principle of "time-to-grounding" is simply to predict when (if) a ship would experience a grounding event if a LOPP (machinery failure) scenario occurs at a given time instant along the planned path, given the current or expected environment (weather) conditions.

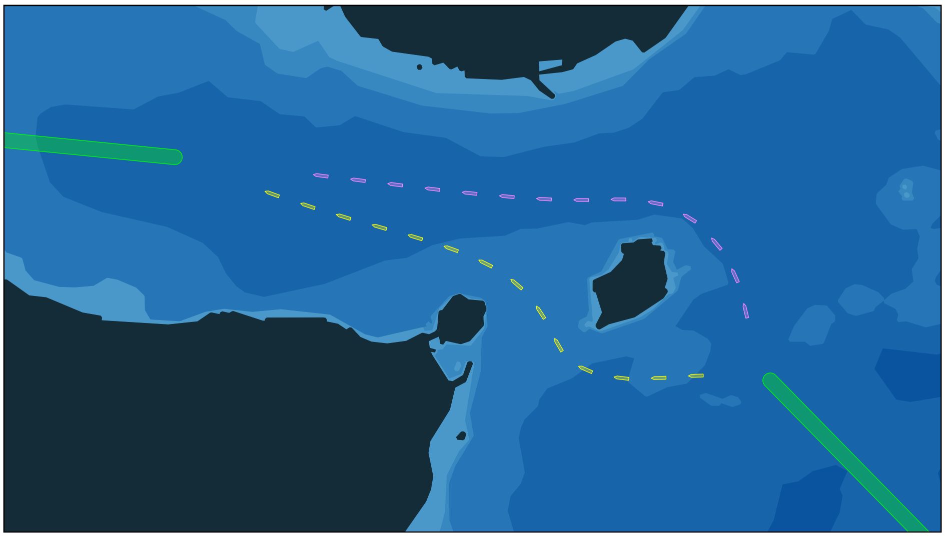


Figure 6.13: Path following simulation example, based on a simple line-of-sight guidance controller.

Figure 6.14 presents a demonstration of the TTG predictions. The wind velocity is 10 m/s, and the currents velocity is 1 m/s. Here, the orange and green collections of ship poses are not simulation trajectories, but rather the ideal ship poses defined for each waypoint distributed evenly along the original paths. This shows how an initial ship yaw angle or heading is needed for each (ideal) waypoint in order to predict TTG. The angles are calculated using the angle between the previous and the next (neighboring) waypoints, for each individual waypoint. The yellow and pink colored ship poses denote the *predicted* trajectories during a LOPP scenario corresponding to each ship pose of the orange and green routes, across a horizon of 10 min.

The predicted future trajectories with no propulsion and steering are simulated by the ship model, and include the ship dynamics and initial ship speed before loss of propulsion power. As expected given a wind direction of 250°, the ship in the orange trajectory is predicted to drift south-west toward the south-west island, and the ship in the green trajectory would firstly hit the smaller island. Moreover, as the accumulated probability of grounding only increases (and is defined) given that a grounding event has *not* occurred, the future predictions are ended if any part of the ship intersects with a grounding obstacle. These intersections are shown as red ship poses. Note that the red grounding events may be asynchronous with respect to the regular sampling intervals of predicted ship state (pose), across the LOPP scenario horizon.

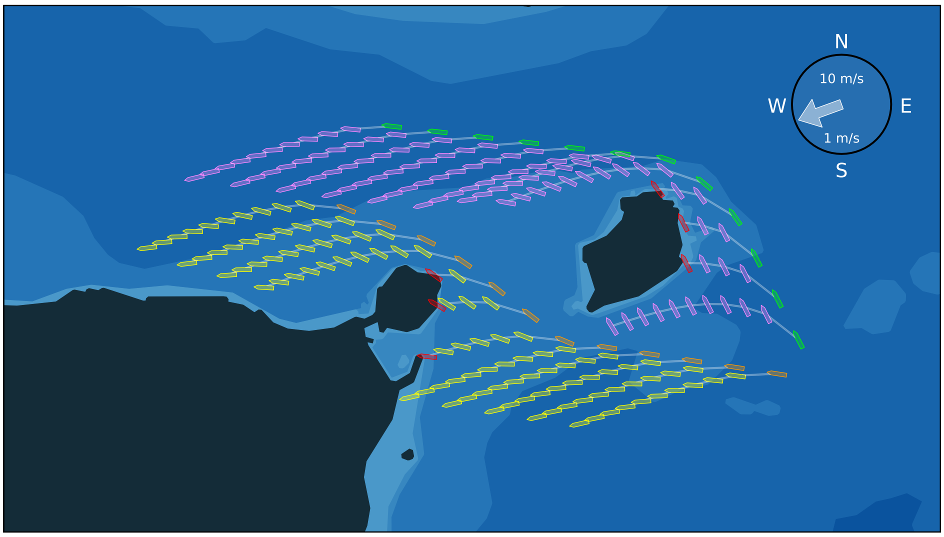


Figure 6.14: Time-to-grounding predictions shown for a LOPP scenario occurring along the trajectories.

6.3.8 MSO mode selection and fuel consumption

The TTG predictions are used to inform the optimization during the PSO run, i.e., to select the most suitable MSO mode as well as the waypoint locations during optimization. This is due to the fact that each MSO mode have different fuel consumption rates when active, and have different restoration properties. The resulting time values of the TTG predictions are subsequently translated into grounding probabilities, and then expected costs through the rate of failure probabilities and restoration rates of Tables 6.2 to 6.4. These costs are ultimately weighted against all other costs defined by the path following cost function, and the PSO outputs three-dimensional (3D) solution particles consisting of the X and Y coordinates of each waypoint, and the selected MSO mode to be used for the following time interval.

Figure 6.15 presents an alternative view of the ideal sailing routes and LOPP blackout predictions of Figure 6.14, in which additional directed arrows denote how the 2D cost *gradients* of the waypoint locations in the horizontal plane are affected by the TTG predictions and resulting estimated costs. The altered locations of the optimized waypoints would in turn *increase* the total fuel consumption, assuming that the original path is near-optimal. It is intuitive that since the risks for a grounding event occurring increases along the direction of the wind disturbance, a purely spatial cost function would move the waypoint locations away from the predicted points of impact [38], [41]. However, in this work, the MSO mode selection also plays an important role.

In this figure, the trajectories are unchanged for the purpose of conceptual demonstration and comparison to later results. For this example, both trajectories ex-

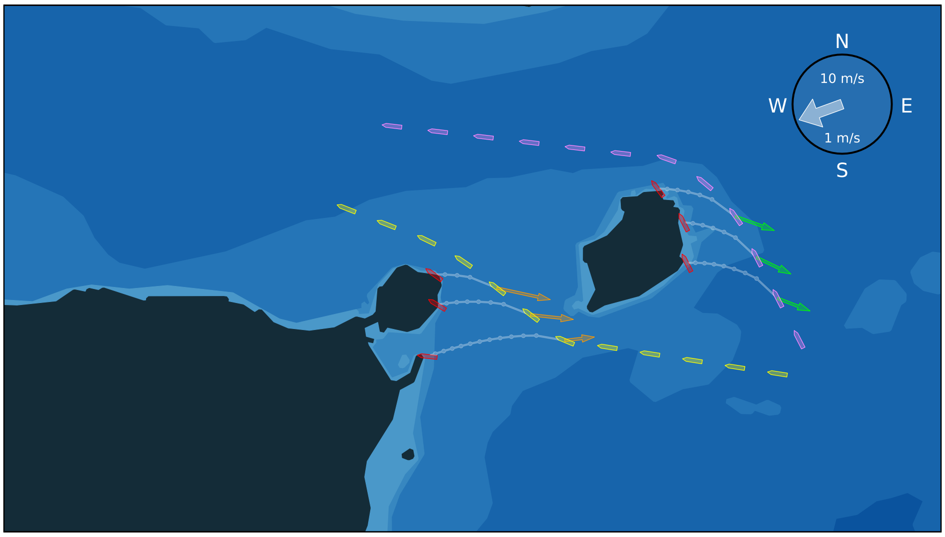


Figure 6.15: Spatial waypoint risk gradients demonstrations.

perience proximity to higher grounding probabilities for an approximately equal amount of time. However, in general, this is not necessarily the case, and the fuel consumption along the complete trajectories are highly dependent on both the specific MSO mode selected, and the accumulated time spent in the mode. Thus, it may sometimes be more economically prudent to simply move the route waypoints further away from the grounding obstacles, as an alternative to disrupt the machinery to go into another (i.e. safer but also more costly) MSO mode. This joint combination of spatial path optimization and operational mode cost minimization during operations is considered the novel contribution of this work.

6.3.9 The complete cost function and simulations

Based on the discussions and intermediate results of the previous sections, the final complete cost function is formulated as follows:

$$C_k(\boldsymbol{\varpi}, m) = C_{path}(\boldsymbol{\varpi}, k) + C_{grounding}(\boldsymbol{\varpi}, k, m) + C_{mso}(\boldsymbol{\varpi}, k, m) \tag{6.11}$$

$$C_{path}(\boldsymbol{\varpi}, k) = \mu_1 ||\boldsymbol{\varpi} - \boldsymbol{\varpi}_k^{ref}||^2 + \mu_2 \Big(||\boldsymbol{\varpi} - \boldsymbol{\varpi}_{k-1}^{ref}|| - ||\boldsymbol{\varpi} - \boldsymbol{\varpi}_{k+1}^{ref}|| \Big)^6 \tag{6.12}$$

$$C_{grounding}(\boldsymbol{\varpi}, k, m) = C_G \Big(\mu_3 \sum_{\sigma \in O} e^{-d_{min}(\boldsymbol{\varpi}, \sigma)\zeta_1} + \mu_4 P(G) \Big) \tag{6.13}$$

$$C_{mso}(\boldsymbol{\varpi}, k, m) = C_{consumption} \cdot \mu_5 ||\boldsymbol{\varpi} - \boldsymbol{\varpi}_{k+1}^{ref}|| \tag{6.14}$$

where $\boldsymbol{\varpi}_k = P(x, y)$ is a 2D waypoint corresponding to the k^{th} line segment along a route alternative, and x, y, k and m are the x- and y-coordinates of a waypoint, the line segment number and the selected MSO machinery mode, respectively.

The second term of C_{path} is raised to a larger (even) power than the first to more strongly encourage distributing the waypoints with equal distances between each other, compared to being close to the ideal reference waypoint along the route. $\sum_{\sigma \in O} e^{-d_{min}(\boldsymbol{\varpi}, \sigma)\zeta_1}$ is the total sum of the negatively scaled minimum distances to every grounding obstacle raised to the power of e, which serves as an exponential barrier function for nearby grounding obstacles irrespective of the heading of the ship or any disturbances.

$P(G)$ is the accumulative grounding probability function from (6.5), and $C_{consumption}$ is the estimated fuel cost per meter traveled. For simplicity, it is for (6.11) assumed that the variable costs defined in (6.10) are held constant for the entire optimization horizon, i.e. C_G is static based on a set of assumptions related to the current surrounding environment. The reference waypoints $\boldsymbol{\varpi}^{ref}$ denote the ideal waypoint locations evenly spread across all line sub-segments along a route alternative if left completely unaltered by grounding risk costs, i.e. $C_{path} = C_{grounding} = 0$ (see the pink waypoints 8 to 19 in Figure 6.12). In this work, the PSO setup used 30 candidate particles in each particle swarm (one for each of the 20 line segments), and was run for 100 iterations. The following hyper-parameters of the PSO was used: The inertia weight was set to 0.75, the cognitive weight was 1.0, the social weight was 2.0, and the velocity limit was 1.0.

Using the path following guidance controller, the resulting optimized WP distributions and trajectories of each route alternative are shown in Figure 6.16. The green trajectory follows the resulting PSO route in pink, and the orange trajectory follows the yellow route. The target ship heading each time interval is calculated by drawing line segments between the optimized waypoints, and extracting the target coordinates by intersecting the resulting path by a circular horizon radius of 200 m. Thus, the generated ship trajectory is entirely independent of the distance between each optimized waypoint of the PSO, and the smoothness of the path to follow may be improved simply by increasing the number of waypoints to optimize.

The cyan waypoints on both routes denote where the most robust but costly MSO mode is selected for a specific WP interval (MEC), and the cyan ship pose shows where the ship has this mode active during its voyage in order to reduce the expected costs of grounding due to the TTG simulations. All other waypoints are given their original colors when using the most economical MSO mode (PTO). These results show how e.g. WP10 of the yellow route and WP6 of the pink route are allowed closer to the nearby obstacles compared to e.g. Figure 6.12 (demonstrating the approach of previous works [41]), as the cost function now integrates and considers the ship dynamics.

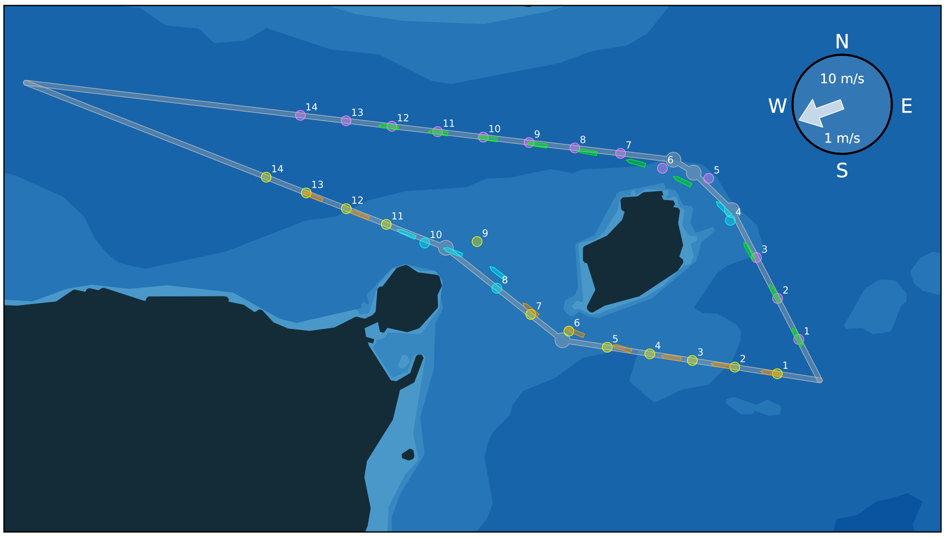


Figure 6.16: The resulting route trajectories using the complete cost function with TTG predictions.

Moreover, it may be noted that the MEC mode is still selected also for the line segment following WP9, for the purpose of demonstration - the MSO mode selection algorithm may utilize more advanced mode management mechanisms than simply choosing the most economical at each interval. It is also apparent that WP9 in this example is moved away from the nearby obstacle, leading to the normal PTO mode being selected. Though such mode switching generally is unwanted due to additional startup/switching costs, this outcome is included here for completeness only; a more sophisticated behavior may be tweaked and fine-tuned as desired.

Graphs of estimated (expected) grounding and fuel costs of each route, as well as the total accumulating costs along each alternative, are presented in Figure 6.17. Expected costs for grounding shows the μ_4 term of $C_{grounding}$ (6.13) for each waypoint, and are shown as blue bars. It may be noted that as the scaling coefficients for grounding events are constant in this work, the value of the blue bars may serve as proxy visualizations for the grounding probabilities P(G) experienced during the TTG simulations of each waypoint - i.e., a taller bar means a larger expected rate of grounding occurrences, which are noticeably different for each mode due to their inherent restoration capabilities. Expected (additional) costs for added fuel consumption with respect to the optimal path are denoted as the green bars on top. The three different MSO modes PTO, MEC and PTI are denoted by zero, halved

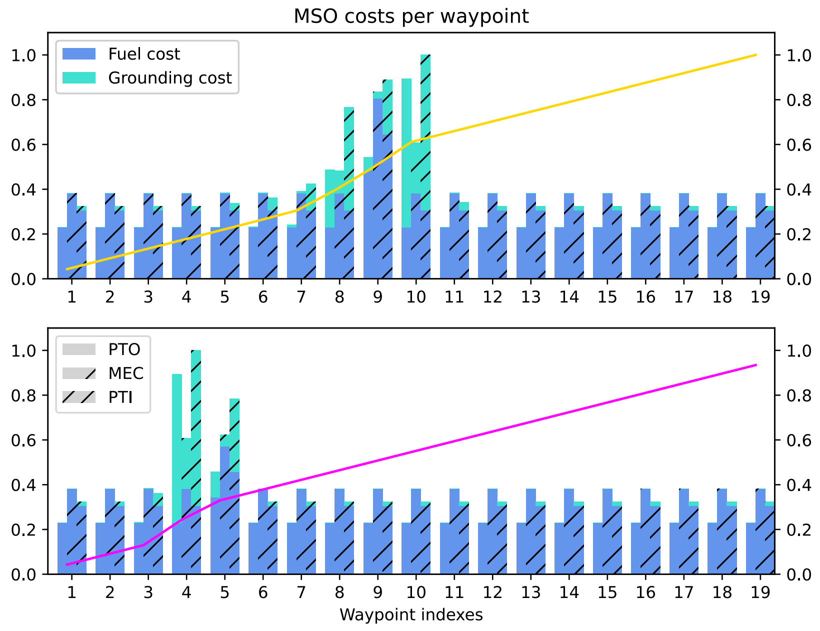


Figure 6.17: The weighted MSO mode costs of each waypoint interval and accumulated route costs.

and fully streaked bars, respectively. The total heights (sum) of these bars are the total expected costs of each MSO mode selection, for each waypoint.

It is apparent how each MSO mode are proportionately related to different fuel cost rates and grounding risk probabilities (due to different restoration rates), e.g. PTO has a lower fuel cost but also a larger grounding risk scaling associated with it, compared to that of MEC. During optimization, the mode with the lowest total cost is simply chosen for each route line segment between optimized waypoints. Note that the cost coefficients used in this work are ad hoc for a proof of concept, and are consequently only meaningful relative to each other. Thus, both Y axes are normalized between 0 and 1.

Definite indications of increased grounding risks and thus expected costs are clearly visible for WP 6, 7, 8, 9 and 10 for the yellow path, and WP 3, 4, and 5 for the pink path. This is in line with the visual information shown of the environment in Figure 6.16, i.e. the nearby grounding obstacles affect the costs as expected. One may also note that despite being as close to the obstacles as the mentioned points, WP 11 of the yellow path and WP 6 and 7 of the pink path are not affected in the same way, due to the general direction of the TTG predictions as a result of the given disturbances. Moreover, there is a noticeable difference between the expected fuel cost of WP 9 in the yellow path compared to its two neighbors. This also corresponds to the visually apparent location shift of the waypoint, in which the increased fuel costs of moving the waypoint in this situation were less expensive than the expected grounding costs for this specific interval. One possible explanation for this result may be that the expected TTG for this interval is less than the shortest minimum time required for all available restoration events, which significantly increases the grounding risk for that initial waypoint location.

The yellow and pink lines are the accumulating costs of each respective route, used to select the most efficient route. Ultimately, the pink route with its resulting green trajectory was chosen due to the lowest total expected cost across the entire (predicted) simulation run. This result shows how the fastest route may not always be considered the most cost-effective within a specific environment and set of conditions, and thus a slightly longer but more effective and/or safer route is generated and selected as the optimal choice.

6.4 Discussion

It is recommended that the methodology and proof of concept presented in this work should be implemented and tested in a practical implementation in future research. Moreover, testing of different combinations of various PSO hyper-parameter settings, number of particle candidates and number of iterations should be investigated and assessed in order to achieve improved performance.

The structure and tuning of specific terms in the complete cost function have a significant impact on the resulting solutions. In fact, the behavior or final waypoint distribution across each route alternative is by definition entirely dependent on both the cost function formulation and its inherent weighting coefficients. As such, the choices made with respect to the individual terms of the cost function must be thoroughly assessed. The sub-parts of (6.10) may in future research be weighted dynamically with respect to the ship state or predicted states. However, C_G is in this proof of concept given a constant value for simplicity.

The C_{path} term (6.12) consists of only path-related costs, which keep the waypoint distribution close to the original route alternative (the first term) as well as distributing the waypoints evenly across the full length of the original route. The values of both the exponents and its weights (μ_1, μ_2) is however highly flexible, and may be adjusted to accommodate various levels of strictness with respect to the path following aspect of the cost function as deemed most fitting by the human operators. Furthermore, it may be noted that though the TTG simulations are time-dependent during the LOPP simulations, C_{path} is not. This is a deliberate choice made in order to enable utilization of parallel computing techniques, due to each waypoint being fully independent from all other variable waypoints during optimization as a result of only using the ideal (static) reference waypoints $\boldsymbol{\varpi}^{ref}$ in this cost term. However, it may be argued that the PSO optimization instead may be structured such that the cost of each WP is dynamically calculated with respect to the best known costs of each PSO WP neighbor instead of the static references. Though this structure is not parallelizable, it may potentially achieve even more optimal waypoint distributions in future works.

Next, the $C_{grounding}$ term (6.13) is comprised of both the exponential anti-grounding costs as well as the weighted TTG-based accumulative probabilities of grounding if a LOPP event occurs. Though these terms are quite different in their form and the resulting effects of each consequently are difficult to compare directly, both are considered necessary to formulate as such in order to achieve desired behavior. The first term is included solely to serve as a strong barrier function for the purpose of extra safety, which may override any insufficient tuning or if any unexpected or unaccounted for events may occur. Thus, the term is exponentially defined, even though the resulting costs close to grounding obstacles are difficult to define explicitly or compare to more practical probability- and expected costs-based terms. This formulation may also be thoroughly examined in later research.

Similarly to the path-related cost formulation, the C_{mso} term (6.14) is based on fuel consumption with respect to the next ideal waypoint, as opposed to a perhaps more intuitive parameters such as total distance traveled, or time. This somewhat indirect form is also chosen in order to enable parallel computing, eliminating the need to include the variable waypoints during optimization, which vastly reduces the computational complexity. It is argued that though the resulting cost is not accurate in terms of actual fuel consumption estimated during the voyage, it is a useful measure of how much *extra* consumption is required for any waypoint change with respect to the ideal route alternative. This is considered appropriate within the scope of this study, due to the pre-computed reference path being assumed near-optimal.

Note that the implementation and cost formulation in this work do not include any additional cost terms for switching of MSO modes, nor any considerations of time delays or other time-dependent variables related to e.g. cold-starting an engine. These factors were not handled in this proof-of-concept study, but are however considered natural and appropriate additions to further research efforts or industrial applications.

There are many uncertainties related to both the measurement of states and predicting future states, and the models used to calculate these states. The management of such uncertainties is an important consideration of optimization problems such as the one presented in this work. In general, one may add additional safety margins to mitigate potential damages if the accumulated errors due to uncertainties lead to an accident. In addition, the models used for environmental factors were kept simple in this study. Future work should also include and implement more comprehensive and accurate models for calculation of environmental forces or physical effects, such as varying ocean depths and disturbance dynamics (winds, waves and currents) in order to reduce the amount of uncertainties present in the system.

Lastly, the results as presented in Figures 6.16 and 6.17 are highly subject to the tuning of the cost function coefficients, and should be acknowledged as such. Moreover, the results are assessed and validated by human evaluation of the behavior of simulated trajectories in a challenging scenario, which is subjective and subject to bias. The environment and cost function weights of this study are to a large degree chosen or established in order to show-case interesting behavior relevant to the proposed methodology, and is consequently biased toward this particular configuration. The method for tuning and assessment should thus be comprehensively investigated in future works.

6.5 Conclusion

This work proposes a methodology and proof-of-concept simulation study which utilizes PSO for simultaneous selection of machinery operational modes combined with waypoint re-planning based on grounding risks calculated from spatial distances and "time-to-grounding" simulations, such that both safety and efficiency may be considered more accurately during optimization compared to previous works. The results show that a slower route with respect to time and distance traveled may still be considered more cost-effective in terms of expected costs when also recognizing grounding risks along a route.

There are nevertheless several limitations that could be addressed in future work. As mentioned, the method could be extended for collision avoidance with dynamic obstacles and following the traffic rules at sea. Moreover, while the choice of PSO as an optimization engine is effective, other methods such as genetic algorithms should be considered as well. Although it is straightforward to define safety margins to account for uncertainty in models and input data, a more systematic method for determining the uncertainty levels and setting the safety margins would be useful. For an industrial implementation, hazard analysis should be used to obtain a more complete overview of the scenarios that potentially can lead to accidents, and to get more insight into which factors may affect the risk, and how. The proposed framework fully supports the implementation of a more comprehensive risk model, and is thus considered a promising approach to serve as the foundation to future works on joint machinery management and autonomous navigation.

6.6 Appendix: The Level-1 route planning algorithm

The following is a summary of concepts from previous works [37]:

A simple path planning algorithm for constructing a tree of possible route alternatives between two waypoints is presented in Algorithm 8. The algorithm is given a set of grounding obstacle polygons G, a safety distance Δd_s, an initial starting waypoint σ, and a single end target waypoint χ to which a path with several potential route alternatives is to be planned. Figure 6.18 shows an example in which a vessel intends to navigate around a collection of smaller isles, i.e. the set of red grounding obstacles G. The start point σ is represented by the vessel hull in white, and the end point χ is denoted by the green disk. The initial route path ρ intersecting G

Algorithm 8 PlanRoutes

```
Input: grounding obstacles $G$, safety distance $\Delta d_s$,
       start point $\sigma$, end point $\chi$
Output: binary tree $R$ of alternative routes from $\sigma$ to $\chi$
  procedure PLANROUTES($G, \sigma, \chi$)
      $H \leftarrow$ convex hulls of all polygons in $G$
      $I \leftarrow$ dilate $H$ by $\Delta d_s$
      $J \leftarrow$ spatial unions of all polygons in $I$
      $K \leftarrow$ convex hulls of unions $J$
      $\rho \leftarrow$ straight line segment from $\sigma$ to $\chi$
      $R \leftarrow$ new tree of line nodes with root $\rho$
      while $\exists P \in K$ intersects $\exists \rho \in R$ do
          $P \leftarrow$ largest intersecting polygon
          $\rho \leftarrow$ remove intersecting line from $R$
          $V \leftarrow$ visible vertices of $P$
          $\Lambda, \Gamma \leftarrow$ group $V$ into left and right wrt. $\rho$
          $\lambda, \gamma \leftarrow$ vertices of $\Lambda$ and $\Gamma$ farthest from $\rho$
          $\delta \leftarrow$ start point of $\rho$
          $\alpha_{1,2} \leftarrow$ linear line segments from $\delta$ to $\chi$ via $\lambda$
          $\beta_{1,2} \leftarrow$ linear line segments from $\delta$ to $\chi$ via $\gamma$
          $R \leftarrow$ add $\alpha_{1,2}$ and $\beta_{1,2}$ as new line nodes
      end while
  end procedure
```

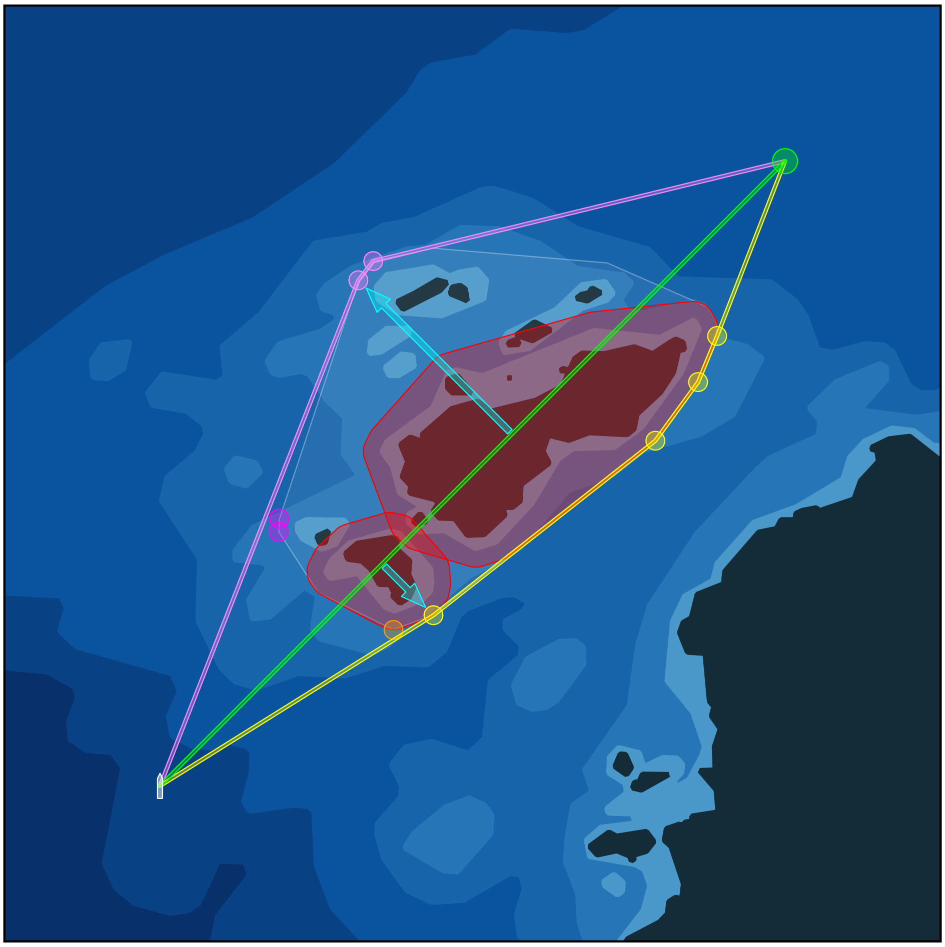


Figure 6.18: Path planning end result visualization of two alternative routes around an obstacle.

is shown as a green line from σ to χ. In this example, G is defined by extracting all nearby areas of seabed depths $< 10\,\text{m}$.

An initialization phase of six steps sets up the algorithm before the main loop is initiated, and consists of the following. The convex hulls H of all polygons in G are computed by accessing the Shapely property **convex_hull**, and the resulting new set of polygons H are subsequently dilated by the safety distance Δd_s (here defined as 50 m), using the Shapely method **buffer** to produce the polygon set of I. The next step calculates the spatial unions J of all polygons in I using the Shapely method **unary_union**, such that any overlapping polygons are merged, producing the highlighted convex polygon. The convex hulls K of J are lastly computed similarly to the first step, yielding the final set of polygons to be used in the main loop. Lastly, the initial green line segment ρ is defined by the start point

σ and the end point χ, and a new binary tree R with ρ as its root node is created.

After initialization, the main loop of the algorithm identifies the largest (if any) polygon $P \in K$ that intersects with any line segment $\rho \in R$ and extracts all *visible* vertices V of P, filtered by line-of-sight checks. Next, these vertices are split into two sets of *left* and *right* (Λ and Γ, respectively) based on their positions with respect to the line segment ρ. These are shown in Figure 6.18, given the colors pink and magenta, and yellow and orange, respectively.

The vertices with the maximum distance from ρ in each group (shown in Figure 6.18 as cyan perpendicular arrows from ρ to each respective vertex) are selected as the new intermediate route waypoints λ and γ, i.e. the minimum distance required to circumnavigate the visible part of the obstacle P at each iteration. These waypoints shown in yellow and pink are used to construct two separate splines of straight lines α and β consisting of two linear line segments each, from σ to χ via λ and γ. These new line segments are subsequently added to the root node of the R tree, leaving two new leaf nodes of line segments sharing the same end target point at χ. If any of the line segments in the resulting tree intersects with any polygon P of K, this process is repeated for that particular line segment, potentially creating more branching nodes along its respective route alternative.

The end result of the algorithm is presented as the pink and yellow line segments with several intermediate waypoints, generated by repeated iterations of the algorithm loop. These path alternatives may subsequently be used by other navigational optimization schemes, e.g. to select the optimal path with respect to resource consumption or time.

Part IV

Concluding remarks

Chapter 7

Conclusion

This chapter is comprised of three concluding sections. Section 7.1 presents a summary focusing on differences and similarities between each contribution, and Section 7.2 discusses each contribution with respect to the proposed research questions and future work. Some final concluding remarks are presented in Section 7.3.

7.1 Summary

This thesis presents a collection of contributions which serve to increase the knowledge within the field of risk-aware decision-making and control of autonomous ships. Three individual parts within this area constitute the main matter: Part I describes an ENC API called *SeaCharts*, which later has been adopted and used by other research and projects. Part II focuses on using MPC for online risk-aware path planning during both normal conditions and machinery faults, and a methodology for transforming the results of an STPA into an optimal control problem to be solved by this MPC approach is presented. Part III considers PSO as an alternative to MPC for path re-planning, leading to plans that can be executed by established guidance algorithms and controllers.

Chapter 2 is considered an independent and more general contribution, that is separate but still utilized by all of the subsequent chapters. Though the SeaCharts API in practice was developed in parallel with some of the other contributions, the published work and resulting API is a standalone product which is not limited to just being a foundation for the rest of this thesis. The chapter discusses its implementation, and demonstrates example usage for a wide range of various applications. The SeaCharts ENC Python package is the end result of this contribution, and has been employed by several works since its development, e.g., [29], [36], [43], [146].

With respect to the chapters presented in Part II, there are intentionally quite a few similarities between Chapter 3 and Chapter 4. The purpose of Chapter 3 is to demonstrate a proof of concept for autonomous ship navigation with anti-grounding and resilience to machinery faults using MPC as a foundation for future work, while the goal of Chapter 4 is to validate the proof of concept by providing a systematized methodology which can be used to formulate a numerical optimal control problem to be solved by the MPC algorithm previously proposed. Specifically, Chapter 3 emphasizes the mathematical definitions and algorithms in order to implement MPC, which the subsequent chapter references directly. It is also heavily focused on demonstrating the considered machinery failures, and the effect they have on the MPC path planner during such scenarios. Moreover, the grounding obstacles are in Chapter 3 approximated to be strictly circular for a simplified proof of concept. In Chapter 4 however, the grounding obstacles are general polygons, and several sections of the chapter are dedicated to the discussion related to this aspect in addition to the general focus on STPA and operational safety. While the result of Chapter 3 is a demonstrated proof of concept using MPC for dynamic path planning during machinery faults, the result of Chapter 4 is a novel methodology which attempts to bridge the gap between qualitative risk analysis and numerical optimal control. Chapter 4 is ultimately considered the natural next iteration of the proof of concept demonstrated in Chapter 3.

Chapter 5 of Part III considers an alternative approach to that of the previous chapter, by investigating the use of a sampling-based optimization method (PSO) for an analogous control problem, for the explicit purpose of categorical comparison. The similarities in results between Part II and Chapter 5 are thus straightforward, and the comparison between the two is indeed the core topic throughout. The differences lies within the approach used to achieve the same autonomous behavior, which again, is the very purpose of Chapter 5. It is demonstrated that the two approaches have different strengths and weaknesses: The MPC approach is generally faster and more robust if a smooth dynamic model is available. PSO is however more suitable for non-smooth and mixed-integer problems, and may lead to more flexible decision-making capabilities. This is considered the main objective of Chapter 5, and the result is a demonstrated proof of concept for dynamic path re-planning which shows that behavior similar to previous works based on MPC may also be achieved using this method. Nevertheless, it is important to consider the available processing power, as the cost function used by the PSO may quickly become expensive based on the chosen (or required) hyperparameters of the solver.

Chapter 6 is the main contribution of Part III, and is along with the ENC API considered one of the two main end results of this thesis. The purpose of this chapter is to illustrate how the versatility of the PSO method may be utilized to achieve higher levels of risk-aware decision-making. Here, model-based predictions are used within the sampling-based PSO to predict grounding hazards along a receding horizon, and this flexibility is acknowledged as one of the main advantages of PSO. The results show that the behavior of the autonomous ship following the path generated from this combined approach is in line with the behavior typical of human operators. It is ultimately proposed that future works investigate the feasibility of using complete MPC algorithms as individual cost sub-processes weighted within in a PSO approach, in order to further increase the capabilities of high-level risk-aware decision-making.

7.2 Discussion on research questions and future work

RQ 1 How can the utilization and visualization of ENC data be made fast and user-friendly in the research and development process for autonomous ship control and decision-making?

Chapter 2 contributes towards RQ 1 by providing an open-source API which enables researchers to access, visualize and utilize ENC data to more efficiently develop algorithms and methods for mission planning and control of autonomous ships in future works.

The ENC API package is however considered only a first draft of what could become a larger software tool for research on autonomous ships, if it is adopted and continuously developed. Examples for future work on the API are additional utility features such as standardized shape drawing methods or map scale legends, adding support for more map features such as fish farms, including navigational lights, indicators or visualization for traffic densities and COLREG compliance, applications for collision avoidance, internal implementations of various guidance and control algorithms for benchmarking, other map projections, simplifying the installation process, as well as expanding the roster of supported spatial database formats. The list of possibilities for further utility development and additional extensions is undoubtedly longer than this.

RQ 2 How can autonomous ships **a)** be proactively controlled such that grounding hazards are predicted and avoided on a voyage while subject to uncertain disturbances, and **b)** be prepared for unexpected machinery faults such as loss of steering or propulsion?

Chapter 3 contributes towards RQ 2 by demonstrating a proof of concept of dynamic risk-based navigation algorithm with inherent emergency management capabilities, which avoids grounding hazards both during normal conditions and when machinery faults unexpectedly occurs, such as impaired steering and loss of propulsion.

This chapter served as a simple proof of concept which laid the groundwork for later research as presented in this thesis. Nevertheless, there are several additions or considerations not covered here which may be investigated as future work. First, the method used simple circles as grounding obstacles, and later chapters utilized polygons for distance calculations. However, there is still unexplored potential in using such circles, given their rapid distance calculations with respect to other points compared to more complex polygons. It may be interesting as a proposal for future work to transform large polygons into smaller tightly packed circles, and e.g. apply a line-of-sight type of filter to eliminate significant amounts of unnecessary data in order to vastly reduce the computational complexity.

Another proposal is to replace the ad hoc risk function presented in the chapter by probabilistic risk models. This is somewhat related to the improvements implemented in Chapter 4 and Chapter 6, in which more complex risk-aware decision-making may be achieved by utilizing suitable models for risk. In a similar vein, there is virtually no limit to the number of additional cost terms (risk-based or otherwise) which may be added to the control problem. One example may be other formulations analogous to the AMM sub-system considered in Chapter 6.

RQ 3 How can the results of qualitative risk analysis methods such as STPA be utilized and systematically structured into a numerical optimal control problem for autonomous ship navigation, such that grounding hazards are identified, mathematically formulated and avoided through numerical optimization?

Chapter 4 contributes towards RQ 3 by developing a step-wise methodology which takes the results of an STPA and transforms them into mathematical equations and inequalities, which subsequently are structured into a numerical optimal control problem to be solved by an MPC approach.

Bridging the cap between qualitative risk analysis methods and quantifiable control problems is a rather unexplored challenge. Thus, the methodology is considered a foundation for future work, rather than a complete framework for constructing e.g. MPC formulations out of STPA results. It is moreover noted that although MPC based on nonlinear programming was used for demonstration in this work, other solvers or algorithms may readily be used to handle the resulting cost function, such as e.g. PSO. The method is moreover applicable to other applications than autonomous ships.

Another obvious challenge is the terminology and/or wording used regarding the concepts for costs or risks. It will be different between e.g. operators, system designers and corporate managers, and it is often difficult to reach consensus. It is suggested that global efforts should be made to standardize and define such critical terminology in future work.

Furthermore, the proposed cost term structure resulting from the methodology should be investigated and tested extensively. Similarly to the previous chapter, the potential for other improvements with respect to the resolution, amount and quality of the environmental data may be examined in order to achieve faster computations without loss of generality. The additions previously mentioned for Chapter 3 may in general also be included as recommendations for future work here as well.

RQ 4 How can the planned routes generated by established voyage planners be modified to give a navigation path during complex real-time transit conditions, such that unexpected hazards and changing environments are accounted for en route?

Chapter 5 contributes towards RQ 4 by presenting a more flexible online path re-planning method utilizing PSO, which may employ the use of established path following or guidance algorithms after re-planning. This alternative method based on PSO is compared to the previous MPC approach of that used nonlinear programming, and demonstrates analogous behavior, enabling the use of more complex (e.g. non-smooth as well as combined continuous and discrete) cost functions in future works.

Future work based on this chapter may include the parameterization proposition as discussed in the contribution, as well as the possibility of using solvers based on e.g. multi-objective evolutionary algorithms [147] or other sampling methods in place of PSO.

Another important consideration is to assess the quality of the solutions generated by the PSO algorithm, as optimal solutions are not guaranteed. Both the initialization of the first (random) solution as well as the PSO algorithm (meta-)parameters are key to achieving satisfactory results. It is recommended to examine other cost formulations than the ones discussed in this work, and to look into distinct planning algorithms which can generate adequate initial solutions when using PSO.

RQ 5 How can model-based predictions for ship dynamics and onboard machinery management systems be combined with a sampling-based path planning algorithm and established path following algorithms in order to reduce overall risks and expected mission costs?

Chapter 6 contributes towards RQ 5 by proposing an approach for combining and utilizing a model-based autonomous machinery management approach with the sampling-based and risk-aware PSO voyage re-planner method from Chapter 5 in the same cost function, which increases the ability of the resulting ANS to consider and optimize resource consumption and grounding risks through both MSO selection and route re-planning or selection.

As mentioned in the discussion of this contribution, it is suggested to apply and test the proposed method in a practical implementation. Similarly to the previous contributions, the tuning of cost coefficients plays an essential part in the performance of the algorithm. This tuning is largely dependent on the model (dynamics) of the system, and is usually different on a case-by-case basis. It is recommended that a systematic approach to this tuning process is developed in future works.

The structure of the presented cost terms are as noted for Chapter 4 highly subject to the specific application, and the choice of structure is explained in the discussion of Chapter 6. It is suggested to add additional cost terms for switching of MSO modes, which consider the physical time delays related to e.g. restarting an engine with respect to its state at that point in time.

Lastly, the effects of measurement and prediction uncertainty for internal states should be examined more systematically. For this purpose, additional safety margins may be added in various ways, more accurate or complex mathematical models may be utilized, and their individual effects on the overall system may be assessed.

7.3 Concluding remarks

The objective of this thesis was to add to the knowledge within the field of risk-aware decision-making and control of autonomous ships. A need for standardized software tools for rapid prototyping, simulation and evaluation of autonomous ships was identified, and an open-source API for ENC was developed. Proof of concept demonstrations for risk-aware decision-making and control of autonomous ships using MPC and PSO is presented, and a novel methodology for transforming the results of qualitative risk analysis into a numerical optimal control problem is proposed. The main result of the thesis is a dynamic joint path planning, route selection and AMM optimization algorithm, which applies simultaneous resource consumption and risk minimization to achieve safer and more efficient autonomous navigation.

It is suggested that further work within the field of autonomous ships may employ the presented ENC API for efficient experimentation, testing and assessment, and that the proposed methodology and risk-aware algorithms may be used as a foundation for the development of standardized optimal control approaches for practical and industrial implementations of decision-making systems for autonomous ships. Specifically, a promising direction may be to investigate the implementation and performance of a hybrid ANS, in which the top-level PSO planner accepts as input and utilizes the real-time optimal solutions of internal MPC solvers as distinct decision variables. Moreover, this thesis presents promising algorithms, methodologies and software which may assist further research considering risk management in conjunction with autonomous systems and optimal control. Ultimately, the proposed methods show significant potential for risk-aware decision-making and control of autonomous ships, and may serve as building blocks for future research efforts.

Appendix A

Risk-based control system for autonomous ships

A.1 Introduction

Although conventional ships have control systems for navigation, maneuvering, and power management, they are designed to rely on human input and supervision onboard. For example, Dynamic Positioning (DP) systems are used to maintain a ship´s position or to maneuver the ship at low speeds with good accuracy. Nevertheless, a human operator must specify the mission and be ready to take over control if the automatic system fails. Power management systems (PMS) also have a high degree of automation to control electric power generation, power distribution, and prevent blackouts on ships.

There is currently no automation system that monitors or controls the complete ship's operation, replacing the crew onboard. For example, engine control systems may monitor the engine and shut it down if there is a failure, even if this compromises the safety and integrity of the ship. An example is the Viking Sky incident, where the diesel generators were automatically shut down due to low lubrication oil levels in a severe sea state, which led to a complete blackout and nearly caused the cruise ship with almost 1400 people onboard to ground in storm conditions [148].

In general, for a ship to operate safely and autonomously, its control systems must be able to assess risk (currently the task of the crew onboard conventional ships). Hence, [7] propose a control system framework that can assess and manage risk, replacing some of the cognitive judgments that the crew would normally make while sailing to improve the autonomous ship´s decision-making. [149] describe how risk analysis methods can be integrated with control systems and identify four areas for implementing this. Another approach is further demonstrated in [150]. A risk model represented by a Bayesian Belief Network (BBN), which is based on a systems theoretic process analysis (STPA), assesses navigational risks for an autonomous cargo ship while sailing as part of a supervisory risk controller (SRC) for high-level control of the ship. This risk model provides information that can be used as a basis for selecting the control mode, machinery mode, and setting control objectives while sailing. [151], [152] presented a similar control system for autonomous underwater vehicles (AUVs) for under ice operations. In this case, the SRC was used to set the altitude set-point, velocity set-point, and control strategy such that the AUV could avoid collision while performing under-ice mapping with sufficient accuracy.

Relevant risk factors have also been discussed in [153]. A framework to identify navigational risk factors for autonomous ships is presented, but without any further application. [154] combine Failure Mode and Effects Analysis (FMEA) with evidental reasoning and Bayesian Networks to quantify the risk level of major hazards related to autonomous ships. [155] propose to use STPA to identify potential hazards for autonomous ships and discuss some methods for finding additional quantitative data to use in a risk model, but without building and using the model. STPA is also used in [156] for hazard analysis on autonomous passenger ferries. This work suggests safety controls to mitigate the identified hazards when designing the ship. [157] use STPA to develop a model to analyze safety and make design recommendations for autonomous vessels. [158] propose a framework to model the ship control structure, based on STPA that can be useful to describe the functionality of the system.

Risk models have also been used to predict the loss of AUVs during missions [159]–[161] and to manage uncertainty in these missions [162]. However, none of these models are connected or implemented as part of the control system. Other papers have discussed risk as part of collision avoidance, but use risk as a very general term and lack a direct link to risk analysis and risk modeling [131], [163]–[167]. Combining some selected risk aspects with Model Predictive Control (MPC) has also been proposed for collision avoidance systems [168], [169] and emergency management but the risk metrics that are used in these studies are not based on risk assessment and are simplified so that they can be used in an MPC application [38].

A quantitative risk model can provide good and useful information about an autonomous control system if it includes reliable information about the ship's position and its surroundings. One option is to use tools such as Simultaneous Localization and Mapping (SLAM) that can be used for AUVs [170]–[172] operating in areas where localization and mapping are challenging. Mapping the environment is unnecessary for autonomous ships because position data are available from global navigational satellite systems (GNSSs), such as position and speed measurements, and electronic navigational charts (ENC) are available. GNSS measurements are already used in control systems, such as in DP controllers to provide position and speed measurements. ENC data have been used in decision-making systems, such as path planners, for ship navigation [173]. The data can then be used directly in the planner, with limitations on extracting and presenting the data. To address these limitations, [37] developed an open-source application programming interface (API) to process and display the data with high accuracy and in short computation time. Their paper shows how the API can be used for certain tasks, such as path planning based on a dynamic risk optimization. A simple risk metric based on wind speed and direction, and the distance to land is used when planning the route.

Developing better control systems is an important step towards realizing autonomous ships, which in turn is expected to improve safety at sea [1], [174]. However, it is important to demonstrate that these ships are safe in operation to achieve approval from the authorities and public acceptance. This means that autonomous ships need to be tested in various scenarios and environmental conditions. Today, verification, validation, and certification in the maritime industry depend on type of ship and operation. On advanced offshore installations and ships, the ship and control systems are thoroughly tested through simulations, scale testing, sea-trials, and Hardware-in-the-Loop (HiL) testing. Extensive and thorough tests are necessary to get the systems approved by class societies and coastal states [175]. Suppliers usually test individual components on less advanced ships during commissioning and sea-trials.

The shift towards autonomous ships presents several challenges with respect to verification and testing. Both the complexity and criticality of the software systems increase. In addition, the control system interacts with a highly dynamic and unstructured operative environment, which causes the span of possible scenarios to become enormous. Autonomous systems typically use machine-learning software to some extent, which introduces its own set of challenges (see [176]). Therefore, there is a need for new methodology to formalize and scale the verification and testing efforts to new levels.

Several recent works have aimed to address these challenges. For example, [8] propose a test system for autonomous navigation systems (ANSs) and show how it can be used to verify the performance of a collision avoidance system. [177] present an Autonomous Simulation-based testing framework and show how it can be used to verify a collision avoidance system. [178] propose a quantitative evaluation method to evaluate obstacle avoidance methods for unmanned ships. These studies indicate that although the test systems work, they only work through testing a very limited part of the control system. They also lack a description of how the testing should be integrated into the design process for autonomous ship control systems.

To summarize the gaps identified in the current literature, it is necessary integrate risk with control systems intended for autonomous ships to improve its high level decision-making. In addition, these control systems need access to data from ENCs, and they need to be verified in a formal and systematic manner to ensure the necessary safety and performance. Hence, the overall objective of this work is to present a novel and interdisciplinary methodology to develop an SRC for high level control of autonomous ships that bridges risk modeling, optimization, ENC, and formalized verification to achieve safer and more intelligent performance of autonomous ships.

The proposed methodology is tested and compared to an existing conventional-manned ship for different coastal routes to assess how the SRC handles failures in the ship's machinery and propulsion system. The main scientific contribution is the demonstration of how the intelligence of an autonomous control system can be improved by combining thorough risk analysis and modeling, detailed data from navigational charts, and novel verification methodology. Compared to existing control systems, this new approach makes it possible to handle a wider range of operations and situations, which reduces the need for human intervention and supervision. Even though the application in this work is focused on autonomous surface ships, it is expected that the methodology will have relevance for other autonomous applications. A similar methodology might also be used to assist operators by providing additional decision support by assessing how the risk level changes leading to safer ship operations.

The rest of this work is organized as follows. Section A.2 presents the methodology for building and setting up the controller. Section A.3 describes the case study. Section A.4.1 and section A.4.2 present the results from the case study. Sections A.4.3-A.4.7 discuss how risk can be included in control systems, how to use ENC data, how to test the system, and it also describes some uncertainties in the controller and risk model. Section A.5 concludes this work and outlines further work towards highly autonomous ships.

A.2 Method

The SRC controller is developed through a five-step process, as shown in Figure A.1. The SRC enables the controller to make risk-informed decisions that emphasize both safety and efficiency when operating the ship. These decisions can (for example) determine the ship's operating machinery mode, control mode, or the speed reference for the proposed control system.

The ship and the operation are first described in detail and analyzed using an extended STPA to identify hazardous events that need to be included in the risk model. Thus, the STPA results are used as the basis for building the online risk model in step 2, which is represented here in terms of a BBN. The justification for using STPA combined with BBN is presented in [7]. For situation awareness, the risk model uses data from the ship's sensors and the control system to assess the current conditions. The ENC module is used to extract data from navigational charts with information about the area surrounding the ship. The ENC model is set up in step 3 based on the design requirements to provide the necessary data to the risk model and SRC. The SRC is then developed in step 4 based on the requirements identified in the system analysis and the STPA (step 1), and using data from both the risk model and ENC. Finally, the controller is verified against the performance requirements using the automatic simulation-based testing methodology.

A.2.1 Step 1: System description and STPA

To setup and build the control system, the ship and operation have to be described and analyzed, such as in terms of a CONOPS (concept of operations). This starts

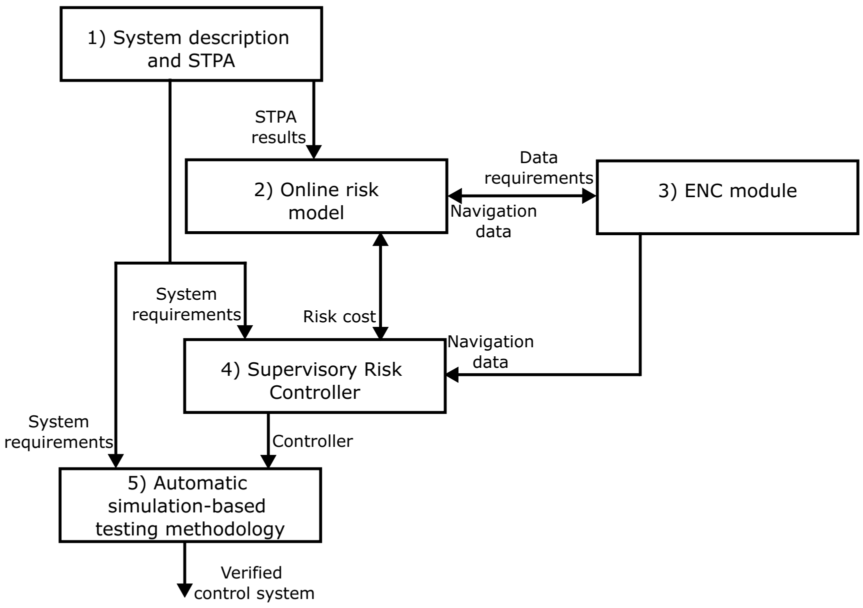


Figure A.1: Methodology flowchart

by clearly describing the ship, how it is controlled, its technical condition, and characterization of the operation that it is used for. In terms of control, it is important to know what type of controllers the ship has or will have, how they are connected, and their different responsibilities. Human operators or supervisors (e.g., onshore in a control center) must also be described with information about how they can control or affect the ship. Describing the ship's operation requires a clear statement of why and where the ship is sailing, as well as its operating modes. For example, a coastal cargo ship sailing along the Norwegian coast may be very different to a passenger ferry sailing between islands in the Mediterranean Sea.

The decisions or control actions relevant for the SRC must also be specified. These are important to consider because they are the only options for the SRC to affect the control of the ship. After describing the ship, STPA can be used to identify potential hazards, causal factors, and safety constraints. The STPA follows the steps defined in [11] but is expanded to also explicitly consider the consequences of the hazardous events and system-level hazards as follows:

a) Define the system
b) Identify hazardous events and system-level hazards
c) Identify unsafe control actions (UCAs)
d) Develop loss scenarios
e) Analyze consequences

The description of the ship can be used as a basis for the first step of STPA, and is a basis for defining the control structure and assigning responsibilities to the different controllers in the system. The next step is to identify hazardous events and to identify UCAs. These are subsequently described in loss scenarios that may lead to UCAs. Scenarios also include how decisions, such as selecting the wrong control mode or using machinery systems with failures, can lead to UCAs. The decisions are included in the same way as risk influencing factors (RIFs). The final part is to describe and classify the potential consequences of the hazardous events (e.g., through cost estimations).

A.2.2 Step 2: Online risk model

The online risk model is built based on the STPA results and follows the emerging top-down structure, like the results of the analysis, as shown in Figure A.2. The BBN has six main types of nodes:

- Consequences
- Hazardous events
- System-level hazards
- UCAs
- RIFs
- Decisions

The end node in the BBN is the consequences. These are caused by the hazardous events, under given conditions. The hazardous events are caused by one or more

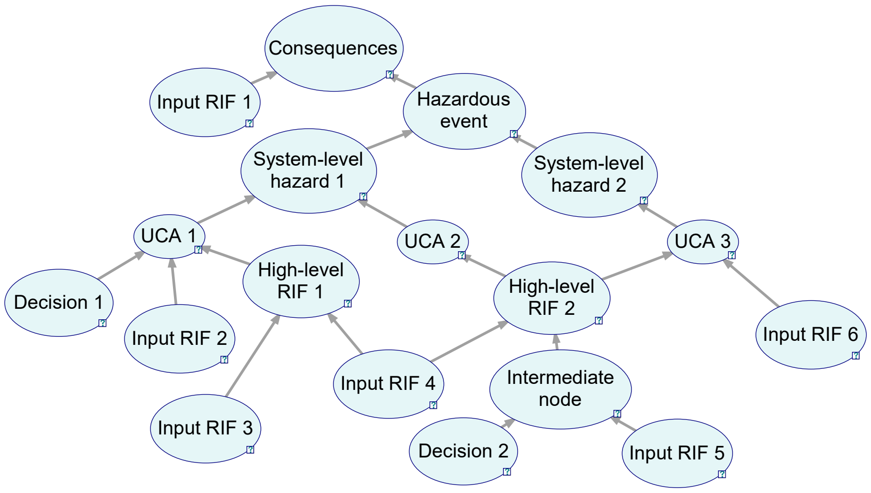


Figure A.2: Example BBN structure, showing how the STPA is linked to the BBN and how different nodes are related (adopted from [7]).

system-level hazards identified in the STPA. The next is the UCAs that lead to system-level hazards. UCAs get an input from RIFs that describe the loss scenarios and the conditions where hazardous events have negative consequences. RIFs can be both high-level RIFs (H-RIFs) and input RIFs (I-RIFs), as shown in Figure A.2. For a more detailed description of mapping STPA results to a BBN, the reader is referred to [7] or [150]. For a detailed description of BBNs in general, the reader is referred to [179].

The BBN is converted to an online risk model by deciding how to update the BBN as the ship sails with online information. This links specific nodes to sensors and systems onboard the ship, and then decides which data are necessary, including the ENC module. Decisions made in the SRC are also included in the BBN to model how they affect the risk picture and consequences. The BBN can also have intermediate nodes to group I-RIFs and decisions to reduce the number of nodes that are connected to each H-RIF. This is more important for larger and more complicated BBNs.

A.2.3 Step 3: ENC module

The ENC module extracts and manipulate data from electronic navigational charts. These data are necessary in the risk model to describe the surroundings and conditions around the ship. The ENC module is based on the open-source Python package SeaCharts [37]. This package use FGDB 10.0 data sets with 2D data of the relevant areas. These are then processed as the application starts, so that they can be stored as shapefiles, where only the relevant depth layers and land areas are stored. This allows for much faster processing because it reduces the time necessary for computation and/or querying. The data is stored as polygons for various water depths and land areas. The stored shapefiles can then be queried to find the distance to points where the ship can collide or ground, and assess how much space the ship needs to maneuver.

The ENC module is set up by first loading the necessary maps for the relevant area. The next step is to define and load relevant layers for the ENC module, depending on the ship and data needed in the control system. This is achieved by defining the minimum water depth that the ship must maintain for safe sailing. To avoid unnecessary quantities of information in the risk model, a planning horizon is set in the ENC to decide how far the ENC should look ahead of the ship. This limits the data size that the ENC must query and reduces the computation time. Connecting the ENC module with the risk model is done by connecting the relevant nodes and updating them with data from the ENC, such as distance to land and shallow areas, combined with position and speed measurements from the GNSS system.

The current ENC module does not account for navigation markers, as this is not currently implemented in the SeaCharts package. This is discussed more in Section A.4.5. For a detailed description of the package and all functions, the reader is referred to [37].

A.2.4 Step 4: Supervisory risk controller

The controller is set up as an SRC to make high-level decisions or set control objectives. One option is to use costs as a means for implementing the inputs from the risk model into the decision-making. For other potential options, see [149].

For an autonomous ship controller, decisions can be made based on four costs: the risk cost from the online risk model, fuel cost based on the expected fuel consumption, operation costs (other than fuel), and the cost of not starting new missions. The total cost is calculated using Equation A.1 as a function of the decisions, d, such as setting the speed reference and deciding how the machinery should be operated:

$$C(d) = R(d) + F(d) + O(d) + L(d) \tag{A.1}$$

The risk cost, $R(d)$, gives the expected cost from the consequences described in the risk model and account for factors such as weather conditions, ship speed, traffic conditions, etc. Fuel cost, $F(d)$, describes the expected cost of fuel of operating the ship under the current conditions. Operation cost, $O(d)$, describes the costs of operating the ship, outside of fuel cost, such as maintenance, insurance, and manning costs. $L(d)$ describes the potential loss of future income caused by the time used. The cost function is set up such that fuel cost, operation cost, and potential loss of future income increase if the ship takes a longer time to reach the final waypoint.

The controller checks each possible set of decisions to find the set with the lowest cost. The decisions can vary depending on the ship and can include selecting what machinery mode to use, how the ship should be controlled, and which speed reference to follow. The SRC configures the control of the ship according to the set with the lowest cost.

A.2.5 Step 5: Automatic simulation-based testing methodology

Step five verifies the controller against a set of design requirements related to safety and efficiency. The verification process is performed using the automatic simulation-based testing methodology from [177]. This methodology automatically runs simulations where the vessel is sailing along its planned route, while varying scenario parameters. The methodology formulates requirements using the Signal Temporal Logic (STL) formal specification language, which enables automatic evaluation of the simulations against the requirements [180]. The result of evaluating a simulation against an STL requirement is an STL robustness score that describes how robustly the requirement is satisfied. If the STL score is greater than zero, then the requirement is satisfied. If it is less than zero, then the requirement is violated.

The methodology selects the simulations to run from a test space that is defined by a set of scenario parameters with corresponding parameter spaces. The test space can, for example, be based on scenarios that are identified in the STPA [13], [35], [181] to test the controller in specific situations. A Gaussian Process (GP) model [182] is used to predict the STL robustness score as an unknown function of the test case parameters. The GP model estimates the expected value and the uncertainty of STL robustness over the entire parameter space of a test case. The GP model is iteratively updated by running simulations and observing the resulting STL robustness score. The estimates of the GP model are then used to adaptively guide the test case selection towards cases with low STL robustness or high uncertainty. This results in efficient coverage of the parameter space or alternatively efficient falsification if the controller does not satisfy the requirements.

The testing terminates in a verified state if the lower confidence interval of the GP is greater than zero for the entire parameter space. For example, using 99% confidence intervals, a verification would indicate that there is at least a 99% probability that the system satisfies the requirement for the entire test space of the test case. Alternatively, if a test case that does not satisfy the requirements is identified, then the verification terminates in a falsified state, returning the corresponding counter-example. For a more detailed explanation of the automatic simulation-based testing methodology, the reader is referred to [177].

A.3 Case study: Supervisory risk control of an autonomous cargo ship

The method for building the SRC is tested in a case study that simulates an autonomous ship operating along the Norwegian coast to assess how the SRC manages and controls the ship in comparison to an existing conventionally-manned ship. The first part of the case study will analyze how the SRC adjusts the speed and configures the ship to maintain control. This is then compared performance-wise to a conventional ship in similar conditions, using position and speed data from the ship navigation system. The second part will study how the SRC handles failures in the machinery and propulsion system.

In the case study, it is assumed that the chart and GNSS measurements are sufficiently accurate to be used in the control system. It is also assumed that the time necessary to start up machinery can be neglected. There are still some delays and thruster dynamics included, such that engines and generators cannot change the load immediately. This is deemed sufficient to show how the SRC functions. Some of the potential ways to include these aspects in the SRC will be discussed in Section A.4.3.

The ship simulation uses a simplified kinetic model without wave forces. This makes it easier to simulate and test the system, while it also changes the ship's movement such that the ship drifts more. This makes it more difficult to control the ship, especially in tight turns, without reducing the speed much more than conventional ships. Although the focus in this work is the design and testing of the SRC, it still provides sufficient results to show that the proposed methodology works.

A.3.1 Step 1: Describing the ship and operation

The autonomous ship that is considered in the case study is an 80 m long and 16 m wide cargo ship that is sailing along the Norwegian coast. Although the ship is operated unmanned, it has a human supervisor onshore that can monitor and take control remotely if necessary. The ship has an autonomous control system, as shown in Figure A.3, with an SRC as the high-level controller, an ANS to control the navigation, and an autonomous machinery management system (AMMS) to manage the machinery. The ANS has two ship operating (SO) modes: (i) DP and (ii) autopilot (AP), with a corresponding controller for each mode. The DP controller is used during low-speed maneuvering and station keeping, while the AP controller is used for transit at higher speeds. When the ship is operated in DP-mode, it utilizes the main propeller, bow tunnel thruster, and aft tunnel thruster to control the ship's speed, position, and heading. The AP controller uses the main propeller and rudder to control the ship.

The ship is equipped with a Liquefied Natural Gas (LNG) fueled main engine, a hybrid shaft generator (HSG), and two diesel generators. The HSG can be used as a generator to produce electricity when the main engine is used or an electric engine when diesel generators can be used to produce electricity.

The AMMS is used to control the machinery system depending on the machinery system operating (MSO) mode. The ship has three MSO-modes: power take out (PTO) mode, where the main engine provide propulsion and the HSG is used as a generator to provide electricity; power take in (PTI) mode, where the diesel generators produce electricity, and the HSG is used as an electrical engine to propel the ship; and the mechanical (Mech) mode is where the main engine provides propulsion and the diesel generators produce electricity.

The SRC is responsible for selecting SO-modes and MSO-modes. It also sets the reference speed for the ANS to follow.

The STPA in the case study is based on a workshop with 12 relevant system experts who identified UCAs for the autonomous cargo ship. The participants have 5-30 years of experience from academia and industry working with risk assessment, testing, verification and validation, marine technology and maritime operation, and ship control system design. The workshop where conducted over three sessions. The first two where used to identify UCAs that were discussed and processed by the participants in the third. The result from the workshop was a report sent out to the participants. The main purpose of the workshop was to not only identify how switching between different machinery modes can lead to insufficient power capacity and power losses but also to identify when the wrong SO-mode used by the ANS could lead to accidents.

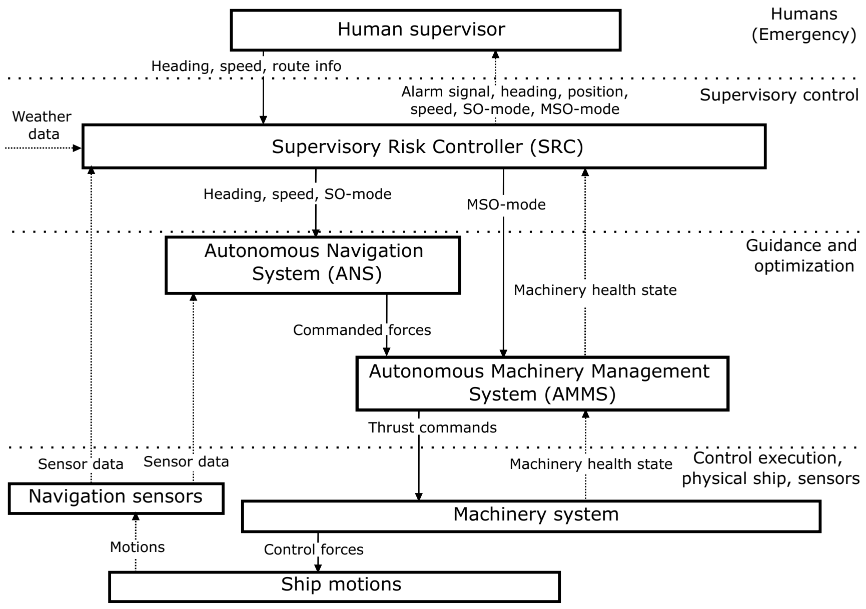

Human supervisor
Humans (Emergency)
Heading, speed, route info
Alarm signal, heading, position, speed, SO-mode, MSO-mode
Supervisory control
Weather data
Supervisory Risk Controller (SRC)
Heading, speed, SO-mode
MSO-mode
Guidance and optimization
Autonomous Navigation System (ANS)
Machinery health state
Commanded forces
Autonomous Machinery Management System (AMMS)
Thrust commands
Sensor data
Sensor data
Machinery health state
Control execution, physical ship, sensors
Navigation sensors
Machinery system
Motions
Control forces
Ship motions

Figure A.3: Hierarchical control structure (adopted from [150])

The STPA in the workshop considered a slightly different control structure with a remote operation center (ROC) that is responsible for planning, monitoring, and supervising the ship. The ANS and AMMS determine the SO- and MSO-mode, respectively, according to the sailing plan. An SRC in the control system was not included. The results from the workshop have therefore been developed further to account for the different ship control structure considered in this case study.

This case study assumes that the human supervisor plans the mission and the SRC then executes this plan. The human supervisor is also responsible for taking remote control of the ship if notified by the SRC. Selecting SO- and MSO-mode is now done by the SRC, and not the ANS and AMMS. The ANS controls the ship in either AP- or DP-mode depending on the SO-mode. The AMMS manages the machinery system according to the MSO-mode decided by the SRC. The AMMS also contains thrust allocation that computes individual thrust commands, based on the commanded forces from the ANS.

Since the workshop did not include an SRC, the control structure is modified to include this with the associated control actions. However, because setting SO-mode, MSO-mode, and the ship speed were considered when identifying UCAs in the workshop, the results can still be used with some modifications to account for the differences.

The SRC has a set of process variables that are used to make decisions, as follows:

- PV-1: Active MSO-mode
- PV-2: Available power and thrust
- PV-3: Machinery system status
- PV-4: Active SO-mode
- PV-5: Ship's navigational states
- PV-6: Weather conditions
- PV-7: Traffic conditions
- PV-8: Route information

The case study focuses on the following hazardous event (HE) and system-level hazards (H), as follows:

- HE1: The ship grounds or has contact with the seafloor
- H1: The ship violates the minimum separation distance to the shore
- H2: The ship sails in water that is too shallow

The workshop identified a total of 60 UCAs. However, including all these would make the risk model more complicated to build and evaluate. Therefore, the case study focuses on five different UCAs, as shown in Table A.1, to reduce the size and complexity of the risk model. These are chosen to have a good basis for specifying scenarios where the decision-making in the SRC, such as setting SO-mode or speed reference, can lead to hazardous events and identify RIFs that affect this.

Nine scenarios are defined to describe the situations that can cause UCAs and hazards, as presented in Table A.2.

Table A.1: Unsafe control actions

UCA	Description
UCA-1	A command is given to change MSO-mode to PTO when the health state of the ME is reduced
UCA-2	A command is given to change MSO-mode to Mech when the diesel generators do not function, or are unable to provide the rated power to the DC bus
UCA-3	A command is given to change MSO-mode to PTI, resulting in insufficient power for the main propulsion
UCA-4	A command is given to change SO-mode to transit/AP when the ship is in harbor/tight areas
UCA-5	A command is given to change SO-mode to maneuvering/DP when the speed is higher than the maximum maneuvering speed

Table A.2: Scenarios

Scenario	Description	UCA
SC-1	MSO changed to PTO because PTI delivers insufficient amount of power but the health state of the ME is reduced, leading to insufficient power production	UCA-1
SC-2	MSO changed to PTO because the extra power in Mech is not necessary but the health state of the ME is reduced, leading to insufficient power production	UCA-1
SC-3	MSO changed to Mech because PTO is not producing sufficient power for propulsion but the diesel generators fail or provide less power than expected, leading to insufficient power on the DC bus	UCA-2
SC-4	MSO-mode is changed to from PTO to PTI due to an underestimate of the power necessary, leading to insufficient power to the ship	UCA-3
SC-5	MSO-mode is changed to from Mech to PTI due to an underestimate of the power necessary, leading to insufficient power to the ship	UCA-3
SC-6	SO-mode is changed to transit while still in harbor due to inaccurate/incorrect measurements of the ship states	UCA-4
SC-7	SO-mode is changed to transit while still in harbor due to wrong understanding of the area around the ship	UCA-4
SC-8	SO-mode is changed to maneuvering with too high speed due to faulty speed estimates/measurements	UCA-5
SC-9	SO-mode is changed to maneuvering with too high speed due to a wrong limit set in the controller	UCA-5

The extended STPA in this work also considers the consequences from the hazardous event and the expected resulting costs. The consequences are divided into damage to own ship, damage to others´ property, and harm to humans. Consequences are classified as either severe, significant, minor, or no consequences [183]. Fatalities or serious injuries to humans or extensive damage to the ship or other ships/objects where assistance is necessary are considered severe consequences. Less serious/minor injuries to humans and damage that needs repairs outside of planned maintenance are considered significant consequences. Insignificant or no injuries to humans and damage that can be fixed in the next planned maintenance are considered minor consequences. Severe consequences cost 4 550 640 USD, significant 455 064 USD, minor 45 506.4 USD, and no consequences lead to zero cost. The costs are estimated based on [184], [185], and [183].

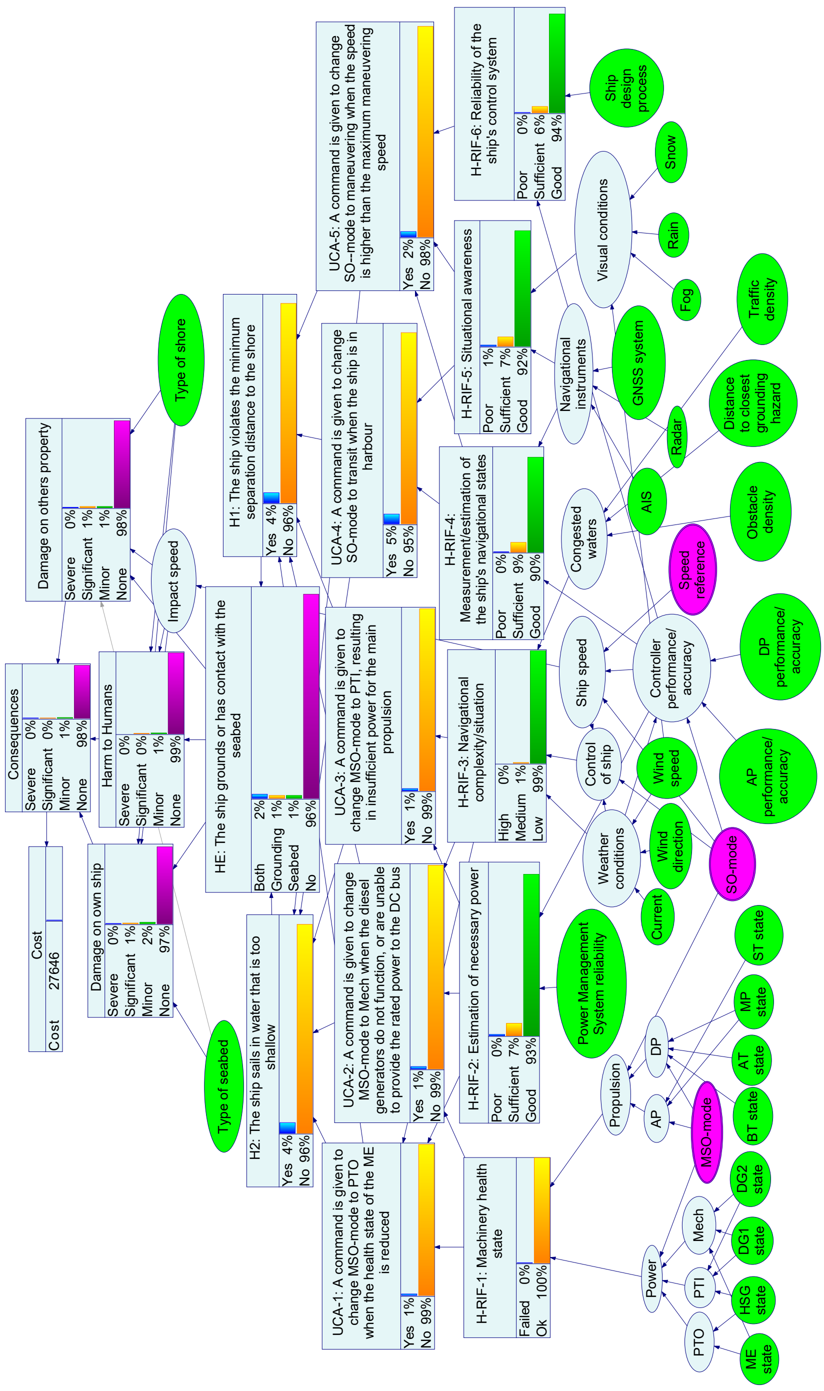

Consequences
Severe 0%
Significant 0%
Minor 1%
None 98%
Cost
Cost 27646
Harm to Humans
Severe 0%
Significant 0%
Minor 1%
None 99%
Damage on others property
Severe 0%
Significant 1%
Minor 1%
None 98%
Damage on own ship
Severe 0%
Significant 1%
Minor 2%
None 97%
Impact speed
Type of shore
Type of seabed
HE: The ship grounds or has contact with the seabed
Both 2%
Grounding 1%
Seabed 1%
No 96%
H1: The ship violates the minimum separation distance to the shore
Yes 4%
No 96%
H2: The ship sails in water that is too shallow
Yes 4%
No 96%
UCA-5: A command is given to change SO–mode to maneuvering when the speed is higher than the maximum maneuvering speed
Yes 2%
No 98%
UCA-4: A command is given to change SO-mode to transit when the ship is in harbour
Yes 5%
No 95%
UCA-3: A command is given to change MSO-mode to PTI, resulting in insufficient power for the main propulsion
Yes 1%
No 99%
UCA-2: A command is given to change MSO-mode to Mech when the diesel generators do not function, or are unable to provide the rated power to the DC bus
Yes 1%
No 99%
UCA-1: A command is given to change MSO-mode to PTO when the health state of the ME is reduced
Yes 1%
No 99%
H-RIF-6: Reliability of the ship's control system
Poor 0%
Sufficient 6%
Good 94%
H-RIF-5: Situational awareness
Poor 1%
Sufficient 7%
Good 92%
H-RIF-4: Measurement/estimation of the ship's navigational states
Poor 0%
Sufficient 9%
Good 90%
H-RIF-3: Navigational complexity/situation
High 0%
Medium 1%
Low 99%
H-RIF-2: Estimation of necessary power
Poor 0%
Sufficient 7%
Good 93%
H-RIF-1: Machinery health state
Failed 0%
Ok 100%
Ship design process
Snow
Rain
Fog
Visual conditions
Traffic density
Navigational instruments
GNSS system
Radar
Distance to closest grounding hazard
AIS
Congested waters
Obstacle density
Speed reference
Ship speed
Controller performance/ accuracy
DP performance/ accuracy
AP performance/ accuracy
Control of ship
Wind speed
Wind direction
Weather conditions
Current
SO-mode
Power Management System reliability
ST state
MP state
AT state
BT state
DP
AP
Propulsion
MSO-mode
DG2 state
DG1 state
HSG state
ME state
Mech
PTI
PTO
Power

Figure A.4: BBN risk model showing an example of the risk cost. For more detailed information about the BBN, please contact the corresponding author.

A.3.2 Step 2: Building the online risk model

The STPA is used as the basis to build the online risk model based on the methodology in [7], as shown in Figure A.4. The output from the risk model is the expected cost from the consequence. The BBN has four nodes describing the consequences: one general consequence node and one for damage to own ship, damage to others property, and harm to humans; one node describes the hazardous event, and one node describes each of the system-level hazards. The two system-level hazards depend on the five UCAs considered in the STPA. Each of these correspond to one node in the BBN.

The nine scenarios described in the STPA are used as the basis to define the six H-RIFs in the BBN. The list of H-RIFs, with the corresponding scenarios are shown in Table A.3. Each of the high-level RIFs are analyzed further to find I-RIFs, as shown in Table A.4.

In addition to the I-RIFs and decisions in Table A.4, the type of seabed and shore affect the consequences directly. Intermediate nodes are used between I-RIFs/decisions and H-RIF nodes to reduce the number of inputs to each node. This reduces the size of conditional probability tables (CPTs) and makes it easier to define these. CPTs and states are defined based on the work in [150], [186], [187], discussions with crew working on different ships, and control engineers from Kongsberg Maritime. A full list of all nodes, with parent nodes, is shown in Tables A.5 and A.6.

The BBN is converted to an online risk model by linking I-RIFs to the control system so they can be updated as the ship sails. Nodes describing the state of machinery parts are updated with information from the AMMS. If the machinery is well functioning and well maintained, then the probability of failure is very low, $9 \cdot 10^{-7}$. In future works, this is intended to be updated as the ship sails since machinery components are more likely to fail as components age, but this is not modeled in the current case study.

Table A.3: Risk influencing factors

High-level RIF	Description	Scenario(s)
H-RIF-1	Machinery health state	SC-1,SC-2,SC-3
H-RIF-2	Estimation of necessary power	SC-1,SC-2,SC-3,SC-4,SC-5
H-RIF-3	Navigational complexity or situation	SC-1,SC-2,SC-3,SC-4,SC-5
H-RIF-4	Measurement/estimation of the ship's navigational states	SC-6, SC-8, SC-9
H-RIF-5	Situation awareness	SC-7, SC-8
H-RIF-6	Reliability of the ship's control system	SC-9

Table A.4: Input to H-RIFs

High-level RIF	Description	Input RIF/Decision
H-RIF-1	Machinery health state	ME state, HSG state, DG1 state, DG2 state, BT state, AT state, MP state, ST state, MSO-mode (Decision node), SO-mode (Decision node)
H-RIF-2	Estimation of necessary power	PMS, AP performance/accuracy, DP performance/accuracy, SO-mode (Decision node)
H-RIF-3	Navigational complexity or situation	Traffic, Obstacles, Current, Distance to grounding hazard, Wind speed, Wind direction, SO-mode (Decision node) Speed reference (Decision node)
H-RIF-4	Measurement or estimation of ship's navigational states	GNSS system, Radar, AIS, SO-mode (Decision node) AP performance or accuracy, DP performance or accuracy
H-RIF-5	Situation awareness	GNSS, Radar, AIS, Visual conditions
H-RIF-6	Reliability of the ship's control system	SO-mode (Decision node), AP performance or accuracy, DP performance or accuracy, Ship design process

Nodes describing the control system and sensors are given a static value based on [150], [186], [187]. Weather nodes are linked to sensors where these exist, such as wind and current, or weather forecast and historical data [188]. These nodes are designed to be updated in real-time depending on the available data. Traffic use data is drawn from the automatic identification system (AIS), which is used to transmit the identity, position, course, and speed to nearby vessels using the very high frequency (VHF) band. Obstacle density and distance to grounding hazards are taken from the ENC. The seabed and shore are described with data from [189] over the relevant area. The values used in input nodes describe the probability over the planned mission.

A.3.3 Step 3: Setting up the ENC module

The ENC module is set up to extract data from electronic navigational charts for use in the online risk model and the rest of the control system. The ENC module here includes charts covering the areas around Brønnøysund and Rørvik in Norway, which are relevant for the type of ship in the case study. The module is set up to consider everything shallower than 5 m as shallow areas or land where the ship cannot navigate safely. The rest of the chart is divided into layers of 10 m, 20 m, 50 m, 200 m, 350 m, and 500 m. This distribution is considered a reasonable combination of chart resolution and efficiency in the control system.

The obstacle density is based on the distance to the closest shallow point (i.e., areas with less than 5 m water depth) and the percentage of obstructed water around the ship. The water depth of 5 m is the same as the max draft of the ship. Using this water depth is considered sufficient for assessing the portion of obstructed water in this work. Shallow areas are consequently areas with too little water depth for the ship to sail in, which should be avoided with sufficient safety margins. The percentage of obstructed water is calculated by considering a disk with radius 1400 m and finding the portion of the disk with land and shallow water. The radius is set through testing to ensure that the disk gives a good picture of the sea area surrounding the ship, without being unnecessarily large.

The ENC module checks the area around the ship every 15 seconds and updates the input to the online risk model. Updating every 15 seconds ensures that the control system has updated data, while limiting the computation time necessary to check the ENC module.

A.3.4 Step 4: Building the supervisory risk controller

The SRC is the high-level controller that manages and controls the ship. The SRC uses data from the risk model and ENC, combined with operational measurements from the ANS and AMMS, such as position, speed, and machinery status to make decisions. The SRC has four main objectives: selecting the SO-mode, selecting the MSO-mode, setting the reference speed for the ship to follow, and notifying the human supervisor when the situation becomes too severe to continue.

The SRC is implemented as a switch that checks the cost function, as shown in Equation A.1, for each set of decisions. The risk cost is calculated using Equation A.2. This takes the probability of the different consequences, $Pr()$, estimated in the online risk model described, multiplied with the expected cost for each consequence, $C()$, as described in Section A.3.1:

$$\begin{aligned} R(d) =& Pr(severe)C_{severe} + Pr(significant)C_{significant} \\ &+ Pr(minor)C_{minor} + Pr(none)C_{none} \end{aligned} \tag{A.2}$$

The fuel cost is calculated as the specific fuel cost (SFC) multiplied by the expected sailing time. The SFC is taken from a look-up table, depending on wind speed, ship speed, current speed, and MSO-mode. The look-up table is made by simulating the machinery under different conditions to estimate how much fuel is used to sail a set distance. The fuel prices are taken from [190] at 1 343.5 USD/ton for LNG and 684.5 USD/ton for diesel. This table provides a cost per distance that is multiplied with the planned sailing distance, as shown in Equation A.3:

$$F(d) = SFC(wind, speed, current, machinery) * distance \tag{A.3}$$

Operation costs are calculated using Equation A.4. This includes manning in the ROC, maintenance from wear and tear on the machinery, insurance of the ship, lubrication oil, spare parts, and logistics. These are estimated based on conventional ships of the similar size and type, and using data from [107] to be 341.3 USD/h for the current ship. This is similar to the fuel cost in normal transit with a speed of $5 - 7$ m/s ($9.7 - 13.6$ knots):

$$O(d) = Cost_{operating} * distance/speed \tag{A.4}$$

The cost of potential future loss is calculated with Equation A.5. This cost is the loss of income if the ship is unable to take on any new missions before finishing the current route, which is set to 910.1 USD/h:

$$L(d) = Cost_{futureloss} * distance/speed \tag{A.5}$$

The cost function, including the ratio between the different terms, is discussed in Section A.4.7. The controller estimates the cost of sailing a distance equal to the initial route distance. This is constant for the whole route which keeps the weight between the different cost terms constant.

The alarm is implemented such that a human supervisor can take over control remotely of the ship if necessary, but unnecessary alarms also need to be avoided. To achieve an acceptable balance, the alarm trips if either the risk cost exceeds 9 267.70 USD, or the probability of the hazardous event exceeds 0.5. The cost limit is set between minor and significant consequences because it is better to have the human supervisor check the ship, than having an emergency later on. The SRC is implemented to lower the speed to limit the risk cost because impact speed directly affects the consequences. However, this can cause situations where the probability of a hazardous event is too high to continue due to environmental conditions, even though the risk cost is low because the speed is reduced to the minimum. Thus, a probability limit of 0.5 is used to notify the human supervisor in these situations.

If the SRC changes the ship's control configuration, then it is paused for 30 seconds before checking again. Implementing a time delay in the switching logic ensures that the controller reacts to changes but avoids situations where it gets stuck switching between different modes (e.g., DP and AP) without stabilizing, which is also called chattering [191].

A.3.5 Step 5: Verifying the control system

After setting up the SRC, verification is done by first determining how to test the system and which requirements to verify against. The autonomous ship should follow the route through Brønnøysund that is shown in Figure A.5. The route follows the same path as a conventional ship and those described in [192]. This is

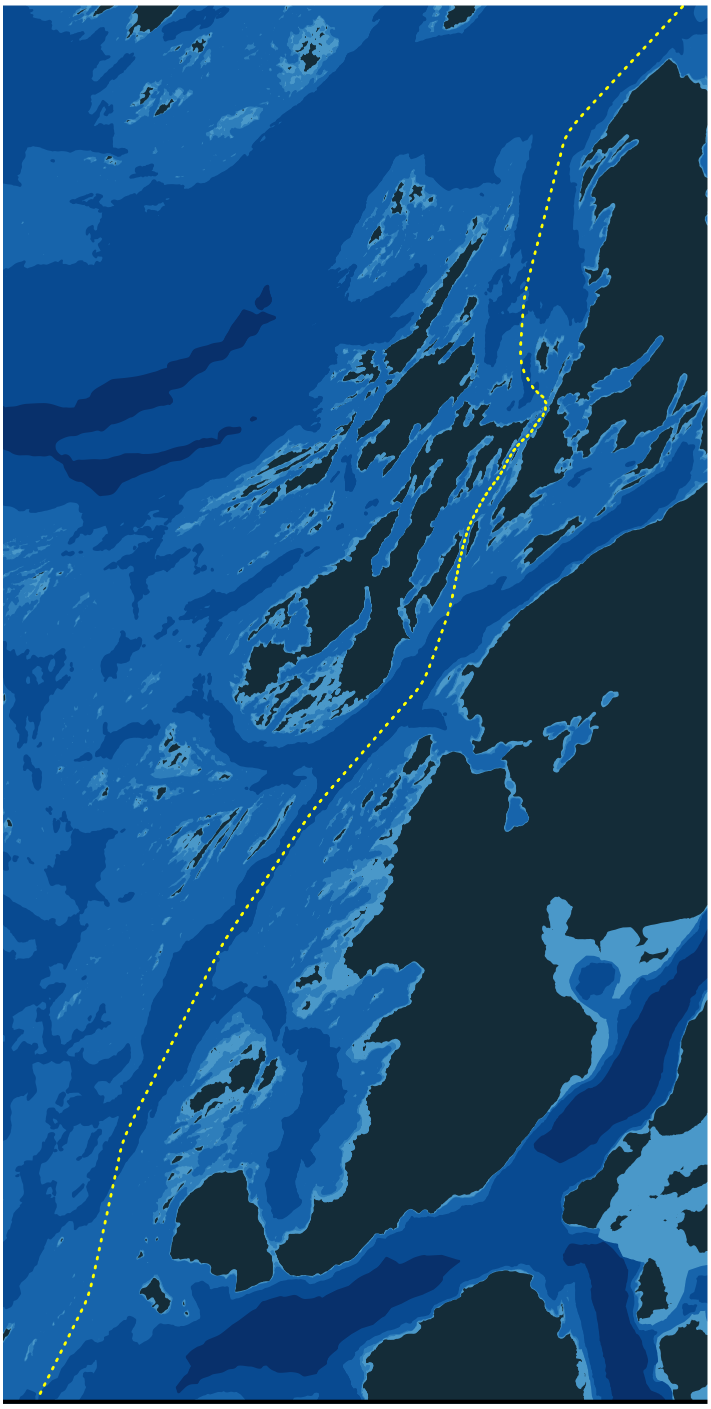


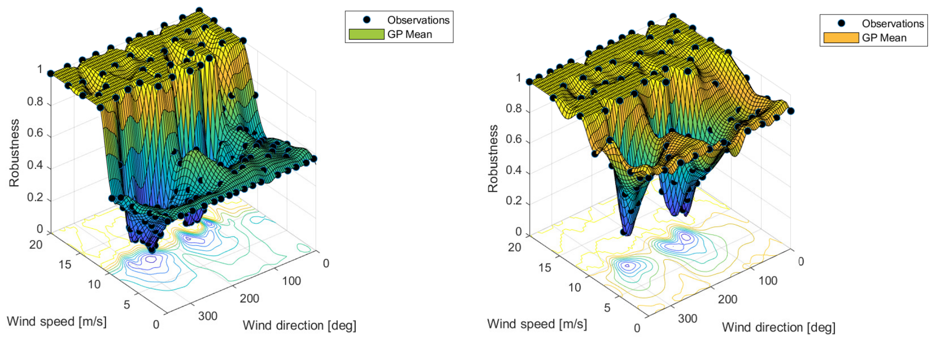
Figure A.5: Route used in the verification process

used to check the ship in situations where the controller is expected to adjust the speed reference, without using much longer time than conventional ships. The ship has to lower the speed reference early enough to slow down when entering narrow and tight areas, and increase it when it opens up again.

To test safety, the ship should maintain a minimum distance of 5 m to shallow areas or provide an alarm to the human supervisor at least 5 min before the minimum distance is violated. Having a minimum distance of 5 m is not realistic for a real ship. However, to account for extra drift caused by simplifications in the simulator this is used to get results reasonable results that can be compared to conventional ships. These assumptions are discussed further in Section A.4.8. The following verification focus on wind and how this affect the ship. However, the process is the same for other disturbances, such as current.

To verify that the controller is efficient, the ship should at maximum use 140 min on the whole route segment under consideration in the case study or provide an alarm to the human supervisor. This time limit is set based on the time existing manned ships used on the same route. Both the safety and efficiency requirements are tested in wind speeds ranging from no wind to 20 m/s and from all directions. Other factors (e.g., current, waves, and machinery failures) are not considered in the verification. This simplifies the verification but still gives sufficient results for further testing of the control system. The route is chosen to get a good variation between open water and more narrow straights with tight turns.

The verification is performed using the automatic simulation-based testing methodology that was introduced in Section A.2.5. This methodology selects and simulates interesting combinations of wind speed and wind direction to verify or falsify the system. The system is verified to satisfy the safety requirement (minimum distance to shallow) in 161 simulations, and the efficiency requirement (maximum allowed sailing time) in 97 simulations. The STL robustness surfaces for safety and efficiency are shown in Figures A.6(a) and A.6(b), respectively. The STL robustness



(a) Safety verification **(b)** Efficiency verification

Figure A.6: Robustness surfaces resulting from the two verification runs

score is normalized to the interval $[-1, 1]$. Figure A.6(a) shows that the robustness score in the case study is always above 0. Similarly, Figure A.6(b) shows that the robustness is always above 0 and is close to 1 when it reaches the final waypoint early or trips an alarm because the risk cost or grounding probability becomes too high.

The verification shows that the control system makes the autonomous ship follow the route and it also reaches the end of the route in reasonable time in wind speeds of up to 8 m/s. Above this, the planned route forces the ship very close to land in certain spots, which means that it notifies the human supervisor. When the wind speed exceeds 10 m/s, the route leaves too little space for the ship to maneuver. This can cause problems with certain wind conditions. However, the control system provides an alarm to the human supervisor with enough time to pass the safety requirement. Overall, the verification shows that the proposed control system works in the planned route but it is limited by not being able to change the route in accordance with the environmental conditions.

A.4 Results and discussion

A.4.1 Comparing the controller with the maneuvering of a conventional ship

After building and setting up the controller, the autonomous ship is simulated along two different routes to compare it against an existing conventional ship. The first route is through Rørvik and the second is through Brønnøysund. The route through Brønnøysund is similar to the one used in the verification (Figure A.5) but with different start and end points. The start and end points are changed because the GNSS data from the conventional ship is only available for part of the route. The purpose is to see how the SRC sets the speed reference, MSO-mode, and SO-mode, and compare this to how conventional ships operate along the same routes in similar weather conditions. The existing ship is equipped with a similar machinery and control system as the autonomous ship but with a crew who decides MOS-mode, SO-mode, and speed reference.

The conventional ship sailed through Rørvik and Brønnøysund in the fall of 2021 with a wind speed between $5-7$ m/s. The routes followed by the conventional ship are plotted with GNSS data taken from the control system aboard the conventional ship. The route through Rørvik is planned by placing waypoints along the route that the autonomous ship can follow. The GNSS data for Brønnøysund contain some measurements that place the route over land. The cause of these are not certain but it only affects the data between point 0.5 and 0.7. Therefore, the route was re-planned by placing waypoints along the same route into Brønnøysund but following the route recommended in [192] through and after Brønnøysund. The routes are shown in Figures A.7 for route one and A.10 for route two with the conventional ship in red and the autonomous ship in yellow.

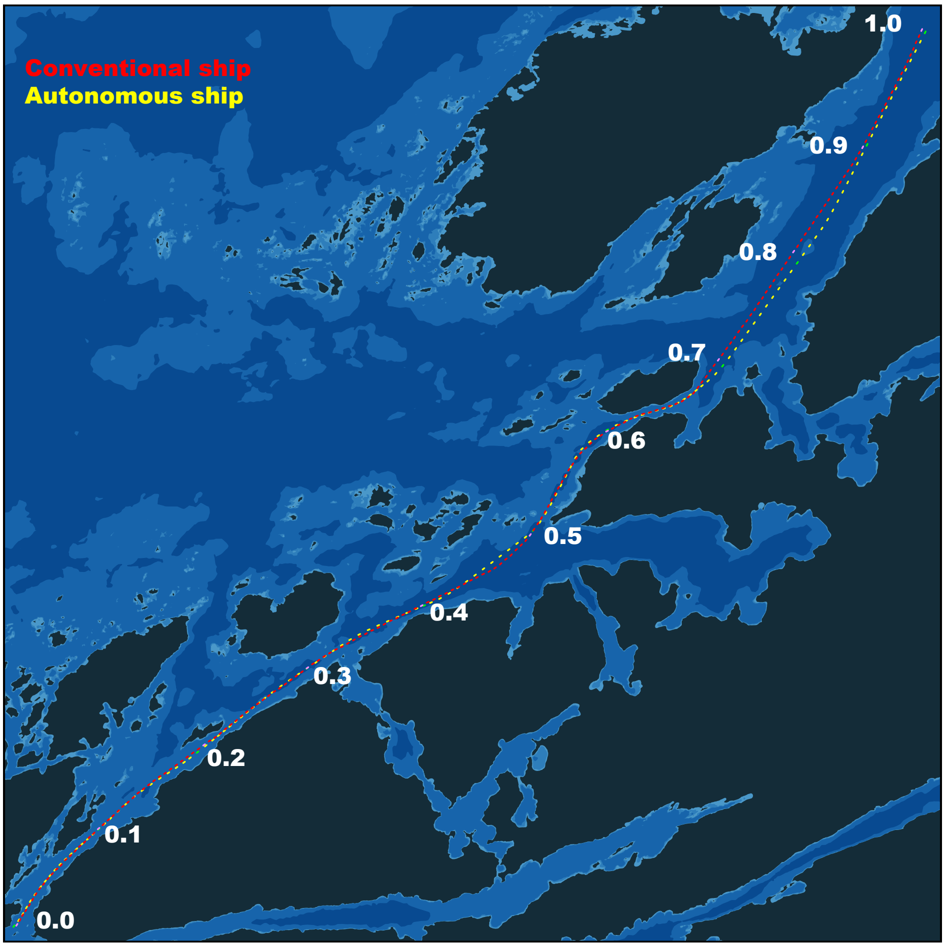


Figure A.7: Map of route one through Rørvik. The conventional ship's route is shown in red and the autonomous ship's route is shown in yellow.

To compare the two ships, the risk model and SRC need position, speed, MSO-mode, and SO-mode from the conventional ship. Position and speed are recorded in the ship's control system. Ship speed is fed directly to the SRC to find the expected fuel cost and is used as input to the risk model. Position data is used in the ENC module to get the distance to the closest grounding hazard and obstacle density. MSO-mode is set to PTO and SO-mode to AP after discussing how the conventional ship is operated with the crew. This provides a cost that can be compared to the autonomous ship. The SRC uses a constant distance when calculating costs, as explained in Section A.3.4. The plots therefore show the costs of sailing a distance equal to the distance of the whole route, d_0, estimated at each point.

Comparison on route one through Rørvik

On route one, the conventional ship starts with a speed of 5.25 m/s, before increasing to 6.5 m/s. The speed is then maintained at $6.5 - 6.75$ m/s the rest of the distance. The autonomous ship starts with a speed of 5 m/s. This is later increased to 7 m/s as the ship sails into more open water. Along the rest of the route, the speed varies between 5 m/s and 7 m/s as it passes through more narrow parts of the route and in more open areas. Overall, the autonomous ship varies the speed more as the environmental conditions change, compared to the conventional ship.

The cost is shown in Figure A.8 for the conventional ship and in Figure A.9 for the

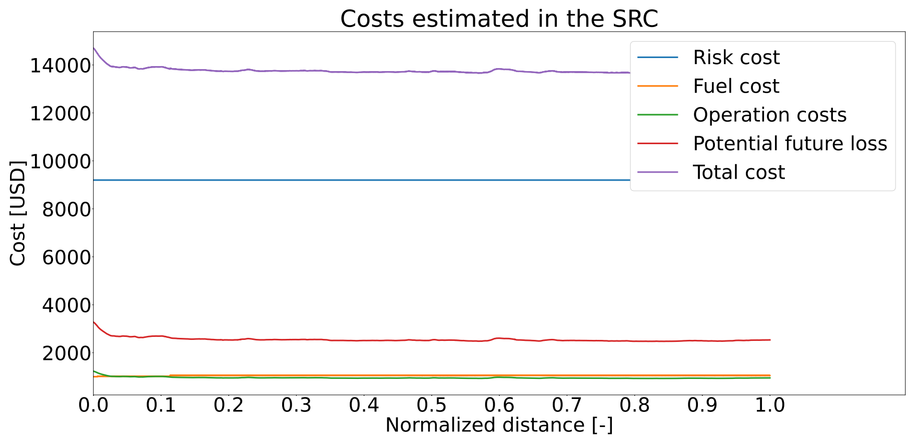


Figure A.8: Conventional ship's costs on route one

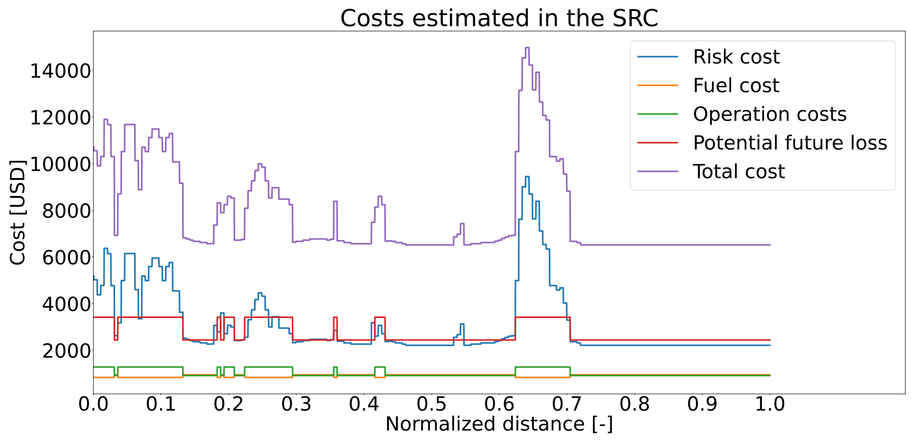


Figure A.9: Autonomous ship's costs on route one

autonomous ship. The plots show the expected costs of sailing the full route, d_0. The conventional ship has a higher risk cost (blue line) because it maintains a higher minimum speed. Fuel (yellow line), operation (green line), and potential future loss (red line) costs are almost the same but they vary more for the autonomous ship because the expected time varies more corresponding to more changes in the speed. For the conventional ship, both fuel and operation costs are almost constant because the speed is kept more or less constant along the whole route. In contrast, the speed of the autonomous ship is changed more, which leads to more changes in fuel and operation costs. The conventional ship uses 96 minutes on the whole route and the autonomous ship uses 103 minutes.

Comparison on route two through Brønnøysund

The routes differ slightly more through Brønnøysund, due to the errors in the position data from the conventional ship. This means that the autonomous ship sails around 1 km longer. The conventional ship maintains a speed of around 6.75 m/s before it reaches the narrow parts of the route between 0.5 and 0.6 on the route shown in Figure A.10. In the narrowest part, the speed is reduced to 3 m/s, it is then increased to $6.75 - 7$ m/s as the area opens up. The autonomous ship has a speed of 7 m/s in open water. This is reduced to 5 m/s when it reaches the first narrow straits between points 0.4 and 0.5. It then returns to 7 m/s for a short time in the more open area, before it is reduced to 4 m/s through the narrow harbour area. Overall, the autonomous ship makes more changes to the speed, but maintains a higher minimum speed.

The cost is shown in Figure A.11 for the conventional ship and Figure A.12 for the autonomous ship. Fuel (yellow line), operation (green line), and potential future loss (red line) costs are virtually the same along the whole route. The risk cost is similar along the first part where both ships follow the same route, but is much higher for the conventional ship in the middle part of the route. This is caused by the inaccuracies in the GNSS data collected on the conventional ship showing the ship sailing very close and over land, and the conventional ship not reducing the speed between points 0.4 and 0.5. This combination results in a significantly higher risk cost compared to the autonomous ship. Fuel cost is similar for both ships with a reduced fuel consumption when the speed is reduced in the most challenging part of the route. Operation cost is also similar, but with a higher top for the conventional ship since it reduces the speed more.

A.4.2 Controlling the ship with machinery and propulsion failures

The second part of the case study tests how the control system manages the autonomous ship when the health of the main engine and steering system is worsened. This is modeled by increasing the probability of failure for these elements in the risk model. The SRC then chooses the best way to operate the ship based on this information. The routes are the same as shown in Figure A.7 for route one and

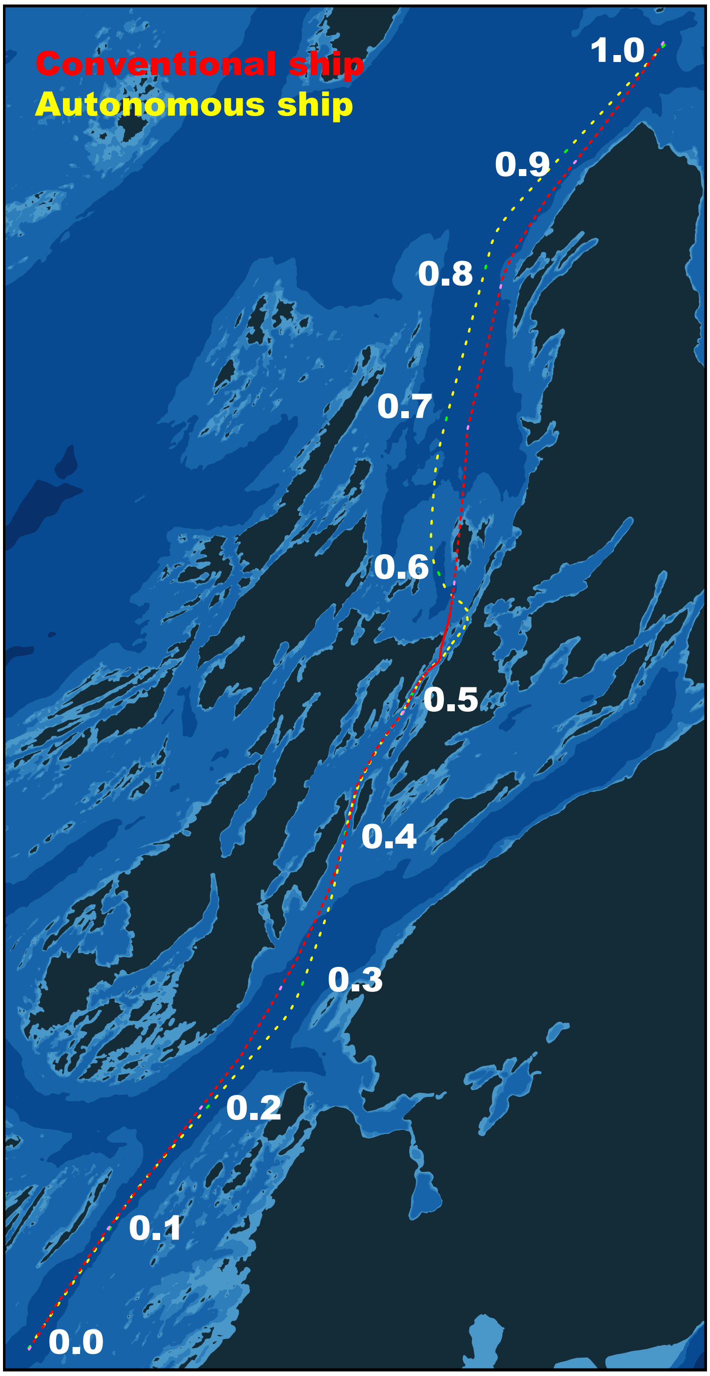


Figure A.10: Map of route two through Brønnøysund. The conventional ship's route is shown in red and the autonomous ship's route is shown in yellow.

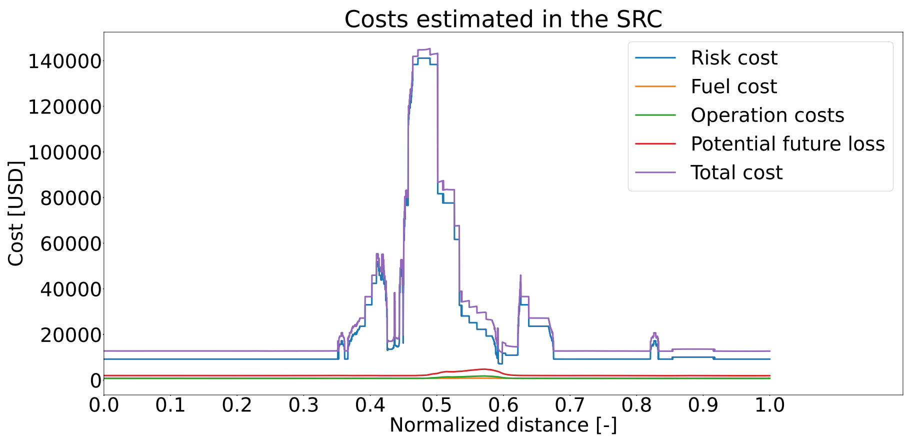


Figure A.11: Conventional ship's costs on route two. The risk cost is here significantly higher since the position data used to estimate the costs include some incorrect measurements placing the ship both very close and on land as shown in Figure A.10.

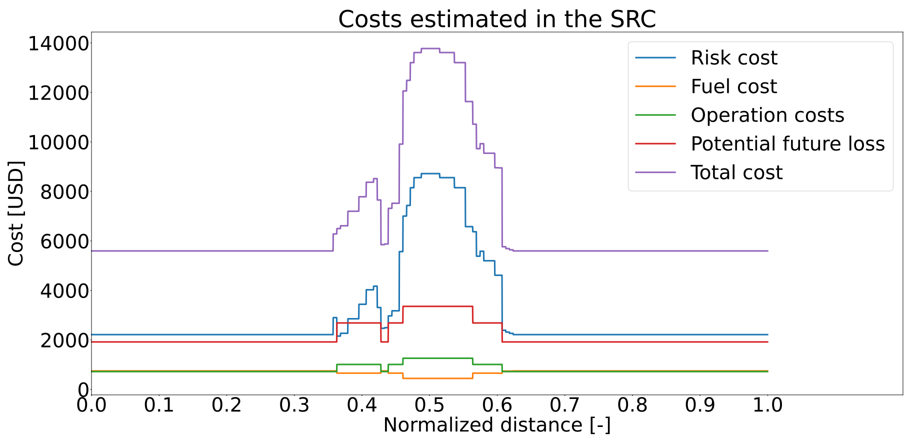


Figure A.12: Autonomous ship's costs on route two

Figure A.10 for route two. The weather is also the same, which ensures that the results can be compared to how the ship is managed when all systems function.

Machinery and propulsion failures on route one through Rørvik

In both cases, the failure happens when the ship has sailed approximately 8 % of the route, close to point 0.1 on the figures. When the main engine fails, the SRC changes MSO-mode to PTI, which only uses the HSG and diesel generators for power production. The speed reference is also reduced to 4 m/s because the diesel generators produce less power than the main engine. This ensures that the ship still has sufficient power to maneuver. The SO-mode is AP along the whole route in this case.

When the steering machinery fails, the speed is lowered significantly such that the tunnel thrusters can provide steering for the ship and SO-mode is changed to DP. The MSO-mode is Mech for the whole route. The speed reference switches between 2 m/s and 3 m/s, depending on the number of islands and obstacles around the ship.

Figures A.13 and A.14 show the costs calculated by the SRC. Overall, the cost is maintained at a similar level as when everything is working by adjusting how the

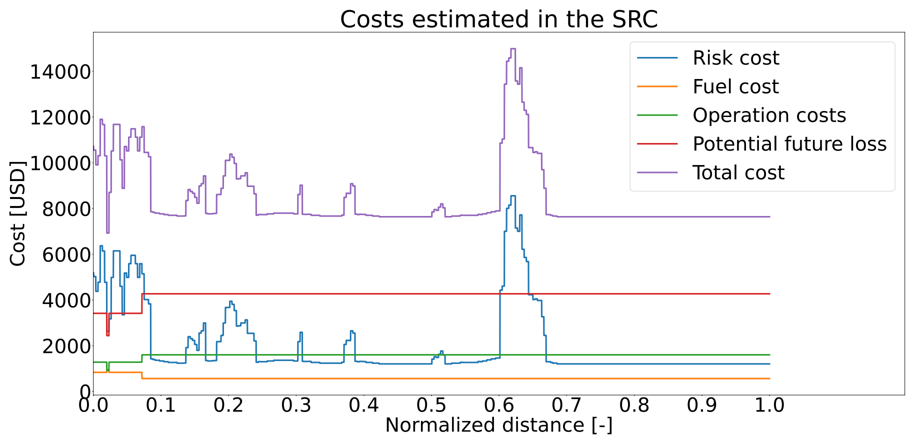


Figure A.13: Costs with failure on main engine on route one

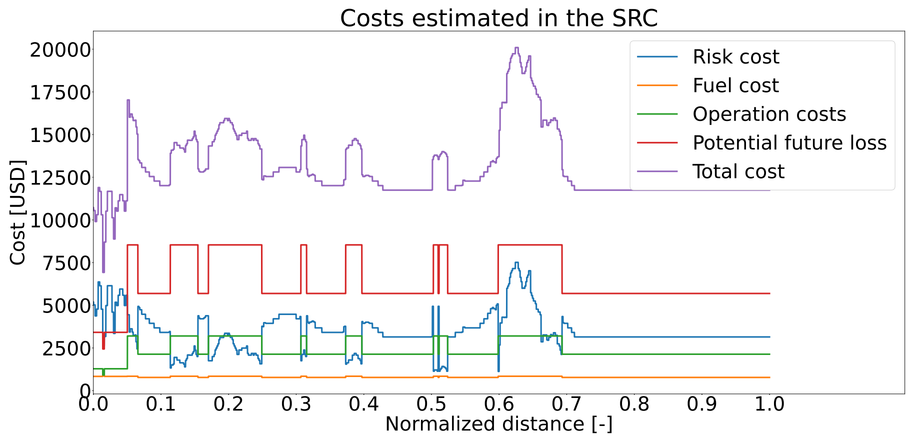


Figure A.14: Costs with failure on steering machinery on route one

speed is operated. The risk cost is controlled by reducing the speed, compared to how the ship is operated when all systems function as intended, and by switching to MSO-modes and SO-modes with functioning components. Operation and potential future loss is increased because the ship uses a longer time with lower speed.

Machinery and propulsion failures on route two through Brønnøysund

The main engine fails between point 0.3 and 0.4, and the steering machinery fails between point 0.2 and 0.3. When the main engine fails, the speed is reduced significantly to account for the reduced power production. The MSO-mode is also changed to PTI, which do not use the main engine. The SO-mode is AP along the whole route.

When the steering machinery fails, the speed is reduced to 2 m/s and SO-mode is changed to DP, to get more effect from the tunnel thrusters and maintain control of the ship. When the ship has passed the narrowest parts of the route, the speed is increased to 3 m/s.

Similar to route one, the costs that are shown in Figure A.15 for the main engine and Figure A.16 are similar as when everything is functioning by reducing the speed and changing MSO-mode and SO-mode. The biggest difference compared to the cost when all systems function is the time used to finish the route. The time and the time dependent costs, operation costs and potential future loss increase when the ship sails at a lower speed. This is most visible after the ship has finished with the most challenging parts of the route, around 0.4-0.5. However, because the speed was reduced in the narrow and tight parts with all systems functioning as well, the max cost is still at the same level.

Data from conventional ships operating with failures but switching to modes that function without the failed components are limited, although this is a logical way to mitigate failures. In a conventional ship, the failed components can be fixed by the crew or the ship can be maneuvered to the closest harbor for repairs. On an autonomous ship without a crew, the only option is to maneuver to harbor and get it fixed there or in case of severe failures transport a repair crew to the ship offshore. Because this route change is not included in the SRC and the redundancy of the machinery systems onboard the autonomous ship was not compromised entirely, the ship continues to sail towards the final waypoint. With the current control system, this is a reasonable solution. Deviating from the planned route to get to shore and repair damaged equipment, which would be viable solutions in case of critical machinery failures and total loss of propulsion, and notifying the human supervisor are topics for further research that could improve the control system further.

A.4.3 Risk modeling and implementation in the control system

The proposed control system uses a BBN-based risk model to assess the risk. The model is based on an extended STPA of the ship. STPA provides a systematic way

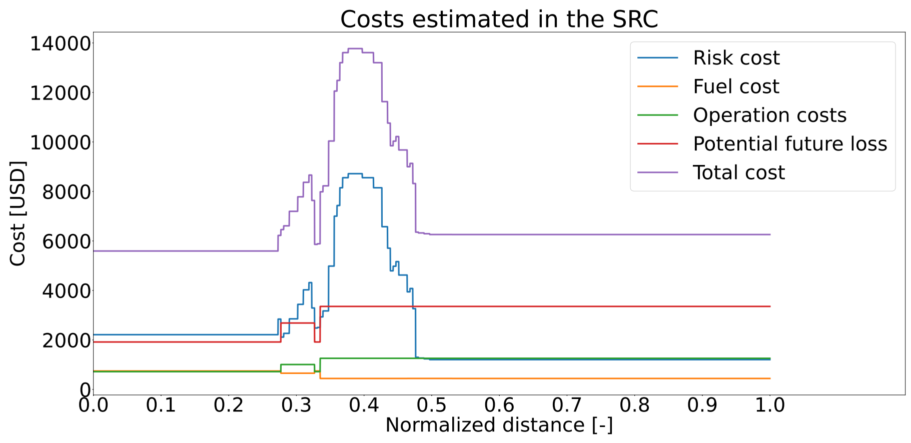


Figure A.15: Costs with failure on main engine on route two

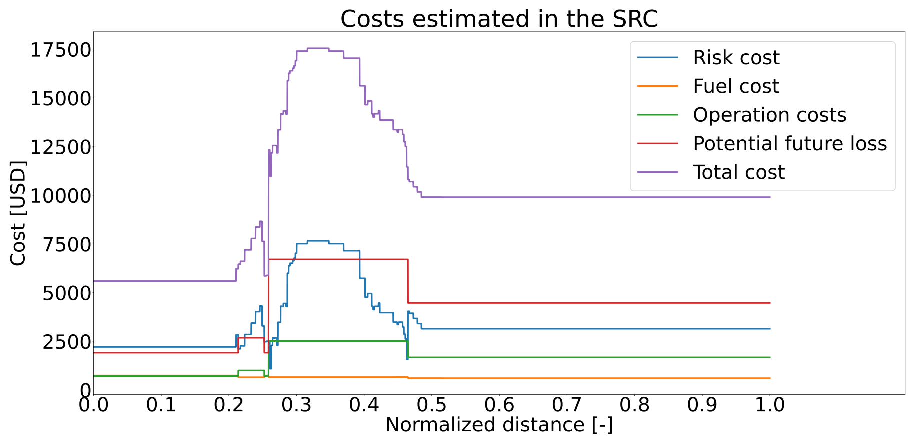


Figure A.16: Costs with failure on steering machinery on route two

to analyze the ship and identify causal factors that can lead to hazardous events. The results of the STPA also provide a logical way to build and structure the BBN. However, the results depend on the data used and the quality of the analysis.

Another potential challenge using STPA is to decide the refinement level. The refinement level generally depends on the purpose of the STPA. More details mean more data, but it can also make the risk model and the corresponding calculations too time consuming. In this current work, the analysis considers one hazardous event only, two system level hazards, and five UCAs. The scenarios include causal factors, such as wind, obstacles, and the main parts of the machinery system. The scenarios could have been more detailed and could have included information about how machinery parts fail. However, because the purpose of the analysis in this work is to build an SRC, the level of detail is considered to be sufficient because the controller does not provide detailed control actions to the different parts of the machinery systems. An example of this could be saying that the main engine can only produce limited power because the cooling system is only partially functioning, although in this situation limited power is necessary to maintain control of the ship. Enabling the controller to make such decisions would be an interesting topic for further research to continue to develop the control system.

When building the BBN risk model, the overall structure is determined by the STPA. However, because the STPA is qualitative, it provides very little data for setting up states defining CPTs. Hence, they are generally based on other sources, such as literature, previous works, and expert judgement. The CPTs can also be adjusted later to put more weight on specific risk factors. Given that the CPTs are based on different sources, they contain a certain degree of uncertainty, as discussed in Section A.4.7.

To convert the risk model into an online risk model, the risk model is connected to the rest of the control system. This means that all of the nodes in the BBN that can be measured by the control system or sensors should be updated when the ship is sailing. The risk model should be updated often to describe the current sailing conditions. However, updating it too often increases the computation time in the control system. There is also a limit to how quickly the controller can update the decisions. In the case study, the risk model and SRC is paused for 30 seconds if the SO-mode, MSO-mode, or speed reference is changed. This delay allows the controller to evaluate if the decisions influence the ship and to avoid chattering, where the controller is stuck switching back and forth between different decisions, such as during DP and AP.

The control system can be expanded further by including more dynamics in the ship model. The case study assumes that machinery parts can be started immediately, which is not the case. Although the specific time necessarily varies for different engines, it will have to be included when making decisions. This type of dynamics could be included in the control system as limits to how often decisions can be changed. The risk model can also be modified to include starters for the different machinery parts. For example, for the main engine to function, both the starter and engine would be necessary.

Similar dynamics can be included for changing load on the machinery and the speed of the ship. In particular, reducing the speed of the ship takes time, depending on the size of the ship. The ship simulator includes a time delay on load changes and uses some time to change the speed of the ship. However, the SRC does not account for this specifically when it makes decisions. Therefore, including more dynamics in the control system and risk model is an interesting topic for further research.

A.4.4 Challenges with measuring risk in the cost function

The proposed control system uses a cost function to make decisions about MSO-mode, SO-mode, and speed reference. This cost function estimates the cost of operating and sailing the ship, and the potential cost of hazardous events. The cost of sailing and operating the ship is straightforward to calculate and use in a cost function because it is already measured as cost. However, to combine this with risk cost is a bigger challenge. The STPA analysis can identify potential hazardous events but is only a qualitative analysis that does not consider likelihood of these events or the following cost.

This work addresses this problem by extending the analysis to consider consequences and classifying these in terms of cost. The STPA results and consequences are modeled in a BBN to give a likelihood of the consequences. The likelihood is multiplied with the consequence cost to give a risk cost to use in the cost function. Decisions are then made based on the current time, without considering how this can change in the future. Risk could be alternatively assessed by simulating how changing conditions and decisions affect the cost over a longer time. This would make the SRC more like an MPC, which could find the optimum set of decisions to minimize the cost over a longer time period. However, this would mean running a lot of simulations to check all potential combinations. Investigating this further could be subject for further research.

A.4.5 Risk modeling and integration with the ENC module

In the proposed control system, information about grounding obstacles is important for the risk model because it allows the model to assess the area around the ship. This information, and other data about the relevant area, is available in ENCs. The ENC module is an efficient tool for extracting and filtering this information to enable it to be used to describe the navigation area in the risk model. The control system uses the distance to the closest area where the ship can ground and the density of such areas as inputs to the risk model. Together with weather and traffic data, this determines how challenging it is to maneuver the ship.

The ENC module used in this work do not account for navigational markers, as this is not currently implemented in SeaCharts. For an autonomous ship, knowing where different navigational markers and their meaning is an important part of operating safely. The proposed control system itself can utilize this information in the risk model to get a better understanding of the environment when this become available with the SeaCharts package. However, the current ENC module is still considered sufficient to demonstrate that the proposed control system works.

The ENC module also provides an efficient way to plot the ship during testing, and is used when testing the control system to see how well the ship follows the route and identifies problems in specific areas. Compared to just using the position data, without grounding obstacles and land, this approach makes it much easier to understand and/or verify how the ship maneuvers.

Data from the ENC module can also be used to add more functions to the control system, such as route planning. In addition, a planning algorithm can use the ENC module to check if the route maintains the necessary distance to land and grounding obstacles. When combined with AIS data, this can enable the planner to account for other ships and use this information to avoid collisions. This is an interesting extension of the control system that would reduce the need for human supervision and control even further. This point is left open as a relevant topic for further research.

A.4.6 The efficiency of testing and verification of control systems in operation

In this work, the proposed control system is verified against the design requirements using the automatic simulation-based testing framework that was introduced in Section A.2.5. Using this approach significantly increases the efficiency of building sufficient verification evidence for the control system. [177] show that this reduces the number of simulations necessary to verify the scenario compared to a regular grid search, which is a large time saver when doing several design iterations and verifying the scenario after each iteration.

The robustness surface resulting from a verification run with the automatic testing framework enables us to quickly get an overview of the performance of the SRC system at different regions of the scenario space. This overview is actively used in the design process to iteratively adjust the control system. Compared to the alternative of running simulations manually and evaluating the resulting time series, this offers a significant reduction in the workload. Furthermore, using STL to evaluate the system also gives a robustness score to show not only that it is verified, but also how well the system performs.

It is also worth noting that the verification process considers a specific route and area. These can be planned such that the route includes different environments, such as open water, coastal waters with many islands, or tight harbor areas. The results from the verification should then be valid for other routes with similar characteristics, as shown in the case study. However, if the system is only tested in a distinct environment, such as open water without obstacles, then it cannot say anything about how the controller handles other environments.

An interesting extension of the automatic testing framework is to also use it in an online setting and integrate it more closely with the SRC system. This online verification system could repeatably start verification runs at fixed time intervals. A verification run would attempt to verify safe operation for a finite time-horizon ahead and for a set of uncertain scenario parameters, such as environmental conditions, traffic, or internal components failures. It would achieve this by running simulations with the current situation as an initial condition and then intelligently selecting the scenarios to simulate using the Gaussian process model. The simulator should have an exact (software-in-the-loop) replica of the SRC system, thereby also evaluating how future choices of the SRC system will affect the performance in the different scenarios. The result from a verification run would be used as a robustness map for future scenarios. This robustness map, when combined with data on the probability of the different scenarios, could then be used by the SRC system to make risk-based decisions. The concept of an online verification system operating in closed loop with the SRC system appears to be very interesting because it enables the SRC system to consider multiple future scenarios and at the same time evaluate how its decisions would affect future behavior.

Another interesting extension is to use the STPA directly to define safety requirements and simulation scenarios; see, for example, [13]. In the current work, the scenarios are set up to test the ship in a wide range of wind conditions and in very different areas. However, testing similar scenarios to those that the STPA identified when controlling the ship is challenging. Therefore, testing in more specific scenarios based on the STPA is left for further research.

A.4.7 Uncertainties and sensitivity in the data and models in the case study

The proposed control system combines existing control systems, such as DP and autopilots, with an online risk model in an SRC. The DP and autopilot are well described in the literature and are used on conventional ships. However, the use of an online risk model in an autonomous ship system and the concept of a cargo ship sailing without humans onboard is a novel concept. This means that data describing this is very limited, and mostly based on concepts and plans for these types of ships.

To get sufficient data in the case study, a combination of data from traditional manned ships, concepts for autonomous ships, geographic, and weather data is used. The quality of geographical and weather data is good with little uncertainty. However, the case study considers a simplified environment and not all conditions that a real ship would experience. For example, the wind measurements are taken over a long period but only at a general location. The wind is therefore assumed to be the same along the whole route, even though it will likely vary significantly between different locations. Similarly, the charts that are used are the same as ships use for navigation today but are simplified to only consider shallow areas and land, and not other ships or navigational marks. Although these simplifications make it possible to test the system, they also lead to uncertainties in the results (e.g., how the system can handle more obstacles such as other ship traffic and more local variations in wind conditions).

The STPA used in the paper is based on a workshop with academic industry experts. This helps to identify relevant information for the case study, but the quantitative risk models and corresponding calculations could still have limitations affecting the risk costs.

The input uncertainty will have a different effect on the overall uncertainty, depending on the sensitivity of each input node. If a node has high sensitivity, then

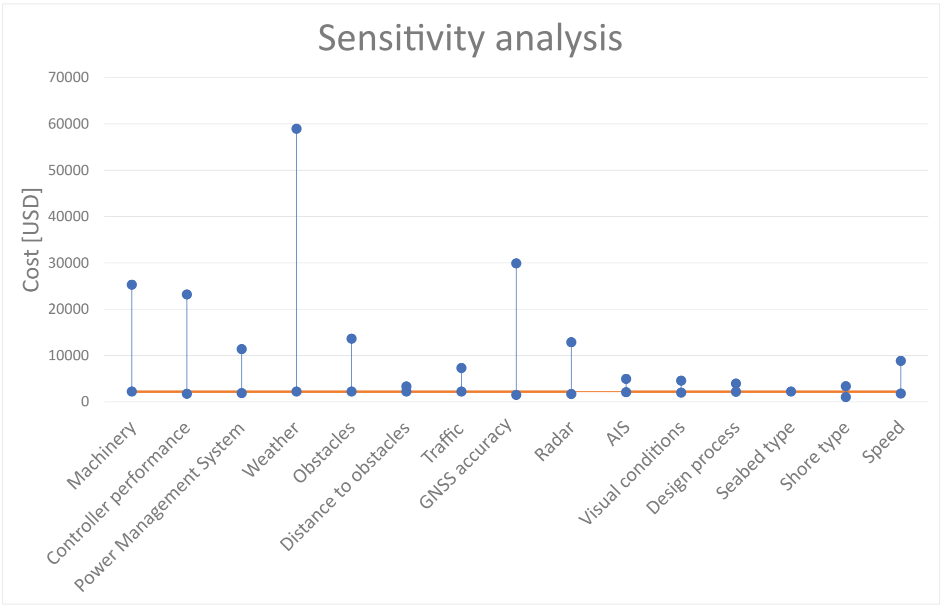


Figure A.17: Sensitivity analysis, showing the effect on the risk cost of setting nodes in the best and worst states.

changing it will change the risk cost more compared to nodes with lower sensitivity. Nodes with high sensitivity have the same effect on the uncertainty in the risk cost. Figure A.17 shows the effect that each node has on the risk cost when setting the node in the best and worst state. This shows that the weather conditions have the biggest potential effect on the risk cost. Other input nodes with a noticeable effect on the risk cost are GNSS accuracy, machinery status, controller performance, and obstacles. However, it is important to note that other factors than weather still give a high risk cost, especially combinations of multiple factors. Figure A.11 shows that the risk cost increases a lot when the GNSS data puts the ship very close to land without reducing the speed. The machinery and control system data are based on multiple sources that describe the system's reliability, and thus have less uncertainty. For weather and obstacles, the main source of uncertainty is the previously mentioned simplifications.

Another source of uncertainty in the risk model is the sensitivity of each input, or how much each input affect the risk cost. It is difficult to say how much weight should be on each input but it is possible to make some general remarks about it based on Figure A.17. For an autonomous ship to function properly, it needs well-functioning machinery, power, and control systems. It also makes sense that sensors providing situation awareness influence the ship, and that weather and obstacles affect the decision-making process. The sensitivity analysis and case study show that all these have a significant effect on the risk cost.

The fuel cost, operation cost, and loss of future income also affect the uncertainty in the case study. Because the SRC makes decisions based on the total cost, the balance between different cost elements affect the decisions and the results. The fuel cost is calculated using a lookup table of how much fuel the ship uses in different environmental conditions and speeds. The table is made by simulating the ship to derive the fuel consumption. These simulations use simplified models of the machinery system, but they still give numbers similar to those for existing ships and engines. Both operation costs and loss of future income are estimated based on the type of ship and operation.

Based on the tests, the balance between safety and efficiency is good. The balance between the different costs is also reasonable. Fuel and operation costs are at the same level. The potential loss of future income is slightly higher than the sum of fuel and operation costs because the ship should have a higher income than just covering the expenses. The results can be improved further by advancing the models, and by getting more and better data, but this is left for future work.

A.4.8 Simplifications in the ship simulator and testing

The proposed methodology and control system is tested using a simplified ship simulator. The simulator is based on the models given in [91]. This provides a good tool to test the ship's control systems. However, the models include simplifications that affect the ship's behavior and control. Not including wave forces is one such simplification. The most commonly used approach to include waves takes a 3D model of the ship and tests it in a hydrodynamic program. However, the data to make this 3D model is missing for the ship in the case study, and therefore the ship is simulated without waves. Similarly, the simulations consider a simplified propulsion system and use approximations in the kinematic and kinetic equations.

In testing, the simulator works sufficiently to test the proposed methodology and SRC. However, the ship is difficult to control when turning, especially using the autopilot. Therefore, the minimum distance used in the safety verification, Section A.3.5, is only 5 m. In real life, the ship should stay further away from land. This would also add more safety margins to the ship draft and more clearance under the keel. Although the system has been tested with a larger minimum distance, it then fails the safety verification at much lower wind speeds. The ship can be operated in DP-mode, which offers much better control at lower speeds using the tunnel thrusters to both control heading and sideways position. However, this would mean sailing at unreasonable low speeds when compared to the conventional ship. To get comparable data, the autonomous ship is allowed to operate with smaller margins in the simulations. Given that the focus of the paper is the method for developing the SRC and how this make high level decisions, this is deemed sufficient. Testing with more accurate ship models is left for further work.

Accuracy in the position data is another challenge when testing the proposed methodology. The case study assumes that the GNSS data is accurate for use in the ship control system. However, GNSS accuracy can be a challenge for autonomous ships, especially when sailing between tall mountains where the signal quality can be affected by bad satellite coverage and signals reflecting off the mountains. How accurate the data is will vary depending on the location, but is something that should be addressed when setting the limits in the system verification and the control system. However, it is still sufficient for testing the SRC and the methodology for building this. Combining GNSS measurements with other sensors, such as radar, LIDAR, sonar, and cameras is an option for improving the accuracy by measuring the distance to land and other objects, instead of just using the GNSS position. However, this is considered to be outside the scope of this work and is left for further work.

A.5 Conclusions

This work presents a control system with risk-based decision-making capabilities to enable the smarter and safer operation of autonomous systems. The proposed control system uses an online risk model, which is represented by a BBN, to evaluate the operational risk, through an SRC. An ENC module is used to provide accurate data of the environment to both the risk model and the rest of the control system. The online risk model provides decision support in the SRC, which can make high level decisions. The control system has been verified against design requirements for safety (minimum distance) and efficiency (maximum time) using a novel formalized verification method. The combination of the SRC with ENC and formalized verification leads to a risk-based control system that can control autonomous ships in a safe and efficient manner, which currently does not exist.

The proposed control system is first compared to experimental data from an existing conventional ship in a case study along two coastal routes. This shows that the novel controller makes similar decisions to adjust the speed and maintain safe operation as the conventional ship, without using significantly more time to reach the end destination. It also shows that the controller took less risks than the conventional ship, mainly by adjusting the speed earlier when maneuvering in narrow areas, while maintaining a higher minimum speed than the conventional ship. This will make a bigger difference for routes that changes a lot, such as the route through Rørvik. However, it will still have an effect on routes with less variation between open water and narrow straits. The second part of the case study tests how the SRC handles failures in the machinery and propulsion system. This shows that the SRC changes MSO-mode and SO-mode to continue safely to the final waypoint.

Further work includes adding more functions to the control system to increase autonomy, such as safe and reliable auto-docking. This will enable the ship to leave harbor, sail to a second location/harbor, deliver goods, and then return and dock in harbor again. This would be a typical cargo ship or passenger operation and would thus be an important step towards achieving highly autonomous ships. Route planning to enable the control system to change route depending on the risk level and environmental conditions, and looking at how a similar system can be used for decision support to human operators are also natural parts of future work.

Appendix: BBN connections

Tables with an overview of child/parent nodes in the BBN.

Table A.5: BBN Nodes, Input-RIFs are only listed as parent nodes

Node description	Parent node(s)
Cost	Consequences
Consequences	Harm to humans, Damage on own ship, Damage on other ships/objects
Damage on other ships or objects	HE, Impact speed, Type of seabed, Type of shore
Damage on own ship	HE, Impact speed, Type of seabed, Type of shore
Harm to humans	HE, Impact speed, Type of shore
HE	H1, H2
H1	UCA-1, UCA-2, UCA-3, UCA-4, UCA-5
H2	UCA-1, UCA-2, UCA-3, UCA-4, UCA-5
UCA-1	H-RIF-1, H-RIF-2, H-RIF-3
UCA-2	H-RIF-1, H-RIF-2, H-RIF-3
UCA-3	H-RIF-2, H-RIF-3
UCA-4	H-RIF-4, H-RIF-5
UCA-5	H-RIF-4, H-RIF-5, H-RIF-6
H-RIF-1	Power, Propulsion
H-RIF-2	Power management system reliability, Controller performance/accuracy
H-RIF-3	Weather conditions, Control of ship, Congested waters
H-RIF-4	Controller performance/accuracy, Navigational instruments
H-RIF-5	Navigational instruments, Visual conditions
H-RIF-6	Controller performance/accuracy Ship design process
Power	PTO, PTI, Mech, MSO-mode
Propulsion	AP, DP
Weather conditions	Current, Wind direction, Wind speed

Table A.6: BBN Nodes, Input-RIFs are only listed as parent nodes

Node description	Parent node(s)
Control of ship	Weather conditions, SO-mode, Ship speed, Propulsion
Congested waters	Obstacle density, Distance to closest grounding hazard, Traffic density
Controller performance or accuracy	AP performance/accuracy, DP performance/accuracy, SO-mode, Weather conditions
Ship speed	Controller performance or accuracy, Speed reference
Navigational instruments	AIS, Radar, GNSS system
Visual conditions	Wind speed, Fog, Rain, Snow
PTO	ME state, HSG state
PTI	HSG state, DG1 state, DG2 state
Mech	ME state, DG1 state, DG2 state
AP	MSO-mode, MP state, ST state
DP	MSO-mode, MP state, BT state, AT state
Impact speed	Ship speed

Appendix B

Ship collision avoidance and anti-grounding

B.1 Introduction

B.1.1 Background

Autonomous ships will require a high level of data processing in order to have adequate situational awareness and to make deliberate decisions. This requires efficient and robust algorithms, and well chosen platforms to enable fast computation. When facing a hazardous situation in e.g. confined space with multiple static and dynamic obstacles, the need to evaluate a larger set of future control behaviours or trajectories for the autonomous ship and other obstacles will be necessary, such that a risk minimizing or collision-free trajectory is possible to find. Furthermore, the collision-free planned trajectory should comply to the Convention on the International Regulations for Preventing Collision at Sea (COLREGS) [193] when possible. However, evaluating the risk associated in any of these control behaviours can be computationally expensive. Thus, to meet run-time requirements, a collision avoidance (COLAV) planning algorithm which scales well in the evaluation of different control behaviours will be both beneficial and necessary in such cases. Increased robustness can then also result as a consequence of being able to evaluate more vessel behaviour scenarios and situational information in the system at run-time.

A Scenario-Based Model Predictive Control (SB-MPC) [194] approach is here a viable option that can incorporate most of the elements needed in a robust COLAV planning algorithm, such as anti-grounding, dynamic obstacle avoidance and multi-ship adherence to COLREGS when possible. This is because of its flexibility in the formulation of its optimization problem, with different control objectives and possible integration of constraints, and which has a rich theoretical foundation. The sampling-based method is also flexible in the prediction models used to generate own-ship and dynamic obstacle prediction scenarios. The problem with this approach however, and especially for the probabilistic version (PSB-MPC) [195], is that the optimization problem in the COLAV planning algorithm scales poorly with an increasing set of considered own-ship avoidance maneuvers, static obstacles and dynamic obstacles with their own alternative prediction scenarios. The prediction of the collision risk with respect to all dynamic and uncertain obstacles involved, and calculating distances to all static obstacles for anti-grounding purposes, has exponentially increasing computational cost as the optimization problem increases.

B.1.2 Literature review

Many studies on maritime collision avoidance exists today, and are mainly summarized in review papers such as [71], [72], [196], [197], whereas we here focus on deliberative COLAV planning methods having dynamic obstacle avoidance and COLREGS adherence in addition to anti-grounding in their algorithms. For a general overview on planning algorithms, see [198].

In this article, deliberative refers to the COLAV algorithm planning efficient trajectories that adheres to the COLREGS when deemed possible, and avoid collision well before risky situations occur. Following the COLREGS blindly in any type of situation will not be sufficient, as was shown in [199] and also discussed in [200], [201]. Thus, the deliberate COLAV planning algorithm should in general also consider the intention uncertainties of nearby dynamic obstacles. Further note that with COLREGS compliance, we mean compliance with the COLREGS rules 8, 13 - 17 on taking early and apparent action, and the correct action in overtaking, head-on and crossing situations with either give-way or stand-on obligations, respectively. These are the rules most relevant and common to consider for automatic COLAV planning. However, a complete COLAV system should consider the full rule set.

A lattice-based trajectory planner using A* search for finding collision-free trajectories is introduced in [202], where non-adherence to the COLREGS, trajectory deviation and collision risk with respect to static and dynamic obstacles is penalized in the cost function. An intention based motion model is used for dynamic obstacles, which relies on learning the positional prediction uncertainty for a given scene when used in calculating collision probabilities. The details on this model is not given, and results on how the planner scales in run-time with increasing lattice grid density, dynamic and static obstacles are however not given.

The work in [203] introduces a hierarchical system with three levels. The top level trajectory planner uses lattice-based A* search combined with an Optimal Control Problem (OCP) method for generating collision-free trajectories with respect to static obstacles. A mid level MPC-based COLAV planning algorithm modifies this trajectory to adhere to the COLREGS and avoid collisions with respect to dynamic obstacles. Lastly, a low-level reactive COLAV sampling-based planning algorithm acts as a fail-safe in case the levels above can not handle the situation. The system does however assume straight line trajectories for dynamic obstacle predictions without uncertainty, which does not coincide with real-time vessel behaviour in hazardous situations. Furthermore, scalability and run-time properties with an increasingly complex situation is not discussed.

In [204], a field-test verified A-star search trajectory planner is developed, which attempts to find a COLREGS-compliant and collision-free trajectory with respect to dynamic and static obstacles in a lattice. To predict nearby dynamic obstacle trajectories, the planner employs Monte-Carlo (MC) simulation using fuzzy logic and the trajectory history of the obstacle to find a set of probable trajectories, where the most probable one is considered for collision avoidance. As in [203], the method does not consider the prediction uncertainty associated with dynamic obstacles. Furthermore, the computational efficiency of the planner only tested for a set of 500 possible own-ship trajectories and one expected dynamic obstacle trajectory, which can be inadequate in highly congested scenarios.

Candeloro et. al. 2017 [23] propose a global and local lattice-based trajectory planner which uses Voronoi Diagrams to generate a set of static obstacle collision-free waypoints, from where a continuous trajectory is generated using Fermat‘s Spiral. The method considers local replanning windows for taking detected dynamic and static obstacles into account, and predicts dynamic obstacle motion with the Constant Velocity (CV) model [205]. A convex hull representing the dynamic obstacle uncertainty up until time to Closest Point of Approach (CPA) is created from using the position estimates and error covariances from a Kalman filter, which is then regarded as an area to avoid in the planner. This may however be overly conservative, due to the unrealistic uncertainty growth in the CV model [206]. How the local replanning run-time scales with increasing windows size, dynamic and static obstacles is not considered.

Nonlinear MPC for static and dynamic obstacle collision avoidance with environmental disturbance rejection was proposed in [207]. A deterministic CV model was used for the dynamic obstacle prediction, which will not be the case in real-time hazardous maritime situations where ships will maneuver. Furthermore, how the MPC scales with static and dynamic obstacles was not considered.

Chiang and colleagues [208] introduces a sampling-based static and dynamic obstacle considerate trajectory planner with COLREGS-compliant COLAV planning algorithm based on Rapidly exploring Random Trees (RRT), where a joint simulator is used to predict both the own-ship and dynamic obstacle motion. Potential fields are used in the prediction to ensure that all the vessels have collision-free trajectories with respect to each other and static obstacles. The method is shown to have beneficial run-times feasible for real-time. However, the underlying assumption in the prediction is however that ships will always perform deterministic COLREGS-compliant maneuvers if possible, which is not necessarily true in practice.

Collision avoidance within a distributed flocking control strategy based on MPC was considered in [209], with respect to nearby dynamic vehicles in the flock and static obstacles. The computational efficiency or scalability of the method was however not discussed, and the states of all vehicles involved are assumed deterministic.

B.1.3 Contributions

In this work, an implementation of the sampling-based PSB-MPC algorithm on a GPU platform which facilitates efficient anti-grounding and dynamic obstacle avoidance is introduced. The main contribution of the article compared to current state-of-the-art static and dynamic obstacle COLAV planning algorithms is the description of a parallelization algorithm for efficient cost evaluation of possible own-ship trajectories in the PSB-MPC, taking into account dynamic obstacle uncertainties and complex static obstacles in maritime hazardous situations. The algorithm is feasible for real-time, as the MPC cost function evaluation scales linearly with increasing numbers of dynamic obstacles with their own prediction scenarios and also static obstacles, due to the parallelization. Static obstacles are read in from Electronic Navigational Chart (ENC) data and processed into simplified polygons using the Ramer-Douglas-Peucker (RDP) algorithm [210]. The efficiency of the parallelized implementation makes it possible for the COLAV planning algorithm to consider more dynamic obstacle prediction scenarios and own-ship trajectories, and more complex static obstacle maps for elevated situational awareness and better trajectory planning. Furthermore, a side contribution of the article is that the dynamic obstacle prediction scheme in [195] is updated to use a kinematic model with incorporated Line-of-Sight (LOS) guidance for more realistic trajectories.

B.1.4 Article structure

The article is organized as follows. Section B.2 gives background information about the PSB-MPC, with prediction models, cost function structure and grounding hazard extraction and representation. An outline of a sequential implementation of the algorithm is also given. A parallelized implementation of the PSB-MPC is given in Section B.3. Finally, Section B.4 show simulation results with the PSB-MPC, and Section B.5 concludes the work.

B.2 The PSB-MPC COLAV planning algorithm

The Probabilistic Scenario-based Model Predictive Control (PSB-MPC) is an optimization-based COLAV planning method that samples a finite set of possible own-ship trajectories, represented by control behaviours. This is illustrated in Figure B.1, where we note that the control behaviours selected are arbitrary.

Formally, a control behaviour l in the PSB-MPC represents a sequence $[(U^l_{m,1}, \chi^l_{m,1})$ $..., (U^l_{m,n_M}, \chi^l_{m,n_M})]$ consisting of speed multiplicative factors U_m and additive course angle offsets χ_m. The sequence represent n_M sequential avoidance maneuvers. The parameter n_M can in general be a variable, but will be considered fixed in this work. The sequence of speed and course modifications are applied to the autopilot references U_d and χ_d in speed and course angle at different time steps in the finite prediction horizon, which in turn generates a specific own-ship trajectory. Each control behaviour is evaluated by a cost function $\mathcal{H}^l(\cdot)$, which penalizes probabilistic collision risk, grounding risk, COLREGS violation and nominal trajectory

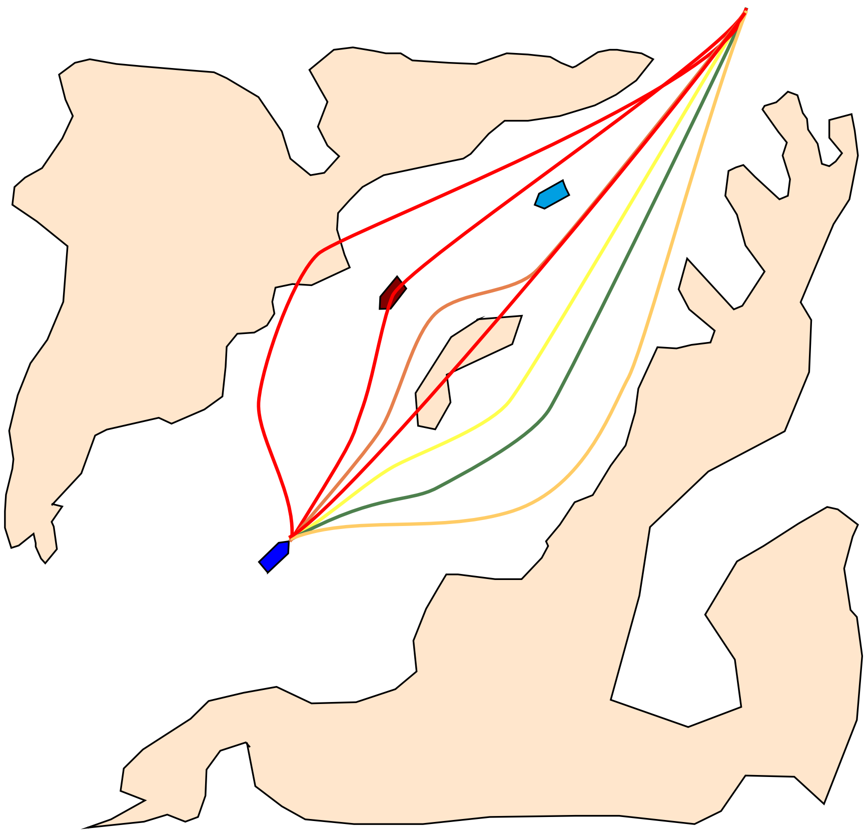


Figure B.1: PSB-MPC illustration, with the own-ship running the algorithm in blue. Nearby dynamic obstacles are shown in cyan and brown. Grounding hazards are shown in beige. Candidate control behaviours predicted in the MPC are also shown, where the color from red to green represents their cost, with green being the lowest. Thus, the green candidate trajectory is the optimal one. The nominal trajectory goes straight north-east through the confined environment.

deviation. The optimal one is selected as

$$l^*(t_0) = \arg\min_{l} \mathcal{H}^l(t_0) \tag{B.1}$$

where t_0 is the current time, and it is the first avoidance maneuver represented by $U^l_{m,1}$ and $\chi^l_{m,1}$ that is applied by the autopilot through the modified guidance references $U_c = U^{l^*}_{m,1} \cdot U_d$ and $\chi_c = \chi^{l^*}_{m,1} + \chi_d$. The open loop optimization based on (B.1) is repeated at regular intervals to account for more information in a moving horizon fashion as is common in MPC, and thus closes the loop. More discussion around feasibility and constraint satisfaction in the PSB-MPC can be found in [168].

B.2.1 Prediction models

For deliberate COLAV planning algorithms, the own-ship prediction model can be selected to be complex or simple depending on how much vessel information one has. The PSB-MPC can easily handle complex ship motion models in its framework. However, as the kinematic uncertainty associated with the ship motion prediction increases substantially with time, having a simple model to capture the approximate own-ship behaviour is often adequate for deliberative COLAV planning algorithms, where the low-level vessel control systems (autopilot) can compensate for model inaccuracies and disturbances. As predictions of vessel motions over longer time horizons are inherently uncertain, due to environmental disturbances, future maneuvering decisions and unforeseen events, especially in hazardous situations, we argue that there is limited gain in using an overly complex model. On the other hand, for reactive collision avoidance methods and lower level motion control with shorter prediction horizons, it will be more important to consider the ship dynamics accurately. Thus, as the PSB-MPC is flexible in the choice of the own-ship prediction model, it will not be described here but in the simulation study in Section B.4.

Therefore, the following text will detail the model used for dynamic obstacles. To create trajectories simulating the ship motion for the own-ship and dynamic obstacles forward in time, we use Euler‘s method for numerical integration. Specifically, the integration is done over the prediction horizon with discrete predicted times $t_k \in D(t_0) = \{t_0, ..., t_0 + k\Delta_{mpc}, ..., T_{mpc}\}$, with Δ_{mpc} as the time step and T_{mpc} as the prediction horizon.

As one most often do not have information on the underlying dynamic obstacle vessel or object, their motion models should be simple. The preliminary PSB-MPC used the Ornstein-Uhlenbeck (OU) process [211] in order to predict the motion of dynamic obstacles, and allows for alternative obstacle prediction scenarios [169]. However, the trajectories only specify a single change in course, and are thus not necessarily realistic. A more realistic approach as shown in Figure B.2 is now used, where more avoidance like maneuvers are used.

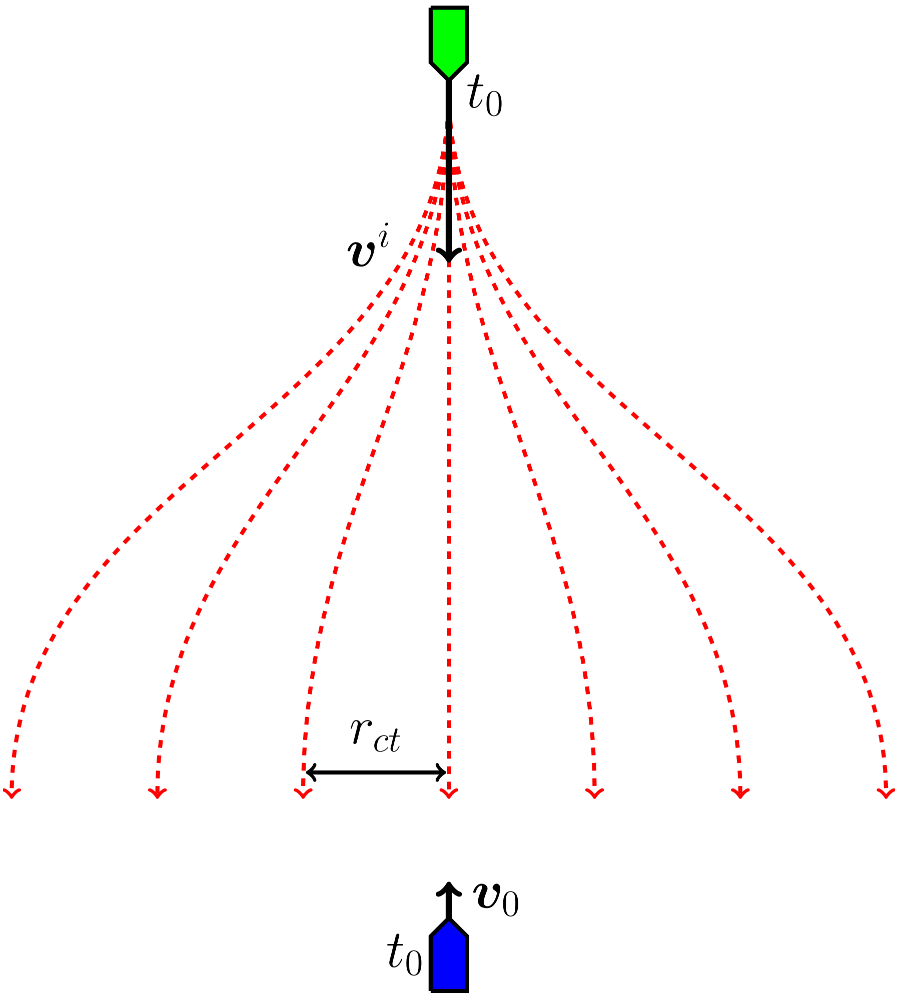


Figure B.2: Head-on scenario with obstacle i in green and own-ship in blue. Their velocity vectors $\boldsymbol{v}^i$ and $\boldsymbol{v}_0$, respectively, are also shown. The updated prediction scheme using LOS guidance allows for the obstacle to make realistic alternative maneuvers to port and starboard. The stationary time spacing between trajectories is determined by r_{ct}.

The predicted obstacle motion is implemented using the following kinematic model

$$\begin{aligned} x_{k+1}^i &= x_k^i + U_k^i cos(\chi_k^i) \\ y_{k+1}^i &= y_k^i + U_k^i sin(\chi_k^i) \\ \chi_{k+1}^i &= \chi_k^i + \frac{1}{T_\chi}(\chi_{d,k}^i - \chi_k^i) \\ U_{k+1}^i &= U_k^i + \frac{1}{T_U}(U_{d,k}^i - U_k^i) \end{aligned} \tag{B.2}$$

where the superscript i is used for dynamic obstacles. The above kinematic model is combined with Line-of-Sight (LOS) guidance [212] to predict the following of a nominal obstacle path parameterized by n_{wps} waypoints WPS: $[\boldsymbol{p}_1, ..., \boldsymbol{p}_z, ..., \boldsymbol{p}_{n_{wps}}]$, where $\boldsymbol{p}_z = [x_z^{wp}, y_z^{wp}]^T$ is waypoint $z \in \{1, 2, ..., n_{wps}\}$. In the case of straight line paths, the LOS guidance method considers waypoint segments from $\boldsymbol{p}_z$ to $\boldsymbol{p}_{z+1}$, and finds the path tangential angle

$$\alpha_z = atan2(y_{z+1}^{wp} - y_z^{wp}, x_{z+1}^{wp} - x_z^{wp}) \tag{B.3}$$

and path-fixed frame referenced path deviation $\boldsymbol{\epsilon}_k$ with rotation α_z as

$$\boldsymbol{\epsilon}_k^i = \boldsymbol{R}_{\alpha_z}^T(\boldsymbol{p}_k^i - \boldsymbol{p}_z) \tag{B.4}$$

where $\boldsymbol{p}_k^i = [x_k^i, y_k^i]^T$ is the position of obstacle i at time t_k. The rotation matrix $\boldsymbol{R}_{\alpha_z}$ is given by

$$\boldsymbol{R}_{\alpha_z} = \begin{bmatrix} cos(\alpha_z) & -sin(\alpha_z) \\ sin(\alpha_z) & cos(\alpha_z) \end{bmatrix} \tag{B.5}$$

The along-track error s_k^i and cross-track error e_k^i, see [91] for more details, makes up the path deviation $\boldsymbol{\epsilon}_k^i = [s_k^i, e_k^i]^T$, where the latter error is used with (B.3) to calculate the desired obstacle COG as

$$\chi_{d,k}^i = \alpha_z + arctan\left(-\frac{e_k^i}{\Delta^i}\right) \tag{B.6}$$

with Δ^i as the lookahead distance, dependent on the obstacle ship type. See [212] for illustrations and more information. The combination of a kinematic model used with LOS guidance allows for a lightweight prediction of alternative dynamic obstacle maneuvering scenarios. By also specifying a speed profile through a desired SOG $U_{d,k}^i$ in addition to the LOS guidance, one goes from path-generation to trajectory generation for the dynamic obstacles [91].

The PSB-MPC normally gets dynamic obstacle information from the tracking system, where their state estimates have an associated kinematic uncertainty, typically represented through a covariance matrix $\boldsymbol{P}^i(t_0)$. The obstacle kinematic uncertainty is here predicted forward in time *heuristically* using an OU-process [206] with mean velocity taken as the current state estimated velocity:

$$\boldsymbol{P}_{k+1}^i = \boldsymbol{P}_0^i + \boldsymbol{\Sigma}_1 \circ \boldsymbol{\Sigma}_2(t_{k+1} - t_0) \tag{B.7}$$

with $\boldsymbol{P}_k^i$ as the predicted covariance and

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} \frac{\sigma_x^2}{\gamma_x^3} & \frac{\sigma_{xy}}{\gamma_x\gamma_y} & \frac{\sigma_x^2}{2\gamma_x^2} & \frac{2\sigma_{xy}}{\gamma_x} \\ \frac{\sigma_{xy}}{\gamma_x\gamma_y} & \frac{\sigma_y^2}{\gamma_y^3} & \frac{2\sigma_{xy}}{\gamma_y} & \frac{\sigma_y^2}{2\gamma_y^2} \\ \frac{\sigma_x^2}{2\gamma_x^2} & \frac{2\sigma_{xy}}{\gamma_y} & \frac{\sigma_x^2}{\gamma_x} & \frac{2\sigma_{xy}}{\gamma_x+\gamma_y} \\ \frac{2\sigma_{xy}}{\gamma_x} & \frac{\sigma_y^2}{2\gamma_y^2} & \frac{2\sigma_{xy}}{\gamma_x+\gamma_y} & \frac{\sigma_y^2}{\gamma_y} \end{bmatrix} \tag{B.8}$$

as the stationary process noise part of the process, with σ_x, σ_{xy} and σ_y as the OU model Wiener process noise parameters. The parameters γ_x and γ_y are the reversion strength parameters, which determines the convergence rate of the OU-process towards its mean velocity. The expression for $\boldsymbol{\Sigma}_2(t_{k+1} - t_k)$ can be found in [211]. The symbol $\circ$ in (B.7) denotes the Hadamard product. The reason behind the usage of the OU-process for uncertainty prediction is its more limited growth in covariance compared to e.g. using a CV model [211]. The 3σ positional uncertainty is heuristically bounded by r_{ct} in the prediction, such that each obstacle trajectory has a tube uncertainty with approximate radius r_{ct}. As for the own-ship, the parameters for the obstacle prediction are all dependent on the type of ship, ship control system, ship captain etc., and should be estimated using available data about the obstacle.

In this article we assume that the nominal obstacle trajectory is a straight line from its current course, and create waypoints on this line. Vessel to vessel communication or e.g. road map methods [213] may be used to predict the nominal obstacle trajectories in more confined spaces where the straight line trajectory assumption is restrictive. An illustration of the uncertainty prediction together with the dynamic obstacle trajectory is shown for a case with a non-straight line obstacle trajectory in Figure B.3.

B.2.2 Grounding hazards

The grounding hazards considered in the PSB-MPC are parameterized as two-dimensional polygons. In this article, polygons are read in from shapefiles using the C based library Shapefile C Library, which are generated using the Electronic Navigational Chart processing module in SeaCharts corresponding to the relevant map region considered [37]. If real-time sensor data is available, this can also be used to update the polygons used in the MPC.

Because electronic map data can have high accuracy, larger polygons extracted can have tens of thousands of vertices. However, for collision avoidance, this level of detail is not necessary, and the polygons should thus be simplified in order to save computation time in the algorithm. One of the earliest and most common curve simplification methods that can be used for polygon simplification is the RDP al-

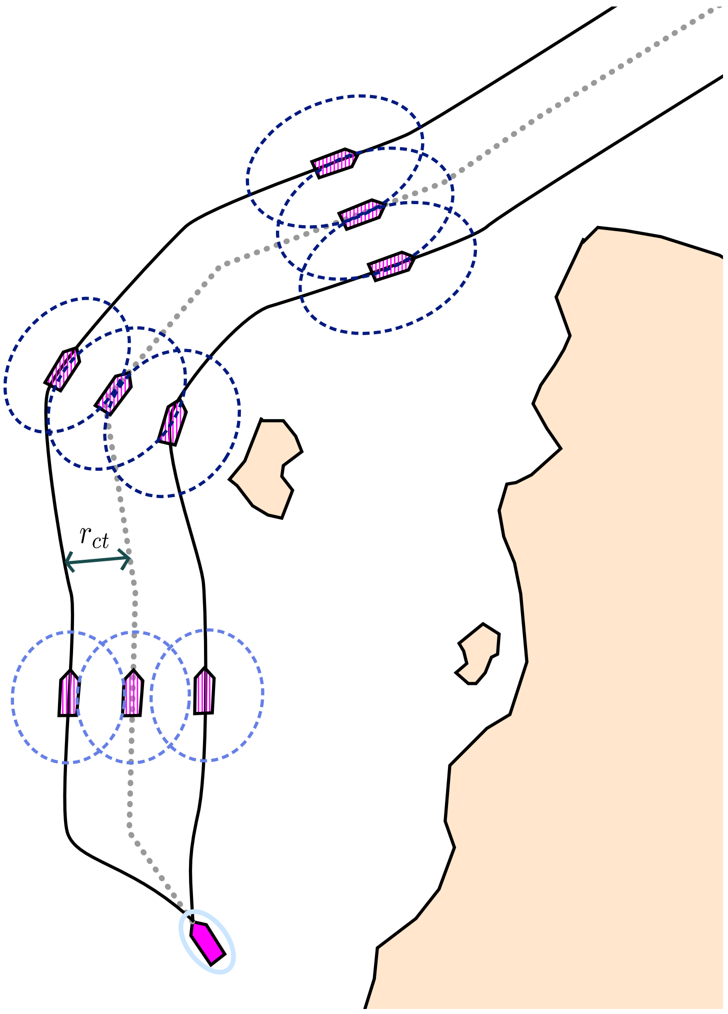


Figure B.3: Dynamic obstacle prediction illustration with an obstacle in purple. Three prediction scenarios are shown, all starting at t_0, where the vessel is depicted in full purple with its tracked estimation error covariance represented around it as a 3σ probability ellipse in light blue. The nominal predicted obstacle trajectory is shown with the grey dotted line, whereas the alternative scenarios are spaced r_{ct} apart.

Algorithm 9 The Ramer-Douglas-Peucker curve simplification algorithm.

```
1: function RDP($Points$, $\epsilon_{rdp}$)
2:     $d_{max} \leftarrow 0$, $j \leftarrow 1$, $end \leftarrow length(Points)$.
3:     for $i = 2, .., end$ do
4:         $d \leftarrow perpendicularDistance(Points[i], Points[1], Points[end])$
5:         if $d > d_{max}$ then
6:             $j \leftarrow i$, $d_{max} \leftarrow d$
7:         end if
8:     end for
9:     if $d_{max} > \epsilon_{rdp}$ then
10:        $rResults1 = \mathrm{RDP}(Points[1, .., j], \epsilon)$
11:        $rResults2 = \mathrm{RDP}(Points[j, .., end], \epsilon)$
12:        $newPoints = \{rResults1[1, .., length(rResults1) - 1], rResults2\}$
13:    else
14:        $newPoints = \{Points[1], Points[end]\}$
15:    end if
16:    return $newPoints$
17: end function
```

gorithm [210]. The method recursively simplifies a curve of points by consecutively considering its line segments, pruning away points which are further away from the considered line segment than a specified threshold ϵ_{rdp}. The distance tolerance parameter ϵ should be chosen as not to overly simplify the polygons, preserving as much structure as possible. The method is summarized in Algorithm 9. A graphical illustration of the algorithm can be found in https://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm. The distance from the own-ship center to nearby polygons is used in the PSB-MPC grounding cost. It is obtained by using a point to polygon calculation method [214], using the ray intersection method for determining if the own-ship center point is inside the polygon, which is suitable for both convex and concave polygons.

B.2.3 Cost function reformulation

We consider the following restructuring of the PSB-MPC cost function

$$\mathcal{H}^l(t_0) = \mathcal{H}^l_{do} + \mathcal{H}^l_{colregs} + \mathcal{H}^l_{so} + \mathcal{H}^l_p \tag{B.9}$$

for a control behaviour l, where the four terms are the cost associated with dynamic obstacles, COLREGS violation, static obstacles or grounding hazards and trajectory tracking, respectively. The dynamic obstacle related cost is here reformulated to

$$\mathcal{H}^l_{do} = \sum_{i=1}^{n_{do}} w^i \mathcal{H}^{l,i}_{do} \tag{B.10}$$

where w^i represent the weight of the cost from obstacle i, in general influenced by factors such as distance, bearing, nearby grounding hazards and vessel-vessel communication. If no prior information is used, it is set to $w^i = 1$. The dynamic obstacle i cost is given by

$$\mathcal{H}_{do}^{l,i} = \sum_{s=1}^{n_{ps}^i} \mathbb{P}_s^i C_s^{l,i} \tag{B.11}$$

where n_{ps}^i is the number of prediction scenarios for the obstacle, $\mathbb{P}_s^i$ represent the associated prediction scenario probabilities from an intention inference module [169], and $C_s^{l,i}$ is the cost involving prediction scenario s for obstacle i, given as

$$C_s^{l,i} = \max_k \zeta_i \mathcal{C}_{i,k}^{l,s} \hat{\mathbb{P}}_{c,k}^{l,i,s} exp(-t_k/T_d) \tag{B.12}$$

which is taken as the maximum of the probabilistic collision risk, involving the relative kinetic energy term $\mathcal{C}_i^{l,s}(t) = K_{coll}||\boldsymbol{v}_k^i - \boldsymbol{v}_k||^2$ between the obstacle i velocity $\boldsymbol{v}_k^i$ and own-ship velocity $\boldsymbol{v}_k$, with parameter K_{coll}, [194]. $\hat{\mathbb{P}}_{c,k}^{l,i,s}$ is the collision probability estimate calculated using the Cross-Entropy method, see [195] for more details. The track loss modifier ζ_i, [215] takes into account cases when dynamic obstacle tracks are lost for some time. Lastly, an exponential discounting term with time constant T_d gives lower weighting of collision events far ahead in the future.

The intention uncertainty of a dynamic obstacle is represented through the scenario probabilities $\mathbb{P}_s^i$ for each considered obstacle prediction scenario. Given a representable set of obstacle prediction scenarios, we are able to cover most anticipated obstacle maneuvering cases because we predict the uncertainty for each of the scenarios. These probabilities of different target ship plans or trajectories are typically inferred by an intention model as in [169], [216], and can be used for having elevated situational awareness in the planner. Furthermore, the probabilities are an adequate way of taking into account intention information, as they are easy to interpret, can be used to define risk and leads to a natural way of weighting the collision risk associated with different decision candidates for an obstacle ship. The downside is that one needs a validated intention inference model, and a sufficient set of dynamic obstacle prediction scenarios in order to have meaningful estimates.

To favor COLREGS compliance in multi-ship situations, the COLREGS related cost is now separated into its own term in the PSB-MPC, and given as

$$\mathcal{H}_{colregs}^{l} = \kappa \sum_{i=1}^{n_{do}} w^i \mu^{l,i} \tag{B.13}$$

where κ is a tuning parameter and

$$\mu^{l,i} = \sum_{s=1}^{n_{ps}^i} \mathbb{P}_s^i \mu_s^{l,i} \tag{B.14}$$

with $\mu_s^{l,i} \in \{0, 1\}$ as the indicator of the own-ship following control behaviour l violating COLREGS with respect to obstacle i in prediction scenario s, calculated as in [194] for head-on, overtaking and crossing situations. The parameters w^i and dynamic obstacle scenario probabilities are again used for weighting purposes. The new formulation now penalizes COLREGS breaches with respect to all dynamic obstacles, and allows for better handling of compliance in multi-ship situations.

The static obstacle related cost or grounding cost is parameterized as

$$\mathcal{H}_{so}^{l} = \max_{j} \mathcal{H}_{so}^{l,j} \tag{B.15}$$

where

$$\begin{aligned}\mathcal{H}_{so}^{l,j} = \max_{k} \quad & (G_1 + G_2 \phi_{j,k}^{l} V_w^2) \\ & \times exp(-(G_3 |d_{0j,k}^{l} - d_{safe}| + G_4 t_k))\end{aligned} \tag{B.16}$$

inspired by [38], where G_1 to G_4 are tuning parameters, V_w the estimated wind speed, $\phi_{j,k}^{l} = \max(0, \boldsymbol{\omega}_j \cdot \boldsymbol{L}_{0j,k}^{l})$ with $\boldsymbol{\omega}_j$ as the wind direction unit vector. $\boldsymbol{L}_{0j,k}^{l}$ is the unit vector pointing from the own-ship to the static obstacle j, and $d_{0j,k}^{l}$ the corresponding distance. d_{safe} is the circular own-ship safety zone.

The trajectory deviation cost is given as

$$\mathcal{H}_{p}^{l} = \frac{1}{n_M} \sum_{M=1}^{n_M} f(\cdot) + \frac{1}{n_M - 1} \sum_{M=2}^{n_M} h(\cdot) \tag{B.17}$$

with $f(\cdot)$ and $h(\cdot)$ as the control deviation and change cost, respectively. More details on the different terms involved in the cost function can be found in [33], [168], [169], [194], [215].

B.2.4 Standard PSB-MPC implementation

As the PSB-MPC is a finite set MPC, the solution to the non-convex Mixed Integer Nonlinear Program (MINLP) in (B.1) is parameterized by the chosen discrete set of own-ship control behaviours. The benefit of the finite-set MPC formulation is that by brute force iterating over the set of control behaviours we are able to find a global solution, which would be hard in the case if numerical optimization was used.

Implementing the cost evaluation in the PSB-MPC on a sequential computing platform will involve loops over the own-ship control behaviours, where loops over static and dynamic obstacles in their set of prediction scenarios are found within. This would look something like the method outlined in Algorithm 10. One can see that this implementation involves several nested for loops, especially the one over dynamic obstacles and their prediction scenarios. In addition, one must also loop over the number of discrete samples $t_k \in D(t_0)$ in the predicted trajectories. Thus, the MPC problem will scale poorly with increasing number of control behaviours, static and dynamic obstacles.

Algorithm 10 Standard PSB-MPC cost evaluation on a sequential processing platform, assuming all obstacle prediction scenarios are generated beforehand.

```
1: Initialize optimal control behaviour to $l^* = 1$.
2: for $l = 1, .., n_{cbs}$ do
3:     Predict the own-ship trajectory following control behaviour $l$.
4:     Calculate the trajectory related cost $\mathcal{H}_p^l$ using (B.17).
5:     for $j = 1, .., n_{so}$ do
6:         Calculate the static obstacle $j$ grounding cost $\mathcal{H}_{so}^{l,j}$ using (B.16).
7:     end for
8:     Calculate total grounding cost $\mathcal{H}_{so}^l$ using (B.15).
9:     for $i = 1, .., n_{do}$ do
10:        for $s = 1, .., n_{ps}^i$ do
11:            Calculate probabilistic collision cost $C_s^{l,i}$ from (B.12) and COLREGS indicator $\mu_s^{l,i}$ in (B.14).
12:        end for
13:        Calculate dynamic obstacle $i$ cost $\mathcal{H}_{do}^{l,i}$ using (B.11).
14:    end for
15:    Calculate total dynamic obstacle cost $\mathcal{H}_{do}^l$ using (B.10).
16:    Calculate control behaviour cost $\mathcal{H}^l(t_0) = \mathcal{H}_{do}^l + \mathcal{H}_{colregs}^l + \mathcal{H}_{so}^l + \mathcal{H}_p^l$.
17:    if $\mathcal{H}^l(t_0) < \mathcal{H}^{l^*}(t_0)$ then
18:        Set $l^* = l$.
19:    end if
20: end for
```

B.3 Parallelized PSB-MPC implementation

The nature of the finite set MPC described in the above section makes it possible to independently evaluate the cost associated with the control behaviours, and thus apply parallelism in the main part of the algorithm. Furthermore, all obstacle prediction scenarios are assumed to be independent of the own-ship control behaviour and can be generated beforehand. This is deemed reasonable as we take into account maneuvering uncertainty in the obstacle prediction.

When considering large amounts of situational information and a dense set of possible own-ship trajectories, evaluating the cost of an own-ship control behaviour sequentially will not make the COLAV planning algorithm real-time feasible. Parallelizing the cost evaluation will allow for more refined own-ship decision making, as more own-ship trajectories can be considered. Also, more static obstacles and prediction scenarios for dynamic obstacles can then be considered, resulting in increased situational awareness for the own-ship. The limiting factor here will then be how many threads that can be scheduled on the parallel computation platform.

A naive way of cost function evaluation parallelization would be to schedule GPU threads to evaluate the cost (B.9). However, this is a big task for a single thread, as it among others involves going through all static and dynamic obstacles in all their prediction scenarios to find the total cost. This equates to a nested for loop over obstacles, prediction scenarios and discrete time samples in the code that implements the MPC as in Algorithm 10, and will scale poorly with an increase in the number of obstacles and number of prediction scenarios n_{ps}^i for dynamic obstacles. As GPU cores have limited processing power compared to CPU cores, their tasks should be as lightweight as possible.

Two of the main bottlenecks in the cost evaluation is calculating the distance to static obstacles and the estimation of collision probabilities. The first bottleneck is readily apparent when considering large polygons with tens of thousands of vertices. However, the RDP algorithm will reduce the number of vertices in a polygon and thus alleviate computational effort. Reducing the number of time steps to evaluate the grounding cost can also aid in fixing this problem.

For the second bottleneck, giving each thread the job of estimating collision probabilities associated with only a pair of trajectories will give higher throughput, at the cost of scheduling more threads on the GPU and therefore having higher memory demands. However, as GPU technology continue to improve with respect to single core processing power and device memory, this is deemed a worthy trade-off. Furthermore, the calculation efficiency using the Cross-Entropy method for collision probability estimation [195] is increased by estimating $\hat{\mathbb{P}}_c^{l,i,s} \approx 0$ when the predicted distance between the own-ship and an obstacle is larger than $d_{safe} + 4\sigma_{largest}^i$, where $\sigma_{largest}^i$ is the standard deviation along the axis where obstacle i has the largest predicted positional uncertainty.

Thus, a way to solve the bottlenecks in (B.1) utilizing parallel processing can be done in two steps: First schedule n_{cbs} threads to predict the own-ship trajectory and calculate the trajectory related cost (B.17) for each control behaviour $l = 1, 2, .., n_{cbs}$. Then, schedule

$$n_{ct} = n_{cbs} \cdot (n_{so} + \sum_{i=1}^{n_{obst}} n_{ps}^i) \tag{B.18}$$

threads that evaluates the cost (B.16), (B.12) and the COLREGS violation indicator in (B.14). The total cost (B.9) is finally stitched together afterwards on the CPU. This way, no GPU thread has run-times dependent on large nested for-loops, and the MPC-problem scales better with increasing number of obstacles and dynamic obstacle prediction scenarios. This approach of using parallelization to solving (B.1) can be summarized in Algorithm 11. Here, “parfor” denotes a parallel for loop.

Note that how the PSB-MPC algorithm is implemented both on the CPU and GPU will have big impacts on the run-time results obtained in this article. Hardware, programming language and software libraries used will be significant factors here. An alternative to the structure in Algorithm 11 would be to have separate kernels to evaluate the static and dynamic obstacle partial costs. This could be better suiting for a setup with multiple GPUs, as the two kernels could then be run concurrently.

Algorithm 11 Parallelized PSB-MPC cost evaluation, assuming all obstacle prediction scenarios are generated beforehand.

```
1: Schedule $n_{cbs}$ GPU threads, transferring all the required data for own-ship trajectory prediction and calculating (B.17).
2: parfor $l = 1, .., n_{cbs}$ do
3:     Predict the own-ship trajectory following control behaviour $l$, save trajectory in GPU memory for use by the subsequent processing.
4:     Calculate the trajectory related cost $\mathcal{H}_p^l$ using (B.17).
5:     Return the results to CPU memory.
6: end parfor
7: Schedule $n_{ct}$ GPU threads, transferring all the required data needed for partial static and dynamic obstacle cost evaluation.
8: parfor $ct = 1, .., n_{ct}$ do
9:     Extract control behaviour $l$, static obstacle $j$ or dynamic obstacle $i$ and prediction scenario $s$ to consider.
10:    Calculate the grounding cost $\mathcal{H}_{so}^{l,j}$ using (B.16) or $C_s^{l,i}$ using (B.12) and the indicator $\mu_s^{l,i}$, depending on if a static or dynamic obstacle is considered in the thread.
11:    Return the results to CPU memory.
12: end parfor
13: Use all the calculated $\mathcal{H}_{so}^{l,j}$ to calculate $\mathcal{H}_{so}^l$ using (B.15).
14: Use all the calculated $C_s^{l,i}$ and $\mu_s^{l,i}$ plus other relevant data to calculate $\mathcal{H}_{do}^l$ using (B.10) and $\mathcal{H}_{colregs}^l$ using (B.13).
15: Finally, calculate (B.9) for all control behaviours using the previously calculated terms $\mathcal{H}_{do}^l$, $\mathcal{H}_{colregs}^l$, $\mathcal{H}_{so}^l$ and $\mathcal{H}_p^l$ of the cost function, and extract the optimal one $l^*$ giving minimal cost.
```

Lastly, because of the extra latency overhead due to porting data from the host (CPU) to the device (GPU), as much memory as possible for the relevant data needed on the GPU should be pre-allocated.

B.4 Simulation study

B.4.1 Own-ship model

In this article we also use a kinematic model with LOS guidance [212] and a constant speed profile for the own-ship, as used for dynamic obstacles in Section B.2.1, to predict any of the candidate trajectories shown in Figure B.1. Specific to the own-ship, the model is restated as

$$\begin{aligned} x_{k+1} &= x_k + U_k cos(\chi_k) \\ y_{k+1} &= y_k + U_k sin(\chi_k) \\ \chi_{k+1} &= \chi_k + \frac{1}{T_\chi}(\chi_{d,k} - \chi_k) \\ U_{k+1} &= U_k + \frac{1}{T_U}(U_{d,k} - U_k) \end{aligned} \tag{B.19}$$

which describes the own-ship state $\boldsymbol{x}_k = [x_k, \quad y_k, \quad \chi_k, \quad U_k]^T$ motion at time t_k. Again, the state consists of the vessel surface position in Cartesian coordinates, course over ground (COG) and speed over ground (SOG), respectively. The time constants T_U and T_χ in speed and course may be found by applying parameter identification methods using motion data from the considered vessel.

For each own-ship control behaviour, the speed modifications $u^l_{m,M}$ and course modifications $\chi^l_{m,M}$ for all n_M sequential maneuvers considered in the PSB-MPC are applied to the LOS guidance references for speed and course at maneuvering times $t_M, M = 1, 2, ..., n_M$, evenly spaced througout the horizon with a time spacing parameter t_{ts} for simplicity.

B.4.2 Setup

The GPU-based PSB-MPC is tested in two situations to illustrate that the COLAV planning algorithm can tackle dynamic obstacles with uncertainties in addition to grounding hazards. The first river scenario is chosen to test how the COLAV planning method handles avoidance in confined spaces, whereas the second scenario aims to test the algorithm performance in a longer time horizon with multiple dynamic obstacles in a mix of an open sea area and a narrow channel. The setup with tracking system and parameters are similar to that in [195], where the obstacle tracker is deliberately tuned conservatively to test the MPC robustness against kinematic uncertainty. The situations are described below, with a number of $N_{MC} = 50$ Monte Carlo simulations used for each situation. A runtime analysis considering the first situation is performed, comparing the CPU and GPU implementations of the PSB-MPC, Algorithm 10 and 11, respectively. The CPU version evaluates the PSB-MPC cost for all own-ship control behaviours sequentially on CPU cores. The simulations are performed on a work station with an Intel(R) Core(TM) i9-10900K 3.70 GHz processor, with 32 GB RAM and an NVIDIA GeForce RTX 3090 GPU. C++ is used to implement the CPU version of the PSB-MPC, whereas C++ and CUDA is used for the GPU version.

1. Head-on scenario in Nidelva in Trondheim, Norway. The own-ship travels upstream with constant speed 2 m/s, whereas two dynamic obstacles travels downstream with constant speed 2 m/s. Vessels of lengths 5 m are here considered, and an own-ship safety zone of $d_{safe} = 5\,\text{m}$ is used.

2. Multi-ship situation with grounding hazards near Sakshaug, Trøndelag in Norway. Dynamic obstacle $i = 1$ is traveling from the south through Straumen with constant speed 5 m/s and ends up in an overtaking situation with respect to the own-ship, whereas dynamic obstacle $i = 2$ travels east-west through Straumen with constant speed 6 m/s and ends up in head-on situations with respect to the other vessels. Obstacle $i = 3$ travels with speed 7 m/s east-west from Straumen towards the own-ship in a head-on situation, and obstacle $i = 4$ just north-east of the own-ship travels south with speed 8 m/s. The own-ship travels with constant speed 7 m/s. Vessels of lengths 10 m are considered, and an own-ship safety zone of $d_{safe} = 10\,\text{m}$ is used. In addition to COLREGS adherence with respect to multiple ships, the challenge here is voyage through the narrow passage in Straumen, beneath the bridge which has two pylons that the vessels have to avoid.

For simplicity, a uniform set of scenario probabilities $\mathbb{P}_s^i$ are defined for the dynamic obstacles, which resembles a conservative case when no prior information from intent inference is available. For the grounding hazards, only polygons within a range d_{so} are considered, to reduce computation time. Waypoints for the own-ship are set in a way that a top level planner could generate, but with small margins to static obstacles, such that the anti-grounding part of the PSB-MPC becomes important. Furthermore, the waypoints are set such that a nominal collision-free trajectory does not exist for all vessels involved.

The MPC is tuned such that anti-grounding and collision avoidance is prioritized over adhering to COLREGS and following the nominal trajectory. Naturally, because river voyage is different from sea voyage, the PSB-MPC has a different tuning for the two situations. Important parameters for the first situation tuning are given in Table B.1.

B.4.3 The Nidelva situation

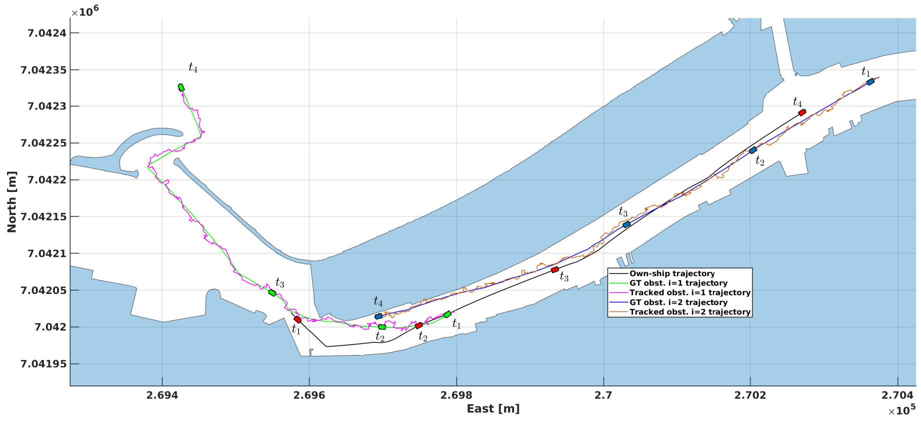
Results for the first situation are given in Figure B.4. The dynamic obstacles are here assumed to be self-governing, running their own PSB-MPC algorithm to simulate human behaviour. The conservative tracking system tuning will create an extra challenge for the COLAV planning algorithm, with higher kinematic obstacle uncertainty. Despite this and nearby grounding hazards, all vessels involved are able to avoid collision and grounding in addition to adhering to the COLREGS rules 8, 13 and 16 related to clear actions, head-on situation and actions for give-way vessels, respectively. The near constant minimum distance to the closest static obstacle in the statistics is because the own-ship is closest to a grounding hazard initially. Note that the map data for the river area do not include the piers at which

Table B.1: Important PSB-MPC parameters for the Nidelva situation.

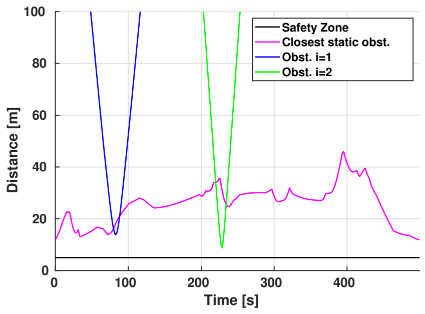
Parameter	**Value**	Comment
ϵ	$2\,\mathrm{m}$	RDP distance threshold
T_{MPC}	$150\,\mathrm{s}$	Prediction horizon
T_s	$0.5\,\mathrm{s}$	Prediction time step
T_d	$100\,\mathrm{s}$	Collision cost time discounting parameter
n_{ps}^{LOS}	5	Number of LOS prediction scenarios
r_{ct}	$10.0\,\mathrm{m}$	Prediction scenario spacing
d_{so}	$200.0\,\mathrm{m}$	Static obstacle consideration range
n_M	2	Number of sequential avoidance maneuvers
$u_{offsets,1}$	$\{1.0, 0.5, 0.0\}$	Surge offsets first maneuver
$u_{offsets,2}$	$\{1.0, 0.5\}$	Surge offsets second maneuver
$\chi_{offsets,1}$	$\{-60, -45, -30, -15, -10, -5, 0, 5, 10, 15, 30, 45, 60\}$	Course offsets first maneuver
$\chi_{offsets,2}$	$\{-60, -45, -30, -15, -10, -5, 0, 5, 10, 15, 30, 45, 60\}$	Course offsets second maneuver

boats are docked, which would be taken into account through e.g. LIDAR data in a real-time application.

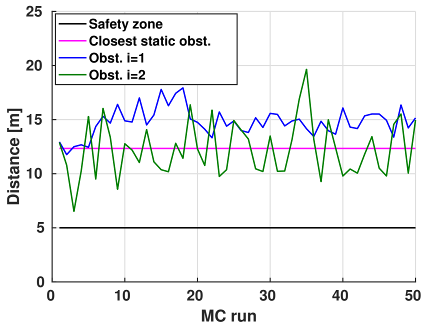
For the situation in Nidelva, a run-time analysis was performed with respect to the number of control behaviours n_{cbs} for the MPC, and the number of dynamic obstacle prediction scenarios n_{ps}^i considered. The number of control behaviours is increased by increasing the number of sequential maneuvers n_M in the horizon, and by expanding the finite set of course and surge modifications. Both the CPU and GPU implementations were run for N_{MC} simulations for each parameter setting. Figures B.5, B.6 and B.7 show a box-plot representation of the results. The GPU-implementation of the PSB-MPC performs better than the CPU-version when the number of control behaviours increase beyond a thousand. With $n_{cbs} < 1000$ and a scheduled number of threads $n_{ct} < 5000$, the overhead of launching the GPU kernels becomes too large compared to the gain of parallelized cost evaluation. This makes the CPU-implementation feasible for cases where typically $n_M = 1$ and a small number of possible course and speed changes is enough, and only a



(a) North east plot at multiple time instants for a sample run. Dynamic obstacles are shown in green ($i = 1$) and blue ($i = 2$). The own-ship is shown in red.



(b) Distance to static and dynamic obstacles for the sample run.



(c) Statistics on the minimum distance to the obstacles over all N_{MC} simulation runs.

Figure B.4: Results for the situation in Nidelva with multiple obstacles.

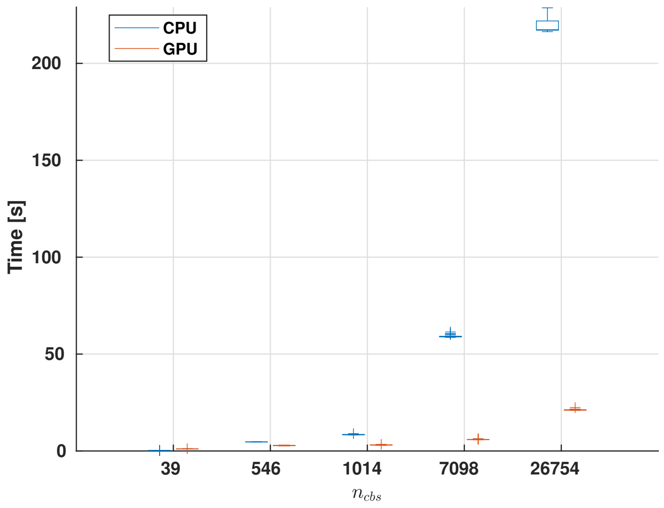


Figure B.5: Box-plot representation of the runtime results with respect to increasing numbers of control behaviours n_{cbs}, when keeping the number of dynamic obstacle prediction scenarios constant at $n_{ps}^i = 1$.

small number of static and dynamic obstacles are considered.

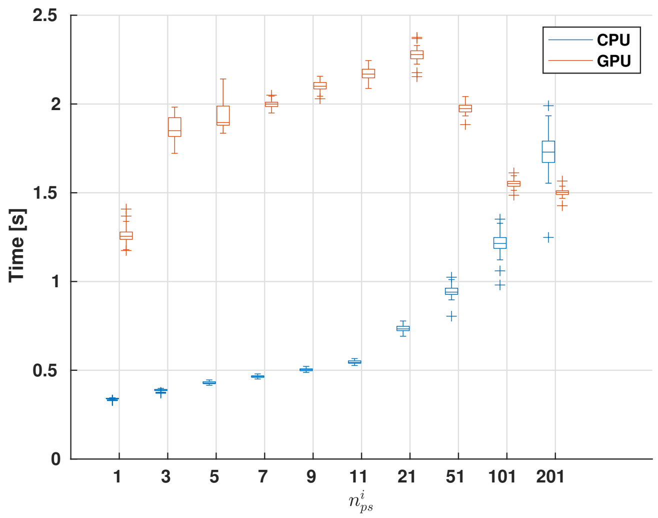


Figure B.6: Box-plot representation of the runtime results with respect to increasing dynamic obstacle prediction scenarios n_{ps}^i, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 1$ and a total number of control behaviours $n_{cbs} = 39$.

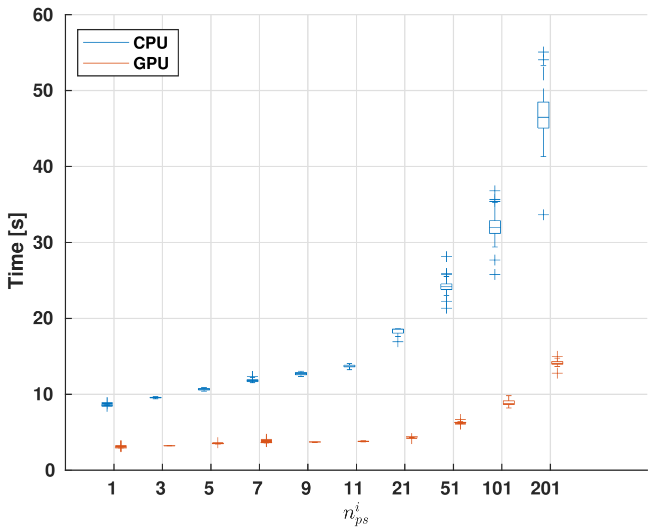


Figure B.7: Box-plot representation of the runtime results with respect to increasing dynamic obstacle prediction scenarios n^i_{ps}, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 2$ and a total number of control behaviours $n_{cbs} = 1014$.

Furthermore, one can see that the CPU implementation performs better than the GPU implementation when considering increasing numbers of prediction scenarios up until $n^i_{ps} = 101$ for dynamic obstacles, when using a low number of control behaviours $n_{cbs} = 39$. In this case, the GPU run-time is mainly caused by the overhead of porting data back and forth between the host and device side. The contrary result is the case when considering $n_{cbs} > 1000$. This is again because a CPU is optimized for fast sequential execution on fewer but more complex tasks, whereas a GPU is optimized for execution of many simple tasks in parallel. A similar result is obtained by increasing n_{obst} while keeping n^i_{ps} constant, but will not be reported here.

From Figures B.5 - B.7, an approximate linear scaling of the MPC run-time complexity with increasing own-ship control behaviours, dynamic obstacle scenarios and static obstacles can be found. For static obstacles represented as polygons, one also have to take into account the added run-time complexity due to the number of vertices in the polygons.

Also, tests to compare the run-time related to calculating predominantly the grounding cost in the MPC on a CPU and GPU platform was performed, when the own-ship is located in Nidelva standing still. No dynamic obstacles are considered, and thus the calculation of the distance to static obstacles will be the bottleneck. The largest static obstacle in the region is a polygon with 21962 vertices originally, and has 1734 vertices after application of the RDP algorithm. The map environment around Nidelva in Trondheim is illustrated in Figure B.8, where the static obstacle

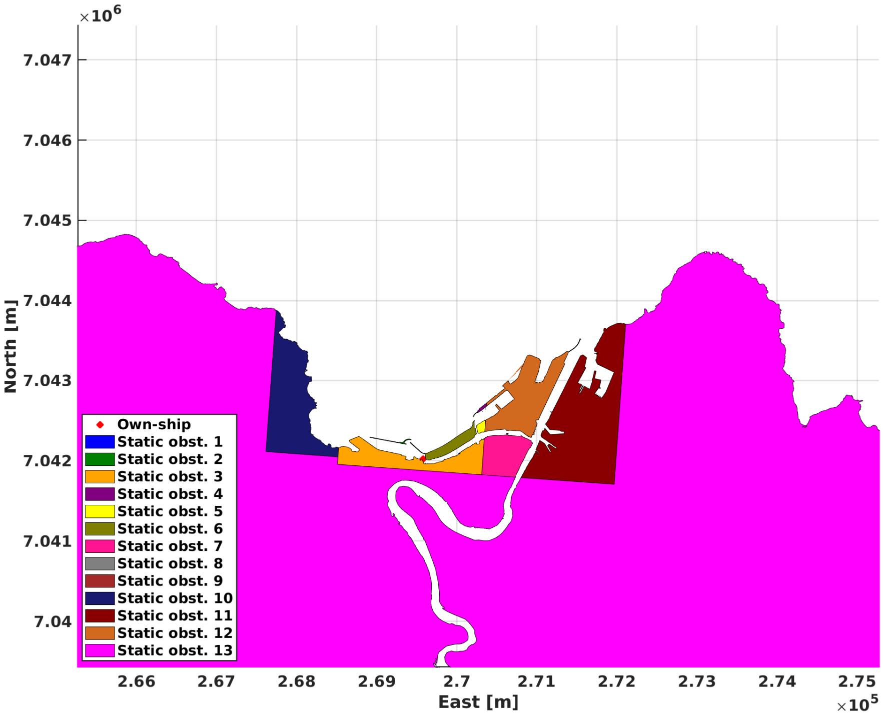


Figure B.8: Map of the Trondheim region with Nidelva in the middle, with all relevant polygons labelled with different colors. The own-ship position is the small red dot in Nidelva in the middle.

$j = 13$ is the largest one with 21962 vertices. Information about the number of vertices for each polygon is given in Table B.2.

The first test compares the run-time when only considering the largest polygon, with and without usage of the RDP algorithm. This is a worst case scenario, as a real-time anti-grounding system should preprocess large polygons such that only the relevant local part is considered. We however include this test for completeness, as it shows the importance of polygon preprocessing. Results are here given in Figure B.9.

The results in Figure B.10 show a run-time analysis for increasing numbers of static obstacles, after using the RDP for polygon simplification. Note that the results considering an increasing number of static obstacles are strongly dependent on the number of vertices for each obstacle, which varies from 3 to 1734 vertices as seen from Table B.2 after using RDP on this environment. This is why there is a sharp increase in average run-time when $n_{so} = 13$, because the largest polygon is then included in the consideration. An approximate linear run-time increase can

Table B.2: Polygon vertices before and after applying RDP on the Nidelva environment.

Polygon	Vertices before	Vertices after
1	8	3
2	207	27
3	649	95
4	322	33
5	140	18
6	890	53
7	207	48
8	8	3
9	8	5
10	1633	187
11	2110	162
12	2483	143
13	21961	1734

however be found when considering polygons of fairly the same complexity. The trend from these results is that the GPU implementation becomes more feasible than the CPU one when the number of scheduled parallel threads n_{ct} surpasses around 5000.

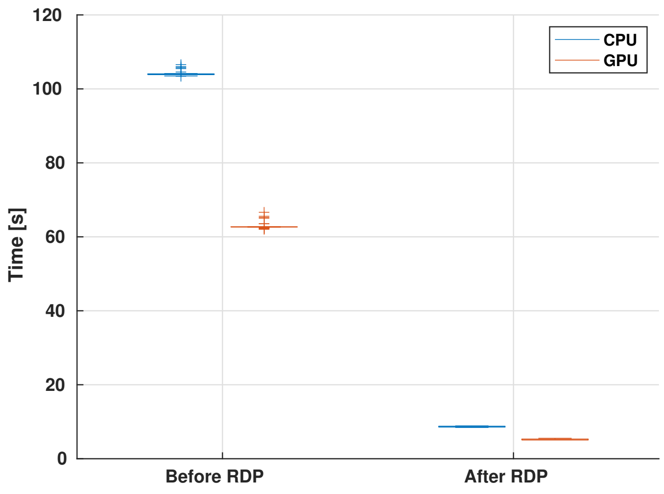


Figure B.9: Box-plot representation of the runtime results with respect to the worst case polygon scenario before and after applying RDP, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 2$ and a total number of control behaviours $n_{cbs} = 1014$.

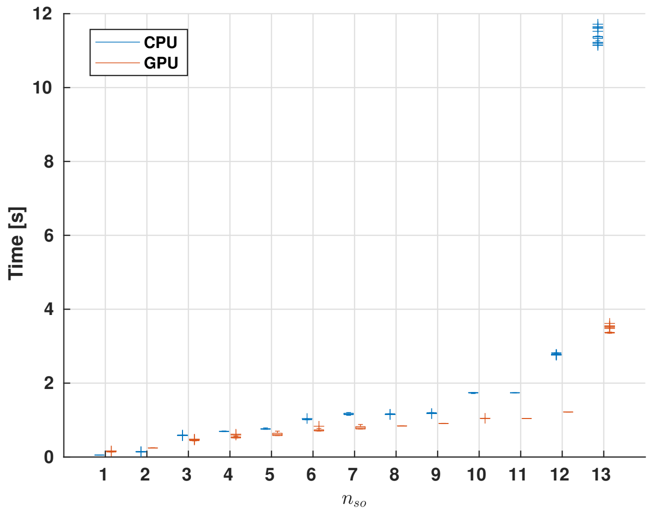


Figure B.10: Box-plot representation of the runtime results with respect to increasing numbers of static obstacles n_{so}, when keeping the number of own-ship avoidance maneuvers constant at $n_M = 2$ and a total number of control behaviours $n_{cbs} = 1014$.

B.4.4 The Sakshaug situation

Important parameters for the tuning are given in Table B.3, with results shown in Figure B.11 and B.12. The first case show results when only the own-ship has a COLAV planning algorithm, whereas the second case show results when all vessels involved use the PSB-MPC. For the first case, waypoints for the obstacles are set such that they will not collide with each other, but would collide with the own-ship if no COLAV planning algorithm was used.

For both the first and second case, the own-ship has difficulties with overtaking purple obstacle $i = 1$ while simultaneously avoid grounding and avoiding blue obstacle $i = 2$ head-on, that adheres to both COLREGS rules 13 and 14 regarding overtaking and head-on. Especially in the time period between t_2 and t_3, the own-ship struggles with figuring out the side to overtake obstacle $i = 1$ on when entering Straumen, hence the oscillations in the trajectory in this period. The black obstacle $i = 3$ and green obstacle $i = 4$ are easier to avoid as the vessels are here less constrained by land.

Thus, the own-ship is in general able to avoid collision with all obstacles in both cases, but COLREGS adherence in the narrow passage is difficult to accomplish with respect to all ships. This is mainly due to constant conservative intent information being used, with uniform prediction scenario probabilities for dynamic obstacle trajectories, essentially assuming that no dynamic obstacle will have specific inclinations towards adhering to the COLREGS. Also needing to avoid grounding in the narrow passage further restricts the PSB-MPC‘s ability to adhere to COL-

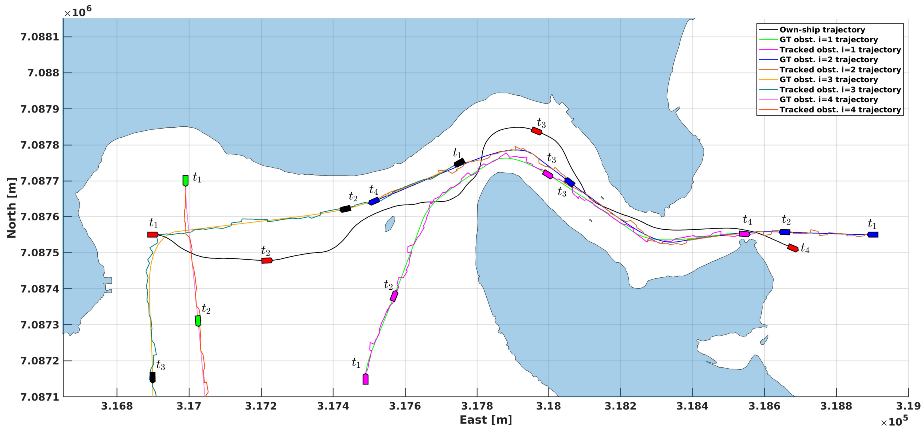
Table B.3: Important PSB-MPC parameters for the Sakshaug situation.

Parameter	**Value**	Comment
ϵ	$2\,\mathrm{m}$	RDP distance threshold
T_{MPC}	$150\,\mathrm{s}$	Prediction horizon
T_s	$1.0\,\mathrm{s}$	Prediction time step
T_d	$100\,\mathrm{s}$	Collision cost time discounting parameter
n_{ps}^{LOS}	5	Number of LOS prediction scenarios
r_{ct}	$20.0\,\mathrm{m}$	Prediction scenario spacing
d_{so}	$800.0\,\mathrm{m}$	Static obstacle consideration range
n_M	2	Number of sequential avoidance maneuvers
$u_{offsets,1}$	$\{1.0, 0.5, 0.0\}$	Surge offsets first maneuver
$u_{offsets,2}$	$\{1.0, 0.5\}$	Surge offsets second maneuver
$\chi_{offsets,1}$	$\{-90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, 90\}$	Course offsets first maneuver
$\chi_{offsets,2}$	$\{-90, -75, -60, -45, -30, -15, 0, 15, 30, 45, 60, 75, 90\}$	Course offsets second maneuver

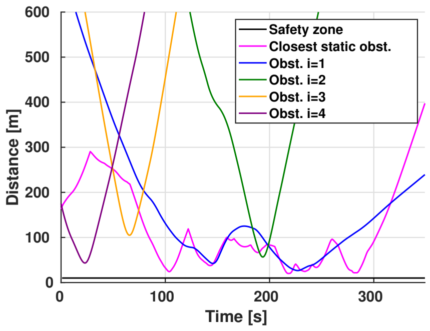
REGS in a safe manner. The algorithm is however able to keep safe distance to all obstacles in all Monte Carlo simulation runs. The diversity of the environment makes algorithm tuning challenging, as one can argue that the COLAV planning algorithm parameters should be adaptive based on changes in the situation.

When the dynamic obstacles do not explicitly follow COLREGS in the first case, the own-ship can be more excused for not doing the same with respect to all vessels. For the second case, one see the potential for vessel-vessel communication to explicitly reduce trajectory uncertainties and adhere to COLREGS, during the passage through Straumen. Addressing these issues is the topic of future research more focused on multi-ship COLREGS compliance in confined waters.

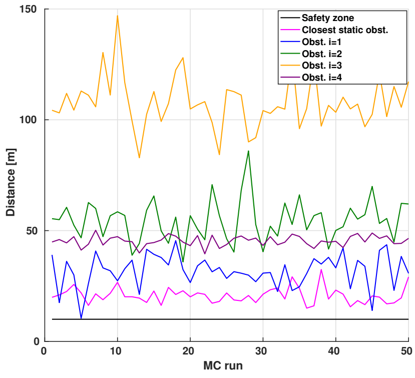
Regarding run-time complexity for this example, it will be similar as for the first situation when considering increasing dynamic obstacles and their prediction scenarios. There will be a small increase in the run-time due to the Sakshaug situation has larger and more complex static obstacles, although a smaller set than for the



(a) North east plot at multiple time instants for a sample run. Dynamic obstacles in purple ($i = 1$), blue ($i = 2$), black ($i = 3$) and green ($i = 4$). The own-ship is shown in red.

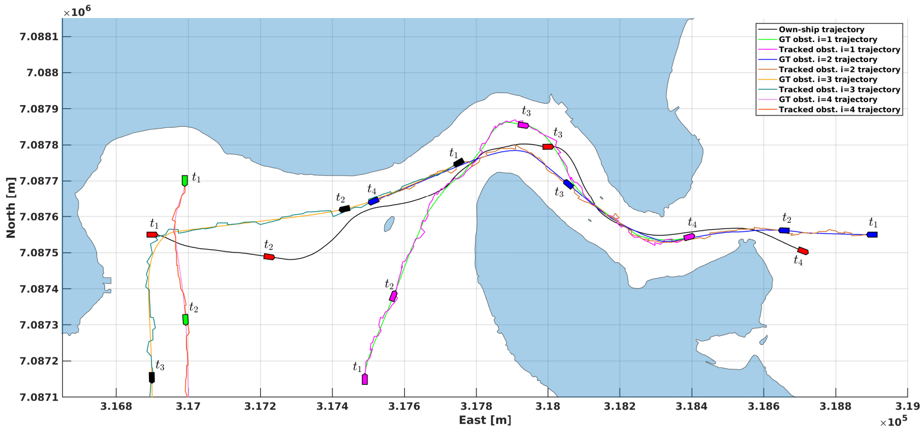


(b) Distance to static and dynamic obstacles for the sample run.

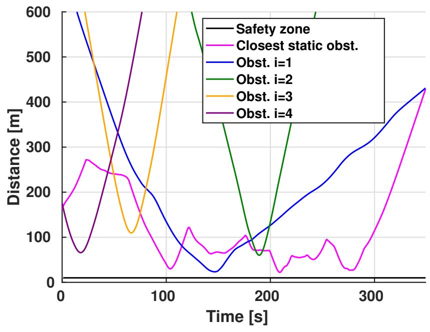


(c) Statistics on the minimum distance to the obstacles over all N_{MC} simulation runs.

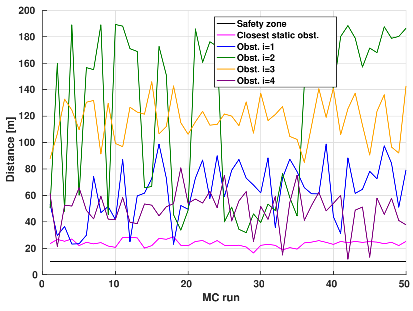
Figure B.11: Results for the situation in Sakshaug with multiple obstacles in the first case.



(a) North east plot at multiple time instants for the sample run. Dynamic obstacles in purple ($i = 1$), blue ($i = 2$), black ($i = 3$) and green ($i = 4$). The own-ship is shown in red.



(b) Distance to static and dynamic obstacles for the sample run.



(c) Statistics on the minimum distance to the obstacles over all N_{MC} simulation runs.

Figure B.12: Results for the situation in Sakshaug with multiple obstacles in the second case.

Nidelva situation is considered in the proximity of the own-ship. In total, run-time results generated for this example would be fairly similar to the first simulation, albeit with a bias on the static obstacle run-time complexity due to larger obstacles considered.

B.5 Conclusion

The PSB-MPC COLAV planning algorithm presented in this article facilitates both dynamic and static obstacle avoidance, with the most performance-critical part of its algorithm implemented on the GPU. What separates it from current state-of-the-art is the computational speed of the algorithm, where the cost evaluation is parallelized such that the MPC problem scales approximately linearly with increasing control behaviours, static and dynamic obstacles and prediction scenarios, as shown in the run-time results presented. This makes the COLAV planner able to consider more control behaviours and dynamic obstacle prediction scenarios efficiently, which results in real-time capabilities and performance gains in cases where large amounts of situational information and possible own-ship decisions have to be considered.

In simulation, the COLAV planning algorithm is shown to handle both grounding hazards and multiple dynamic obstacles in a safe manner, both in a narrow river environment, and also in a mix of more open sea and narrow waters. However, there is an inherent challenge in finding parameters that will make the algorithm work robustly and adhere to COLREGS for multiple types of situations, especially when the environmental constraints vary a lot.

Future work will involve making the PSB-MPC adaptive to the environment faced, and utilize historical data for tuning the algorithm. Also, the dynamic obstacle prediction and COLREGS penalization cost evaluation should be extended to consider static obstacles, for better applicability in confined spaces.

References

[1] K. Wróbel, J. Montewka, and P. Kujala, "Towards the assessment of potential impact of unmanned vessels on maritime transportation safety," *Reliability Engineering & System Safety*, vol. 165, pp. 155–169, 2017.

[2] I. B. Utne, A. J. Sørensen, and I. Schjølberg, "Risk management of autonomous marine systems and operations," in *International Conference on Offshore Mechanics and Arctic Engineering*, American Society of Mechanical Engineers, vol. 57663, 2017, V03BT02A020.

[3] F. Goerlandt, "Maritime autonomous surface ships from a risk governance perspective: Interpretation and implications," *Safety science*, vol. 128, p. 104 758, 2020.

[4] IMO, *Outcome of the Regulatory Scoping Exercise for the Use of Maritime Autonomous Surface Ships (MASS)*, 2021.

[5] G. DNV, "Remote-controlled and autonomous ships," *Position Paper DNV GL Høvik, Norway*, 2018.

[6] K. Wróbel, P. Krata, J. Montewka, and T. Hinz, "Towards the Development of a Risk Model for Unmanned Vessels Design and Operations," *The International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 10, no. 2, pp. 267–274, 2016.

[7] I. B. Utne, B. Rokseth, A. J. Sørensen, and J. E. Vinnem, "Towards supervisory risk control of autonomous ships," *Reliability Engineering & System Safety*, vol. 196, p. 106 757, 2020.

[8] T. A. Pedersen, J. A. Glomsrud, E.-L. Ruud, A. Simonsen, J. Sandrib, and B. H. Eriksen, "Towards simulation-based verification of autonomous navigation systems," *Safety Science*, vol. 129, p. 104 799, 2020.

[9] T. A. Johansen, "Toward dependable embedded model predictive control," *IEEE Systems Journal*, vol. 11, no. 2, pp. 1208–1219, 2017.

[10] T. A. Johansen, A. Cristofaro, and T. Perez, "Ship collision avoidance using scenario-based model predictive control," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 14–21, 2016.

[11] N. Leveson, *Engineering a safer world: Systems thinking applied to safety.* MIT press, 2011.

[12] B. Rokseth, I. B. Utne, and J. E. Vinnem, "A systems approach to risk analysis of maritime operations," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 231, no. 1, pp. 53–68, 2017.

[13] B. Rokseth, I. B. Utne, and J. E. Vinnem, "Deriving verification objectives and scenarios for maritime systems using the systems-theoretic process analysis," *Reliability Engineering & System Safety*, vol. 169, pp. 18–31, 2018.

[14] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[15] H. S. Dewang, P. K. Mohanty, and S. Kundu, "A robust path planning for mobile robot using smart particle swarm optimization," *Procedia computer science*, vol. 133, pp. 290–297, 2018.

[16] M. S. Alam, M. U. Rafique, and M. U. Khan, "Mobile robot path planning in static environments using particle swarm optimization," *arXiv preprint arXiv:2008.10000*, 2020.

[17] F. Ding, Z. Zhang, M. Fu, Y. Wang, and C. Wang, "Energy-efficient path planning and control approach of usv based on particle swarm optimization," in *OCEANS 2018 MTS/IEEE Charleston*, IEEE, 2018, pp. 1–6.

[18] H. Xue, "A quasi-reflection based sc-pso for ship path planning with grounding avoidance," *Ocean Engineering*, vol. 247, p. 110 772, 2022.

[19] X. Guo, M. Ji, Z. Zhao, D. Wen, and W. Zhang, "Global path planning and multi-objective path control for unmanned surface vehicle based on modified particle swarm optimization (pso) algorithm," *Ocean Engineering*, vol. 216, p. 107 693, 2020.

[20] J. Mkaka and J. Magaj, "Data extraction from an electronic S-57 standard chart for navigational decision systems," *Zeszyty Naukowe/Akademia Morska w Szczecinie*, pp. 83–87, 2012.

[21] Y. Yu, H. Zhu, L. Yang, and C. Wang, "Spatial indexing for effective visualization of vector-based electronic nautical chart," in *2016 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, IEEE, 2016, pp. 323–326.

[22] M. Wlodarczyk-Sielicka and N. Wawrzyniak, "Interpolating Bathymetric Big Data for an Inland Mobile Navigation System," *Information Technology and Control*, vol. 47, no. 2, pp. 338–348, 2018.

[23] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Engineering Practice*, vol. 61, pp. 41–54, 2017.

[24] T. Wilson and S. B. Williams, "Adaptive path planning for depth-constrained bathymetric mapping with an autonomous surface vessel," *Journal of Field Robotics*, vol. 35, no. 3, pp. 345–358, 2018.

[25] 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, 2019.

[26] J. Larson, M. Bruch, and J. Ebken, “Autonomous navigation and obstacle avoidance for unmanned surface vehicles,” in *Unmanned systems technology VIII*, International Society for Optics and Photonics, vol. 6230, 2006, p. 623 007.

[27] R. Szlapczynski and J. Szlapczynska, “A method of determining and visualizing safe motion parameters of a ship navigating in restricted waters,” *Ocean Engineering*, vol. 129, pp. 363–373, 2017.

[28] A. Bakdi, I. K. Glad, E. Vanem, and Ø. Engelhardtsen, “AIS-based multiple vessel collision and grounding risk identification based on adaptive safety domain,” *Journal of Marine Science and Engineering*, vol. 8, no. 1, 2020.

[29] T. Tengesdal, T. A. Johansen, T. D. Grande, and S. Blindheim, “Ship collision avoidance and anti grounding using parallelized cost evaluation in probabilistic scenario-based model predictive control,” *IEEE Access*, vol. 10, pp. 111 650–111 664, 2022.

[30] Z. Li and J. Sun, “Disturbance Compensating Model Predictive Control With Application to Ship Heading Control,” *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 257–265, Jan. 2012.

[31] H. Zhou, L. Guvenc, and Z. Liu, “Design and evaluation of path following controller based on MPC for autonomous vehicle,” in *Chinese Control Conference, CCC*, 2017, pp. 9934–9939.

[32] B. O. H. Eriksen and M. Breivik, “MPC-based mid-level collision avoidance for ASVs using nonlinear programming,” in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2017, pp. 766–772.

[33] D. K. M. Kufoalor, T. A. Johansen, E. F. Brekke, A. Hepsø, and K. Trnka, “Autonomous maritime collision avoidance: Field verification of autonomous surface vehicle behavior in challenging scenarios,” *Journal of Field Robotics*, vol. 37, no. 3, pp. 387–403, 2020.

[34] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, “Two-stage optimized trajectory planning for asvs under polygonal obstacle constraints: Theory and experiments,” *IEEE Access*, vol. 8, pp. 199 953–199 969, 2020.

[35] T. A. Pedersen, Å. Neverlien, J. A. Glomsrud, I. Ibrahim, S. M. Mo, M. Rindarøy, T. Torben, and B. Rokseth, “Evolution of safety in marine systems: From system-theoretic process analysis to automated test scenario generation,” *International Conference on Maritime Autonomous Surface Ships*, 2022.

[36] B. Rokseth and I. B. Utne, “A risk-based autonomous mode control system for the hybrid-electric machinery system of an autonomous ship,” *Submitted for publication to Safety Science*, 2023.

[37] S. Blindheim and T. A. Johansen, "Electronic Navigational Charts for Visualization, Simulation, and Autonomous Ship Control," *IEEE Access*, vol. 10, pp. 3716–3737, 2022. DOI: https://doi.org/10.1109/access.2021.3139767.

[38] S. Blindheim, S. Gros, and T. A. Johansen, "Risk-Based Model Predictive Control for Autonomous Ship Emergency Management," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 524–14 531, 2020, 21st IFAC World Congress. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1456.

[39] J. Andersson, J. Åkesson, and M. Diehl, "Casadi: A symbolic package for automatic differentiation and optimal control," in *Recent advances in algorithmic differentiation*, Springer, 2012, pp. 297–307.

[40] S. Blindheim, I. B. Utne, and T. A. Johansen, "Risk-Based Supervisory Control for Autonomous Ship Navigation," *Journal of Marine Science and Technology*, pp. 1–25, 2023. DOI: 10.1007/s00773-023-00945-6.

[41] S. Blindheim and T. A. Johansen, "Particle Swarm Optimization for Dynamic Risk-Aware Path Following for Autonomous Ships," *IFAC-PapersOnLine*, 2022, IFAC CAMS. DOI: https://doi.org/10.1016/j.ifacol.2022.10.411.

[42] S. Blindheim, B. Rokseth, and T. A. Johansen, "Autonomous Machinery Management for Supervisory Risk Control Using Particle Swarm Optimization," *Journal of Marine Science and Engineering*, vol. 11, no. 2, p. 327, 2023. DOI: https://doi.org/10.3390/jmse11020327.

[43] T. Johansen, S. Blindheim, T. R. Torben, I. B. Utne, T. A. Johansen, and A. J. Sørensen, "Development and testing of a risk-based control system for autonomous ships," *Reliability Engineering & System Safety*, vol. 234, p. 109 195, 2023.

[44] B. D. MacRae, R. Stephenson, T. Leadholm, and I. Gonin, "Digital chart database conversion into a system electronic navigational chart," Environmental Research Institute of Michigan Ann Arbor, Tech. Rep., 1992.

[45] A. Weintrit, "The electronic chart systems and their classification," *Annual of Navigation*, pp. 127–140, 2001.

[46] A. Weintrit, "Clarification, systematization and general classification of electronic chart systems and electronic navigational charts used in marine navigation. Part 1-electronic chart systems," *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 12, 2018.

[47] A. Weintrit, "Clarification, systematization and general classification of electronic chart systems and electronic navigational charts used in marine navigation. Part 2-electronic navigational charts," *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 12, 2018.

[48] A. Palikaris and A. K. Mavraeidopoulos, "Electronic Navigational Charts: International Standards and Map Projections," *Journal of Marine Science and Engineering*, vol. 8, no. 4, p. 248, 2020.

[49] M. R. Mahmud, N. Ibrahim, A. A. RAHMAN, R. Othman, U. Din, and A. H. Omar, "The development of a low-cost integrated marine navigation system for leisure crafts and small boats," *Universiti Teknologi Malaysia*, 2006.

[50] F. Zhu, Y. Zhang, and W. Sang, "Web Marine Spatial Information Service Based on Electronic Nautical Charts," in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, IEEE, vol. 3, 2007, pp. 131–136.

[51] G. Park, D. Park, and S. Park, "Design and implementation of display module for electronic navigational chart data," in *2014 International Conference on IT Convergence and Security (ICITCS)*, IEEE, 2014.

[52] A. Weintrit, "Radar Image Overlay in ECDIS Display Versus Electronic Navigational Chart Overlay on Radar Screen," *Prace Wydziału Nawigacyjnego*, no. 22, 2008.

[53] M. Waz and K. Naus, "Electronic Navigational Chart in aid of generation of multi-dimensional radar display," *International Journal of Circuits and Electronics*, vol. 2, 2017.

[54] K. Naus and A. Makar, "Conception of spatial presentation of ENC," in *XIV International Scientific and Technical Conference "The Part of Navigation in Support of Human Activity on the Sea". Naval University of Gdynia, Gdynia*, 2004.

[55] L. Hui, X. Shengwei, and Z. Yingjun, "Inland waterway three-dimensional visualization based on 3D-GIS technology," in *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, IEEE, vol. 1, 2008, pp. 564–568.

[56] T. Liu, D. Zhao, and M. Pan, "Generating 3D depiction for a future ECDIS based on digital earth," *The Journal of Navigation*, vol. 67, no. 6, p. 1049, 2014.

[57] J.-C. Morgère, J.-P. Diguet, and J. Laurent, "Electronic navigational chart generator for a marine mobile augmented reality system," in *2014 Oceans-St. John's*, IEEE, 2014.

[58] M. Lager, E. A. Topp, and J. Malec, "Remote Operation of Unmanned Surface Vessel through Virtual Reality-a low cognitive load approach," in *Proceedings of the 1st International Workshop on Virtual, Augmented, and Mixed Reality for HRI (VAM-HRI)*, 2018.

[59] S. M. Smith, L. Alexander, and A. A. Armstrong, "The navigation surface: A new database approach to creating multiple products from high-density surveys," *The International Hydrographic Review*, 2002.

[60] C. Shi, M. Zhang, and J. Peng, "Harmonic potential field method for autonomous ship navigation," in *2007 7th International Conference on ITS Telecommunications*, IEEE, 2007.

[61] Y. Liu and R. Bucknall, “Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment,” *Ocean engineering*, vol. 97, pp. 126–144, 2015.

[62] R. Song, Y. Liu, and R. Bucknall, “A multi-layered fast marching method for unmanned surface vehicle path planning in a time-variant maritime environment,” *Ocean Engineering*, vol. 129, pp. 301–317, 2017.

[63] Y. Liu and R. Bucknall, “Efficient multi-task allocation and path planning for unmanned surface vehicle in support of ocean operations,” *Neurocomputing*, vol. 275, pp. 1550–1566, 2018.

[64] Y. Ma, M. Hu, and X. Yan, “Multi-objective path planning for unmanned surface vehicle with currents effects,” *ISA transactions*, vol. 75, pp. 137–156, 2018.

[65] B. Shah and S. Gupta, “Speeding up A* search on visibility graphs defined over quadtrees to enable long distance path planning for unmanned surface vehicles,” in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 26, 2016.

[66] Y. Singh, S. Sharma, R. Sutton, D. Hatton, and A. Khan, “A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents,” *Ocean Engineering*, vol. 169, pp. 187–201, 2018.

[67] R. Song, Y. Liu, and R. Bucknall, “Smoothed A* algorithm for practical unmanned surface vehicle path planning,” *Applied Ocean Research*, vol. 83, pp. 9–20, 2019.

[68] R. Goralski, C. Ray, and C. Gold, “Applications and benefits for the development of cartographic 3D visualization systems in support of maritime safety,” *International Recent Issues about ECDIS, e-Navigation and Safety at Sea: Marine Navigation and Safety of Sea Transportation*, vol. 77, 2011.

[69] X. Gao, S. Shiotani, and H. Makino, “The Study of Effective Communication of Water Depth Information for Prevention of Accidents in Marine Traffic,” in *2012 Fifth International Conference on Emerging Trends in Engineering and Technology*, IEEE, 2012, pp. 265–269.

[70] O. A. V. Banda, S. Kannos, F. Goerlandt, P. H. van Gelder, M. Bergström, and P. Kujala, “A systemic hazard analysis and management process for the concept design phase of an autonomous vessel,” *Reliability Engineering & System Safety*, vol. 191, 2019.

[71] 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,” *Journal of Marine Science and Technology*, 2021.

[72] A. Vagale, R. T. Bye, R. Oucheikh, O. L. Osen, and T. I. Fossen, “Path planning and collision avoidance for autonomous surface vehicles II: a comparative study of algorithms,” *Journal of Marine Science and Technology*, 2021.

[73] M. P. Vitus, S. L. Waslander, and C. J. Tomlin, "Locally optimal decomposition for autonomous obstacle avoidance with the tunnel-MILP algorithm," in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 540–545.

[74] R. Zhen, M. Riveiro, and Y. Jin, "A novel analytic framework of real-time multi-vessel collision risk assessment for maritime traffic surveillance," *Ocean Engineering*, vol. 145, pp. 492–501, 2017.

[75] E. F. Brekke, E. F. Wilthil, B. H. Eriksen, D. Kufoalor, Ø. K. Helgesen, I. B. Hagen, M. Breivik, and T. A. Johansen, "The Autosea project: Developing closed-loop target tracking and collision avoidance systems," in *Journal of Physics: Conference series*, IOP publishing, vol. 1357, 2019.

[76] J. Zhou, C. Wang, and A. Zhang, "A COLREGs-Based Dynamic Navigation Safety Domain for Unmanned Surface Vehicles: A Case Study of Dolphin-I," *Journal of Marine Science and Engineering*, vol. 8, no. 4, p. 264, 2020.

[77] J. M. Cordero and C. Kastrisios, "Characterizing free and open-source tools for ocean-mapping," *ResearchGate*, 2020.

[78] C. Barry, N. P. H. Branch, S. Legeer, G. Parker, N. A. H. Branch, and K. VanSant, "Us office of coast survey's re-engineered process for application of hydrographic survey data to noaa charts," in *The 10th International User Group Conference and Educational Sessions, Nova Scotia, Canada*, Citeseer, 2005.

[79] L. Alexander and M. Huet, "Relationship of marine information overlays (mios) to current/future iho standards," *International Hydrographic Organization*, 2007.

[80] G. Masetti, B. R. Calder, and M. J. Wilson, *Pydro white paper*, 2017.

[81] Python Software Foundation, *Python Package Index - PyPI.*

[82] R. Renger, A. Cimetta, S. Pettygrove, and S. Rogan, "Geographic information systems (GIS) as an evaluation tool," *American Journal of Evaluation*, vol. 23, no. 4, pp. 469–479, 2002.

[83] S. Gillies *et al.*, *Shapely: Manipulation and analysis of geometric objects*, toblerity.org, 2007.

[84] P. Wu, S. Xie, H. Liu, M. Li, H. Li, Y. Peng, X. Li, and J. Luo, "Autonomous obstacle avoidance of an unmanned surface vehicle based on cooperative manoeuvring," *Industrial Robot: An International Journal*, 2017.

[85] T. D. Grande, "PSB-MPC Collision Avoidance with Anti-Grounding," *NTNU ITK*, 2021.

[86] Intel Corporation, *Intel oneAPI Math Kernel Library*, 2021.

[87] T. Keviczky, P. Falcone, F. Borrelli, J. Asgari, and D. Hrovat, "Predictive control approach to autonomous vehicle steering," in *2006 American Control Conference*, Aug. 2006, 6 pp.

[88] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, "Design and Development of an Optimal-Control-Based Framework for Trajectory Planning, Threat Assessment, and Semi-autonomous Control of Passenger Vehicles in Hazard Avoidance Scenarios," in *Springer Tracts in Advanced Robotics (2011) 70(STAR)*, Springer, Berlin, Heidelberg, 2011.

[89] S. Samuelson and I. Yang, "Safety-aware optimal control of stochastic systems using conditional value-at-risk," in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 6285–6290.

[90] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 194–206, Jan. 2018.

[91] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011.

[92] D. D. Morrison, J. D. Riley, and J. F. Zancanaro, "Multiple shooting method for two-point boundary value problems," *Communications of the ACM*, vol. 5, no. 12, pp. 613–614, 1962.

[93] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," in *6th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2003), Girona, Spain*, 2003.

[94] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control*, Springer, 2017, pp. 45–69.

[95] I. B. Utne, I. Schjølberg, and E. Roe, "High reliability management and control operator risks in autonomous marine systems and operations," *Ocean Engineering*, vol. 171, pp. 399–416, 2019.

[96] S. A. Shappell and D. A. Wiegmann, *The human factors analysis and classification system–hfacs*, 2000.

[97] M. A. Ramos, I. B. Utne, and A. Mosleh, "Collision avoidance on maritime autonomous surface ships: Operators' tasks and human failure events," *Safety science*, vol. 116, pp. 33–44, 2019.

[98] L. Layman, V. R. Basili, and M. V. Zelkowitz, "A methodology for exposing risk in achieving emergent system properties," *ACM Trans. Softw. Eng. Methodol.*, vol. 23, no. 3, 22:1–22:28, Jun. 2014.

[99] K. Øien, I. Utne, and I. Herrera, "Building safety indicators: Part 1 – theoretical foundation," *Safety Science*, vol. 49, no. 2, pp. 148–161, 2011.

[100] K. Øien, I. Utne, R. Tinmannsvik, and S. Massaiu, "Building safety indicators: Part 2 – application, practices and results," *Safety Science*, vol. 49, no. 2, pp. 162–171, 2011.

[101] T. I. Bø and T. A. Johansen, "Dynamic safety constraints by scenario-based economic model predictive control of marine electric power plants," *IEEE Transactions on Transportation Electrification*, vol. 3, no. 1, pp. 13–21, 2016.

[102] A. Veksler, T. A. Johansen, F. Borrelli, and B. Realfsen, "Dynamic positioning with model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1340–1353, Jul. 2016.

[103] M. Rausand, *Risk assessment: theory, methods, and applications*. John Wiley & Sons, 2013, vol. 115.

[104] M. Ludvigsen and A. J. Sørensen, “Towards integrated autonomous underwater operations for ocean mapping and monitoring,” *Annual Reviews in Control*, vol. 42, pp. 145–157, 2016.

[105] B. Rokseth, O. I. Haugen, and I. B. Utne, “Safety verification for autonomous ships,” in *MATEC Web of Conferences*, EDP Sciences, vol. 273, 2019, p. 02 002.

[106] N. G. Leveson and J. P. Thomas, “Stpa handbook,” *Cambridge, MA, USA*, 2018.

[107] M. Stopford, *Maritime economics*, 3rd ed. Routledge, 2009.

[108] H. Kim, M. A. Lundteigen, A. Hafver, and F. B. Pedersen, “Utilization of risk priority number to systems-theoretic process analysis: A practical solution to manage a large number of unsafe control actions and loss scenarios,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 235, no. 1, pp. 92–107, 2021.

[109] D. Kiran, “Chapter 26 - failure modes and effects analysis,” in *Total Quality Management*, D. Kiran, Ed., Butterworth-Heinemann, 2017, pp. 373–389.

[110] S. Wright and J. Nocedal, “Numerical optimization,” *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[111] E. J. Mishan, “Evaluation of life and limb: A theoretical approach,” *Journal of Political Economy*, vol. 79, no. 4, pp. 687–705, 1971.

[112] F. Ackerman and L. Heinzerling, “Pricing the priceless: Cost-benefit analysis of environmental protection,” *University of Pennsylvania Law Review*, vol. 150, no. 5, pp. 1553–1584, 2002.

[113] A. Richards, “Fast model predictive control with soft constraints,” *European Journal of Control*, vol. 25, pp. 51–59, 2015.

[114] M. Alhajeri and M. Soroush, “Tuning guidelines for model-predictive control,” *Industrial & Engineering Chemistry Research*, vol. 59, no. 10, pp. 4177–4191, 2020.

[115] W. Edwards, G. Tang, G. Mamakoukas, T. Murphey, and K. Hauser, “Automatic tuning for data-driven model predictive control,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 7379–7385.

[116] S. V. Rothmund and T. A. Johansen, “Risk-based obstacle avoidance in unknown environments using scenario-based predictive control for an inspection drone equipped with range finding sensors,” in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2019, pp. 221–230.

[117] A. B. Martinsen, A. M. Lekkas, and S. Gros, “Optimal model-based trajectory planning with static polygonal constraints,” *IEEE Transactions on Control Systems Technology*, 2021.

[118] H. Niu, Y. Lu, A. Savvaris, and A. Tsourdos, "An energy-efficient path planning algorithm for unmanned surface vehicles," *Ocean Engineering*, vol. 161, pp. 308–321, 2018.

[119] R. Zaccone, "Colreg-compliant optimal path planning for real-time guidance and control of autonomous ships," *Journal of Marine Science and Engineering*, vol. 9, no. 4, p. 405, 2021.

[120] T. T. Enevoldsen and R. Galeazzi, "Grounding-aware RRT* for Path Planning and Safe Navigation of Marine Crafts in Confined Waters," *IFAC-PapersOnLine*, vol. 54, no. 16, pp. 195–201, 2021.

[121] A. Dallolio, T. K. Bergh, R. Pedro, H. Øveraas, and T. A. Johansen, "Enc-based anti-grounding and anti-collision system for a wave-propelled usv," *IEEE/MTS OCEANS Conference, India*, 2022.

[122] A. P. Engelbrecht, "Particle swarm optimization: Global best or local best?" In *2013 BRICS congress on computational intelligence and 11th Brazilian congress on computational intelligence*, IEEE, 2013, pp. 124–135.

[123] Y.-I. Lee and Y.-G. Kim, "A collision avoidance system for autonomous ship using fuzzy relational products and colregs," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3177, pp. 247–252, 2004.

[124] R. Zaccone and M. Martelli, "A collision avoidance algorithm for ship guidance applications," *Journal of Marine Engineering and Technology*, vol. 19, no. sup1, pp. 62–75, 2020.

[125] J. Koszelew, J. Karbowska-Chilinska, K. Ostrowski, P. Kuczyński, E. Kulbiej, and P. Wołejsza, "Beam search algorithm for anti-collision trajectory planning for many-to-many encounter situations with autonomous surface vehicles," *Sensors (Switzerland)*, vol. 20, no. 15, pp. 1–17, 2020.

[126] P. Chen, Y. Huang, E. Papadimitriou, J. Mou, and P. van Gelder, "Global path planning for autonomous ship: A hybrid approach of fast marching square and velocity obstacles methods," *Ocean Engineering*, vol. 214, 2020.

[127] T. Statheros, G. Howells, and K. McDonald-Maier, "Autonomous ship collision avoidance navigation concepts, technologies and techniques," *Journal of Navigation*, vol. 61, no. 1, pp. 129–142, 2008.

[128] Y.-I. Lee, S.-G. Kim, and Y.-G. Kim, "Fuzzy relational product for collision avoidance of autonomous ships," *Intelligent Automation and Soft Computing*, vol. 21, no. 1, pp. 21–38, 2015.

[129] T. 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.

[130] G. Li, H. Hildre, and H. Zhang, "Toward time-optimal trajectory planning for autonomous ship maneuvering in close-range encounters," *IEEE Journal of Oceanic Engineering*, 2019.

[131] H. Lyu and Y. Yin, "COLREGS-Constrained Real-Time Path Planning for Autonomous Ships Using Modified Artificial Potential Fields," *Journal of Navigation*, vol. 72, no. 3, pp. 588–608, 2019.

[132] L. Song, H. Chen, W. Xiong, Z. Dong, P. Mao, Z. Xiang, and K. Hu, "Method of emergency collision avoidance for unmanned surface vehicle (usv) based on motion ability database," *Polish Maritime Research*, vol. 26, no. 2, pp. 55–67, 2019.

[133] L. Zhao and M.-I. Roh, "Colregs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, 2019.

[134] X. Liu and Y. Jin, "Reinforcement learning-based collision avoidance: Impact of reward function and knowledge transfer," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 2020.

[135] T. G. Fowler and E. Sørgård, "Modeling ship transportation risk," *Risk Analysis*, vol. 20, no. 2, pp. 225–244, 2000.

[136] K. Bergman, O. Ljungqvist, and D. Axehill, "Improved path planning by tightly combining lattice-based path planning and optimal control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, pp. 57–66, 2020.

[137] H. Lyu and Y. Yin, "Fast path planning for autonomous ships in restricted waters," *Applied Sciences (Switzerland)*, vol. 8, no. 12, 2018.

[138] F. Xiao and Y. Ma, "Artificial forces for virtual autonomous ships with encountering situations in restricted waters," *Maritime Policy and Management*, vol. 47, no. 5, pp. 687–702, 2020.

[139] J. Zheng, W. Sun, Y. Li, and J. Hu, "A receding horizon navigation and control system for autonomous merchant ships: Reducing fuel costs and carbon emissions under the premise of safety," *Journal of Marine Science and Engineering*, vol. 11, no. 1, 2023.

[140] S.-W. Ohn and H. Namgung, "Requirements for optimal local route planning of autonomous ships," *Journal of Marine Science and Engineering*, vol. 11, no. 1, 2023.

[141] X. Chen, M. Gao, Z. Kang, J. Zhou, S. Chen, Z. Liao, H. Sun, and A. Zhang, "Formation of mass collision avoidance and path following based on artificial potential field in constrained environment," *Journal of Marine Science and Engineering*, vol. 10, no. 11, 2022.

[142] A. Hovenburg, F. Andrade, C. D. Rodin, T. A. Johansen, and R. Storvold, "Contingency path planning for hybrid-electric uas," in *Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linköping*, 2017.

[143] A. R. Hovenburg, F. A. de Alcantara Andrade, R. Hann, C. D. Rodin, T. A. Johansen, and R. Storvold, "Long range path planning using an aircraft performance model for battery powered sUAS equipped with icing protection system," *IEEE J. Miniaturization for air and space systems*, vol. 1, pp. 76–89, 2020.

[144] B. Rokseth, “Ship in transit simulator,” https://github.com/BorgeRokseth/ship_in_transit_simulator, Dec. 2022.

[145] A. J. Sørensen, S. I. Sagatun, and T. I. Fossen, “Design of a dynamic positioning system using model-based control,” *Control Engineering Practice*, vol. 4, no. 3, pp. 359–368, 1996.

[146] I. B. Hagen, “On maritime collision avoidance,” 2022.

[147] J. Szlapczynska and R. Szlapczynski, “Preference-based evolutionary multi-objective optimization in ship weather routing,” *Applied Soft Computing*, vol. 84, p. 105 742, 2019.

[148] NSIA, “Interim report 12 november 2019 on the investigation into the loss of propulsion and near grounding of viking sky, 23 march 2019,” Norwegian Safety Investigation Authority, Tech. Rep., 2019.

[149] C. A. Thieme, B. Rokseth, and I. B. Utne, “Risk-informed control systems for improved operational performance and decision-making,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, p. 1748006X211043657, 2021.

[150] T. Johansen and I. B. Utne, “Supervisory risk control of autonomous surface ships,” *Ocean Engineering*, vol. 251, p. 111 045, 2022.

[151] J. E. Bremnes, C. A. Thieme, A. J. Sørensen, I. B. Utne, and P. Norgren, “A bayesian approach to supervisory risk control of AUVs applied to under-ice operations,” *Marine Technology Society Journal*, vol. 54, no. 4, pp. 16–39, 2020.

[152] J. E. Bremnes, P. Norgren, A. J. Sørensen, C. A. Thieme, and I. B. Utne, “Intelligent risk-based under-ice altitude control for autonomous underwater vehicles,” *OCEANS 2019 MTS/IEEE Seattle*, pp. 1–8, 2019.

[153] C. Fan, K. Wróbel, J. Montewka, M. Gil, C. Wan, and D. Zhang, “A framework to identify factors influencing navigational risk for maritime autonomous surface ships,” *Ocean Engineering*, vol. 202, p. 107 188, 2020.

[154] C. H. Chang, C. Kontovas, Q. Yu, and Z. Yang, “Risk assessment of the operations of maritime autonomous surface ships,” *Reliability Engineering & System Safety*, vol. 207, 2021.

[155] T. Johansen and I. B. Utne, “Risk analysis of autonomous ships,” *e-proceedings of the 30th European Safety and Reliability Conference and 15th Probabilistic Safety Assessment and Management Conference (ESREL2020 PSAM15)*, pp. 131–138, 2020.

[156] O. A. Valdez Banda, S. Kannos, F. Goerlandt, P. H. A. J. M. van Gelder, M. Bergström, and P. Kujala, “A systemic hazard analysis and management process for the concept design phase of an autonomous vessel,” *Reliability Engineering & System Safety*, vol. 191, p. 106 584, 2019.

[157] K. Wróbel, J. Montewka, and P. Kujala, “Towards the development of a system-theoretic model for safety assessment of autonomous merchant vessels,” *Reliability Engineering & System Safety*, vol. 178, pp. 209–224, 2018.

[158] M. Chaal, O. A. Valdez Banda, J. A. Glomsrud, S. Basnet, S. Hirdaris, and P. Kujala, "A framework to model the stpa hierarchical control structure of an autonomous ship," *Safety Science*, vol. 132, 2020.

[159] M. Brito and G. Griffiths, "A bayesian approach for predicting risk of autonomous underwater vehicle loss during their missions," *Reliability Engineering & System Safety*, vol. 146, pp. 55–67, 2016.

[160] T. Y. Loh, M. P. Brito, N. Bose, J. Xu, N. Nikolova, and K. Tenekedjiev, "A hybrid fuzzy system dynamics approach for risk analysis of auv operations," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 24, no. 1, pp. 26–39, 2020.

[161] T. Y. Loh, M. P. Brito, N. Bose, J. Xu, and K. Tenekedjiev, "Fuzzy system dynamics risk analysis (fusdra) of autonomous underwater vehicle operations in the antarctic," *Risk Analysis*, vol. 40, no. 4, pp. 818–841, 2020.

[162] M. Brito, "Uncertainty management during hybrid autonomous underwater vehicle missions," *Autonomous Underwater Vehicles 2016, AUV 2016*, pp. 278–285, 2016.

[163] L. Hu, W. Naeem, E. Rajabally, G. Watson, T. Mills, Z. Bhuiyan, and I. Salter, "Colregs-compliant path planning for autonomous surface vehicles: A multiobjective optimization approach," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13 662–13 667, 2017.

[164] H. Wang, F. Guo, H. Yao, S. He, and X. Xu, "Collision Avoidance Planning Method of USV Based on Improved Ant Colony Optimization Algorithm," *IEEE Access*, vol. 7, pp. 52 964–52 975, 2019.

[165] J. Woo and N. Kim, "Collision avoidance for an unmanned surface vehicle using deep reinforcement learning," *Ocean Engineering*, vol. 199, p. 107 001, 2020.

[166] M. Li, J. Mou, L. Chen, Y. He, and Y. Huang, "A rule-aware time-varying conflict risk measure for mass considering maritime practice," *Reliability Engineering & System Safety*, vol. 215, 2021.

[167] M. Gil, "A concept of critical safety area applicable for an obstacle-avoidance process for manned and autonomous ships," *Reliability Engineering & System Safety*, vol. 214, 2021.

[168] T. Tengesdal, E. F. Brekke, and T. A. Johansen, "On collision risk assessment for autonomous ships using scenario-based MPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 509–14 516, 2020, 21st IFAC World Congress.

[169] T. Tengesdal, T. A. Johansen, and E. F. Brekke, "Risk-based autonomous maritime collision avoidance considering obstacle intentions," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020.

[170] J. Yin, Y. Wang, J. Lv, and J. Ma, "Study on underwater simultaneous localization and mapping based on different sensors," *Proceedings of 2021 IEEE 10th Data Driven Control and Learning Systems Conference, DDCLS 2021*, pp. 728–733, 2021.

[171] J. S. Willners, Y. Carreno, S. Xu, T. Luczynski, S. Katagiri, J. Roe, È. Pairet, Y. Petillot, and S. Wang, “Robust underwater slam using autonomous relocalisation,” *IFAC-PapersOnLine*, vol. 54, no. 16, pp. 273–280, 2021.

[172] S. S. Sandøy, J. Hegde, I. Schjølberg, and I. B. Utne, “Polar map: A digital representation of closed structures for underwater robotic inspection,” *Aquacultural Engineering*, vol. 89, p. 102 039, 2020.

[173] J. Mąka and J. Magaj, “Data extraction from an electronic s-57 standard chart for navigational decision systems,” *Proc. Zeszyty Naukowe/Akademia Morska Szczecinie*, pp. 83–87, 2012.

[174] J. de Vos, R. G. Hekkenberg, and O. A. Valdez Banda, “The impact of autonomous ships on safety at sea – a statistical analysis,” *Reliability Engineering & System Safety*, vol. 210, 2021.

[175] IMO, *MSC.1/Circ.1580: Annex: Guidelines for vessels with dynamic positioning systems*, IMO, 2017.

[176] T. Torben, Ø. Smogeli, I. B. Utne, and A. J. Sørensen, “On formal methods for design and verification of maritime autonomous surface ships,” *Proceedings of the World Maritime Technology Conference.*, 2022.

[177] T. Torben, J. A. Glomsrud, T. A. Pedersen, I. B. Utne, and A. J. Sørensen, “Automatic simulation-based testing of autonomous ships using gaussian processes and temporal logic,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, p. 1748006X211069277, 2022.

[178] G. Xiao, B. Ren, C. Tong, and X. Hong, “A quantitative evaluation method for obstacle avoidance performance of unmanned ship,” *Journal of Marine Science and Engineering*, vol. 9, no. 10, p. 1127, 2021.

[179] Fenton, N. and Neil, M., *Risk Assessment and Decision Analysis with Bayesian Networks.* CRC Presss, 2019.

[180] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Lecture Notes in Computer Science*, 2004, pp. 152–166.

[181] B. Rokseth and I. B. Utne, “Deriving safety requirement hierarchies for families of maritime systems,” *Transactions of the Royal Institution of Naval Architects Part A: International Journal of Maritime Engineering*, vol. 161, A229–A243, 2019.

[182] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[183] IMO, “Revised Guidelines for Formal Safety Assessment (FSA) for Use in the IMO Rule-Making Process,” IMO, Tech. Rep., 2018.

[184] EfficienSea, “Methods to Quantify Maritime Accidents for Risk-based decision making,” EfficienSea, Tech. Rep., 2012.

[185] The Norwegian Agency for Public and Financial Management, *Guide in socio-economic analysis*, 2018.

[186] DNVGL, “DNV Report no 2003-0277 Annex II FSA 2003,” DNVGL group technology & research, Tech. Rep., 2003.

[187] M. Hassel, I. B. Utne, and J. E. Vinnem, “An allision risk model for passing vessels and offshore oil and gas installations on the norwegian continental shelf,” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 235, no. 1, pp. 17–32, 2021.

[188] Norwegian Meteorological Institute, *Met*, 2021.

[189] Norwegian Mapping Authority, *Norgeskart*, 2021.

[190] Ship & Bunker, *Rotterdam Bunker Prices*, 2022.

[191] V. Utkin and H. Lee, “Chattering problem in sliding mode control systems,” *IFAC Proceedings Volumes*, vol. 39, no. 5, p. 1, 2006, 2nd IFAC Conference on Analysis and Design of Hybrid Systems.

[192] Norwegian Hydrographic Service, *The Norwegian Pilot Guide*, 2018.

[193] I. M. O. (IMO), “COLREGS - International Regulations for Preventing Collisions at Sea,” *Convention on the International Regulations for Preventing Collisions at Sea, 1972*, 1972.

[194] 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, Dec. 2016.

[195] T. Tengesdal, T. A. Johansen, and E. F. Brekke, “Ship collision avoidance utilizing the cross-entropy method for collision risk assessment,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 148–11 161, 2022.

[196] C. Tam, R. Bucknall, and A. Greig, “Review of collision avoidance and path planning methods for ships in close range encounters,” *The Journal of Navigation*, vol. 62, no. 3, pp. 455–476, 2009.

[197] Y. Huang, L. Chen, P. Chen, R. R. Negenborn, and P. van Gelder, “Ship collision avoidance methods: State-of-the-art,” *Safety Science*, vol. 121, pp. 451–473, 2020.

[198] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[199] C. Chauvin and S. Lardjane, “Decision making and strategies in an interaction situation: Collision avoidance at sea,” *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 11, no. 4, pp. 259–269, 2008.

[200] S. R. Clawson Jr, *Overtaking or crossing? Don’t assume what other ship will do*, 2013.

[201] K. Woerner, M. R. Benjamin, M. Novitzky, and J. J. Leonard, “Quantifying protocol evaluation for autonomous collision avoidance,” *Autonomous Robots*, vol. 43, no. 4, pp. 967–991, Apr. 2019.

[202] B. C. Shah, P. Švec, I. R. Bertaska, A. J. Sinisterra, W. Klinger, K. von Ellenrieder, M. Dhanak, and S. K. Gupta, “Resolution-adaptive risk-aware trajectory planning for surface vehicles operating in congested civilian traffic,” *Autonomous Robots*, vol. 40, no. 7, pp. 1139–1163, 2016.

[203] B.-O. H. Eriksen, G. I. Bitar, M. Breivik, and A. M. Lekkas, “Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13-17,” *ArXiv*, vol. abs/1907.00198, 2019.

[204] P. Agrawal and J. M. Dolan, “COLREGS-compliant target following for an unmanned surface vehicle in dynamic environments,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1065–1070.

[205] X. Rong Li and V. P. Jilkov, “Survey of maneuvering target tracking. part i. dynamic models,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, Oct. 2003.

[206] L. M. Millefiori, P. Braca, K. Bryan, and P. Willett, “Long-term vessel kinematics prediction exploiting mean-reverting processes,” in *2016 19th International Conference on Information Fusion (FUSION)*, Jun. 2016, pp. 232–239.

[207] M. Abdelaal, M. Fränzle, and A. Hahn, “Nonlinear model predictive control for trajectory tracking and collision avoidance of underactuated vessels with disturbances,” *Ocean Engineering*, vol. 160, pp. 168–180, 2018.

[208] H. 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, Jul. 2018.

[209] Y. Lyu, J. Hu, B. M. Chen, C. Zhao, and Q. Pan, “Multivehicle flocking with collision avoidance via distributed model predictive control,” *IEEE Transactions on Cybernetics*, vol. 51, no. 5, pp. 2651–2662, 2021.

[210] D. H. Douglas and T. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112–122, 1973.

[211] L. M. Millefiori, P. Braca, K. Bryan, and P. Willett, “Modeling vessel kinematics using a stochastic mean-reverting process for long-term prediction,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2313–2330, 2016.

[212] M. Breivik and T. I. Fossen, “Guidance laws for planar motion control,” in *Proc. 47th IEEE Conf. Decision and Control*, Dec. 2008, pp. 570–577.

[213] B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[214] C.-W. Huang and T.-Y. Shih, “On the complexity of point-in-polygon algorithms,” *Computers and Geosciences*, vol. 23, no. 1, pp. 109–118, 1997.

[215] D. K. M. Kufoalor, E. Wilthil, I. B. Hagen, E. F. Brekke, and T. A. Johansen, “Autonomous COLREGs-compliant decision making using maritime radar tracking and model predictive control,” in *Proc. 18th European Control Conf. (ECC)*, Jun. 2019, pp. 2536–2542.

[216] S. V. Rothmund, T. Tengesdal, E. F. Brekke, and T. A. Johansen, “Intention modeling and inference for autonomous collision avoidance at sea,” *Ocean Engineering*, vol. 266, p. 113 080, 2022.

ISBN 978-82-326-7102-1 (printed ver.)
ISBN 978-82-326-7101-4 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU
Norwegian University of
Science and Technology