Abstract

This Ph.D. thesis presents developments towards energy-optimized trajectory planning and automatic docking for surface vessels. It comprises eight peer-reviewed research papers, in addition to the introductory chapters that provide context to the thesis' main topics and a summary of the contributions.

Autonomy can lead to safer and more energy-efficient maritime operations and enable new, low-cost opportunities within the maritime domain. Commercial actors have already utilized autonomous technology for maritime transportation, surveillance, and research operations. The Norwegian government and several commercial actors are currently exerting efforts to electrify and bring autonomy to car and passenger ferries. Through the Norwegian Research Council's project "Energy-optimized concept for all-electric, emission-free and autonomous ferries in integrated transport and energy systems," this thesis focuses on motion control and planning during the three phases of automatic crossing: undocking, transit, and docking.

The methods developed in this thesis are focused on practical applications and are tested in full-scale experiments. The thesis contributes to model-based, energy-optimized trajectory planning, a collision-avoidance system compliant with the COLREGs (International Regulations for Preventing Collisions at Sea), a method for automatic docking, and a system for performing automatic crossing, which comprises the phases mentioned above: undocking, transit, and docking. Additionally, the thesis contributes to development and improvements for the experimental autonomous test ferry *milliAmpere*.

As part of this thesis, a method for optimization-based model identification for *milliAmpere* has been developed. The result is a nonlinear three-degree-of-freedom dynamic model, plus a set of models for thruster dynamics, thruster force mapping, and wind effects. Additionally, the Ph.D. work has contributed to a web-based graphical user interface for *milliAmpere* and a software-in-the-loop simulator. These contributions have helped lower the threshold for experimental testing of autonomous

technology.

Most of the academic literature on trajectory planning is towards time- and distance-optimized trajectories, with little focus on energy-optimization and closed-loop results. This thesis presents four model-based, energy-optimized trajectory planning methods for autonomous surface vehicles (ASVs), with various warm-starting techniques, obstacle representations, and solver designs. The most recently proposed method can consider arbitrary maps and disturbances in the form of wind and has been shown to produce dynamically feasible trajectories in validation experiments.

In order to consider moving obstacles in accordance with the COLREGs, dynamic collision avoidance methods must recognize various situations when encountering other vessels and produce behaviors that fulfill the rules of those regulations. In this regard, a hierarchical collision-avoidance system that complies with rules 8 and 13–17 of the COLREGs was developed as part of this thesis. The collision-avoidance system is a three-layered architecture that handles nominal, global trajectory planning, long-term dynamic collision-avoidance that considers the COLREGs, and short-term, safe maneuvering. The system has been tested in simulations, where it safely handled a wide range of situations. This work is among the first in academia that complies with the most critical parts of the COLREGs regarding motion planning and control.

The literature on automatic docking in the maritime domain is mostly limited to underwater vehicles. The few publications that deal with surface vessels fail to consider obstacles and the harbor layout explicitly. This thesis presents work on an optimization-based method for automatic docking of ASVs, which takes into account harbor layouts and obstacles, using a priori map information and information from exteroceptive sensors. The method is successfully tested in several full-scale experiments.

While some commercial actors have performed sea trials with surface ships crossing automatically from one dock to another, they do not publish information about the methods used in such operations. There are also few results from automatic crossing in the academic literature. In this thesis, a framework for automatic crossing for a surface vessel is presented. The framework combines different trajectory planners and control modules to automate the three phases of undocking, transit, and docking. The framework has been tested experimentally and has resulted in successful automatic dock-to-dock crossing operations.

Development and improvement of technologies such as trajectory planning, COLREGs-compliant collision avoidance systems, and automated docking methods are small steps towards an autonomous future at sea.

Preface

I have worked with the material in this thesis since I started on my master's thesis in 2017, until the end of my Ph.D. in 2020. During this time at the Department of Engineering Cybernetics (ITK) at the Norwegian University of Science and Technology (NTNU), I have been supervised by Dr. Morten Breivik and Prof. Anastasios "Tasos" M. Lekkas. My Ph.D. was a part of the Norwegian Research Council (NRC) project "Energioptimalisert konsept for hel-elektriske, utslippsfrie og autonome ferjer i integrerte transport og energisystemer," which translates to "Energy-optimized concept for fully electric, emission-free, and autonomous ferries in integrated transport and energy systems," henceforth known as the *Autonomous Ferry project* (project number 269116). The Autonomous Ferry project was an innovation project that was part of the NRC's Pilot-E program, and had the industrial partners Kongsberg Maritime, Grenland Energy, Fjellstrand, and Grønn Kontakt. I am grateful to have been a part of the project. Furthermore, I have been affiliated with the Centre for Autonomous Marine Operations and Systems (AMOS), which is an NRC Centre of Excellence (project number 223254). AMOS has provided me with many good networking opportunities.

Tired of school at 16, I would never have thought to end up doing a Ph.D. in anything. Instead, I chose to attend a vocational school in automation and to subsequently do an apprenticeship. Curiosity got the better of me after that, which led me to pursue a bachelor's degree in computer science and industrial automation at Telemark University College in Porsgrunn. Student life was delightful, which made continuing with a master's degree in cybernetics and robotics at NTNU in Trondheim an easy decision. There I got in touch with Morten, who provided an opportunity to begin a Ph.D. in autonomous ferries. Having practical experience has led me to focus on applicable and practical solutions to the research challenges I have encountered. I hope that this focus is reflected in the thesis and the related publications. Right now, I am relieved to be at the end of 23 years of elementary, secondary, and higher education. Although I have

enjoyed being a student and will cultivate the friendships, curiosity, and independence I have gained along the way, I'm relieved to be thrown back into the real world.

During this academic journey, and most significantly during my Ph.D., there have been times where I have had the need for motivation, support, and love. I am very lucky to have many to thank for all of the above. Morten and Tasos, thank you for guiding me through the academic, technical, and personal challenges that have emerged throughout the last three years by being my supervisors. I believe you understand that this would not be possible without you.

My friends at ITK, I am truly proud to know all of you. I have been spoiled by our excellent work environment, and I'm afraid to never find such a good group of friends and coworkers again.

Filippo, Giorgio, and Emil, thanks for sharing offices with me. It's not just about sharing space, but also ideas, troubles, and joys.

Bjørn-Olav and Andreas, I appreciate that you have wanted to collaborate on projects with me. Our discussions and teamwork have been incredibly valuable to me and have given me that sorely needed motivation. Thank you very much.

I appreciate having gotten to know all the members of the coffee-break club during these years. Thanks for all the useless and hilarious discussions, bouldering sessions, and ski trips. I really hope that we will keep in touch! Gunhild, Håkon, and Inger, I am very happy that I was placed in the office next to yours when I started at ITK. You have been invaluable to me.

Erlend and Viggo, thanks for staying behind in Trondheim after the master's, and for being great friends and discussion partners.

Adrian, thanks for keeping me in touch with real life. I really wish Trondheim was closer to Skien!

Mom and dad, obviously, without you, there would be no me. And I believe you deserve much more credit than that. Credit for giving me so many opportunities—many times by making personal sacrifices, and many times by just being great—and for supporting everything that I've done. Thank you. Martin, I love that you are always so creative, friendly, open, and funny. Emma, I'm proud of your hard work and for always standing up for what you believe. Thank you both so much for supporting your annoying big brother.

Eirin, you keep giving me so many reasons to look forward to tomorrow. Du er best.

Contents

Acronyms

ANN artificial neural network

APF artificial potential field

ASV autonomous surface vehicle

AUV autonomous underwater vehicle

BC-MPC branching-course MPC

CG center of gravity

CO control origin

COLREGs International Regulations for Preventing Collisions at Sea

DOF degree-of-freedom

DP dynamic positioning

DW dynamic window

IMO International Maritime Organization

MPC model predictive control

NED North-East-Down

NLP nonlinear program

OCP optimal control problem

ODE ordinary differential equation

PDE partial differential equation

PRM probabilistic roadmap

ROS Robot Operating System

RRT rapidly-exploring random tree

SB-MPC scenario-based MPC

UML-SD Unified Modeling Language state diagram

USV unmanned surface vehicle

VO velocity obstacle

Chapter 1

Introduction

1.1 Motivation

Automation and robotics are increasingly a part of our daily lives. Computer-controlled robotic systems have automated dull, dirty, and dangerous tasks across many industries. As computers and algorithms increase in power and sensors and data are becoming more available, autonomy is increasing in prevalence. The automotive industry might be the most available example that has seen massive leaps towards autonomy in recent years, with large, visible players such as Waymo and Tesla working hard to develop completely autonomous cars and trucks. The maritime industry is also increasingly developing automated and autonomous systems to perform transportation, surveillance, and marine operations.

The distinction between automated and autonomous systems is blurred, and many actors have different definitions. The definitions are also dependent on the context of the domain and if the topic is communicated to the general public or to an expert audience. In the automotive domain, SAE International calls self-driving cars *automated vehicles* and has published a scale of levels of automation [1]. In the maritime domain, Rødseth and Vagia [2] provide a useful survey and discussion on distinguishing the terms. In this thesis, the distinction is made as follows: In contrast to *automated systems*, which can handle simple tasks with limited scope, *autonomous systems* can plan, schedule, and execute tasks to complete a general goal without human intervention.

The motivation for increased autonomy at sea is threefold. First, the potential for safer maritime operations is considerable, as some reports state that human errors cause more than 75 % of maritime accidents [3, 4]. Implementing safe and robust autonomous systems for motion control and navigation at sea, compliant with the International Regulations for Preventing Collisions at Sea (COLREGs)[1] [5], can make maritime

Figure 1.1: Illustration of an autonomous passenger ferry in an urban setting, which is a novel concept enabled by autonomous technology. © 2020, Zeabuz.

operations more reliable, repeatable, and safe.

Second, automation and autonomy can make maritime operations more energy-efficient and reduce environmental impact. Automation can, e.g., make the operations more consistent, and optimization-based methods can reduce energy consumption. The CO_2 footprint of the world's maritime industry is significant. For example, 9 % of the Norwegian domestic CO_2 emissions are caused by ships [6], and the maritime industry is responsible for 2.5 % of global greenhouse gas emissions.[2] Therefore, a reduction in general energy consumption for ships will have an immense impact on the environment.

Third, in addition to improvements in safety and energy consumption in existing maritime industries, autonomy can also enable new opportunities. For example, unmanned surface vehicles (USVs) and autonomous underwater vehicles (AUVs) have been used to increase ocean floor surveys' efficiency and range.[3] Another example is low-cost, autonomous passenger ferries that can revitalize hard-to-access areas in cities divided by water-

[1]The COLREGs convention document is available at `https://www.imo.org/en/About/Conventions/Pages/COLREG.aspx` (accessed November 3, 2020).

[2]European Commission policy on reducing emissions from the shipping sector: `https://ec.europa.eu/clima/policies/transport/shipping_en` (accessed November 3, 2020).

[3]Unmanned Systems Technology article on automated mapping of ocean floors: `https://www.unmannedsystemstechnology.com/2018/01/combined-usv-auv-system-maps-ocean-floor` (accessed November 3, 2020).

ways. A company called Zeabuz, based in Trondheim, Norway is working to deliver services for such ferries, which are a novel concept enabled by autonomy, see Figure 1.1.[4]

Commercial actors have already utilized autonomous technology for maritime transportation. NYK is a Japanese shipping company that completed autonomous surface ship trials in 2019.[5] The transportation ship *Iris Leader* navigated autonomously between Xinsha in China to Nagoya in Japan, which was a big step towards NYK's goal of manned autonomous shipping operations. In the Nordic countries, several actors have contributed to automating car and passenger ferries. In 2018, both Wärtsilä and Rolls-Royce Marine (later acquired by Kongsberg Maritime) demonstrated autonomous capabilities with the ferries *Folgefonn* and *Falco*, respectively.[6] Both tests included automatic transit and docking.

This Ph.D. thesis fits into the efforts made by the Norwegian government and several commercial actors to automate car ferries. It is a part of the Norwegian Research Council (NRC) *Autonomous Ferry project*,[7] which has focused on exploring, researching, and developing energy-optimized solutions for electric, autonomous car ferries. Autonomous ships must safely handle planning and maneuvering during the following phases without human intervention:

Undocking: Moving at slow speeds from a docked position along a quay to more open waters.

Transit: Moving at higher speeds towards the destination while adhering to the COLREGs.

Docking: Moving at slow speeds in the proximity of the destination to a docked position along a quay.

Enabling technologies include path or trajectory planning, COLREGs-compliant collision avoidance, and automated docking, which are the main topics of this thesis. Since the Autonomous Ferry project focuses on energy-optimized solutions, this thesis' objective has been to develop methods

[4]CNN Travel article on Zeabuz and self-driving ferries: `https://edition.cnn.com/travel/article/norway-self-driving-ferries-zeabuz-spc-intl/index.html` (accessed November 3, 2020).

[5]NYK press release on their autonomous surface ship trial: `https://www.nyk.com/english/news/2019/20190930_01.html` (accessed August 31, 2020).

[6]Maritime Executive article on Rolls-Royce and Wärtsilä autonomous ferry trials: `https://www.maritime-executive.com/article/rolls-royce-and-wartsila-in-close-race-with-autonomous-ferries` (accessed September 14, 2020).

that produce energy-optimized maneuvers.

1.2 Contributions at a glance

The main contributions in this thesis are towards energy-optimized trajectory planning for autonomous surface vehicles (ASVs) and automatic docking. Work has also been done towards combining these elements in a system for automatic crossing. Also, there are contributions to COLREGs-compliant collision avoidance. Lastly, significant work has been put into further developing the experimental platform *milliAmpere*. The contributions can be summarized as follows, with references to papers listed in Section 1.3:

- Identification of model parameters for *milliAmpere* using a data-driven, optimization-based method.
- Development of a software-in-the-loop simulator, a user interface, and safety functions for *milliAmpere*.
- Development of a model-based energy-optimized path planner using pseudospectral optimal control. This method is presented in Paper A and takes into account ocean currents and static obstacles in the form of ellipses.
- Development of three warm-started model-based energy-optimized trajectory planners that use multiple-shooting approaches. The planners use various methods for warm starting and obstacle representation. The methods are presented in Paper C, [7], and Paper F.
- Development of a three-layered hierarchical collision avoidance architecture that takes into account the COLREGs rules 8 and 13–17. A two-layered implementation is presented in Paper B, while the three-layered implementation that takes into account said rules is presented in Paper D.
- Development and experimental validation of a method for automatic docking of an ASV. The method comprises a trajectory planner that takes into account static obstacles and replans at a set interval, and a trajectory-tracking controller. It is developed and implemented in Paper E and extended with exteroceptive sensors in Paper G.

[7] This Ph.D. is part of the NRC project "Energioptimalisert konsept for hel-elektriske, utslippsfrie og autonome ferjer i integrerte transport og energisystemer," which translates to "Energy-optimized concept for fully electric, emission-free, and autonomous ferries in integrated transport and energy systems." It has the NRC project number 269116. The project is here abbreviated as the *Autonomous Ferry project*.

- Development and experimental validation of an architecture for automatic crossing. The architecture combines trajectory planning and docking to perform the three phases of a crossing operation automatically: undocking, transit, and docking. The architecture is presented in Paper H.

A more in-depth discussion of these contributions is provided in Chapter 3.

1.3 Publications

This thesis is based on eight peer-reviewed papers enumerated as Paper A through Paper H in chronological order of publication. Paper H is currently under review for presentation at the European Control Conference in 2021. Additionally, the Ph.D. work has involved co-supervision of three master students who have worked on model identification, path and trajectory planning, and automatic docking. Lastly, the thesis presents contributions from an unpublished paper that represents improvements upon the work in Paper C. The contributions of the research papers and M.Sc. theses can be illustrated in a control system block diagram as seen in Figure 1.2.

Peer-reviewed papers

Paper A [8]

G. Bitar, M. Breivik, and A. M. Lekkas. "Energy-optimized path planning for autonomous ferries". In: *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. Opatija, Croatia, 2018, pp. 389–394. DOI: 10.1016/j.ifacol.2018.09.456.

Paper B [9]

G. Bitar, B.-O. H. Eriksen, A. M. Lekkas, and M. Breivik. "Energy-optimized hybrid collision avoidance for ASVs". In: *Proceedings of the 18th European Control Conference (ECC)*. Naples, Italy, 2019, pp. 2522–2529. DOI: 10.23919/ECC.2019.8795645.

Paper C [10]

G. Bitar, V. N. Vestad, A. M. Lekkas, and M. Breivik. "Warm-started optimized trajectory planning for ASVs". In: *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. Daejeon, South Korea, 2019, pp. 308–314. DOI: 10.1016/j.ifacol.2019.12.325. arXiv: 1907.02696 [eess.SY].

Paper D [11]

B.-O. H. Eriksen, **G. Bitar**, M. Breivik, and A. M. Lekkas. "Hybrid

collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17". In: *Frontiers in Robotics and AI* 7 (2020). DOI: 10.3389/frobt.2020.00011. arXiv: 1907.00198 [eess.SY].

Paper E [12]

G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik. "Trajectory planning and control for automatic docking of ASVs with full-scale experiments". In: *Proceedings of the 1st Virtual IFAC World Congress*. 2020. arXiv: 2004.07793 [eess.SY].

Paper F [13]

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". In: *IEEE Access* 8 (2020), pp. 199953–199969. DOI: 10.1109/ACCESS.2020.3035256.

Paper G [14]

A. B. Martinsen, **G. Bitar**, A. M. Lekkas, and S. Gros. "Optimization-based automatic docking and berthing of ASVs using exteroceptive sensors: theory and experiments". In: *IEEE Access* 8 (2020), pp. 204974–204986. DOI: 10.1109/ACCESS.2020.3037171.

Paper H [15]

G. Bitar, B.-O. H. Eriksen, A. M. Lekkas, and M. Breivik. *Three-phase automatic crossing for a passenger ferry with field trials*. Submitted for publication.

Master theses co-supervised

- A. A. Pedersen. "Optimization based system identification for the milliAmpere ferry". MA thesis. Norwegian University of Science and Technology, 2019. URL: http://hdl.handle.net/11250/2625699 [16].
- V. N. Vestad. "Automatic and practical route planning for ships". MA thesis. Norwegian University of Science and Technology, 2019. URL: http://hdl.handle.net/11250/2625697 [17].
- E. D. Molven. "Optimal control-based docking for autonomous ferries". MA thesis. Norwegian University of Science and Technology, 2020 [18].

Unpublished

- **G. Bitar**, A. M. Lekkas, and M. Breivik. *Improvements to warm-started optimized trajectory planning for ASVs*. arXiv: 1908.07311

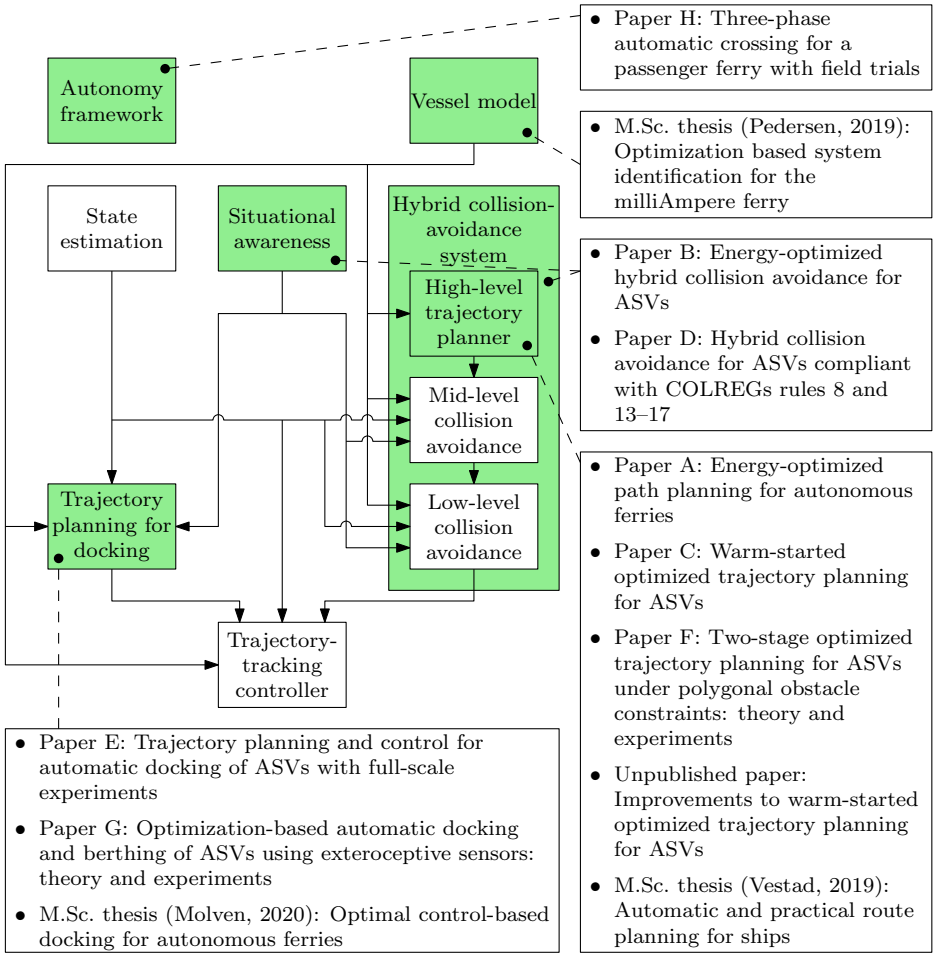


Figure 1.2: The research papers and M.Sc. theses arranged by topic in a control system block diagram for automatic docking and transit. The highlighted blocks are topics within which this Ph.D. thesis has significant contributions.

`[eess.SY]` [7].

1.4 Outline

The rest of this thesis is structured as follows: Background material on experimental platforms, vessel modeling, collision avoidance, trajectory planning, and automatic docking is presented in Chapter 2. A more detailed presentation of the thesis' contributions is provided in Chapter 3. Chapter 4 provides concluding remarks and recommendations for further work. The publications written as part of the Ph.D. work are reprinted in Chapter 5.

Chapter 2

Background

2.1 Experimental platform: *milliAmpere*

Several simulations and full-scale experiments have been performed as part of this thesis. Most of the simulations and all of the full-scale experiments have used *milliAmpere* as the experimental platform. Figure 2.1 shows *milliAmpere*, which is an experimental autonomous passenger ferry developed at the Norwegian University of Science and Technology (NTNU). Its specifications are listed in Table 2.1.

The *milliAmpere* is controlled by an onboard computer running Linux Ubuntu, and its control system software is built around the Robot Operating System (ROS), a software framework that enables inter-process communication and logging.[1] The *milliAmpere*'s two azimuth thrusters are controlled using the optimization-based thrust-allocation algorithm from [19]. That algorithm receives force and moment commands from a trajectory-tracking DP controller. The control setup is shown in Figure 2.2.

[1] See `https://www.ros.org` for information about ROS.

Table 2.1: *milliAmpere* specifications.

Dimensions	5 m by 2.8 m symmetric footprint
Position and heading reference system	Vector VS330 dual GNSS with RTK capabilities
Thrusters	Two azimuth thrusters on the center line, 1.8 m aft and fore of center
Existing control modules	Trajectory-tracking DP controller and thrust allocation system

Figure 2.1: The experimental electric autonomous passenger ferry *milliAmpere*.

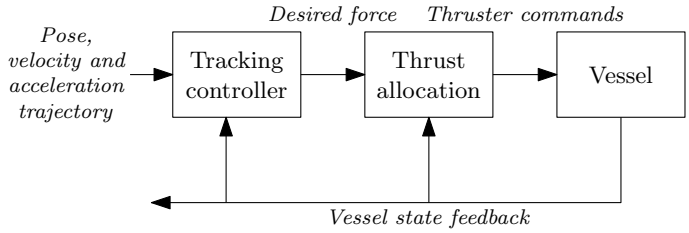


Figure 2.2: The control setup of *milliAmpere*, with the trajectory-tracking dynamic positioning (DP) controller and thrust allocation algorithm.

The *milliAmpere* is a part of the Autoferry research project at NTNU.[2] The project aims to develop concepts and methods to enable autonomous passenger ferries for transport in urban waterways. The project is also building a larger version of *milliAmpere*, named *milliAmpere 2*, which is intended to be in public traffic in a canal in Trondheim, Norway. A spin-off company from NTNU called Zeabuz is related to Autoferry and is working to commercialize the idea of small, urban passenger ferries.[3]

2.2 Vessel modeling and model identification

Many control approaches for maritime vessels are model-based. So are all the planners that are developed as part of this thesis. Model-based methods are beneficial for control and planning since inherent damping can

[2]Read more about Autoferry at `https://www.ntnu.edu/autoferry`.
[3]Read about Zeabuz at `https://zeabuz.com`.

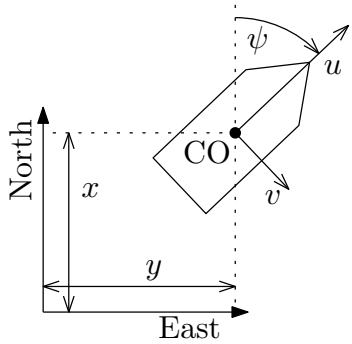


Figure 2.3: Illustration of the NED and body coordinate frames.

Table 2.2: Notation used for forces, positions, and velocities.

Symbol	Description
x	Distance North from NED origin to the CO
y	Distance East from NED origin to the CO
ψ	Heading angle relative to North (yaw angle)
u	Surge velocity
v	Sway velocity
r	Yaw rate
X	Surge force
Y	Sway force
N	Yaw moment

be exploited, and the necessary compensation needed to reach the control target can be provided in a feed-forward manner. Conversely, inaccurate models can lead to poorly performing controllers and may cause more harm than good. These benefits and drawbacks emphasize the need for accurate model identification.

Notation from [20] is used in this thesis and its associated papers. A three-degree-of-freedom (DOF) model that describes motion in the North-East plane is considered sufficient for trajectory planning and docking. Heave, pitch, and roll motion are out of scope for the applications presented here. The coordinate frame used for positional coordinates is called North-East-Down (NED), which is Earth-tangential and has its origin at a point on the Earth's surface. Vessel poses are denoted $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times S$, where x and y are distances North and East from the NED origin to the vessel's control origin (CO) (often its center of gravity (CG)), and ψ is the vessel's heading (also called yaw angle), where $\psi = 0^\circ$ points to North, and is increasing with clockwise rotation, seen from above. An illustration of these coordinates is provided in Figure 2.3. Table 2.2 lists the variables

that are used in the model.

A vessel's 3-DOF velocity vector is denoted $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$ and describes the body-fixed velocity in surge (along the vessel's longitudinal axis), sway (across that axis), and yaw rate (angular velocity), respectively. See Figure 2.3 for an illustration of the surge and sway velocities. The kinematic relationship between the velocity and pose vectors is described with a transformation matrix, in this 3-DOF case, the rotational matrix $\mathbf{R} : S \mapsto SO(3)$:

$$\dot{\boldsymbol{\eta}}(t) = \mathbf{R}(\psi(t))\boldsymbol{\nu}(t)\,. \tag{2.1}$$

The rotational matrix is

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

The time dependence is often omitted for brevity.

Vessel kinetics can be described with the following set of ordinary differential equations (ODEs):

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}\,. \tag{2.3}$$

The system inertia matrix $\mathbf{M} \in \mathbb{R}^{3\times3}$ is positive definite and determines mass, hydrodynamically added mass, and coupling effects. The Coriolis and centripetal effects are captured in $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, and the linear and nonlinear hydrodynamic damping effects are captured in $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$. The control forces are $\boldsymbol{\tau} = [X, Y, N]^\top \in \mathbb{R}^3$, collected in a vector composed of the force in surge and sway, and the moment in yaw, respectively. The disturbances caused by environmental effects, such as waves, currents, and wind, are captured in $\boldsymbol{\tau}_{\text{env}} \in \mathbb{R}^3$, which has the same composition as $\boldsymbol{\tau}$.

The Coriolis and centripetal matrix $\mathbf{C}(\boldsymbol{\nu})$ is dependent on the elements of the system inertia matrix

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix}, \tag{2.4}$$

so that

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -c_1(\boldsymbol{\nu}) \\ 0 & 0 & c_2(\boldsymbol{\nu}) \\ c_1(\boldsymbol{\nu}) & -c_2(\boldsymbol{\nu}) & 0 \end{bmatrix}, \tag{2.5}$$

where

$$c_1(\boldsymbol{\nu}) = m_{22}v + \frac{1}{2}(m_{23} + m_{32})r \tag{2.6a}$$

$$c_2(\boldsymbol{\nu}) = m_{11}u\,. \tag{2.6b}$$

This is one of several possible parametrizations of $\mathbf{C}(\boldsymbol{\nu})$. Alternative parametrizations can be found in, e.g., [20].

The damping matrix $\mathbf{D}(\boldsymbol{\nu})$ can be parametrized in many ways, with one possible parametrization being the one used in simulations of *milliAmpere*:

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} d_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d_{22}(\boldsymbol{\nu}) & d_{23}(\boldsymbol{\nu}) \\ 0 & d_{32}(\boldsymbol{\nu}) & d_{33}(\boldsymbol{\nu}) \end{bmatrix}, \tag{2.7}$$

where

$$d_{11}(\boldsymbol{\nu}) = -X_u - X_{|u|u}|u| - X_{uuu}u^2 \tag{2.8a}$$
$$d_{22}(\boldsymbol{\nu}) = -Y_v - Y_{|v|v}|v| - Y_{|r|v}|r| - Y_{vvv}v^2 \tag{2.8b}$$
$$d_{23}(\boldsymbol{\nu}) = -Y_r - Y_{|v|r}|v| - Y_{|r|r}|r| \tag{2.8c}$$
$$d_{32}(\boldsymbol{\nu}) = -N_v - N_{|v|v}|v| - N_{|r|v}|r| \tag{2.8d}$$
$$d_{33}(\boldsymbol{\nu}) = -N_r - N_{|v|r}|v| - N_{|r|r}|r| - N_{rrr}r^2\,. \tag{2.8e}$$

The identification of these model parameters is not trivial. The Marine Systems Simulator toolbox[4] for Matlab has models and parameters for some specific vessel hulls, but an effort to identify such parameters must generally be undertaken. Model identification procedures for dynamical systems are available in [21].

To identify the diagonal damping parameters, one can use the steady-state version of (2.3) and perform linear regression to measured velocity and force. The steady-state model becomes

$$\mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2.9}$$

if we assume no disturbances and uncoupled motion, i.e., clean surge, sway, or yaw motion. If $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$ are measured, we must only fit the parameters of $\mathbf{D}(\boldsymbol{\nu})$, which are linear in the resulting equation.

Identifying coupling and inertia parameters is more complicated. Parameter fitting by optimization is a feasible approach. If $\tilde{\boldsymbol{y}}(\cdot)$ is a time-parametrized trajectory of measurements, and $\boldsymbol{y}(\cdot)$ is the trajectory resulting from the propagation of

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \tilde{\boldsymbol{u}}, \boldsymbol{p}) \tag{2.10a}$$
$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})\,, \tag{2.10b}$$

where

[4]The Marine Systems Simulator toolbox is developed by T. I. Fossen and T. Perez, and is available at `https://github.com/cybergalactic/MSS`.

- $\boldsymbol{x}$ are system states,
- $\tilde{\boldsymbol{u}}$ are the input commands used to get the measurements,
- and $\boldsymbol{f}$ and $\boldsymbol{h}$ are model and measurement functions,

we can find the model parameters $\boldsymbol{p}$ by solving

$$\min_{\boldsymbol{p}} \int_0^{t_f} L(\boldsymbol{y}(t), \tilde{\boldsymbol{y}}(t), \boldsymbol{p})\, \mathrm{d}t\,. \tag{2.11}$$

Here, L is a loss function designed to minimize the difference between simulated and measured data, $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$.

2.3 The optimal control problem

Much of the work in this thesis revolves around the optimal control problem (OCP). An OCP is an optimization problem that attempts to minimize an objective functional by finding an optimal system input trajectory, i.e., an optimal control trajectory, while maintaining dynamic and algebraic constraints. A general formulation of the OCP is

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),t_f} \int_0^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, \mathrm{d}t + E(\boldsymbol{x}(t_f), t_f) \tag{2.12a}$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \qquad \forall t \in [0, t_f] \tag{2.12b}$$

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq \boldsymbol{0} \qquad \forall t \in [0, t_f] \tag{2.12c}$$

$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \boldsymbol{0} \qquad \forall t \in [0, t_f] \tag{2.12d}$$

$$\boldsymbol{e}(\boldsymbol{x}(0), \boldsymbol{x}(t_f), t_f) = \boldsymbol{0}. \tag{2.12e}$$

Symbol definitions are found in Table 2.3.

The OCP can be designed to solve many practical problems, including path and trajectory planning, by designing the functions in (2.12) to suit the problem. However, this is a continuous OCP and is, in general, not possible to solve analytically. Transcribing the continuous OCP into a discrete problem and subsequently solving the resulting nonlinear program (NLP) is the most common way to find a solution to the OCP in practice. There are several ways to do this, including *single shooting* and *multiple shooting*.

In single shooting, the state trajectory is entirely defined by its initial conditions and the input trajectory, and thus, the variables that define the state trajectory are not a part of the solution space:

$$\min_{\boldsymbol{u}_{[0:N-1]}} \sum_{i=0}^{N-1} \hat{L}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) + E(\boldsymbol{x}_N, t_N) \tag{2.13a}$$

Table 2.3: Nomenclature for the OCP in (2.12).

Symbol	Description
$\boldsymbol{x}(\cdot)$	Time-parametrized trajectory of *states*.
$\boldsymbol{u}(\cdot)$	Time-parametrized trajectory of *inputs*.
t_f	End time.
$L(\boldsymbol{x}, \boldsymbol{u}, t)$	*Cost-to-go* function.
$E(\boldsymbol{x}, t)$	Cost of terminal state.
$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$	Right-hand side of ODEs representing dynamic model.
$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t)$	Function representing inequality constraints.
$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t)$	Function representing equality constraints.
$\boldsymbol{e}(\boldsymbol{x}(0), \boldsymbol{x}(t_f), t_f)$	Function representing initial and terminal constraints.

subject to

$$\boldsymbol{x}_{i+1} = \hat{\boldsymbol{f}}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \qquad i = 0, \dots, N-1 \tag{2.13b}$$
$$\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \leq \boldsymbol{0} \qquad i = 0, \dots, N-1 \tag{2.13c}$$
$$\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) = \boldsymbol{0} \qquad i = 0, \dots, N-1 \tag{2.13d}$$
$$\hat{\boldsymbol{e}}(\boldsymbol{x}_N, t_N) = \boldsymbol{0} \tag{2.13e}$$
$$t_i = h \cdot i. \tag{2.13f}$$

Here, functions denoted with a hat $\hat{(\cdot)}$ are discretized versions of the functions from (2.12), and h denotes the temporal discretization step length. In this example, the discrete states $\boldsymbol{x}_i, i = 1, \dots, N$ are propagated from the provided initial condition $\boldsymbol{x}_0$ with the discretized dynamic function $\hat{\boldsymbol{f}}$. The decision variables are, therefore, only the input series $\boldsymbol{u}_i, i = 0, \dots, N-1$. This can cause sensitivity issues when propagating, as the cost function and its gradient are highly sensitive to the inputs at the beginning of the time series. Detailed symbol descriptions are available in Table 2.4.

To address the sensitivity issues apparent in single shooting methods, it is possible to split the state and input trajectories into segments. A set of decision variables represents each segment, and equality constraints connect neighboring segments. This approach is called multiple shooting:

$$\min_{\boldsymbol{x}_{[0:N]}, \boldsymbol{u}_{[0:N]}} \sum_{i=0}^{N} \hat{L}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) + E(\boldsymbol{x}_N, t_N) \tag{2.14a}$$

Table 2.4: Nomenclature for the single shooting problem in (2.13).

Symbol	Description
$\boldsymbol{x}_i$	States at discrete points in time, for $i = 1, \ldots, N$. These are not part of the decision space, but values propagated from the provided initial state $\boldsymbol{x}_0$ by discrete integration.
$\boldsymbol{u}_i$	Inputs at discrete points in time for $i = 0, \ldots, N-1$. These are part of the decision space.
t_i	Points in time for $i = 0, \ldots, N$. Separated by h.
h	Discretization interval length.
$\hat{L}(\boldsymbol{x}, \boldsymbol{u}, t)$	Stage cost—discrete version of L in Table 2.3.
$E(\boldsymbol{x}, t)$	Cost of terminal state. Same as in Table 2.3.
$\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u}, t)$	Discrete integration function of ODEs.
$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t)$	Function representing inequality constraints.
$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t)$	Function representing equality constraints.
$\hat{\boldsymbol{e}}(\boldsymbol{x}_N, t_N)$	Discrete version of terminal constraints.

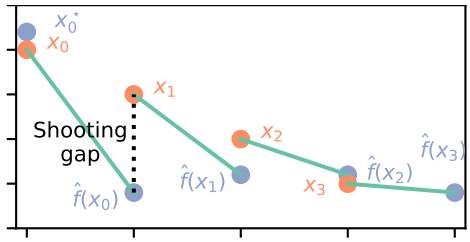


Figure 2.4: Multiple shooting of a scalar state trajectory. During multiple shooting optimization, the solver attempts to close the gaps between where a shooting interval ends $\hat{f}(x_i)$ and the beginning of the next interval x_{i+1}. These are called shooting gaps.

subject to

$$\boldsymbol{x}_0 = \boldsymbol{x}_0^\star \tag{2.14b}$$

$$\boldsymbol{x}_{i+1} = \hat{\boldsymbol{f}}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \qquad i = 0, \ldots, N-1 \tag{2.14c}$$

$$\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) \leq \boldsymbol{0} \qquad i = 0, \ldots, N \tag{2.14d}$$

$$\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i) = \boldsymbol{0} \qquad i = 0, \ldots, N \tag{2.14e}$$

$$\hat{\boldsymbol{e}}(\boldsymbol{x}_N, t_N) = \boldsymbol{0} \tag{2.14f}$$

$$t_i = h \cdot i. \tag{2.14g}$$

This problem is almost equivalent to (2.13). The critical difference is that each of the state trajectory points is included in the decision variables. Equation (2.14c) is, in this case, not a propagation of numerical values, but an equality constraint that attempts to close shooting gaps, i.e., a connection of each segment start $\boldsymbol{x}_{i+1}$ to the endpoint of the previous

Table 2.5: Nomenclature for the multiple shooting problem in (2.14).

Symbol	Description
$\boldsymbol{x}_i$	States at discrete points in time, for $i = 0, \ldots, N$. These are part of the decision space.
$\boldsymbol{u}_i$	Inputs at discrete points in time for $i = 0, \ldots, N$. These are part of the decision space.
t_i	Points in time for $i = 0, \ldots, N$. Separated by h.
h	Discretization interval length.
$\hat{L}(\boldsymbol{x}, \boldsymbol{u}, t)$	As in Table 2.4.
$E(\boldsymbol{x}, t)$	As in Table 2.4.
$\boldsymbol{x}_0^\star$	Numerical value of initial conditions.
$\hat{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u}, t)$	As in Table 2.4.
$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t)$	As in Table 2.4.
$\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, t)$	As in Table 2.4.
$\hat{\boldsymbol{e}}(\boldsymbol{x}_N, t_N)$	As in Table 2.4.

segment $\hat{\boldsymbol{f}}(\boldsymbol{x}_i, \boldsymbol{u}_i, t_i)$. Equation (2.14b) is a lifting of the initial condition $\boldsymbol{x}_0^\star$. The multiple shooting concept is illustrated in Figure 2.4, and symbol descriptions are found in Table 2.5.

The integration method in the regular multiple shooting concept (2.14c) can be simple, e.g., a forward-Euler method. Direct collocation methods for integration can be more efficient for the solver [22]. The direct collocation method involves representing the state and input intervals using, e.g., Lagrange polynomials and discretizing them in turn at Legendre collocation points. This representation introduces more variables to represent states and inputs per shooting interval, but the resulting NLP benefits from sparse, stable structures.

2.4 Path and trajectory planning

In robotics, the word "path" is used to describe a curve in the workspace or configuration space that connects a starting point with the desired endpoint, i.e., the goal. A path carries no information about time or velocity and can not say much about dynamic feasibility or energy consumption. On the other hand, a trajectory is a path with a velocity profile, i.e., a time-parametrized path. Numerous methods for path and trajectory planning are available in the robotics literature. LaValle [23] has written a general introduction to path planning in robotics from the perspective of computer science. In his work, he introduces widespread notation and nomenclature.

A useful categorization of planning methods is by the continuity of the

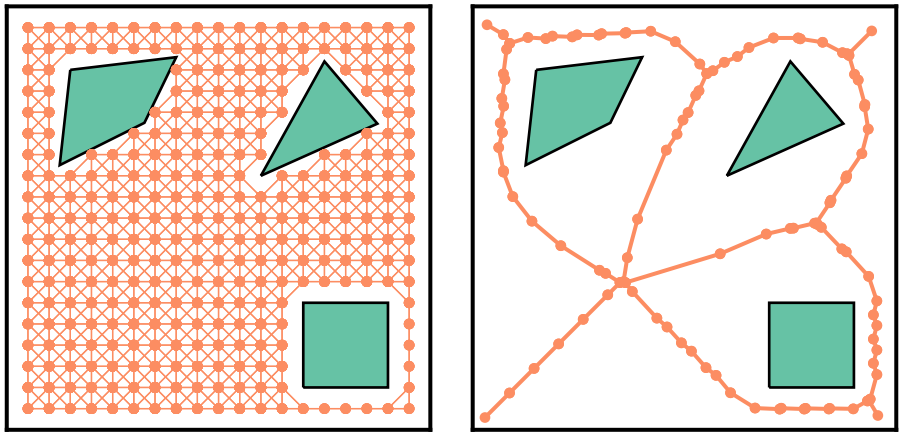


Figure 2.5: An illustration of two different discretization techniques. The map to the left is uniformly discretized, and each node is connected to its eight Moore-neighbors. The map to the right is discretized using a Voronoi diagram, and the set of available routes between the regions is sparse compared to the uniform discretization on the left.

resulting paths. *Discrete methods* explore points in the configuration space that lead to a piecewise linear path from start to goal when connected. A subsequent smoothing process is often necessary to ensure that a robot or vehicle can follow this path without large deviations. Discrete methods include *combinatorial methods* that accurately capture the continuous configuration space and subsequently search the resulting graph; *approximate methods* that discretize the configuration space in a practical manner and similarly search the resulting graph; and *sampling-based methods*, where points in the continuous configuration space are randomly sampled, with connections to their closest neighbors.

Some combinatorial path planning methods are covered in, e.g., [24, Chapter 6]. A* is a graph search algorithm commonly used in path planning to search discretized configuration spaces [25]. The configuration space can be uniformly discretized or more sparsely discretized using, e.g., a Voronoi diagram [26]. A Voronoi diagram is a partitioning of a space into regions where all points in each region are closer to the region's *generator point* than any other generator point. If generator points are placed along obstacle edges, the boundaries between the Voronoi regions can be used to build a roadmap for path planning. An illustration of the difference between uniform discretization and a Voronoi-diagram approach is illustrated in Figure 2.5. Implementations of Voronoi diagrams for path planning are found in, e.g., [27, 28].

A discretized configuration space may also be searched by *genetic algorithms* [29–31], which may be better if the configuration space's dimen-

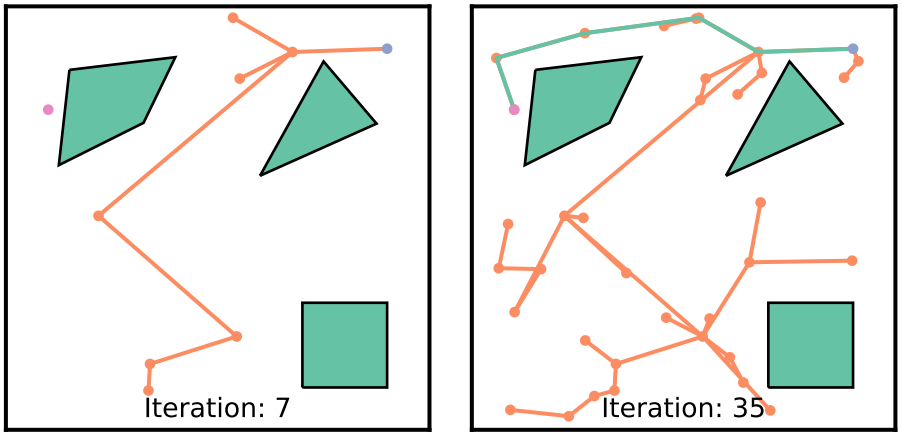


Figure 2.6: An illustration of an RRT for a simple, holonomic agent, at the beginning and end of its development. Each new node is randomly generated and connected to its closest neighbor in the tree. The blue node indicates the starting point, the pink node is the goal, and the green curve is the path found between start and goal.

sionality is high. Genetic algorithms in path planning is an optimization technique that produces and searches among candidate solutions encoded by sequences in the discretized configuration space. These algorithms use genetic operators to combine candidate solutions and produce mutations to evolve the solution space towards an optimized path.

Fast marching methods can also be classified as discrete since they operate on a discretized grid in the configuration space [32, 33]. The fast marching method was developed to solve the eikonal equation; a partial differential equation (PDE) encountered in wave-propagation problems. It can also be used to generate shortest-time paths in a map of obstacles by using the fast marching solution map for a gradient-descent search. It is advantageous compared to a uniform-grid graph search since the solution does not have a directional bias defined by the grid connectivity.

Sampling-based methods include the probabilistic roadmap (PRM) [34] and the rapidly-exploring random tree (RRT) [35]. These types of methods are known to be beneficial when searching in high-dimensional configuration spaces. PRM consists of the two phases *learning* and *query*. During the learning phase, nodes are placed randomly in the free configuration space with a given distribution. These are connected, and the corresponding edges in the resulting tree are feasible paths. During the query phase, the start and end configurations are connected to the tree, which is subsequently searched for an optimized path. Various PRM implementations are studied in [36].

RRT is a method that grows a tree in a configuration space by randomly

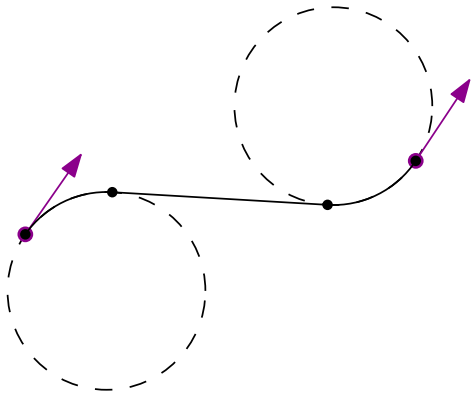


Figure 2.7: Example of a right-straight-left Dubins path.

adding new nodes and connecting them to their nearest existing neighbor. Collision checking is performed during sampling and connection, and the connection process may be quite involved, depending on the model used for path planning. Once the RRT can connect to the goal configuration, the process is terminated. An illustration of the development of an RRT for a simple two-dimensional holonomic agent is shown in Figure 2.6. An overview of RRT variations is available in [37].

Continuous methods work directly with the continuous configuration space and return paths that are at least continuous in its parametrization. These methods are either used to smooth the result of a discrete method or used directly to generate a path. Analytical methods can find solutions to a limited class of problems, e.g., the obstacle-free shortest path for a Dubins car [38] and a Reeds-Shepp car [39]. These are models of cars that travel by unit speed and constrained curvature on a plane. The Dubins path gives the shortest path for a Dubins car between two points in a plane and consists of straight segments and right and left circular arcs. An example is illustrated in Figure 2.7. The Reeds-Shepp car model is similar to the Dubins car but includes reversing.

Approximate optimization methods can be developed by specifying an OCP to formulate a path or trajectory planning problem and solve it by direct or indirect methods. Variations of these methods can take into account dynamical models, constraints, and obstacles. Arbitrary objective functions can be implemented to give, e.g., energy or time-optimal paths or trajectories. Two challenges with optimization methods are providing a good initial guess, i.e., *warm starting*, and representing obstacles accurately and efficiently. Gong et al. [40] present an approximate continuous planning method based on pseudospectral optimal control. They use simple obstacle representations such as rectangles and ellipses. Zhang et al. [41] present another approximate continuous optimization

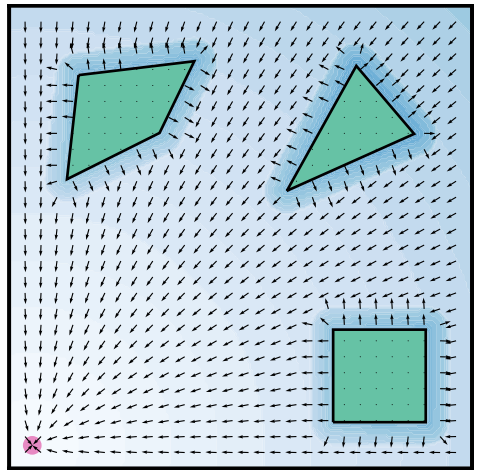


Figure 2.8: Illustration of an APF. The color contours show the potential, and the arrows show the gradient descent directions. The pink node indicates the goal.

method, where they are able to use exact polygonal constraints with the help of auxiliary optimization variables. Another approximate continuous optimization method is developed by Bergman et al. [42], where the authors produce sequences of convex areas to represent obstacles. In that paper, the authors also use a discrete search to warm-start the optimization method, which efficiently solves the warm-starting issue.

Another type of continuous method is the artificial potential field (APF), developed by Khatib [43]. A potential field is generated in the configuration space by adding an attractive potential to the goal and repulsive potentials to the obstacles. A path is then generated by iteratively following the gradient descent directions from the start. An illustration of an APF is shown in Figure 2.8. An issue with APF methods is the possibility of getting stuck in cyclic behavior caused by equilibria near obstacle boundaries [44].

Some methods fall outside the two categories. E.g., the hybrid A* method [45] performs a graph search in a discretized configuration space, however, with continuous motion primitives. Other examples include methods that combine graph searches and approximate optimization techniques, such as the one presented in [41], where the result of the hybrid A* algorithm warm-starts an optimization algorithm. Other examples are found in, e.g., [42, 46].

2.5 Collision avoidance at sea

The methods presented in Section 2.4 are all long-term, global planning methods. These can ensure collision avoidance in a known, static map of the world. Most of them can also be used to take into account moving

obstacles by adding a temporal dimension and including their projected positions. However, some approaches are developed explicitly with moving or initially unknown obstacles in mind. They are often local methods that plan on shorter-term horizons.

Fox et al. [47] introduce the dynamic window (DW) method for collision avoidance. The method avoids collision with obstacles by first generating a finite set of the agent's possible velocities. This set is then pruned for unsafe velocities, i.e., velocities that cause collisions with obstacles, and for velocities that are unreachable with admissible accelerations within the allotted time interval, i.e., the dynamic window. The velocities are then searched for the ones that generate optimal trajectories based on a blend of optimization criteria, e.g., path following, clearance to obstacles, and desired velocities. Since this is a finite set of velocities, searching among them is fast, and thus the method can be used to avoid collision by immediately reacting to new sensor information about obstacles. An application of the DW method to ASVs with moving obstacles is found in [48].

Velocity obstacles (VOs) is a collision-avoidance method introduced by Fiorini and Shiller [49] for robots in a non-static environment. The method calculates the sets of relative velocities between the agent and obstacles that avoid collisions when projected ahead in time. This set of non-colliding velocities is intersected with the set of the agent's reachable velocities, taking into account the agent's dynamics. A tree is built from successive applications of this operation and searched for an optimal maneuver based on some objective criterion.

Other collision avoidance methods for static and moving obstacles are, e.g., the branching-course MPC (BC-MPC) method [50, 51], which handles vessel dynamics and parts of the COLREGs, and the scenario-based MPC (SB-MPC) method [52, 53], which has similar properties.

The COLREGs is a set of maritime regulations, also called "rules of the road," published by the International Maritime Organization (IMO). Cockroft and Lameijer [5] provide a guide to the regulations. In addition to rules that regulate navigation lights, visibility, and other aspects of maritime operations, the part of the COLREGs that is most relevant to motion control is Part B, where rules 8 and 13 to 17 are of particular interest:

Rule 8: Action to avoid collision

> Any action taken to avoid collision should be positive and readily observable for other vessels. This rule implies that, e.g., large, abrupt course changes should be preferred to small, smooth changes.

Rule 13: Overtaking

> Overtaking is initiated when a vessel approaches another vessel from

behind and requires the overtaking vessel to stay clear of the overtaken vessel.

Rule 14: Head on
If two power-driven vessels are traveling towards each other with opposite or nearly-opposite courses, they are in a head-on situation, and both vessels should maneuver to starboard.

Rule 15: Crossing
When two vessels approach each other in other ways than described in rules 13 and 14, they are in a crossing situation. The vessel having the other vessel on its starboard side is deemed the give-way vessel. The other is the stand-on vessel.

Rule 16: Action by the give-way vessel
The give-way vessel must take early and substantial action to avoid collision, preferably by maneuvering towards starboard, to pass behind the stand-on vessel.

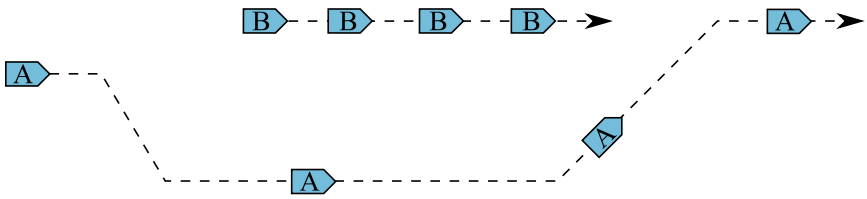
Rule 17: Action by the stand-on vessel
The stand-on vessel is required to keep its current speed and course. However, if the give-way vessel does not take appropriate action, the stand-on vessel must avoid the collision, preferably by maneuvering towards starboard, slowing down, or both.

Illustrations of overtaking, head-on, and crossing situations are provided in Figure 2.9.

Collision-avoidance methods must comply with these rules. Some of the methods mentioned so far are partly compliant; however, it is difficult for a single method to take all of the COLREGs into account. Much of the COLREGs is derived from practical seamanship and is hard to quantify in a computer algorithm. Combinations of long-term and short-term collision-avoidance methods may be a more appropriate paradigm for solving COLREGs-compliance. Hierarchical approaches, also called hybrid approaches, allow for flexibility when designing collision-avoidance systems [55–57]. Eriksen and Breivik [58] present a three-layered collision-avoidance architecture where the top-level layer deals with nominal path planning, the mid-level layer performs long-term, COLREGs-compliant trajectory planning, and the bottom-level layer performs short-term collision-avoidance that ensures safe maneuvering also in fast-paced, close-range encounters.

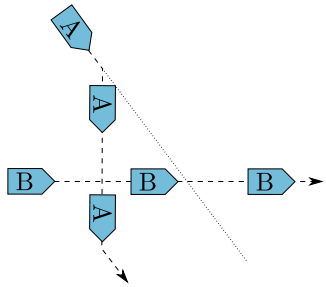
2.6 Automatic docking

Docking a vessel to a quay in a confined harbor area requires precisely planned and executed movements and careful consideration of how the



(a) Overtaking. Vessel A can overtake Vessel B on either side.

(b) Head on. Vessels A and B encounter each other head on, and both have to maneuver towards starboard before resuming to their course.



(c) Crossing. In this encounter, Vessel A's original path would cause a collision with Vessel B. Since Vessel A has Vessel B on its starboard side, Vessel A is the give-way vessel and should manever towards starboard and pass behind Vessel B. Vessel B is the stand-on vessel and should maintain its course.

Figure 2.9: The three main types of COLREGs situations, with maneuvers that correctly solve the encounters. © 2019, B. O.-H. Eriksen [54].

ship is affected by wind and hydrodynamic effects [59]. There is a limited amount of academic literature that presents practical, automatic docking for surface vessels. Most of the literature comes from the AUV domain, where the scenario is often to dock to an underwater docking station in the absence of obstacles. Rae and Smith [60] and Teo et al. [61] present fuzzy rule-based docking methods for AUVs. In these methods, obstacles are not handled explicitly or automatically, but manually constructed regions that define the AUV's guidance actions. Hong et al. [62] present a docking procedure for an AUV that combines optimization-based path generation, sensor information from an acoustic positioning system, and a camera that tracks the position of the docking station. This method also does not consider obstacles.

There has been some research on automatic docking methods for surface vessels, but no prominent examples that explicitly handle obstacles. Tran and Im [63] have developed a docking method for a surface vessel supported by a tug, which uses an artificial neural network (ANN) trained on data from manual docking operations. Other methods based on ANNs are

developed by Hasegawa and Fukutomi [64] and by Ahmed and Hasegawa [65]. Alternative methodologies include a method based on fuzzy logic by Nguyen and Im [66] and one based on APFs by Woo and Kim [67]. None of these handle obstacles explicitly or automatically. However, the APF-based method has a manually constructed potential field based on the specific harbor area the vessel operates in, which should result in collision-free maneuvers.

Optimal control-based methods may be more flexible in that it is possible to include obstacle constraints directly; however, most of the research on these types of methods do not do so. Djouani and Hamam [68] present an optimization-based trajectory planner for docking a surface vessel that takes into account vessel dynamics. A tracking controller tracks the trajectory, and the method is tested in simulations. A similar method is developed by Mizuno et al. [69]. These methods do not perform explicit obstacle handling. A promising optimization-based method is also presented by Li et al. [70], who develop a multi-objective nonlinear model predictive control (MPC) method for the docking of a surface vessel. The vessel's actuators are included in the MPC, and the inputs are used directly. This method also does not take into account obstacles explicitly but uses a reference path manually designed to avoid collision with the harbor and other vessels.

In optimization-based techniques, representing obstacles efficiently seems to be a challenge. Ideas from, e.g., [40–42] can be used to generate constraints that represent obstacles in optimization problems. Martinsen et al. [71] have developed an MPC-based docking method for a surface vessel that represents obstacles by automatically forming an obstacle-free, convex area around the vessel's position in a map of polygonal obstacles. The method also considers the vessel's dynamical model, thruster mapping, and limitations on the actuators' dynamics.

Of the methods mentioned so far, only [69] has been through experimental validation. Automatic docking is clearly a topic where the potential is high for practical methods that can be tested in full-scale experiments.

Commercial solutions for automatic from the maritime industry do exist. The technology companies Wärtsilä and Rolls-Royce Marine (acquired by Kongsberg Maritime in 2019) have both performed automatic docking with car ferries.[5] The details of their solutions are not publicly available, but both claim to have performed full crossing operations, including docking, automatically. Figure 2.10 depicts a docking operation by Bastø Fosen's ferry *Bastø VI* using technology from Kongsberg Maritime.

In addition to the industrial docking solutions, both Raymarine and

[5]See `https://www.maritime-executive.com/article/rolls-royce-and-wartsila-in-close-race-with-autonomous-ferries` (accessed September 14, 2020).

Figure 2.10: *Bastø VI* docking automatically in Horten, Norway in 2019 by using technology from Kongsberg Maritime.

Volvo Penta have developed solutions for leisure boats to assist with or automatically handle docking, which is often stressful for the boat operator.[6] Raymarine's solution uses infrared cameras to create a map of the harbor area, while Volvo Penta's solution uses an onshore reference system.

[6]See Raymarine's product website `https://www.raymarine.com/assisted-docking` and Volvo Penta's press release `https://www.volvopenta.com/about-us/news-media/press-releases/2018/jun/volvo-penta-unveils-pioneering-self-docking-yacht-technology` (accessed December 4, 2020).

Chapter 3

Contributions

In the Autonomous Ferry project presented in Chapter 1, the objective was to develop solutions for energy-efficient, electrical autonomous car ferries. A plethora of solutions must be in place; this includes ticketing, passenger registration, docking, automatic charger connections, navigation, target detection, identification and tracking, motion control, and motion planning. As it is impossible to work on all subjects in the course of one Ph.D. thesis, an early decision was made to work on motion planning and docking. During the research, it was found that significant work on auxiliary topics must be performed in order to perform experimental testing. These topics include model identification, development of a user interface for the experimental platform *milliAmpere*, and development of a software-in-the-loop simulator. Additionally, structural and safety-related improvements to the experimental platform were necessary. The contributions, including the auxiliary work, are summarized in this chapter.

3.1 Development and improvements on *milliAmpere*

As stated in Section 2.1, *milliAmpere* has been the platform used for experimental testing during the Ph.D. work. This platform, built around ROS, had its software, control systems, and navigation solutions developed by many master and Ph.D. students. To facilitate planning, control, and experiments, substantial work in this thesis has gone into model identification, development of a user interface, a software-in-the-loop simulator, and safety functions.

A method for optimization-based model identification for *milliAmpere* was developed in collaboration with Pedersen [16], who provided models for kinetics, thruster dynamics, thruster-to-force mapping, and wind effects. The propeller dynamics were modeled with linear models from the desired

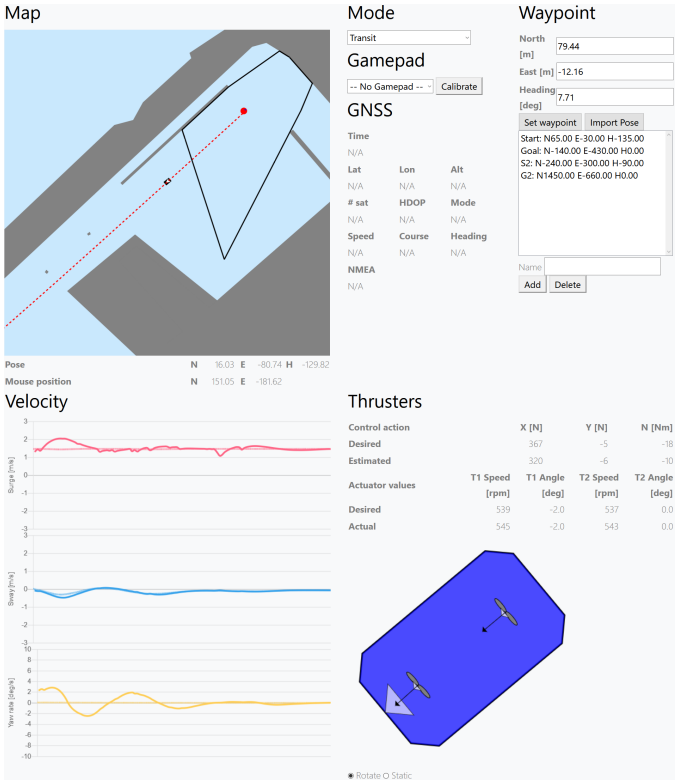


Figure 3.1: Screenshot of the user interface developed for *milliAmpere*. In this view, the user sees a map of the surroundings including a planned trajectory, and various input controls. Plots of velocities, and information and visualization of thruster information is also available.

rotational velocity to the actual velocity. The dynamics of the azimuth angles were modeled with saturated terms from the desired to the actual angle. The mapping from propeller rotational velocity was a polynomial fitted to data gathered by measuring force with a bollard pull experiment. For the kinetic model, an OCP was designed to fit the model described in Section 2.2 to data recorded during identification experiments. The OCP was transcribed using a multiple-shooting approach and solved using Casadi and Ipopt [72, 73]. These models have since been used in numerous master theses and research papers [12–14, 74, 75].

A web-based graphical user interface for *milliAmpere* was also developed in this thesis, in cooperation with Andreas B. Martinsen. The interface is shown in Figure 3.1. It provides access to *milliAmpere*'s position, heading, and velocity, a map of its surroundings, and the thruster states. It is also possible to set waypoints and change control modes. The graphical user interface has made debugging and monitoring easy to perform during experimental testing.

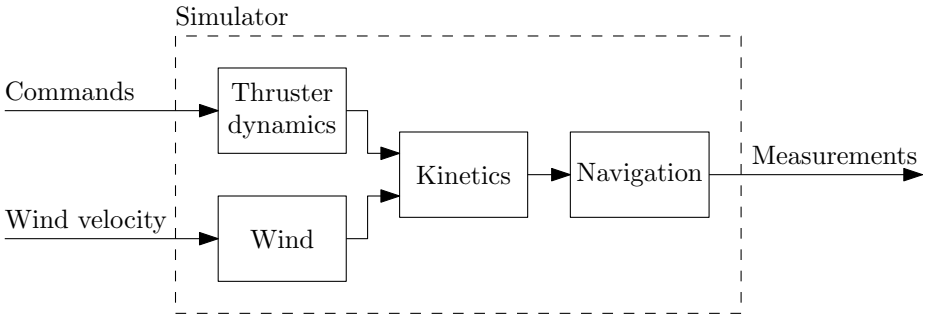


Figure 3.2: Block diagram of the developed software-in-the-loop simulator for *milliAmpere*. For the full-scale experimental tests, the vessel itself replaces the components in the simulator, with equivalent interfaces.

In order to facilitate rapid development on *milliAmpere*, a software-in-the-loop simulator was developed as part of this thesis. It is based on the models of [16]. As with the *milliAmpere* itself, the simulator was developed using ROS and provides the same interfaces as the physical vessel, illustrated in Figure 3.2. It simulates kinetics, wind, thruster dynamics, and navigation. The simulator makes it possible to use the same code for planning and control in both simulation and experimental tests and is in daily use by students and researchers involved with *milliAmpere*.

A fail-safe emergency stop feature was lacking on *milliAmpere*, which is critical when performing full-scale experiments. As part of this thesis, a fail-safe holding circuit for emergencies, which cuts power to the actuators, was designed and developed. The installation was done in cooperation with Alexander B. Holmen, who was an automation apprentice with the Department of Engineering Cybernetics at NTNU at the time.

3.2 Energy-optimized trajectory planning

Using pseudospectral optimal control

Work in optimization-based path and trajectory planning for ASVs is mostly limited to time and distance-optimized trajectories, with few attempts towards energy-optimization. Furthermore, there are few results from trajectory planning for ASVs with disturbances and closed-loop simulations. Pseudospectral optimal control has been used for motion planning on several classes of vehicles in [40], but without closed-loop simulations and with a generic quadratic cost functional. Practical minimum-time implementations for spacecraft are also presented in [76], and for an ASV in [77], but closed-loop minimum-energy implementations for ASVs are not available.

The development of a model-based energy-optimized planner for ASVs using pseudospectral optimal control is presented in Paper A. This was a continuation of the work in [78]. It is titled a *path planner* in the paper, but it is really a *trajectory planner* since it includes a velocity profile. Inspired by work in [77, 79], the pseudospectral planner was developed to include a full-state ASV model and an energy-optimization objective.

Pseudospectral optimal control is a computational method for solving OCPs [76]. It is similar to a direct collocation method in that it uses Lagrange polynomials to represent states and inputs, with Legendre collocation points [40]. In the method from Paper A, the software framework Dido for Matlab by Elissar Global was used.[1] Dido has been efficient at solving some OCPs, but can be hard to work with due to the inherent opacity of proprietary software, which makes the code difficult to debug.[2]

For a gradient-based OCP solver to be efficient, it is helpful that the constraint functions are continuously differentiable. In Paper A, obstacles were approximated by ellipses, which can be represented by smooth functions. The drawback is that complex maps, which are often polygonal, cannot be accurately represented by ellipses. Additionally, providing the solver with a feasible initial guess, i.e., a warm start, is not trivial, and without it, the solver is slow and will likely lead to undesired solutions due to the inherent nonconvexity of trajectory planning.

The method from Paper A took ocean currents into account by formulating the dynamical model in terms of relative velocity. For simulation results, the guidance system from [80] was implemented to track the optimal trajectory. Simulations showed that the generated trajectories were dynamically feasible and that the guidance and control system could track them well. Additionally, the use of accurate information about ocean currents significantly reduced energy usage. In practice, however, ocean current velocities are not uniform, and the effect of ocean currents on the vessel requires accurate measurements and models, which are hard to obtain.

Warm-starting techniques

As stated above, optimization-based planners benefit significantly from a feasible initial guess. If the planning occurs in a complex map, finding a good initial guess is completely necessary. A good initial guess has a geometry close to the global optimum and guides the OCP solver away from

[1] See `http://www.elissarglobal.com/academic/products/` for more information (accessed October 15, 2020).

[2] Open-source implementations of pseudospectral optimal control do exist. See, e.g., `https://github.com/danielrherber/basic-multiple-interval-pseudospectral` or `https://github.com/mpopt/mpopt` (accessed October 15, 2020).

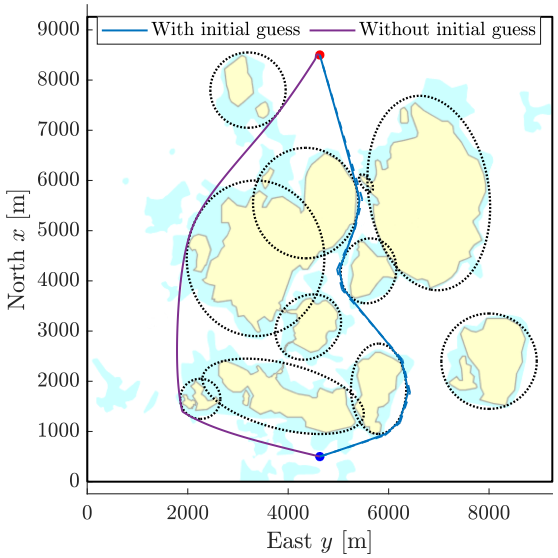


Figure 3.3: The map used for planning trajectories in Paper C. It shows that when the OCP solver is not provided with a feasible initial guess, it converges to an undesired trajectory.

undesired routings. In the literature, approaches to providing optimization-based trajectory planners with initial guesses found using discrete searches are found for vehicles [41, 46], but not for energy-optimized trajectory planning for ASVs. In Paper C, a practical, three-step process to generate such an initial guess was developed to solve an OCP for trajectory planning. Figure 3.3 shows how the OCP benefits from an initial guess. The three-step process can be summarized as:

1. Generate the shortest path on a uniformly discretized grid with the A* search algorithm.

2. Convert the shortest path into a trajectory by performing a waypoint reduction, connecting the waypoints with circle arcs, and adding artificial dynamic information.

3. Solve the energy-optimizing trajectory-planning OCP using the trajectory from Step 2 as an initial guess.

The OCP solver in Paper C is built using Casadi [72] with Ipopt [73] for Matlab, and the transcription is formed using multiple shooting with Runge-Kutta integration, rather than with a pseudospectral method. By exploiting the speed of an A* search, the results showed significant run-time improvements when using the initial guess, compared to cold-starting the

OCP solver. The optimality is also much improved since the global A* method guides the OCP solver to the shortest routing in the discrete search space.

Further improvements to this method were developed in [7], the unpublished paper discussed in Section 1.3. In that paper, the uniform discretization in Step 1 was replaced with one based on Voronoi diagrams. The waypoint reduction and trajectory generation algorithms from Step 2 were also improved. For the example presented in the paper, the Voronoi discretization resulted in the discrete search having 95 % fewer discrete nodes available to explore than the uniform discretization, which reduced the run time of Step 1 by 80 %. The improvements to the waypoint reduction and trajectory generation algorithms caused Step 2's run time to drop by 95 %. Step 3 was equivalent in the two approaches and accounted for most of the run time, so the overall improvement in run time was 35 %.

While the methods from Paper C and [7] are promising, they are still limited by using simple elliptic obstacle representations. For the method to be practical, that limitation needs to be addressed. In Paper F, a similar trajectory planning algorithm was developed, with two significant changes:

- The discrete search was replaced with a hybrid A* implementation that 1) inherently results in a time-parametrized, feasible trajectory, since it is propagated by motion primitives, and 2) supports arbitrary cost functions so that the optimality criterion for this search is equivalent to the optimality criterion in the OCP.

- Instead of representing obstacles as ellipses in the OCP, a sequence of convex permissible areas is generated along the hybrid A* search solution, wherein the OCP can freely move the positional states. This representation is smooth, which is beneficial for the gradient-based optimization solver.

- The transcription of the OCP was changed to use direct collocation with Legendre collocation points.

This development resulted in a flexible planner that can consider arbitrary maps, i.e., maps with polygonal obstacle descriptions. Experimental validation of the method was presented in Paper F. These experiments showed that *milliAmpere* was able to track the resulting trajectory and that proper distance was kept from obstacles. The method can be further improved by, e.g.,

- including surge velocity in the discrete search space, which would allow the hybrid A* search to look for variations in the speed profile which can improve energy efficiency;

- increasing computational efficiency by improving the software implementation;
- including other types of external disturbances, such as waves or ocean currents;
- and by including aspects of the COLREGs relevant to trajectory planning.

Pros and cons of model-based trajectory planning

So far the thesis has presented four model-based trajectory planning methods for ASVs. These types of methods are advantageous for three main reasons:

Dynamic feasibility The resulting trajectory will be dynamically feasible. Geometrical paths generated by discrete methods do not explicitly consider dynamical models, making them overly conservative or not dynamically feasible, meaning that they cannot be followed tightly by the physical vehicle. Model-based trajectory planning methods are constrained by dynamical models, which inherently produces such feasibility.

Energy optimization Energy consumption is tightly connected to vessel speed or actuator input, which requires a temporal dimension to characterize accurately. Using model-based optimization to generate trajectories gives a straight-forward way of including energy terms in the objective function.

Environmental disturbances Environmental effects, such as winds or ocean currents, can be included. Using dynamical models makes it possible to include effects caused by winds and ocean currents to consider both for energy consumption and dynamical feasibility.

Since this thesis focuses on energy-optimized planning and control, trajectory planning is preferred to path planning for these reasons. However, model-based methods also have some disadvantages that must be considered:

Model dependency A model-based method is dependent on a sufficiently accurate dynamical model. Such a model is not trivial to produce, and since the performance of model-based methods is dependent on the model's accuracy, using inaccurate models can lead to poor results.

Fading accuracy Including, e.g., environmental effects to make the trajectory more energy efficient will only be accurate over a limited temporal horizon. This limitation implies that the trajectory requires regular replanning with updated information to maintain its optimality.

Long running times Running times of optimization-based trajectory planning methods are significantly longer than discrete methods.

In conclusion, it is helpful to utilize model-based methods when performing energy-optimized planning, but it places significant requirements for the accuracy of the dynamical ship model and the data and measurements used for planning.

3.3 Hybrid collision-avoidance architecture

A path or trajectory planner is not the only necessary part of a collision-avoidance solution for ASVs. In addition to low-level control, navigation, and support functions, the motion control system must consider moving obstacles in compliance with the COLREGs. Several collision-avoidance approaches attempt to consider the COLREGs by using a single method to capture all relevant rules [52, 81, 82]. As an alternative, a hierarchical or hybrid approach is suggested in, e.g., [55–57]. In these works, the authors present two and three-layered collision-avoidance architectures, where the high-level layers perform nominal, long-term planning among static obstacles, and lower levels deal with short-term planning while including moving obstacles. Eriksen and Breivik [58] present a three-level hybrid architecture, where the high-level layer deals with nominal planning, the mid-level layer deals with protocol-based dynamic collision avoidance, making efforts to adhere to the COLREGs, and the low-level layer deals with short-term emergency handling. This architecture was further developed in Paper B and Paper D. A block diagram showing the architecture is available in Figure 3.4.

In Paper B, the main contribution was to combine a high-level planner, which was the pseudospectral trajectory planner from Paper A, with a mid-level dynamic collision-avoidance method, which was the MPC-based trajectory planner from [58]. The high-level planner planned a collision-free trajectory in a static map, taking into account ocean currents, which the mid-level method attempted to follow while taking into account detected moving obstacles, and adhering to Rule 8 of the COLREGs. Rule 8 requires that maneuvers are positive and readily observable for other vessels, which disallows slow course changes. The paper's simulations showed that this type of architecture enabled the ASV to avoid static and moving obstacles

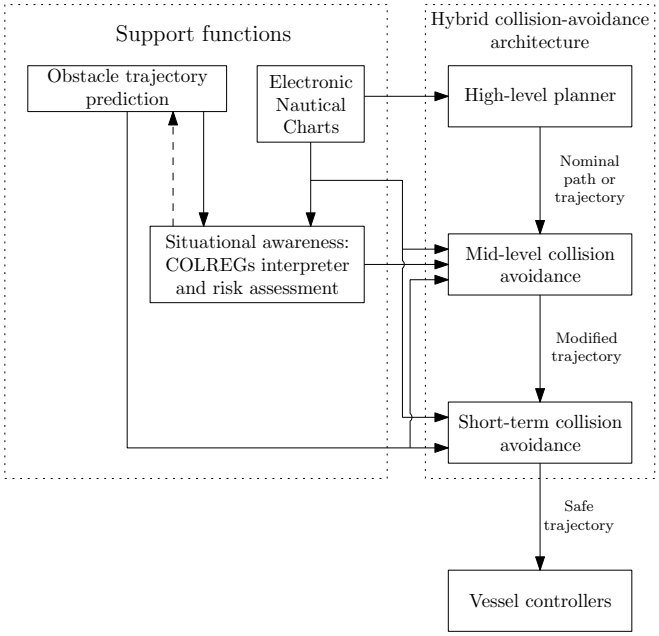


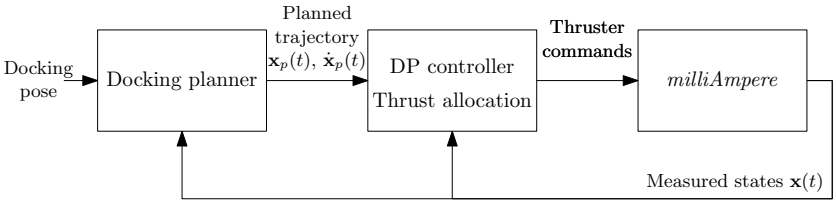
Figure 3.4: The hybrid, or hierarchical, collision-avoidance architecture from Paper B and Paper D.

in accordance with Rule 8 and otherwise tracked the energy-optimized trajectory. However, that work did not include the low-level layer, which deals with short-term emergency handling.

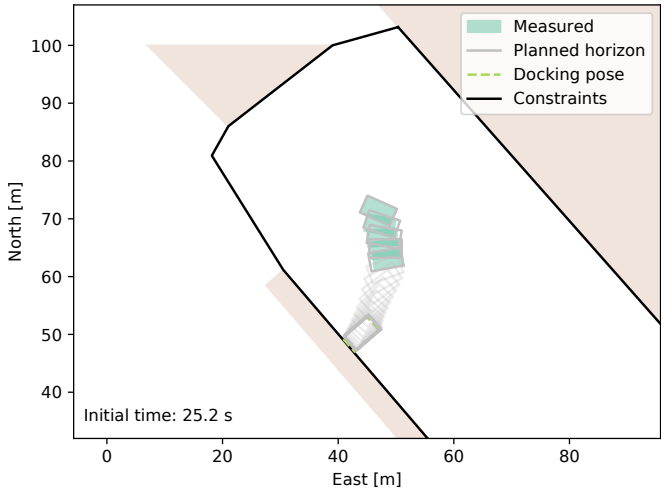
In Paper D, the three-layered hierarchy was completed by including the branching-course model-predictive controller from [50, 51] as the low-level layer. In addition to Rule 8, the architecture complies with the COLREGs rules 13 to 17. These rules describe how to act during overtaking, head-on, and crossing situations. They also describe what to do if another vessel does not comply with the maneuvering part of the COLREGs. This compliance was achieved by designing a classifier that determined the active COLREGs situation, with respect to each of the other detected vessels, and by designing cost functions in the mid-level MPC that elicit behavior following the COLREGs. Moreover, the pseudospectral planner used in Paper B and Paper A was replaced with the one developed in Paper C. Simulations showed that this hybrid collision-avoidance architecture could handle a wide range of situations while maintaining energy-efficiency.

3.4 Automatic docking

While the contributions presented so far have been focused on planning and collision avoidance during transit, this thesis also includes work on



(a) Block diagram of the docking control system setup.



(b) Planned and measured positions for *milliAmpere* during a docking experiment. The measured and planned poses are shown in two-second intervals along with the permissible convex set.

Figure 3.5: Block diagram and experiment plot from Paper E.

automatic docking for ASVs. As touched upon in Section 2.6, academic work on this topic lacks explicit handling of obstacles and experimental validation.

Martinsen et al. [71] developed an MPC-based method for automatic docking of ASVs. The method takes into account the vessel's dynamic model, thruster mapping, and limited thruster dynamics. It also takes into account obstacles by forming a permissible convex set that encapsulates the vessel and the docking target location, which allows the MPC to plan collision-free trajectories. This convex set excludes static obstacles that are defined by a polygonal map of the harbor area. The method has been tested in simulations.

As part of this thesis' work, the method from [71] has been further developed to turn it into a practical implementation that works in full-scale experiments with the *milliAmpere* autonomous ferry from Section 2.1. That work is presented in Paper E, which has the following key contributions:

- In the traditional sense, an MPC controller feeds its generated input

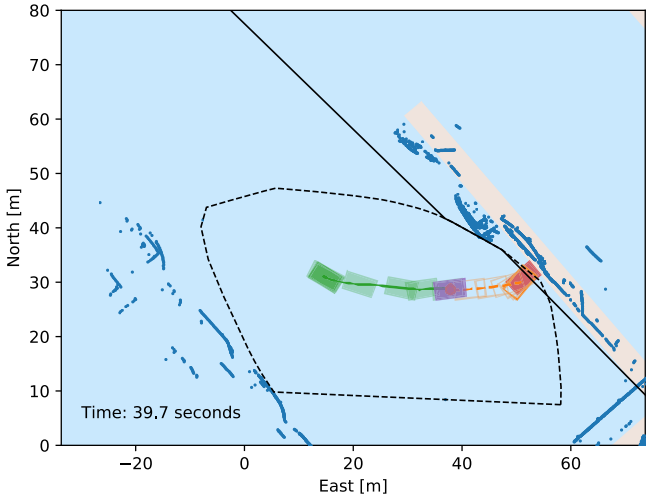


Figure 3.6: Plot of an experiment from Paper G. The plot shows measured and planned poses of the vessel during docking, a point cloud from the on-board lidar sensor, and the resulting permissible convex set.

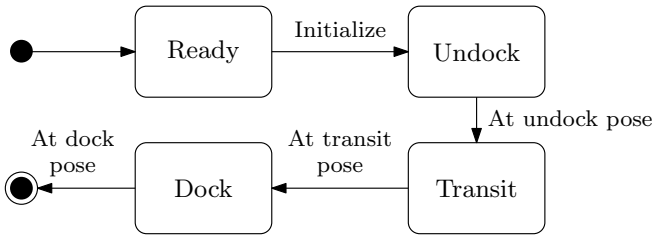
directly to the ASV's actuators. However, in this method, the state trajectory is sent to a trajectory-tracking DP controller. This separation is done to account for external disturbances and model errors, and the concept is illustrated in Figure 3.5a.

- The optimization problem is modified with slack variables to deal with feasibility issues when implementing the method in a real-world scenario.

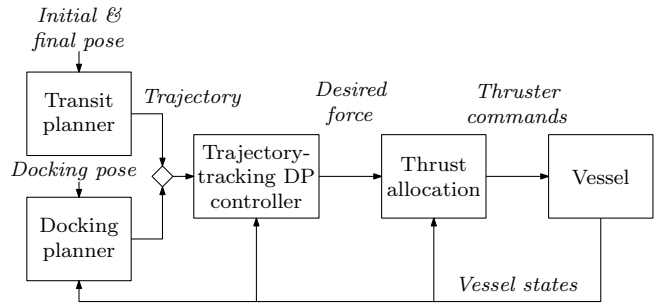
- The method is validated through full-scale experiments with *milliAmpere*.

Additionally, the method uses an algorithm that dynamically updates the convex set based on *milliAmpere*'s current location every 10 s, although it is not detailed in Paper E. That algorithm is presented subsequently in Paper G.

The automatic docking method was tested in experiments performed with the *milliAmpere*, in which the method produced collision-free docking maneuvers. A snapshot of one of the experiments is shown in Figure 3.5b, where measured and planned vessel poses are plotted inside the permissible convex set. The method is general and can interface with any vessel capable of tracking a time-parametrized trajectory. It requires a mathematical vessel model and a geographical map of the harbor area of sufficient resolution. The method may be extended with exteroceptive sensors such as cameras, lidars, radars, and ranging systems.



(a) UML-SD of simplified automatic crossing operation.

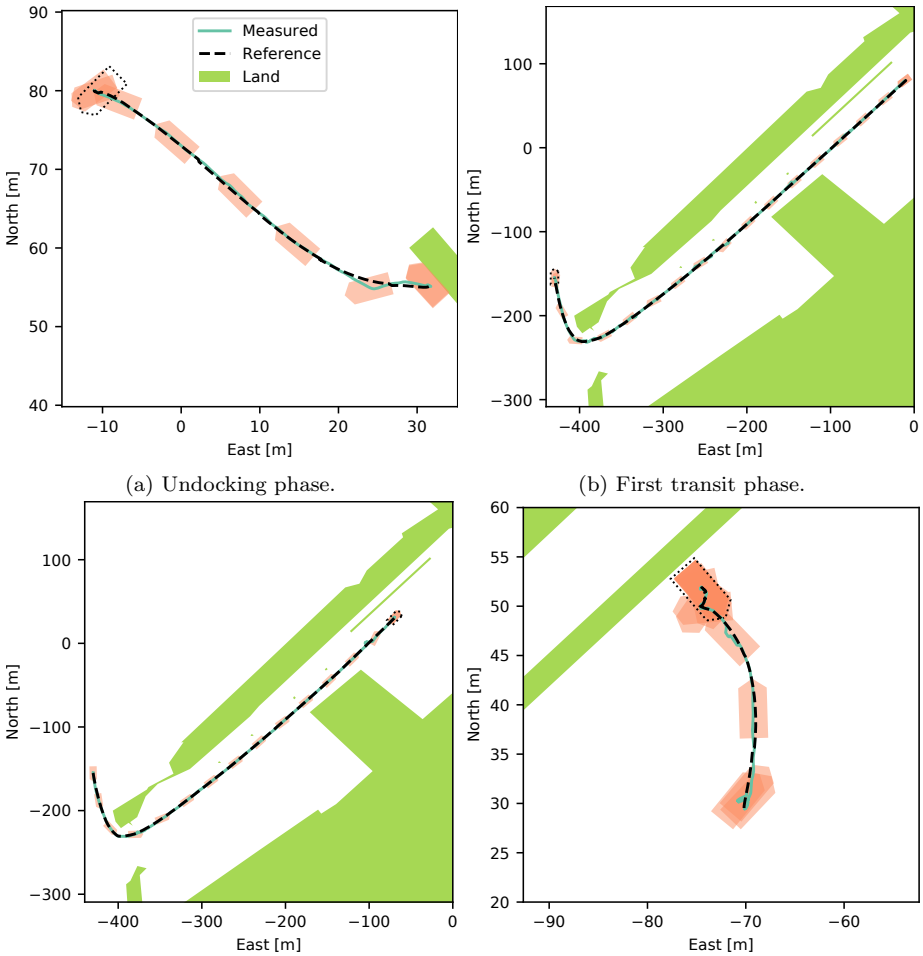


(b) Block diagram of control setup for automatic crossing.

Figure 3.7: UML-SD and control setup from Paper H.

In Paper G, the docking method is improved by including data from a lidar sensor and ultrasonic range sensors. These sensors provide points representing obstacles in the area around the vessel, which are used to further constrain the permissible convex set. The integration of exteroceptive sensors allows the method to consider data about the physical environment in real-time. Improvements are also made to the optimization problem's cost function and the interplay between the tracking controller and trajectory planner. The improved method was successfully tested in full-scale experiments, which showed that it generated collision-free docking maneuvers.[3] The method avoids collision both with static obstacles known a priori and with obstacles not included in the static map, observed with the exteroceptive sensors. A plot of an experiment is provided in Figure 3.6, including planned and measured poses, the point cloud generated by the lidar sensor, and the resulting permissible convex set.

These contributions are among the first academic literature results demonstrating experimental validation of collision-free automatic docking.



(a) Undocking phase.

(b) First transit phase.

(c) Second transit phase.

(d) Docking phase.

Figure 3.8: Maps and trajectories during the main phases of the experiment in Paper H. For practical reasons, this experiment had two transit phases.

3.5 Framework for automatic crossing

The prominent examples of automatic or autonomous crossing from one dock to another are commercial. As mentioned in Section 1.1, Wärtsilä and Rolls-Royce Marine performed autonomous crossings in 2019. There are few results for such implementations in the academic literature, however. Some long-term model-based trajectory planners may be suitable for the task, such as [42, 83]. These types of methods can plan a single trajectory that handles the undocking, transit, and docking phases as a single process. An approach that gives more flexibility and modularity is to handle the

[3] A video of one of the experiments is available at `https://youtu.be/AyaWlJvI6K8`.

phases with separate, fit-for-purpose planners and controllers.

The automatic docking method from Paper G was combined with the trajectory planner from Paper F to create a system for three-phase automatic crossing in Paper H. The three phases are undocking, transit, and docking. The methods are combined in an architecture designed as a Unified Modeling Language state diagram (UML-SD) [84], which switches between the methods depending on the active phase. In the undocking and docking phases, the automatic docking method is used, and in the transit phase, the trajectory planner is used. In all phases, the trajectories generated by the active method are fed to a trajectory-tracking DP controller. The UML-SD and control setup are illustrated in Figure 3.7. In Paper H, experiments with *milliAmpere* show that the architecture is a simple and effective way to achieve automatic crossing. Maps and trajectories for an experiment are shown in Figure 3.8.

While this method for automatic crossing does not include the COLREGs, the results are an important stepping stone towards fully autonomous crossing operations for ASVs. By integrating the transit control module with a hybrid collision-avoidance system, proposed in, e.g., Paper D, the method would comply with the COLREGs.

Using an UML-SD to design a sequential switching control architecture is advantageous since it is an open standard and several open-source libraries are available for implementation. This type of explicit programming to produce automatic behavior is useful in many scenarios, but has limitations when it comes to autonomous systems. For unsupervised operations, explicit programming might not be sufficient, since it is impossible to account for all types of unforeseen incidents. Other methodologies could allow for unsupervised operations, such as T-REX from the Monterey Bay Aquarium Research Institute [85], HAL from the Norwegian Defence Research Establishment [86], CARACaS from the US Office of Naval Research [87], GenoM from the Laboratory for Analysis and Architecture of Systems [88], and MOOS-IvP from the Massachusetts Institute of Technology [89]. These methodologies have advanced capabilities such as replanning and deliberative task-level planning, which could take us closer to unsupervised, autonomous operations.

Chapter 4

Conclusions and further work

To make maritime operations safer, more energy-efficient, and to provide low-cost commercial and social opportunities, substantial effort is being put into increasing the level of automatic control and autonomy at sea. This thesis represents an effort to develop and improve energy-efficient planning and motion control methods, which again enable autonomy. The contributions are towards several topics:

- The development of models and auxiliary functions for the experimental vessel *milliAmpere*, which are used in subsequent simulations and experiments, and by other researchers and students.
- The development and experimental validation of several model-based, energy-optimized trajectory planning methods that take into account external disturbances and complex static maps.
- The development of a hierarchical, hybrid collision-avoidance architecture that takes into account the International Regulations for Preventing Collisions at Sea (COLREGs).
- The development and experimental validation of methods for automatic docking.
- The combination of purpose-fit methods for the three phases of automatic crossing for autonomous surface vehicles (ASVs): undocking, transit, and docking.

It is easy to underestimate the amount of work required to perform experimental validation. Auxiliary topics, such as visualization and model identification, are necessary prerequisites. This realization has incited

much of the work presented in this thesis. The result is an ecosystem around *milliAmpere*, where the threshold for performing experiments has been lowered. This ecosystem has allowed for experimental validation of the methods developed in this thesis.

The model-based trajectory planning methods in this thesis have been developed incrementally. All of them consider static obstacles and are based on the optimal control problem with various solving processes, warm-starting techniques, and obstacle representations. The first method from Paper A is based on pseudospectral optimal control, has no warm-start technique, and represents obstacles with ellipses. The most recent method from Paper E is based on multiple shooting with direct collocation, is warm-started with a hybrid A* search, and represents obstacles with arbitrary polygonal shapes. This method has been experimentally validated to produce collision-free, dynamically feasible trajectories.

A global trajectory planning method that only considers static obstacles is insufficient to avoid collisions and travel safely in practical applications of ASVs—the ASV must also consider moving obstacles, adhere to the COLREGs, and perform safe maneuvers. The development of a hybrid or hierarchical collision-avoidance system has been presented in this thesis. In this system, the global trajectory planning method constitutes the top layer and gives a nominal trajectory that is tracked in the absence of moving obstacles. The middle layer tracks the nominal trajectory and takes into account moving obstacles as well. It plans COLREGs-compliant trajectories on a medium temporal horizon. The bottom layer tracks the COLREGs-compliant trajectories on a short temporal horizon and ensures that the ASV performs safe, collision-free, and dynamically feasible maneuvers. In Paper D, the collision-avoidance system is tested in simulations and is shown to produce collision-free, COLREGs-compliant maneuvers that track the nominal trajectory in the absence of moving obstacles.

The methods presented so far have been focused on the transit-phase of a crossing operation. Work on methods for automatic docking of ASVs is also presented in this thesis. The docking methods from Paper E and Paper G are based on periodic trajectory generation within collision-free convex sets. They consider a static polygonal obstacle map and real-time exteroceptive data using lidar and ultrasonic sensors. The methods are tested experimentally and produce collision-free trajectories that successfully dock the ASV.

With methods developed for trajectory planning for transit and automatic docking in place, the final contribution presented in this thesis, Paper H, is an architecture for combining the undocking, transit, and docking phases. The architecture combines the docking method from Paper G to handle the undocking and docking phases with the trajectory

planning method from Paper F to handle the transit phase. The methods are combined using an architecture based on Unified Modeling Language state diagrams (UML-SDs), and the architecture has been experimentally validated through successful full-scale experiments in the Trondheim harbor area.

Further work

While we are getting closer to autonomous maritime operations, there is still substantial room for further work on several related topics. Some of the topics within maritime autonomy that require more attention are:

Situational awareness This is a prerequisite to, e.g., collision avoidance and trajectory planning. Detection, classification, and tracking of other objects in an autonomous vessel's vicinity are necessary to avoid collision and adhere to the COLREGs. So is the inference of other vessels' intentions, which is regularly done by humans.

Natural language processing Vessels operating in the vicinity of other vessels are, in many cases, dependent on verbal radio communication. For an autonomous, unmanned vessel to operate alongside manned vessels in a busy area, transmission and comprehension of verbal information may be necessary.

COLREGs-compliant collision avoidance Very few available collision-avoidance methods can take into account all relevant parts of the COLREGs. Consideration of the COLREGs is necessary when autonomous vessels encounter other vessels at sea. Continued work on collision-avoidance methods that can interpret COLREGs encounters and handle them according to the regulations is necessary.

Robust, redundant, and real-time capable planning and collision-avoidance methods In addition to having collision-avoidance capabilities taking the COLREGs into account, methods that enable autonomous operations must be reliable in all situations. Most methods in the literature and in this thesis do not have these capabilities, which are essential to real-world implementations.

Autonomy engines While for simple operations, explicitly programming automatic behavior may be sufficient, it is not possible to account for all conceivable scenarios, which may be necessary for unmanned, autonomous vessels. The use of autonomy engines can enable dynamic replanning at the task level, which adds flexibility, and can handle a broader range of scenarios without explicit programming.

Thanks to entrepreneurs, researchers, and commercial institutions' significant efforts, we are well on our way to close the gap to autonomy at sea.

Chapter 5

Publications

This chapter contains reprints of the publications that are a part of this thesis.

Paper A Energy-optimized path planning for autonomous ferries

Published paper by **G. Bitar**, M. Breivik, and A. M. Lekkas. "Energy-optimized path planning for autonomous ferries". In: *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. Opatija, Croatia, 2018, pp. 389–394. DOI: 10.1016/j.ifacol.2018.09.456.

Bibliography entry [8].

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-29 (2018) 389–394

IFAC Papers Online CONFERENCE PAPER ARCHIVE

Energy-Optimized Path Planning for Autonomous Ferries

Glenn Bitar * **Morten Breivik** * **Anastasios M. Lekkas** *

* *Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway. E-mail: {glenn.bitar,anastasios.lekkas}@ntnu.no, morten.breivik@ieee.org*

Abstract: The topic of energy-optimized path planning using pseudospectral optimal control is considered. An optimal control problem (OCP) is formulated to produce an energy-optimized path between two points among static obstacles. A nonlinear 3-degree-of-freedom underactuated ship model is considered in the OCP with an energy-based cost function. The ship model is affected by external disturbances in the form of ocean currents. The OCP is solved using a pseudospectral method, which has proven successful in real-world applications. Additionally, the method is not as sensitive to dimensionality compared to Hamilton-Jacobi-Bellman methods. Position and speed information from the OCP solution is used in a closed-loop simulation with a guidance controller to show feasibility of the path. The simulation results show that the path is feasible, and that including ocean-current information in the path planning significantly reduces energy consumption. The paper also gives a short overview and classification of alternative path-planning methods. Finally, a proposal for how the path planner fits in a complete motion-control architecture is provided.

Keywords: Path planning, energy optimization, autonomous ships, optimal control

1. INTRODUCTION

Being a coastal country with a large prevalence of mountains and fjords, transportation has always been a challenge in Norway. Ferries are commonly used across large bodies of water, where construction of bridges is not cost effective or even possible. In 2017, Norway had around 150 ferry connections, where most of these are marked in Fig. 1. Approximately 9 % of the Norwegian domestic CO_2 emissions are caused by ships, where a significant portion is caused by passenger ships (DNV GL, 2014). As the focus on emission-free transportation increases, so does the need for energy-efficient solutions. Battery-powered ferries are low-emission solutions tested in Norwegian waters, and will be an important factor towards emission-free transportation, since the country's goal is to have a zero-emission maritime industry by 2050 (Criscione, 2017). However, battery-powered vessels are more limited in range than those powered by fossil fuel. This is due to challenges related to cost and weight of the energy storage (Kongsberg Maritime et al., 2017). Optimizing motion control may increase the operation envelope for battery solutions. Increasing the autonomy level of motion control systems will to a greater extent facilitate energy optimization. Rolls-Royce are among the industrial initiators for autonomy in ferries with their "Auto-Crossing" system, aimed to make energy consumption predictable (Rolls-Royce, 2016). Furthermore, the work presented in this paper is part of another industrial initiative for energy-optimized autonomous ferries, with project partners Kongsberg Maritime, Fjellstrand, Grenland Energy,

Fig. 1. Overview of Norwegian ferry connections.

Grønn Kontakt and NTNU in a Pilot-E funded initiative (Kongsberg Maritime et al., 2017).

This work presents a method for performing energy-optimized path planning using pseudospectral (PS) optimal control. A nonlinear, underactuated 3-degree-of-freedom (DOF) ship model affected by external disturbances in the form of ocean currents is used in both planning and simulation. The planner finds an optimized path using an energy-based cost function, which is then used by a guidance controller. The optimization is performed on

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.09.456

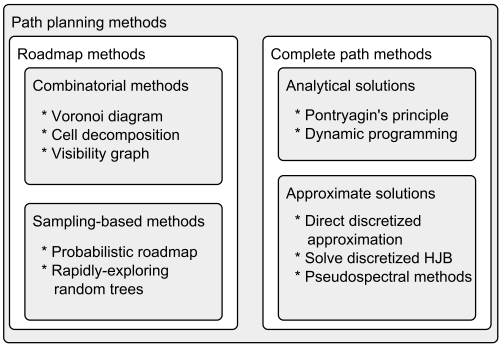


Fig. 2. Hierarchy of path-planning methods.

the three pose states, namely north and east position and heading, and on three velocity states, namely surge, sway and yaw rate. The work is a continuation of the master's thesis (Bitar, 2017), where several aspects of autonomy in ferries are considered, including path-planning methods, collision avoidance (COLAV), automatic docking and industrial control systems. Although the method presented in this work is designed for use with ferries, it can also be applied to other underactuated ships. The purpose of this paper is a show of concept, and not an extensive study of the method, therefore aspects such as failure handling, actuator limitations and state estimation are not considered.

Extensive research has been performed on path planning and motion planning in robotics literature. Useful references include (Wolek and Woolsey, 2017), where several model-based methods are reviewed. These methods include optimal control, level set methods, roadmap methods and others which are suitable for marine vehicles. Two distinct categories of path-planning methods are *roadmap methods* and *complete path methods*. Roadmap methods produce waypoints which form a feasible path if connected. Complete path methods generate a continuous parametrized path, often by optimization or optimal control. Figure 2 classifies some of the most important path-planning methods in a hierarchical manner.

Roadmap methods may be further divided by computational complexity. The two main subcategories are *combinatorial* and *sampling-based* methods. Combinatorial planning is also called exact planning, and considers the whole continuous space for roadmap vertices, which is inherently computationally expensive. Sampling-based methods are probabilistic and consider only an amount of sampled points in the continuous space, and thus have shorter running times for high-dimensional problems.

Optimization-based complete path methods are usually based on optimal control. A state-trajectory is found by using a solver, e.g. dynamic programming, which involves solving the Hamilton-Jacobi-Bellman (HJB) partial differential equation (PDE), or by using a pseudospectral algorithm, which is the solution presented in this paper. Other optimization-based methods include e.g. model predictive control (MPC).

Several researchers demonstrate the use of pseudospectral methods for motion planning and path planning. Lekkas et al. (2016) use pseudospectral optimal control to perform time-optimized path planning for a 3-DOF ship model. Re-planning at set intervals is also performed, such that updated estimates of environmental forces are taken into account. However, the authors use a reduced-order model, assume zero sideslip, while yaw-rate is treated as an input. Gong et al. (2009) perform motion planning on several kinds of vehicles using pseudospectral optimal control. These experiments involve complex environments and models, but no guidance simulations with disturbances are performed. Ross and Karpenko (2012) mention several successful demonstrations of the use of pseudospectral methods, among them a minimum-time rotational maneuver of a space telescope in orbit, and a zero-propellant maneuver of the International Space Station.

The main contribution of this paper is a method for performing model-based energy-optimized path planning using pseudospectral optimal control with an energy-based cost function. While pseudospectral optimal control has been used to generate motion trajectories in other references, the use of an energy-based cost function to create the path is novel. Path feasibility is shown by performing simulations with a curved-path guidance algorithm. The simulations indicate that energy can indeed be saved by performing path planning with up-to-date ocean current information.

The paper is organized as follows: Section 2 proposes a control system architecture for transit operations, including blocks for path planning, guidance, COLAV, low-level control and sensors; Section 3 treats ship modeling and control; Section 4 introduces the optimization-based path planning method; Section 5 discusses the planning and simulation results, while Section 6 concludes the paper.

2. MOTION CONTROL ARCHITECTURE

Motion control in ferry operations consist of undocking, a controlled maneuver away from the dock area, transit towards the destination dock, and docking. A proposed control system architecture for transit is illustrated in Fig. 3, where the highlighted block is the path planner which is the focus of this paper. Other important blocks are the mid-level and reactive COLAV subsystems, as defined by Eriksen and Breivik (2017).

COLAV is the task of avoiding collisions with static and moving obstacles. The task may be split into three levels: High-level global path planning, mid-level protocol-based COLAV, and low-level reactive COLAV. Low-level COLAV is responsible for avoiding immediate collision, and does not care about e.g. the International Regulations for Preventing Collisions at Sea (COLREG). Mid-level COLAV is intended to prevent collisions by following a set of rules, e.g. the COLREG. Methods at this level should also perform distinct maneuvers that communicate intended actions to onlookers and other vessels. High-level planning is responsible for creating paths that avoid known static obstacles such as land and reefs.

The low-level and mid-level methods rely upon local sensor information about dynamic and static obstacles. Target

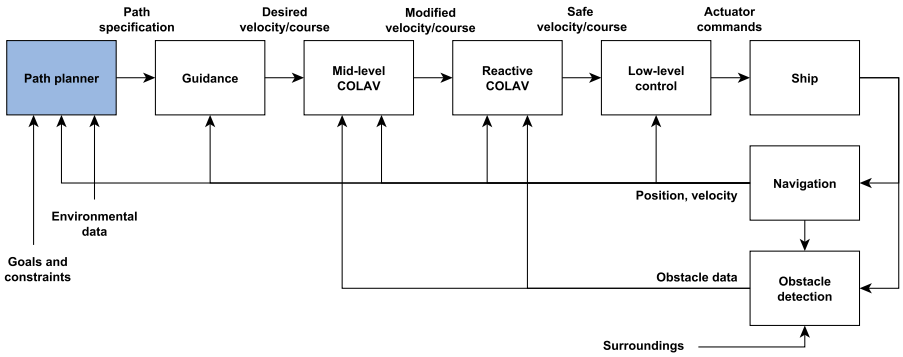


Fig. 3. Block diagram of motion control architecture during transit.

tracking and related sensory technology are key to retrieving this information. Wilthil et al. (2017) present a method for tracking targets with radar based on the probabilistic data association filter (PDAF). More references to target tracking methods are found therein. Eriksen and Breivik (2017) use nonlinear MPC to perform mid-level COLAV which is partly COLREG compliant. A cost function penalizing gentle turns and small speed changes is implemented to conform to the COLREG rule 8, which states that "action taken to avoid collision should be positive, obvious and made in good time." The high-level planner relies on static information about the area, such as a map. Various techniques may be implemented to provide a two-dimensional feasible path for the ship to follow using a guidance system. One such technique is explored in Section 4.

3. SHIP MODELING AND CONTROL

3.1 Modeling

We use an underactuated 3-DOF nonlinear ship model for path planning and simulation. The modeling techniques and notation are retrieved from (Fossen, 2011). The ship is called Cybership II, and is a 1:70 scale replica of a supply ship, with a length of $L = 1.255\,\text{m}$ and mass $m = 23.8\,\text{kg}$. Information about this model ship and its physical parameters is found in (Skjetne et al., 2005). To scale velocities and other variables up to its full-scale equivalent, one may use the bis scaling system, described in e.g. (Fossen, 2011). The scaling factor for linear velocities for this ship is $\sqrt{\frac{L_f}{L}}$, where L_f is the length of the full-scale ship, resulting in a factor of $\sqrt{70}$. The model is written on the following form, with the time dependency omitted for notational brevity:

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{1a}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}}_r + \mathbf{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau}. \tag{1b}$$

Here $\boldsymbol{\eta} = [x, y, \psi]^\top$ is the ship's pose, where x and y are the north and east position, respectively, and ψ is the ship's heading. The rotation matrix $\mathbf{R}(\psi)$ is defined as

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

The ship's velocity over ground in BODY coordinates $\{b\}$ is $\boldsymbol{\nu} = [u, v, r]^\top$, where u is surge speed, v is sway speed, and r is the yaw rate. The ship's velocity relative to water is used to model currents: $\boldsymbol{\nu}_r = [u_r, v_r, r]^\top = \boldsymbol{\nu} - \boldsymbol{\nu}_c$. We have denoted the current velocity in BODY as $\boldsymbol{\nu}_c = [u_c, v_c, 0]^\top$. The current velocity in NED coordinates $\{n\}$ is $\boldsymbol{\nu}_c^n = [V_x, V_y, 0]^\top$, and the relationship between those is $\boldsymbol{\nu}_c = \mathbf{R}^\top(\psi)\boldsymbol{\nu}_c^n$.

In (1), $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the symmetric, positive definite system inertia matrix. We denote the Coriolis and centripetal matrix $\mathbf{C}(\boldsymbol{\nu}_r) \in \mathbb{R}^{3\times3}$, and the damping matrix $\mathbf{D}(\boldsymbol{\nu}_r) \in \mathbb{R}^{3\times3}$. The vector $\boldsymbol{\tau} = [X, Y, N]^\top$ contains the control forces. We use an actuator model with two input signals $\boldsymbol{u} = [F, \delta]^\top$, where F is the force produced by a rear-mounted azimuth thruster, and δ is its angle:

$$\boldsymbol{\tau}(\boldsymbol{u}) = \left[F\cos\delta,\ 0,\ -l_{th}F\sin\delta\right]^\top, \tag{3}$$

where l_{th} is the length from the BODY origin to the azimuth thruster. The second element $\tau_2 = Y$ is set to zero, because it is possible to do a coordinate transformation on any ship model to achieve this property (Fredriksen and Pettersen, 2004).

3.2 Guidance controller

The guidance controller used in simulation is a curved-path line-of-sight (LOS) controller developed by Breivik and Fossen (2004). The key geometric variables are illustrated in Fig. 4. The particle $\boldsymbol{p}_d(\theta)$ is the origin of the path-parallel frame $\{p\}$. This particle exists on the path $\mathcal{P} = \{\boldsymbol{p} \in \mathbb{R}^2 | \boldsymbol{p} = \boldsymbol{p}_d(\theta)\,\forall\,\theta \in [0, \theta_{\max}]\}$. The path variable θ is interpreted as some distance along the path, and is the time variable from the path planner, as discussed in Section 4. The along-track error $s(t)$ is the tangential distance from the path particle, whereas $e(t)$ is the cross-track error. The relationship between the ship's position $\boldsymbol{p}(t) = [x(t), y(t)]^\top$ and the error variables is

$$\boldsymbol{\varepsilon}(t) = \left[s(t),\ e(t)\right]^\top = \mathbf{R}_t^\top(\chi_t(\theta))(\boldsymbol{p}(t) - \boldsymbol{p}_d(\theta)), \tag{4}$$

where $\mathbf{R}_t \in \mathbb{R}^{2\times2}$ is equivalent to the upper-left block in (2).

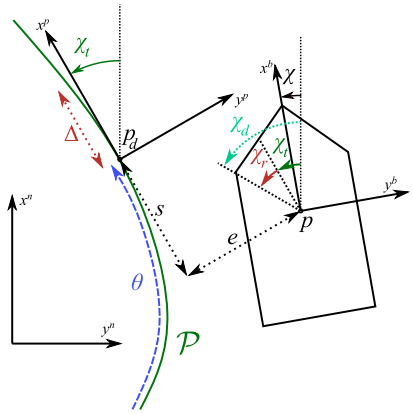


Fig. 4. Geometric variables used in guidance.

The path-tangential angle is $\chi_t(\theta)$, and the ship's course is $\chi(t)$. A control law is applied to drive $e(t)$ asymptotically to 0: $\chi_d(t) = \chi_t(\theta) + \chi_r(t)$, where $\chi_d(t)$ is the desired course angle, and $\chi_r(t) = \arctan\left(-\frac{e(t)}{\Delta}\right)$. The tuning parameter $\Delta > 0$ may be interpreted as the ship's lookahead distance. Finally, dynamics are added to the path parameter to drive it forwards:

$$\dot{\theta} = \frac{U(t)\cos(\chi(t) - \chi_t(\theta)) + \gamma s(t)}{\sqrt{x_d'(\theta)^2 + y_d'(\theta)^2}}, \tag{5}$$

where $U(t) = \sqrt{u(t)^2 + v(t)^2}$ and $\gamma > 0$ is a tuning parameter deciding the particle's convergence speed to the ship's position. Prime notation for derivatives is used: $x_d'(\theta) = \frac{\mathrm{d}x_d}{\mathrm{d}\theta}(\theta)$.

3.3 Surge and heading controller

The low-level controller is also retrieved from Breivik and Fossen (2004). A short summary of the control law follows:

$$\boldsymbol{\tau} = \mathbf{M}\dot{\boldsymbol{\alpha}} + \mathbf{N}(\boldsymbol{\nu})\boldsymbol{\alpha} - \mathbf{R}(\psi)^\top \hat{\boldsymbol{b}} - \boldsymbol{h} z_1 - \mathbf{K}_2 \boldsymbol{z}_2, \tag{6}$$

where

$$\mathbf{N}(\boldsymbol{\nu}) = \mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu}) \qquad \boldsymbol{h} = [0,0,1]^\top$$
$$\boldsymbol{\alpha} = [u_d, \alpha_2, -k_1 z_1 + r_d]^\top \qquad z_1 = \psi - \psi_d$$
$$\dot{\hat{\boldsymbol{b}}} = \boldsymbol{\Gamma}\mathbf{R}(\psi)\boldsymbol{z}_2 \qquad \boldsymbol{z}_2 = \boldsymbol{\nu} - \boldsymbol{\alpha}\,.$$

The variable α_2 is determined by a differential equation generated by the dynamic constraint $\tau_2 = Y = 0$. The constant parameter $k_1 > 0$ and matrix $\mathbf{K}_2 = \mathrm{diag}\{k_{21}, k_{22}, k_{23}\} > 0$ are tuning parameters of the controller. The constant matrix $\boldsymbol{\Gamma} > 0$ tunes the convergence rate of the estimator $\hat{\boldsymbol{b}}$, which estimates disturbances in the NED frame.

The reference signal for surge speed is $u_{\mathrm{ref}}(\theta(t))$, which is the planned surge speed at the path parameter $\theta(t)$ mentioned in Section 4. A reference model is used to generate smooth derivatives:

$$(u_d/u_{\mathrm{ref}})(s) = \omega_{n,u}^2/(s^2 + 2\zeta_u\omega_{n,u}s + \omega_{n,u}^2), \tag{8}$$

which is converted to a state-space representation, with saturation on the acceleration $\dot{u}_d(t)$.

The reference signal for heading is retrieved from the guidance controller: $\psi_{\mathrm{ref}} = \chi_d(t) - \beta(t)$, where $\beta(t)$ is the ship's sideslip angle $\beta(t) = \arcsin\frac{v(t)}{U(t)}$. The reference model in yaw is defined by the transfer function

$$\frac{\psi_d}{\psi_{\mathrm{ref}}}(s) = \frac{\omega_{n,\psi}^3}{(s + \omega_{n,\psi})(s^2 + 2\zeta_\psi\omega_{n,\psi}s + \omega_{n,\psi}^2)}, \tag{9}$$

which is converted to a state-space representation, with saturation on angular acceleration and velocity $\dot{r}_d(t)$ and $r_d(t)$.

4. PATH PLANNING BY PSEUDOSPECTRAL OPTIMAL CONTROL

PS methods may be used for solving the HJB equation, but for optimal control, a different approach is commonly used. The PS method discussed in this paper instead approximates the states and inputs as Lagrange polynomials, which are subsequently discretized. A discrete representation of the dynamics is applied as constraints, in addition to the equality and inequality constraints. The new optimization problem is then solved using a spectral algorithm (Gong et al., 2009).

Optimization-based complete path methods are based on the OCP:

Minimize

$$J(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), t_f) = E(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F(\boldsymbol{x}(t), \boldsymbol{u}(t), t_f)\,\mathrm{d}t \tag{10a}$$

subject to

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{10b}$$
$$\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0 \tag{10c}$$
$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = 0 \tag{10d}$$
$$\boldsymbol{e}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) = 0\,, \tag{10e}$$

where $\boldsymbol{x}(t)$ is the full state vector of the dynamic system, and $\boldsymbol{u}(t)$ is the control vector. The sense of optimality is defined in the cost function (10a), and the dynamics of the vehicle are included in the OCP as (10b). Path constraints (static obstacles), velocity and control constraints may be represented as (10c). Equality constraints may be included as (10d), and start and end conditions may be represented as (10e).

The cost function used in path planning is energy-based, and is defined as the work done by the actuators on the water:

$$J(\boldsymbol{x}, \boldsymbol{u}, t_f) = \int_{t_0}^{t_f} \left(|Xu_r| + |Yv_r| + |Nr|\right)\mathrm{d}t\,. \tag{11}$$

Since this energy function contains the non-smooth absolute value function, an approximation of this function is used for numerical efficiency: $|(\cdot)| = \sqrt{(\cdot)^2 + c}$, where $c > 0$ is a small constant, in our case $c = 0.001$.

The map where we perform path planning is defined by *path constraints*. For this we use elliptic shapes for virtual obstacles. Obstacle i is represented as

$$c_i(\boldsymbol{\eta}) = ((x - x_i)/x_{s,i})^2 + ((y - y_i)/y_{s,i})^2 \geq 1\,. \tag{12}$$

The center of the ellipse is x_i and y_i, and the latitudinal and longitudinal radii are $x_{s,i}$ and $y_{s,i}$. To avoid large numbers which may lead to numerical issues when the ship is far away from the obstacle, we take the logarithm of both sides of the inequality (12). The planning is in

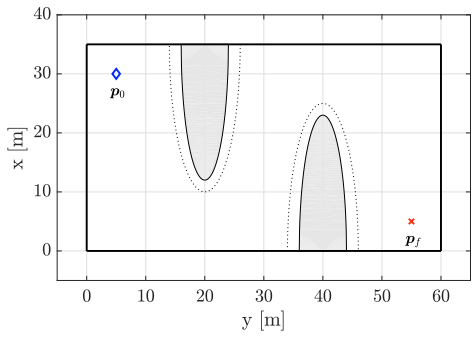


Fig. 5. Map used in path planning. The map shows virtual obstacles as dotted lines enclosing the actual obstacle with a safety margin.

this paper performed with two elliptic obstacles, with the parameters $x_1 = 35\,\text{m}$, $y_1 = 20\,\text{m}$, $x_{s,1} = 25\,\text{m}$, $y_{s,1} = 6\,\text{m}$; and $x_2 = 0\,\text{m}$, $y_2 = 40\,\text{m}$, $x_{s,2} = 25\,\text{m}$, $y_{s,2} = 6\,\text{m}$. These virtual obstacles should enclose actual obstacles in the map with a safety margin.

In addition to the elliptic obstacles, the search space is limited to the box $x \in [0, 35]\,\text{m}$, $y \in [0, 60]\,\text{m}$. This gives the map depicted in Fig. 5. The figure also shows the ship's start and goal locations as a blue diamond and red cross, respectively. Their positions are: $\boldsymbol{p}_0 = [30, 5]^\top\,\text{m}$ and $\boldsymbol{p}_f = [5, 55]^\top\,\text{m}$. The actuator values are limited to $F_{\min} \leq F(t) \leq F_{\max}$ where $F_{\min} = 0$ and $F_{\max} = 8\,\text{N}$, and $-\delta_{\max} \leq \delta(t) \leq \delta_{\max}$ where $\delta_{\max} = 30°$. The start time is $t_0 = 0$, and the final time t_f is limited to $0 \leq t_f \leq 150\,\text{s}$.

To generate a path with this method, the ship's pose $\boldsymbol{\eta}(t)$ and speed $\boldsymbol{\nu}(t)$ from (1) are changed to desired states, $\boldsymbol{\eta}_d(\theta)$ and $\boldsymbol{\nu}_d(\theta)$, respectively. Additionally, the time variable t in the OCP is changed to the path parameter θ. All related information retrieved from the planning will have the subscript $(\cdot)_d$, except for the planned course, which will be called the tangential course angle and have the notation $\chi_t(\theta)$, and the reference surge speed, which will have the notation $u_{\text{ref}}(\theta)$. The planning information used in simulation is the desired position $\boldsymbol{p}_d(\theta) = [x_d(\theta), y_d(\theta)]^\top$ and its derivative, the path-tangential angle $\chi_t(\theta)$ and the desired surge speed $u_{\text{ref}}(\theta)$.

The resulting OCP is solved using a software package for MATLAB: DIDO for PS optimal control by Elissar Global, on a computer with an Intel Core i7-7700HQ processor.

5. PLANNING AND SIMULATION RESULTS

Two different scenarios are explored in the results:

Scenario 1: (S1) Path planned using no current information.

Scenario 2: (S2) Path planned using correct current information.

After planning, both scenarios are simulated using a south-to-north current: $V_x = 0.1\,\text{m}\,\text{s}^{-1}$ and $V_y = 0$, equivalent to $0.84\,\text{m}\,\text{s}^{-1}$ for the full-scale ship. The ocean current

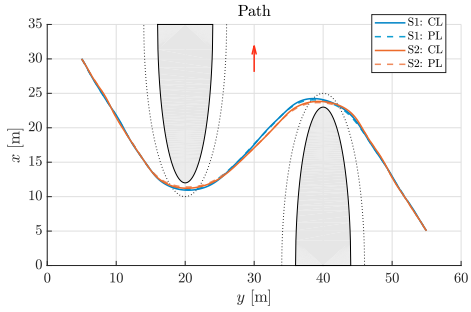


Fig. 6. Planned (PL) and closed-loop simulated (CL) paths from the two scenarios. The red arrow shows the resultant current direction.

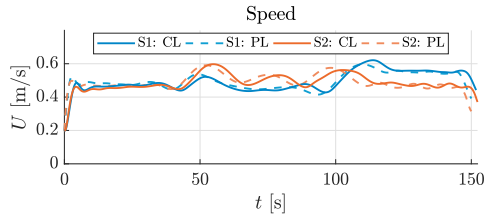


Fig. 7. Differences in speed between S1 and S2.

magnitude is a significant perturbation to the ship, and the direction is selected to be perpendicular to the main direction of travel. This is to demonstrate that using ocean current information in planning reduces the energy spent in transit. Closed-loop simulations are performed with the controllers detailed in Section 3. The results from planning are labeled PL, while the closed-loop simulation results are labeled CL.

The scenarios take place in the map shown in Fig. 5. The initial velocity is set to $\boldsymbol{\nu}_0 = [0.2, 0, 0]^\top\,\text{m}\,\text{s}^{-1}$ ($1.7\,\text{m}\,\text{s}^{-1}$ for the full-scale ship). The initial and final headings are free, but otherwise, the start and end conditions are set as mentioned in Section 4. The OCP solutions of S1 and S2 were found in 25 s and 21 s, respectively.

Figure 6 shows both the planned and simulated paths of S1 and S2. The planned paths are quite similar, however, especially when the ship maneuvers along the current direction, the path differs. This is also evident from Fig. 7, which shows significant speed differences between the scenarios after 50 s. These differences lead to significant changes in energy consumption between the two scenarios. An 8 % reduction of consumed energy is seen in Table 1, with the same time to completion t_f.

Table 1. Scenario results.

		Scenario 1		Scenario 2	
PL	J	124	J	153	J
CL	J	164	J	**151**	**J**
PL	t_f	150	s	150	s
CL	t_f	152	s	152	s

Figure 8 shows performance metrics for the two scenarios: On top is shown the cross-track error $e(t)$ from (4), while

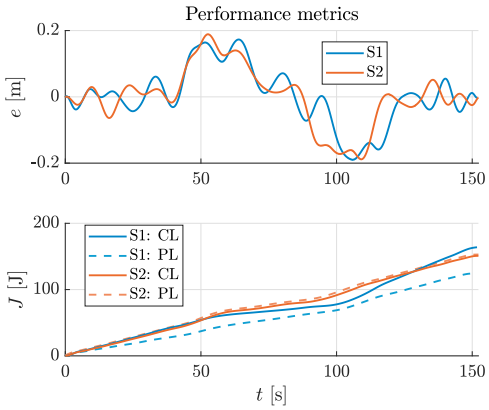


Fig. 8. Performance for both scenarios, both planned and simulated.

the accumulated energy consumption $J(t)$ from (11) is shown below. The cross-track error stays within 0.2 m for both scenarios, which is satisfactory and corresponds to 1.7 m for a full-scale ship. The simulated energy consumption in S1 surpasses the planned consumption, however, since this scenario is planned with no current information, the discrepancy is to be expected. In S2, the simulated energy consumption stays close to the planned consumption, and is significantly lower than in S1. An explanation to why the simulated consumption is slightly lower than planned may be that the integration method used in the PS algorithm differs from the simulation.

A notable observation from the resulting paths seen in Fig. 6 is that the planned (and the simulated) paths cross the obstacle boundaries. This is common with the PS method because the method only enforces constraints at the collocation points, of which there are only 30 in this case. One solution to avoid this is to increase the number of nodes at the expense of computational time, while another is to create the obstacle boundary with a safety margin outside the actual obstacle. The latter is preferable, because these are areas that are not desirable to enter.

6. CONCLUSIONS AND FUTURE WORK

A method for finding an energy-optimized path using pseudospectral optimal control has been proposed. Through closed-loop simulation with a guidance controller, this method has been verified to produce feasible and energy-efficient paths for a given underactuated 3-DOF ship model. Additionally, using up-to-date ocean current information helps to reduce energy spent. The method is suitable to use in combination with arbitrary curved-path guidance algorithms and low-level controllers. Several COLAV methods are also suitable to use with the path-planning method. Moreover, an overview of several path-planning methods is provided.

Exploring how the results from the PS path-planning method compares to other traditional roadmap planners is suggested for further work. Integrating the method with a complete COLAV system is also desirable, both in simulation and in real-world testing. Combining a COLAV system with regular re-planning might be necessary to retain optimality when the ship strays off course.

ACKNOWLEDGEMENTS

This work is funded by the Research Council of Norway and Innovation Norway with project number 269116. The work is also supported by the Centers of Excellence funding scheme with project number 223254.

REFERENCES

Bitar, G.I. (2017). *Towards the Development of Autonomous Ferries*. Master's thesis, NTNU.

Breivik, M. and Fossen, T.I. (2004). Path following for marine surface vessels. In *Proc. of the IEEE/MTS Oceans '04 Techno-Oceans Conf., Kobe, Japan*.

Criscione, V. (2017). Norway's greener future fleet. *Norway Exports*. URL `http://www.norwayexports.no/`.

DNV GL (2014). Sammenstilling av grunnlagsdata om dagens skipstrafikk og drivstofforbruk. Technical Report 2014-1667, Klima- og miljødepartementet.

Eriksen, B.O.H. and Breivik, M. (2017). MPC-based mid-level collision avoidance for ASVs using nonlinear programming. In *Proc. of the IEEE '17 CCTA, Mauna Lani, HI, USA*.

Fossen, T.I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley-Blackwell.

Fredriksen, E. and Pettersen, K.Y. (2004). Global κ-exponential way-point manoeuvering of ships. In *Proc. of the 43rd IEEE CDC, Nassau, Bahamas*.

Gong, Q., Lewis, R., and Ross, I.M. (2009). Pseudospectral motion planning for autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 32(3), 1039–1045.

Kongsberg Maritime, Grenland Energy, NTNU, Fjellstrand, and Grønn Kontakt (2017). Lessons learned — best available technology. Technical Report 269116/E20, Pilot-E.

Lekkas, A.M., Roald, A.L., and Breivik, M. (2016). Online path planning for surface vehicles exposed to unknown ocean currents using pseudospectral optimal control. In *Proc. of the 10th IFAC CAMS, Trondheim, Norway*.

Rolls-Royce (2016). Rolls-Royce to supply first automatic crossing system to Norwegian ferry company Fjord1. *Rolls-Royce*. URL `https://www.rolls-royce.com/media/press-releases/yr-2016/18-10-2016-rr-to-supply-first-automatic-crossing-system-to-norwegian-ferry-company-fjord1.aspx`.

Ross, I.M. and Karpenko, M. (2012). A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2), 182–197.

Skjetne, R., Fossen, T.I., and Kokotović, P.V. (2005). Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory. *Automatica*, 41(2), 289–298.

Wilthil, E.F., Flåten, A.L., and Brekke, E.F. (2017). A target tracking system for ASV collision avoidance based on the PDAF. In *Sensing and Control for Autonomous Vehicles*. Springer International Publishing.

Wolek, A. and Woolsey, C.A. (2017). Model-based path planning. In *Sensing and Control for Autonomous Vehicles*. Springer International Publishing.

Paper B Energy-optimized hybrid collision avoidance for ASVs

Published paper by **G. Bitar**, B.-O. H. Eriksen, A. M. Lekkas, and M. Breivik. "Energy-optimized hybrid collision avoidance for ASVs". In: *Proceedings of the 18th European Control Conference (ECC)*. Naples, Italy, 2019, pp. 2522–2529. DOI: 10.23919/ECC.2019.8795645.

Bibliography entry [9].

2019 18th European Control Conference (ECC)
Napoli, Italy, June 25-28, 2019

Energy-Optimized Hybrid Collision Avoidance for ASVs

Glenn Bitar, Bjørn-Olav H. Eriksen, Anastasios M. Lekkas and Morten Breivik

***Abstract*— This paper considers the development of a hybrid planning and collision avoidance architecture for autonomous surface vehicles (ASVs). The proposed architecture combines a high-level optimization-based planning algorithm with a mid-level collision avoidance (COLAV) algorithm based on model-predictive control (MPC). The high-level planner produces an energy-optimized trajectory by solving an optimal control problem via a pseudospectral method, taking into account known static obstacles and ocean currents. The mid-level algorithm performs MPC by solving a nonlinear program (NLP) to produce a collision-free local trajectory, also taking into account dynamic obstacles. In particular, the NLP optimizes for a combination of following the energy-optimized trajectory with performing readily observable maneuvers, as defined by Rule 8 of the International Regulations for Preventing Collisions at Sea (COLREGs). Numerical simulations are used to verify that the hybrid architecture produces safe, efficient and readily observable trajectories.**

I. Introduction

Autonomous ships are currently being explored for transportation and marine operations. The Yara Birkeland project in Norway [1] is an example of this, where the aim is to replace 40 thousand truck journeys of fertilizer per year with an autonomous cargo ship. Specifically, autonomous surface vehicles (ASVs) can provide reduced costs, risks and environmental impact, increased operational windows and introduce new opportunities in ocean operations. In excess of $75\,\%$ of maritime accidents are caused by human errors, which emphasizes the potential for increased safety by autonomous technology [2]. Approximately $9\,\%$ of Norwegian domestic CO_2 emissions originate from ships, which signifies the potential impact of energy-optimization in ship technology [3].

A fundamental requirement for ASVs is a robust and safe collision avoidance (COLAV) system. The ASV must be able to avoid static obstacles, such as land and shallow waters, as well avoid dynamic obstacles, including other ships. In addition, the ASV must perform its maneuvers in compliance with the International Regulations for Preventing Collisions at Sea (COLREGs) [4], which are often referred to as the "rules of the road" for marine vessels.

There exist numerous COLAV algorithms for marine applications. They may be divided into terms of the temporal planning horizon, e.g. long-term and short-term algorithms [5]. Long-term COLAV algorithms include offline planners as well as algorithms intended to run online, with a large spatial and temporal horizon. Bitar *et al.* present a long-term path and trajectory planning algorithm based on pseudospectral optimal control [6], inspired by findings in e.g. [7]. The algorithm produces energy-optimized trajectories and takes into account environmental factors, such as ocean current and static obstacles. Eriksen and Breivik have developed a long-term COLAV algorithm which takes into account Rule 8 of COLREGs, which requires that maneuvers are readily observable for other vessels [8]. The algorithm is based on model predictive control (MPC) and avoids static and dynamic obstacles by planning local trajectories via a nonlinear program (NLP). The NLP objective function includes terms to penalize slow changes in course and speed, hence making the resulting maneuvers observable. Other MPC-based COLAV algorithms include [9]–[11]. Other long-term algorithms include a *Voronoi diagram*-based method [12], *cell decomposition* [13], *rapidly exploring random trees (RRTs)* [14], with a COLREGs-compliant implementation in [15].

Short-term COLAV algorithms take into account a limited spatial and temporal horizon, and plan maneuvers to avoid immediate collision. Eriksen *et al.* have developed

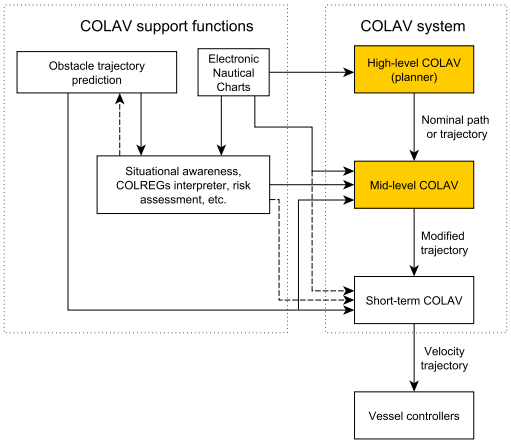


Fig. 1. Suggested hybrid COLAV architecture with three layers. A high-level planner is combined with a mid-level COLAV algorithm to form parts of a hybrid architecture for an ASV. A complete hybrid architecture would also include short-term COLAV for immediate obstacle avoidance. The hybrid COLAV architecture should be supported by data from electronic nautical charts, represented in a suitable manner for the algorithms, as well as situational awareness functions that track and predict obstacles, perform risk assessment, etc.

The authors are with the Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway. Email: {glenn.bitar, anastasios.lekkas}@ntnu.no, {bjorn-olav.h.eriksen, morten.breivik}@ieee.org

Fig. 2. The *Telemetron* ASV. Courtesy of Maritime Robotics.

the *Branching-Course MPC Algorithm*, which produces collision-free, dynamically feasible local trajectories, aligned with an input desired trajectory, and is partially COLREGs compliant [5]. Other examples of short-term COLAV algorithms include *dynamic window* [16], and *velocity obstacles* [17].

Loe [18] and Casalino *et al.* [19] propose a hybrid COLAV architecture where the COLAV task is split into two or three layers to exploit the complementary strengths of long-term and short-term algorithms. We base our hybrid architecture on [8], with the structure illustrated in Fig. 1, where the COLAV system is divided into three layers. The top-level layer is the path or trajectory planner, which is intended to run offline prior to a mission, with global information about static obstacles. To be able to optimize with respect to energy, it is necessary to use a trajectory rather than a path, since energy consumption is tightly coupled with speed and acceleration. The mid-level layer is responsible for adhering to the parts of COLREGs that are related to maneuvering, and takes into account information about local dynamic and static obstacles. The short-term level is intended to execute dynamically feasible maneuvers and avoid immediate collision situations. This makes the hybrid COLAV architecture comply with Rule 17 of COLREGs, stating that the stand-on vessel (which has right of way) should take action if the give-way vessel does not, by having the ability to ignore the COLREGs considerations from the mid-level.

The authors have worked extensively on several parts of the hybrid COLAV architecture in Fig. 1. Examples include long-term COLAV algorithms intended to run offline as high-level planners [6], [12], or as mid-level algorithms [8], and short-term algorithms [5], [16]. The short-term algorithms have been tested in several full-scale experiments with radar-based obstacle detection and tracking. The COLAV algorithms are dependent on the performance of the vessel controllers. Therefore, the authors have also put significant effort into modeling, identification and control of ASVs [20], [21]. In this paper, we develop parts of the hybrid architecture shown in Fig. 1, by connecting and further developing two existing algorithms [6], [8]. Several improvements have been made to both algorithms:

- The high-level planner [6] has been extended to work with a non-first-principles model.
- The mid-level algorithm [8] is extended with relative velocities as a way to include ocean currents,
- has improved numerical properties, and
- has been extended with an interface for *relative* trajectory tracking, enabling the ASV to track a desired trajectory with a time-varying time offset.

Both algorithms utilize a mathematical model of the ASV *Telemetron*, depicted in Fig. 2.

The rest of the paper is organized as follows: The updated mathematical model of the ASV is introduced in Section II. The high-level planner and the mid-level algorithm are presented in Section III and Section IV, respectively. A description of the scenario used to test the approach is presented in Section V, along with a discussion of our results. Section VI concludes the paper.

II. ASV MODELING

In [20], Eriksen and Breivik developed a nonlinear dynamic model structure for high-speed ASVs operating in the displacement, semi-displacement and planing regions. They identified parameters for the *Telemetron* ASV, which is the vessel considered in this paper. The model has the form

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\chi)\boldsymbol{x} \tag{1a}$$

$$\dot{\chi} = r + \dot{\beta} \tag{1b}$$

$$\mathbf{M}(\boldsymbol{x})\dot{\boldsymbol{x}} + \boldsymbol{\sigma}(\boldsymbol{x}) = \boldsymbol{\tau}\,. \tag{1c}$$

Here, the pose vector $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times S$ contains the ASV's position and heading angle in the Earth-fixed North-East-Down (NED) frame $\{n\}$. The vector $\boldsymbol{x} = [U, r]^\top \in \mathbb{X} \subset \mathbb{R}^2$ contains the ASV's speed over ground (SOG) U and the rate of turn (ROT) r. The matrix $\mathbf{R}(\chi)$ transforms the SOG and ROT to kinematic velocities:

$$\mathbf{R}(\chi) = \begin{bmatrix} \cos\chi & 0 \\ \sin\chi & 0 \\ 0 & 1 \end{bmatrix}. \tag{2}$$

The course rate $\dot{\chi}$ is determined by the ROT r and the sideslip rate $\dot{\beta}$. The matrix $\mathbf{M}(\boldsymbol{x}) \in \mathbb{R}^{2\times 2}$ represents velocity-dependent inertia, and $\boldsymbol{\sigma}(\boldsymbol{x}) \in \mathbb{R}^2$ is the damping effect. The ASV is controlled by the control vector $\boldsymbol{\tau} = [\tau_m, \tau_\delta]^\top \in \mathbb{U} \subset \mathbb{R}^2$, which contains normalized values for throttle and rudder, respectively.

The states and controls are restricted to the sets $\mathbb{X}$ and $\mathbb{U}$ which are where the model (1) is valid. Detailed information about the valid sets are found in [20].

A. Extension to relative velocities

The model for *Telemetron* (1) does not take into account ocean currents, nor does it separate surge and sway motion. In this paper, we change (1c) to include relative velocities in order to account for ocean currents.

The following assumptions are made to include ocean current disturbances.

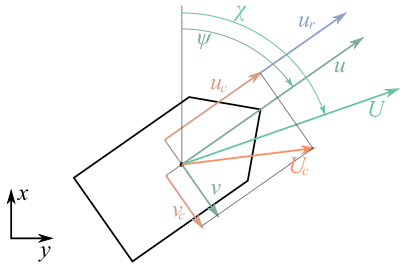


Fig. 3. An illustration of Assumption 2 with ocean currents. The ship's absolute surge velocity u is larger than the ocean current velocity in the surge direction u_c, giving a positive relative surge velocity $u_r = u - u_c$. However, in the sway direction, there is no model of the ship's movement. Thus we assume that the ship's relative sway velocity v_r is zero, implying that the sway velocity v is equal to the ocean current velocity in the sway direction v_c. The figure also illustrates the heading and course angles, ψ and χ.

Assumption 1. *The model for SOG U from* (1c) *is valid for relative surge velocity u_r.*

Assumption 2. *The relative sway velocity v_r is identically equal to zero.*

The reasoning for Assumption 1 is that the majority of the damping effects during model identification experiments were due to relative velocity. Assumption 2 may seem counterintuitive, but since we don't know anything about the sway dynamics of the vessel, we assume that the ship's sway motion follows the ocean currents, implying that the sway velocity v is equal to the sway-component of the ocean current v_c. Fig. 3 illustrates this idea.

Under these assumptions we change the model (1c) from a SOG and ROT model to a relative-velocity model without loss of generality for the rest of this paper. In fact, any ordinary ship model may be used for the hybrid architecture. The new model has the form

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{x}_r + \begin{bmatrix}\boldsymbol{V}_c^\top & 0\end{bmatrix}^\top \tag{3a}$$

$$\mathbf{M}(\boldsymbol{x}_r)\dot{\boldsymbol{x}}_r + \boldsymbol{\sigma}(\boldsymbol{x}_r) = \boldsymbol{\tau}\,, \tag{3b}$$

where $\boldsymbol{x}_r = [u_r, r]^\top \in \mathbb{X}_r \subset \mathbb{R}^2$ and $\boldsymbol{V}_c = [V_x, V_y]^\top \in \mathbb{R}^2$ describes the ocean current velocity in NED. The set $\mathbb{X}_r$ is equivalent to $\mathbb{X}$, based on the assumptions made for the ocean current extension. The rest of the symbols are the same as in (1). The relationship between the absolute surge u and sway v velocities and the relative surge u_r and sway v_r velocities is described by

$$\begin{bmatrix}u\\v\end{bmatrix} = \begin{bmatrix}u_r + u_c\\v_r + v_c\end{bmatrix} = \begin{bmatrix}u_r + u_c\\v_c\end{bmatrix}\,, \tag{4}$$

where u_c and v_c are the surge and sway components of the ocean current velocity.

The ocean current velocity is assumed to be constant and known in the NED frame. The body-fixed frame $\{b\}$ has origin and orientation fixed in the ASV, which results in the following ocean-current representation:

$$\begin{bmatrix}u_c\\v_c\end{bmatrix} = \begin{bmatrix}\cos\psi & \sin\psi\\-\sin\psi & \cos\psi\end{bmatrix}\boldsymbol{V}_c\,. \tag{5}$$

III. High-level planner

The high-level planner solves an optimal control problem (OCP) by using a pseudospectral method to find an energy-optimized trajectory, as described in [6]. The OCP is given by

$$\min_{\boldsymbol{z}_{r,d},\boldsymbol{\tau}_d,t_f} \int_0^{t_f} F(\boldsymbol{z}_{r,d}(t),\boldsymbol{\tau}_d(t),t)\,\mathrm{d}t \tag{6a}$$

subject to

$$\dot{\boldsymbol{z}}_{r,d}(t) = \boldsymbol{f}(\boldsymbol{z}_{r,d}(t),\boldsymbol{\tau}_d(t))\ \forall t \in [0,t_f] \tag{6b}$$

$$\boldsymbol{h}_p(\boldsymbol{z}_{r,d}(t),\boldsymbol{\tau}_d(t),t) \le \mathbf{0}\ \forall t \in [0,t_f] \tag{6c}$$

$$\boldsymbol{e}_p(\boldsymbol{z}_{r,d}(0),\boldsymbol{z}_{r,d}(t_f),t_f) = \mathbf{0}\,. \tag{6d}$$

The symbols used in (6) will be further explained in Section III-A. We subscript the states and inputs with $(\cdot)_d$ to emphasize that these are desired values.

A. Cost functional, dynamics, inequality constraints and boundary conditions

To obtain an energy-optimized trajectory, we utilize an energy-based cost functional in (6a), with

$$F(\boldsymbol{z}_{r,d}(t),\boldsymbol{\tau}_d(t),t) = n(\tau_{m,d}(t))^2\cdot \Big(\big|\cos\delta(\tau_\delta(t))\,u_{r,d}(t)\big| + \big|L_e\sin\delta(\tau_\delta(t))\,r\big|\Big)\,, \tag{7}$$

where $n(\tau_m)$ is a function that maps the control input τ_m to propeller rounds per minute (RPM), $\delta(\tau_\delta)$ maps the control input τ_δ to engine angle, and L_e is the length between the engine and the axis of rotation. Force is often modeled as being proportional to the square of propeller rate, and the product of force and velocity gives a measure proportional to power [22].

The dynamics (6b) are the same as (3), collected in the model

$$\dot{\boldsymbol{z}}_r = \boldsymbol{f}(\boldsymbol{z}_r,\boldsymbol{\tau})\,, \tag{8}$$

where $\boldsymbol{z}_r = [\boldsymbol{\eta}^\top, \boldsymbol{x}_r^\top]^\top$, and

$$\boldsymbol{f}(\boldsymbol{z}_r,\boldsymbol{\tau}) = \begin{bmatrix}\mathbf{R}(\psi)\boldsymbol{x}_r + [\boldsymbol{V}_c^\top, 0]^\top\\ \mathbf{M}(\boldsymbol{x}_r)^{-1}(-\boldsymbol{\sigma}(\boldsymbol{x}_r) + \boldsymbol{\tau})\end{bmatrix}\,. \tag{9}$$

Static obstacles are represented in (6c) as elliptic inequalities. The basis for the elliptic inequality is

$$\left(\frac{x - x_c}{x_a}\right)^2 + \left(\frac{y - y_c}{y_a}\right)^2 \ge 1\,, \tag{10}$$

where x_c and y_c describe the ellipse center in NED, and x_a and y_a describe the sizes of the two elliptic axes. To allow angled ellipses, the differences $x - x_c$ and $y - y_c$ are rotated by the angle α between the direction of x_a and north. Applying the logarithmic function to both sides of the inequality avoids quadratic growth in x and y, which is convenient for the solvers, and adding a small value $\epsilon > 0$ improves the numerical properties when $x \to x_c$ and $y \to y_c$,

without changing the inequality. The resulting inequality in (6c) is:

$$h_o(x, y, x_c, y_c, x_a, y_a, \alpha) = \\ - \log \left[\left(\frac{(x - x_c)\cos\alpha + (y - y_c)\sin\alpha}{x_a} \right)^2 \right. \\ \left. + \left(\frac{-(x - x_c)\sin\alpha + (y - y_c)\cos\alpha}{y_a} \right)^2 + \epsilon \right] \\ + \log(1 + \epsilon) \leq 0\,. \quad (11)$$

In addition to the static obstacles, the state constraints $\boldsymbol{x}_{r,d} \in \mathbb{X}_r$ and control constraints $\boldsymbol{\tau}_d \in \mathbb{U}$ are also enforced in (6c).

Boundary conditions (6d) are employed to decide initial position and velocity, final position, and to set a maximum end time $t_f \leq t_{f,\max}$. The maximum end-time limitation is necessary for convergence of the OCP, since without it, the cost functional is minimized by not using any control action, resulting in the ship following the ocean currents.

B. Planner output

The OCP solution yields a trajectory $\boldsymbol{z}_{r,d}(\cdot)$ which describes the motion of the ASV over time t. The notation $\boldsymbol{z}_{r,d}(\cdot)$ emphasizes that this is not a point $\boldsymbol{z}_{r,d}(t)$ at a certain time t, but describes a trajectory of states. The planner is run offline ahead of time, and the x, y coordinates of the trajectory are used as input to the mid-level MPC algorithm, i.e. $\boldsymbol{p}_d(t) = [x_d(t), y_d(t)]^\top$, as illustrated in Fig. 4. Fig. 1 shows this interface in the context of the hybrid architecture.

IV. Mid-level COLAV

The mid-level COLAV algorithm is an MPC-based algorithm intended for long-term COLAV with respect to both static and dynamic obstacles, originally presented in [8]. The algorithm complies with Rule 8 of COLREGs, producing maneuvers that are readily observable for other vessels while following a reference trajectory specified by the trajectory planner.

In this section, we further develop the algorithm to be able to perform *relative* trajectory tracking and improve on the numeric properties of the algorithm. By relative trajectory tracking, we mean that we want the algorithm to track a desired trajectory with a time offset $t_b \in \mathbb{R}$. This implies that if the ASV for some reason lag behind the desired trajectory $\boldsymbol{p}_d(t) = [x(t), y(t)]^\top$, we may adjust t_b to offset the desired trajectory rather than speeding up to catch up with $\boldsymbol{p}_d(t)$. We define the relative desired trajectory as

$$\bar{\boldsymbol{p}}_d(t) = \boldsymbol{p}_d(t + t_b), \quad (12)$$

where t_b is the time offset, which is computed each time the mid-level algorithm is run. For notational simplicity, we omit t_b from the function parameters since (12) at each MPC run should be interpreted as a time-shifted trajectory only dependent on the time t. At a time step t_0, denoting the current time when the MPC is called, the time offset t_b is computed by the optimization problem

$$t_b(t_0) = \arg\min_{t_b} \left\| \boldsymbol{p}_d(t_0 + t_b) - \boldsymbol{p}(t_0) \right\|_2, \quad (13)$$

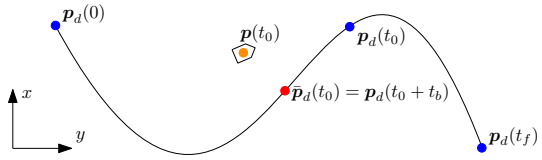


Fig. 4. Illustration of relative trajectory tracking. The black line shows the desired trajectory, with the blue circles marking the start position $\boldsymbol{p}_d(0)$, current position $\boldsymbol{p}_d(t_0)$ and goal desired position $\boldsymbol{p}_d(t_f)$. The orange circle marks the vessel position at time t, while the red point shows the relative desired position $\bar{\boldsymbol{p}}_d(t) = \boldsymbol{p}_d(t_0 + t_b)$.

which can be solved by a simple line search algorithm. Here, we find the offset t_b which minimizes the Euclidean distance between the current relative desired trajectory position $\bar{\boldsymbol{p}}_d(t_0)$ and the current vessel position $\boldsymbol{p}(t_0)$. The concept is illustrated in Fig. 4. In general, other paradigms can be used to find t_b based on the desired behavior. Using the Euclidean distance makes the algorithm track the desired trajectory from the closest point.

A similar concept can be employed if the input to the mid-level algorithm is a geometric path $\boldsymbol{p}_d(\theta)$ with an associated along-path speed $U_d(\theta)$, where $\theta \in \mathbb{R}$ is a path parameter. In this case, we compute an initial path parameter

$$\theta_0(t_0) = \arg\min_{\theta} \left\| \boldsymbol{p}_d(\theta) - \boldsymbol{p}(t_0) \right\|_2, \quad (14)$$

and generate a map from time to path parameter $\theta : \mathbb{R} \to \mathbb{R}$ by integrating

$$\dot{\theta} = \frac{U_d(\theta)}{\sqrt{\left(\frac{\partial x_d(\theta)}{\partial \theta}\right)^2 + \left(\frac{\partial y_d(\theta)}{\partial \theta}\right)^2}} \quad \theta(t_0) = \theta_0(t_0)\,. \quad (15)$$

This gives $\theta(t)$ valid for $t \geq t_0$. This map is then used to compute the relative desired trajectory as $\bar{\boldsymbol{p}}_d(t) = \boldsymbol{p}_d(\theta(t))$. This would for instance be useful if we would like the mid-level algorithm to input a waypoint-defined path.

The mid-level algorithm is formalized as the OCP:

$$\min_{\boldsymbol{\eta}, \boldsymbol{x}_r} \ \phi(\boldsymbol{\eta}(\cdot), \boldsymbol{x}_r(\cdot)) \quad (16a)$$

subject to

$$\dot{\boldsymbol{\eta}}(t) = \mathbf{R}(\psi(t))\boldsymbol{x}_r(t) + \begin{bmatrix} \boldsymbol{V}_c \\ 0 \end{bmatrix} \ \forall t \in [t_0, t_0 + T_h] \quad (16b)$$

$$\boldsymbol{h}_m(\boldsymbol{\eta}(t), \boldsymbol{x}_r(t), t) \leq \mathbf{0} \ \forall t \in [t_0, t_0 + T_h] \quad (16c)$$

$$\boldsymbol{e}_m(\boldsymbol{\eta}(t_0)) = \mathbf{0}\,, \quad (16d)$$

where $T_h > 0$ is the prediction horizon. Notice that we use a purely kinematic model (16b) in the mid-level algorithm, which is done to reduce the computational load, ensuring that the algorithm can be run in real time as an MPC algorithm. In the hybrid architecture in Fig. 1, the output of the algorithm is passed to the short-term algorithm, which takes the vessel dynamics into account, ensuring that only feasible trajectories are fed to the vessel controllers and justifying neglecting kinetic feasibility in the mid-level algorithm. The constraint $x_r \in \mathbb{X}_r$ as well as static and dynamic obstacles

are enforced in (16c), while the boundary constraints (16d) ensures that the trajectory starts at the current pose $\boldsymbol{\eta}(t_0)$.

To solve the OCP (16), we discretize it over $t \in [t_0, t_0 + T_h]$ using multiple shooting with N_p time steps. The vessel model (16b) is discretized using 4$^{\text{th}}$-order Runge-Kutta, while the cost functional is discretized using forward Euler. For more details on the discretization, see [8]. This results in the NLP:

$$\begin{aligned} &\min_{\boldsymbol{w}} \ \phi_p(\boldsymbol{w}, \bar{\boldsymbol{p}}_{d,1:N_p}) + \phi_c(\boldsymbol{w}) \\ &\text{subject to} \\ &\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) = \boldsymbol{0} \\ &\boldsymbol{h}(\boldsymbol{w}) \leq \boldsymbol{0}\,, \end{aligned} \tag{17}$$

where $\boldsymbol{w} = [\boldsymbol{\eta}_0^\top, \boldsymbol{x}_{r,0}^\top, \ldots, \boldsymbol{\eta}_{N_p-1}^\top, \boldsymbol{x}_{r,N_p-1}^\top, \boldsymbol{\eta}_{N_p}^\top]^\top \in \mathbb{R}^{5N_p+3}$ is a vector of $5N_p + 3$ decision variables, $\bar{\boldsymbol{p}}_{d,1:N_p} = [\bar{\boldsymbol{p}}_{d,1}, \bar{\boldsymbol{p}}_{d,2}, \ldots, \bar{\boldsymbol{p}}_{d,N_p}]$ is a sequence of desired positions and $\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) \in \mathbb{R}^{3N_p+3}$ is a vector of $3N_p + 3$ shooting and boundary constraints.

A. Objective function

The objective function consists of two terms, where $\phi_p(\boldsymbol{w}, \bar{\boldsymbol{p}}_d(t))$ ensures convergence to the relative trajectory and $\phi_c(\boldsymbol{w})$ ensures that the resulting trajectory is readily observable. The functions are given as

$$\phi_p(\boldsymbol{w}, \bar{\boldsymbol{p}}_{d,1:N_p}) = \sum_{k=1}^{N_p} K_p q_p\left(\boldsymbol{p}_k, \bar{\boldsymbol{p}}_{d,k}\right) \tag{18a}$$

$$\phi_c(\boldsymbol{w}) = \sum_{k=0}^{N_p-1} K_{\dot{U}} q_{\dot{U}}(\dot{U}_k) + K_{\dot{\chi}} q_{\dot{\chi}}(\dot{\chi}_k), \tag{18b}$$

where the function $q_p(\cdot, \cdot)$ penalizes trajectories deviating from the desired trajectory, while and $q_{\dot{U}}(\cdot)$ and $q_{\dot{\chi}}(\cdot)$ penalize speed and course changes that are not readily observable to other vessels. The values $K_p, K_{\dot{U}}, K_{\dot{\chi}} > 0$ are tuning parameters, while $\bar{\boldsymbol{p}}_{d,k} = \bar{\boldsymbol{p}}_d(t_k)$ denotes the desired position at time t_k. Notice that neither the speed U or the course χ are elements in $\boldsymbol{w}$, but they can be computed using $U = \sqrt{u^2 + v^2}$ and $\chi = \psi + \arcsin\frac{v}{U}$, respectively. Furthermore, their derivatives are computed using finite differencing.

To measure the deviance from the desired trajectory, we want to use the Huber loss function, which is quadratic around the origin and resembles the absolute value function above a threshold $\sigma > 0$:

$$H(\rho) = \begin{cases} \frac{1}{2}\rho^2 & |\rho| \leq \sigma \\ \sigma(|\rho| - \frac{1}{2}\sigma) & |\rho| > \sigma\,. \end{cases} \tag{19}$$

Using the Huber function rather than a quadratic cost avoids the issue of the position error dominating over the other terms in the objective function when the position error is large, ensuring a similar behavior close and far from the desired trajectory [8]. The Huber function (19) can be used to define $q_p(\boldsymbol{p}, \boldsymbol{p}_d)$ as:

$$q_p(\boldsymbol{p}, \bar{\boldsymbol{p}}_d) = H(x - \bar{x}_d) + H(y - \bar{y}_d) \tag{20}$$

to evaluate a 2-dimensional loss [23].

The Huber function is, however, only C^1, resulting in (17) having a discontinuous Hessian matrix, making the NLP difficult to solve. In [8], this was circumvented using the pseudo-Huber function, which is a smooth approximation of (19). This can, however, cause numerical issues since the pseudo-Huber function introduces a square root in the objective function. To remedy this, the Huber function (19) can be implemented as a quadratic program (QP) [23]:

$$\begin{aligned} H(\rho) = &\min_{\omega,\mu} \ \sigma\omega + \frac{1}{2}\mu^2 \\ &\text{subject to} \\ &-\mu - \omega \leq \rho \leq \mu + \omega \\ &\omega \geq 0\,. \end{aligned} \tag{21}$$

This avoids the discontinuity issues by utilizing slack variables. To use (21) rather than (19) to implement the Huber function, we combine (17) and (21) to define a new NLP which is equivalent to (17) [23]:

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\omega},\boldsymbol{\mu}} \ \bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}) + \phi_c(\boldsymbol{w}) \\ &\text{subject to} \\ &\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) = \boldsymbol{0} \\ &\boldsymbol{h}(\boldsymbol{w}) \leq \boldsymbol{0} \\ &\bar{\boldsymbol{h}}_k(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \bar{\boldsymbol{p}}_{d,k}) \leq \boldsymbol{0}\ \forall k \in \{1, \ldots, N_p\}\,, \end{aligned} \tag{22}$$

where $\boldsymbol{\omega} = [\boldsymbol{\omega}_1^\top, \boldsymbol{\omega}_2^\top, \ldots, \boldsymbol{\omega}_{N_p}^\top]^\top \in \mathbb{R}^{2N_p}$ and $\boldsymbol{\mu} = [\boldsymbol{\mu}_1^\top, \boldsymbol{\mu}_2^\top, \ldots, \boldsymbol{\mu}_{N_p}^\top]^\top \in \mathbb{R}^{2N_p}$ are slack variables. The function $\bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$ is

$$\bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}) = \sum_{k=1}^{N_p} K_p \left(\sigma \boldsymbol{1}^\top \boldsymbol{\omega}_k + \frac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k\right), \tag{23}$$

and $\bar{\boldsymbol{h}}_k(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}) \in \mathbb{R}^{6N_p}$ encodes the constraints in (21):

$$\bar{\boldsymbol{h}}_k(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}, \bar{\boldsymbol{p}}_{d,k}) = \begin{bmatrix} \boldsymbol{v_k} + \boldsymbol{\mu_k} + \boldsymbol{p}_k - \bar{\boldsymbol{p}}_{d,k} \\ \boldsymbol{v_k} + \boldsymbol{\mu_k} - (\boldsymbol{p}_k - \bar{\boldsymbol{p}}_{d,k}) \\ -\boldsymbol{\omega}_k \end{bmatrix}. \tag{24}$$

The penalty terms in speed and course change motivates the algorithm to perform readily observable maneuvers by penalizing maneuvering with small speed and course changes more than using large changes. This is done by using a nonlinear cost based on a combination of a quadratic and decaying exponential function:

$$q(\rho; a, b) = a\rho^2 + (1 - e^{-\frac{\rho^2}{b}}), \tag{25}$$

where $a > 0$ and $b > 0$ are parameters. Using this function, the penalty terms in speed and course change are defined as

$$q_{\dot{U}}(\dot{U}) = \frac{100}{q(\dot{U}_{\max}; a_{\dot{U}}, b_{\dot{U}})} q(\dot{U}; a_{\dot{U}}, b_{\dot{U}}) \tag{26a}$$

$$q_{\dot{\chi}}(\dot{\chi}) = \frac{100}{q(\dot{\chi}_{\max}; a_{\dot{\chi}}, b_{\dot{\chi}})} q(\dot{\chi}; a_{\dot{\chi}}, b_{\dot{\chi}})\,. \tag{26b}$$

This results in changing course or speed with a high acceleration or turn rate is preferred over changes with low rates of change, see [8] for more details on the speed and course change penalty terms.

B. Obstacle handling and steady-state feasibility

The constraint $\boldsymbol{h}(\boldsymbol{w}) \leq \boldsymbol{0}$ ensures COLAV and that only steady-state feasible trajectories are specified.

As in the trajectory planning algorithm, static obstacles are avoided by using elliptic inequalities to define constraints. For the i-th static obstacle, we define the inequality

$$\boldsymbol{h}_{s_i}(\boldsymbol{w}) = \begin{bmatrix} h_o(x_1, y_1, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \\ \vdots \\ h_o(x_{N_p}, y_{N_p}, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \end{bmatrix} \leq \boldsymbol{0}, \tag{27}$$

where $h_o(\cdot)$ is defined in (11) and $x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}$ and α_i denote the x and y center coordinates, major and minor axis lengths and major axis orientation of the i-th static obstacle, respectively.

Dynamic obstacles are handled by letting the ellipse parameters be time varying. For the i-th dynamic obstacle, we define the inequality

$$\boldsymbol{h}_{m_i}(\boldsymbol{w}) = \begin{bmatrix} h_o(x_1, y_1, x_{c,i}(t_1), y_{c,i}(t_1), \\ x_{a,i}, y_{a,i}, \alpha_i(t_1)) \\ \vdots \\ h_o(x_{N_p}, y_{N_p}, x_{c,i}(t_{N_p}), y_{c,i}(t_{N_p}), \\ x_{a,i}, y_{a,i}, \alpha_i(t_{N_p})) \end{bmatrix} \leq \boldsymbol{0}, \tag{28}$$

where $x_c(t), y_c(t), x_{a,i}, y_{a,i}$ and $\alpha_i(t)$ denote the time-varying x and y center coordinates, major and minor axis lengths and major axis orientation of the i-th dynamic obstacle, respectively. We assume, without loss of generality, that the minor and major axis lengths are constant.

Given S static and M dynamic obstacles, we define the inequality

$$\begin{aligned} \boldsymbol{h}_o(\boldsymbol{w}) = \Big[& \boldsymbol{h}_{s_1}(\boldsymbol{w})^\top \quad \dots \quad \boldsymbol{h}_{s_S}(\boldsymbol{w})^\top \\ & \boldsymbol{h}_{m_1}(\boldsymbol{w})^\top \quad \dots \quad \boldsymbol{h}_{m_M}(\boldsymbol{w})^\top \Big]^\top \leq \boldsymbol{0}, \end{aligned} \tag{29}$$

which ensures avoidance of both static and dynamic obstacles.

Similarly as in [8], we ensure steady-state feasibility at each time step through a constraint $\boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_r) \leq \boldsymbol{0} \in \mathbb{R}^4$, which encodes the constraint $\boldsymbol{x}_r \in \mathbb{X}_r$. To ensure feasibility for the entire prediction horizon, we define the inequality

$$\boldsymbol{h}_{\boldsymbol{x}_r}(\boldsymbol{w}) = \Big[\boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_{r,0})^\top \quad \dots \quad \boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_{r,N_p-1})^\top \Big]^\top \leq \boldsymbol{0}. \tag{30}$$

Finally, the inequality constrains are combined as

$$\boldsymbol{h}(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{h}_o(\boldsymbol{w}) \\ \boldsymbol{h}_{\boldsymbol{x}_r}(\boldsymbol{w}) \end{bmatrix} \in \mathbb{R}^{(M+S+4)N_p}, \tag{31}$$

which is used in (22).

V. Simulation scenario and results

The scenario used for testing the hybrid architecture is shown in Fig. 5. It consists of two static obstacles (a and b) and one dynamic obstacle (c) with parameters listed in Table I. The dynamic obstacle (c) has the start position listed in the table, and continues with a speed of $5\,\mathrm{m/s}$ and course $-11.25°$. It comes in the way of the ship in the middle of the nominal trajectory, which requires the mid-level algorithm to plan a trajectory around it.

TABLE I

Obstacle parameters.

Obstacle	x_c	y_c	x_a	y_a	α
a	$5500\,\mathrm{m}$	$2000\,\mathrm{m}$	$4000\,\mathrm{m}$	$1000\,\mathrm{m}$	$-5°$
b	$2000\,\mathrm{m}$	$6500\,\mathrm{m}$	$4000\,\mathrm{m}$	$1000\,\mathrm{m}$	$5°$
c*	$500\,\mathrm{m}$	$4800\,\mathrm{m}$	$1000\,\mathrm{m}$	$400\,\mathrm{m}$	$-11.25°$

* Obstacle c continues with speed $5\,\mathrm{m/s}$ and course $-11.25°$ after initialization.

TABLE II

Simulation and tuning parameters.

Param.	Value	Comment
$t_{f,\max}$	$1500\,\mathrm{s}$	High-level planner time constraint
$[V_x, V_y]$	$[2.5, 0]\,\mathrm{m/s}$	Ocean current velocity
N_s	165	Number of simulation steps
h	$10\,\mathrm{s}$	Mid-level step size
N_p	36	Mid-level prediction steps
K_p	$2 \cdot 10^{-2}$	Position error scaling
$K_{\dot{U}}$	2	SOG-derivative penalty term scaling
$K_{\dot{\chi}}$	7.5	Course-derivative penalty term scaling
$[a_{\dot{U}}, b_{\dot{U}}]$	$[8, 2.5{\cdot}10^{-4}]$	SOG-derivative penalty term parameters
$[a_{\dot{\chi}}, b_{\dot{\chi}}]$	$[112, 1.875{\cdot}10^{-4}]$	Course-derivative penalty term parameters

The algorithms are run with the parameters listed in Table II. The high-level algorithm is implemented using the pseudospectral optimal control package DIDO for MATLAB on a desktop computer [7]. The mid-level algorithm is implemented using CASADI [24] and IPOPT [25] for MATLAB on a desktop computer. Since the mid-level algorithm is MPC-based, a new solution is computed at every time step, where only the first step of the solution is implemented.

Fig. 5 shows how the mid-level algorithm modifies the nominal trajectory to produce collision-free and observable maneuvers. Fig. 6 shows speeds and course angle plots from the mid-level algorithm. From Fig. 5, we see that when the nominal trajectory is free from obstacles, the mid-level algorithm tracks it well, while performing readily observable maneuvers. Deviations from the nominal trajectory occur from time stamp t_2, where the mid-level algorithm chooses to keep a different course to pass behind the dynamic obstacle. Fig. 6 shows that the SOG U is held at a lower value to be able to pass behind the obstacle between $400\,\mathrm{s}$ and $650\,\mathrm{s}$, which also makes the maneuver more observable, according to the objective function (18) which discourages small changes in SOG. The figure also shows that course changes are performed in large steps, in excess of $30°$, which is accepted as *readily observable* maneuvers even in restricted visibility [4]. When the obstacle is passed, the mid-level algorithm again tracks the reference speed and moves

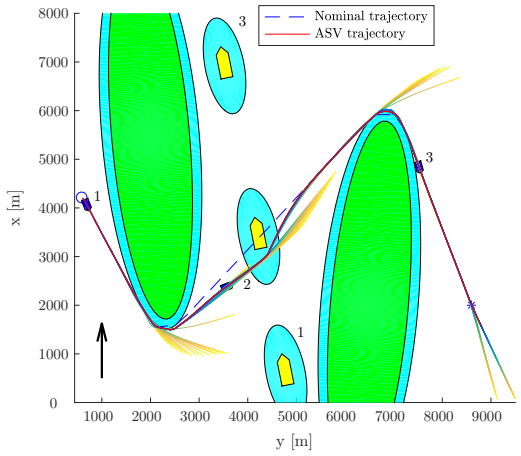


Fig. 5. Test scenario with static and moving obstacles. Nominal trajectory (striped blue) and resulting trajectory (red) together with predicted trajectory at every time step of the MPC. The predicted trajectories start with blue color at t_0, and gets more yellow towards the end of the prediction horizon $t_0 + T_h$. The green ellipses are the "real" static obstacles, contained in blue ellipses which are safety margins. The moving obstacle is a yellow patch encapsulated by a blue ellipse representing its safety margin. The time stamps (1, 2, 3) represent times 20 s, 540 s and 1190 s, respectively. The arrow in the lower-left corner is the ocean current direction.

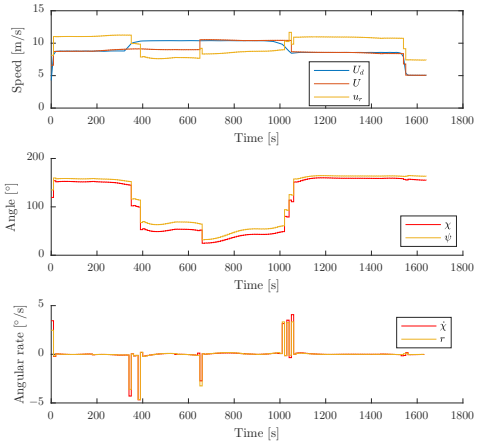


Fig. 6. Desired SOG, actual SOG and relative surge velocity (top), course and yaw angle (middle), and course and turn rate (bottom) over the duration of the scenario.

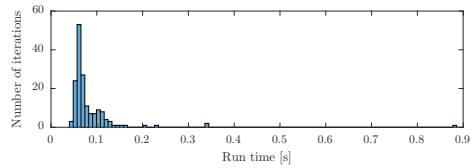


Fig. 7. Run times for the mid-level algorithm.

back to the nominal trajectory. In addition, notice that the relative surge velocity u_r is actively controlled to ensure readily observable changes in the SOG, especially as the ASV changes course at around 350 s and 1050 s.

The run time for the high-level planner is between 150 s and 200 s, depending on the scenario. Since the planner is run offline before the start of the scenario, high run times are not detrimental. For the mid-level algorithm, the majority of the algorithm steps are completed in less than 0.1 s, as seen in Fig. 7. There are some outliers, but the maximum observed run time is approximately 0.9 s, which with a step size of 10 s is considered to be real-time feasible. Ideally, the mid-level algorithm provides safe commands in a timely manner, but the hybrid architecture allows the short-term COLAV algorithm to serve as a backup in case of a malfunction above it in the hierarchy.

VI. Conclusion

We have developed and verified parts of a hybrid COLAV architecture for ASVs. The simulation results show that the architecture enables an ASV to avoid both static and dynamic obstacles and produces readily observable maneuvers in compliance with Rule 8 of COLREGs. In the absence of dynamic obstacles, the ASV tracks an energy-optimized trajectory.

Possibilities for further work include:

- Complete the development and implementation of the three-layered hybrid architecture by including a short-term COLAV algorithm, e.g. [5], and perform full-scale experiments.
- Investigate the possibility of performing periodic high-level re-planning to increase optimality of the trajectory by taking into account updated information about environmental factors and ship state.
- Investigate if an alternative OCP solver can decrease run time for the high-level algorithm.
- Include more COLREGs rules in the mid-level algorithm, specifically the required behavior in *overtaking*, *crossing* and *head-on* situations.

Acknowledgments

This work was supported by the Research Council of Norway through project number 269116, project number 244116, as well as the Centers of Excellence funding scheme with project number 223254.

References

[1] Kongsberg Maritime. (2018). Autonomous ship project, key facts about YARA Birkeland, [Online]. Available: `https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/4B8113B707A50A4FC125811D00407045` (visited on 2018-11-10).

[2] O. Levander, "Autonomous ships on the high seas," *IEEE Spectrum*, vol. 54, no. 2, pp. 26–31, 2017-02.

[3] DNV GL, "Sammenstilling av grunnlagsdata om dagens skipstrafikk og drivstofforbruk, Miljøtiltak for maritim sektor," Norwegian, Klima- og miljødepartementet, Tech. Rep. 2014-1667, 2014.

[4] A. N. Cockcroft and J. N. F. Lameijer, *A Guide to the Collision Avoidance Rules*. Elsevier Butterworth-Heinemann, 2004, ISBN: 0-7506-6179-8.

[5] B.-O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. Brekke, "The branching-course MPC algorithm for maritime collision avoidance," Submitted to Journal of Field Robotics.

[6] G. Bitar, M. Breivik, and A. M. Lekkas, "Energy-optimized path planning for autonomous ferries," in *Proc. of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)*, Opatija, Croatia, 2018, pp. 389–394.

[7] I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory to flight," *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.

[8] B.-O. H. Eriksen and M. Breivik, "MPC-based mid-level collision avoidance for ASVs using nonlinear programming," in *Proc. of the IEEE Conference on Control Technology and Applications (CCTA)*, Mauna Lani, HI, USA, 2017, pp. 766–772.

[9] I. B. Hagen, D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "MPC-based collision avoidance strategy for existing marine vessel guidance systems," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 7618–7623.

[10] P. Švec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. von Ellenrieder, M. Dhanak, and S. K. Gupta, "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013, pp. 3871–3878.

[11] M. Abdelaal and A. Hahn, "NMPC-based trajectory tracking and collision avoidance of unmanned surface vessels with rule-based colregs confinement," in *Proc. of the IEEE Conference on Systems, Process and Control (ICSPC)*, 2016.

[12] 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.

[13] A. Tsourdos, B. White, and M. Shanmugavel, *Cooperative Path Planning of Unmanned Aerial Vehicles*. John Wiley & Sons, Inc., 2010, 190 pp., ISBN: 978-0-470-74129-0.

[14] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path plannning," Tech. Rep., 1998.

[15] H.-T. L. Chiang and L. Tapia, "COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018.

[16] B.-O. H. Eriksen, E. F. Wilthil, A. L. Flåten, E. F. Brekke, and M. Breivik, "Radar-based maritime collision avoidance using dynamic window," in *Proc. of the IEEE Aerospace Conference*, Big Sky, MT, USA, 2018.

[17] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGs, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2014.

[18] Ø. A. G. Loe, "Collision avoidance for unmanned surface vehicles," Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2008.

[19] G. Casalino, A. Turetta, and E. Simetti, "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields," in *Proc. of the IEEE Oceans Conference*, IEEE, Bremen, Germany, 2009, pp. 1–8.

[20] B.-O. H. Eriksen and M. Breivik, "Modeling, identification and control of high-speed ASVs: Theory and experiments," in *Sensing and Control for Autonomous Vehicles*, T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, Eds., Springer International Publishing, 2017, pp. 407–431.

[21] B.-O. H. Eriksen and M. Breivik, "A model-based speed and course controller for high-speed ASVs," in *Proc. of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)*, Opatija, Croatia, 2018, pp. 317–322.

[22] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. John Wiley & Sons, 1994, ISBN: 0-471-94-113-1.

[23] S. Gros and M. Zanon, "Penalty functions for handling large deviation of quadrature states in NMPC," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3848–3860, 2017.

[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[25] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2005.

Paper C Warm-started optimized trajectory planning for ASVs

Published paper by **G. Bitar**, V. N. Vestad, A. M. Lekkas, and M. Breivik. "Warm-started optimized trajectory planning for ASVs". In: *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. Daejeon, South Korea, 2019, pp. 308–314. DOI: 10.1016/j.ifacol.2019.12.325. arXiv: 1907.02696 [eess.SY].

Bibliography entry [10].

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-21 (2019) 308–314

Warm-Started Optimized Trajectory Planning for ASVs

Glenn Bitar * **Vegard N. Vestad** * **Anastasios M. Lekkas** * **Morten Breivik** *

* *Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway.*
E-mails: {glenn.bitar,anastasios.lekkas}@ntnu.no, vegardnittervestad@gmail.com, morten.breivik@ieee.org

Abstract: We consider warm-started optimized trajectory planning for autonomous surface vehicles (ASVs) by combining the advantages of two types of planners: an A* implementation that quickly finds the shortest piecewise linear path, and an optimal control-based trajectory planner. A nonlinear 3-degree-of-freedom underactuated model of an ASV is considered, along with an objective functional that promotes energy-efficient and readily observable maneuvers. The A* algorithm is guaranteed to find the shortest piecewise linear path to the goal position based on a uniformly decomposed map. Dynamic information is constructed and added to the A*-generated path, and provides an initial guess for warm starting the optimal control-based planner. The run time for the optimal control planner is greatly reduced by this initial guess and outputs a dynamically feasible and locally optimal trajectory.

Keywords: Trajectory planning, energy optimization, autonomous surface vehicles, optimal control

1. INTRODUCTION

Motivated by potential for reduced costs, as well as safer and more environmentally friendly operations, technology for autonomous surface vehicles (ASVs) is being developed at a rapid pace. Several commercial actors are spearheading the search for solutions for safe, collision-free and reliable autonomous operations. Rolls-Royce and Finferries demonstrated the world's first autonomous car ferry "Falco" in 2018 (Jallal, 2018), which navigated autonomously between two ports in Finland by combining advanced sensor technology and collision avoidance algorithms.

A prerequisite for safe and efficient operation is a well-functioning path or trajectory planning method. Such a method is responsible for providing the ASV with a safe trajectory that avoids static obstacles such as land and shallow waters. Depending on the type of operation, one might want to optimize the trajectory for various objectives, such as energy efficiency, speed or trajectory length.

Numerous path and trajectory planning algorithms have been researched and are available for marine applications. One may categorize these planning algorithms as being roadmap-based or optimization-based. Figure 1 gives an overview of the categorization of some planning algorithm types. *Roadmap methods* are based on exploring points in the geometric space in order to build a path between the start and goal positions. There are two subcategories in roadmap methods. *Combinatorial* methods decompose an obstacle map using a preferred strategy, and perform a search in the resulting graph. The decomposition strategies include e.g. uniform grids, Voronoi diagrams and visibility graphs. The combinatorial methods explore the entire geometric space. The graph search is often performed using A*, which is an efficient and well-known search algorithm widely used to solve path planning problems (Hart et al., 1968). A* guarantees to find the shortest path when using an admissible heuristic function. Hybrid A* extends the A* algorithm by generating dynamic trajectories to connect nodes, thus adding dynamic information to the search (Dolgov et al., 2010). As opposed to combinatorial methods, *sampling-based* methods randomly explores points in the map to build a path towards the goal. Probabilistic roadmap (PRM) is a sampling-based planning method that draws samples from the configuration space and connects them using a local planner (Kavraki et al., 1996). A graph search algorithm is applied to find the minimum cost path from start to goal in the resulting graph. Rapidly-

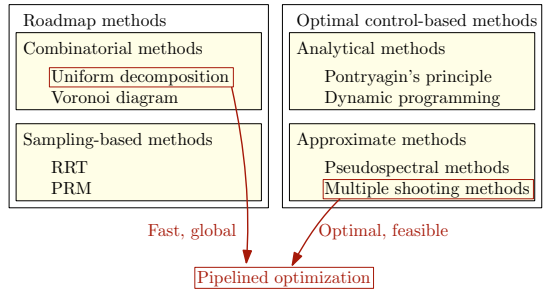


Fig. 1. Categorization of some planning algorithms.

2405-8963 © 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2019.12.325

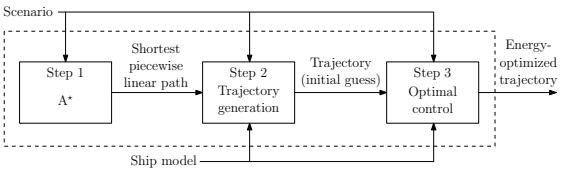


Fig. 2. Pipelined path planning concept.

exploring random tree (RRT) is another sampling-based method which calculates input trajectories between randomly sampled points and connects them in a tree until the start and goal positions are connected (LaValle, 1998). Although RRT uses a cost function, the method is not optimal and will lock into the first connection between start and goal. Various flavors of RRT are developed to amend this, e.g. RRT* (Karaman and Frazzoli, 2011). This method continuously performs tree rewiring and has probabilistic completeness, but converges slowly.

The other group of planning methods contains algorithms based on *optimal control*. This group may further be divided into analytical and approximate methods. Analytical methods such as Pontryagin's principle are only able to find solutions in very simple cases and are generally unpractical. Approximate methods such as e.g. pseudospectral optimal control (Bitar et al., 2018; Ross and Karpenko, 2012) are highly sensitive to initial guesses of the solution and will converge to a local optimum close to this guess. Without a good initial guess, they also experience long run times and are sensitive to problem dimensionality.

Zhang et al. (2018) plan trajectories for parking autonomous cars by combining hybrid A* with an optimal control-based method. Motivated by the same goals of exploiting the strengths and mitigate the weaknesses of optimal control-based algorithms, we here attempt to solve the long-term trajectory planning problem for ASVs as a transcribed optimal control problem (OCP), and warm start the solver using the smoothed solution of an A* geometric planner. In this three-step pipelined approach, the A* planner swiftly provides a set of waypoints representing the shortest path as Step 1. This path is converted into a full state trajectory by adding artificial and nearly feasible temporal information as Step 2. Step 3 takes this trajectory and uses it as the initial guess for an OCP solver, which finds an optimized trajectory near the globally shortest path. The structure of this pipelined concept is illustrated in Figure 2. The method is an off-line global planner, which assumes that information about the map and environment is known a priori.

The rest of this paper is organized as follows: Section 2 presents the mathematical model of the ASV used in simulations and planning. Finding the waypoints describing the shortest path with A* is described in Section 3, and Section 4 explains how the A* solution is converted to a trajectory. Section 5 shows how the OCP is transcribed to an nonlinear program (NLP), which yields an optimized trajectory when solved. Simulation scenarios and results are presented in Section 6, while Section 7 concludes the paper.

2. ASV MODELING AND OBSTACLES

In (Loe, 2008), a simple nonlinear 3-degree-of-freedom ship model is identified to approximate the dynamics of the ASV Viknes 830. Without loss of generality for the method described in this paper, we use that model to perform trajectory planning. The model has the form

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{1a}$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}(\boldsymbol{u})\,. \tag{1b}$$

The pose vector $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times \mathrm{S}$ contains the ASV's position and heading angle in the Earth-fixed North East Down (NED) frame. The velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$ contains the ASV's body-fixed velocities: surge, sway and yaw rate, respectively. The rotation matrix $\mathbf{R}(\psi)$ transforms the body-fixed velocities to NED:

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

The matrix $\mathbf{M} \in \mathbb{R}^{3\times3}$ represents system inertia, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ Coriolis and centripetal effects, and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ represents damping effects. The ASV is controlled by the control vector $\boldsymbol{u} = [X, N]^\top \in \mathbb{R}^2$, which contains surge force and yaw moment. The control vector is mapped to a force vector $\boldsymbol{\tau}(\boldsymbol{u}) = [X, 0, N]^\top$. The ASV's states are collected in the vector $\boldsymbol{x} = [x, y, \psi, u, v, r]^\top$, and we collect the dynamic model (1) in the following compact form for notational ease in the remainder of the paper:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \mathbf{R}(\psi)\boldsymbol{\nu} \\ \mathbf{M}^{-1}\left(-\mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau}(\boldsymbol{u})\right) \end{bmatrix}. \tag{3}$$

3. STEP 1: A* PATH PLANNER

To quickly find the global shortest collision-free path between a start and goal position, we use an A* implementation on a uniformly decomposed grid. The A* implementation is standard, and details may be found in e.g. (Hart et al., 1968). The search algorithm looks for collision-free paths between nodes in the uniform grid, and uses Euclidean distance as cost and heuristic functions.

The decomposition of the map affects the solution space and the run time for Step 1. Using a uniform grid with grid size $\Delta d > 0$ too large will take paths going through narrow passages away from the solution space, and the desired shortest path may not be found. A smaller grid size will explore more options, but requires more evaluation, giving a longer run time. This uniform grid is in our case chosen for simplicity, however exploring other decompositions such as Voronoi diagrams or a non-uniform grid might be desirable for performance reasons.

4. STEP 2: TRAJECTORY GENERATION

In order to use the shortest path generated by Step 1 as an initial guess for the OCP, we convert it to a trajectory based on straight segments and circle arcs using a nominal forward velocity $u_{\text{nom}} > 0$. The trajectory generation consists of three sub-steps: waypoint reduction, waypoint connection, and adding dynamic information.

4.1 Waypoint reduction

Algorithm 1 is employed to reduce the A* path from Step 1 to a minimum number of waypoints. The algorithm outputs a reduced path as an ordered set of waypoints $\mathbb{P} = \{\boldsymbol{p}_k \in \mathbb{R}^2 \mid k = 1, \ldots, N_r\}$ where N_r is the number of waypoints. The A* waypoints are denoted $\boldsymbol{p}_k^\star$ for $k = 1, \ldots, N_\star$, ordered from start to goal, where $N_\star$ is the number of waypoints.

Algorithm 1 Waypoint reduction algorithm.

```
procedure REDUCE
    i ← N_⋆; P ← InitializePath(p*_i)
    do
        for j = 1 to i − 1 do
            if ¬Collision(p*_i, p*_j) then
                AddPoint(P, p*_j)
                i ← j
                break
    while i > 1
```

4.2 Waypoint connection

The waypoints in the reduced path $\boldsymbol{p}_k \in \mathbb{P}$ are connected with straight segments and circle arcs to increase geometric feasibility. This is done by calculating the parameters of a circle based on a radius of acceptance $R_{\text{acc}} > 0$. The result is a path with discontinuous turn rate since the turn rate of such a curve will experience jumps at the beginning and end of the circle arcs. However, if the circle arcs have a turning radius $R_{\text{turn}} > 0$ larger than the minimum turning radius of the ASV $R_{\text{turn,min}} > 0$, the resulting geometry of the path can be followed tightly. Additional information about such a path waypoint connection is available in (Fossen, 2011).

For each straight segment, the turn rate is 0. For the circle arcs, the turn rate is ${}^{u_{\text{nom}}}/_{R_{\text{turn},k}}$, where $R_{\text{turn},k} > 0$ is the turning radius for arc k. The tangent angles for the straight segments are $\gamma_k \in \text{S}$, and for the circle arcs, the tangent angles move between γ_k and γ_{k+1}, depending on how far along the curve it is evaluated.

Using this information, we can concatenate a path consisting of alternations of straights and circle arcs, and construct a path function parametrized by length with position

$$\boldsymbol{p}_g : [0, L_{\text{path}}] \to \mathbb{R}^2 \,, \tag{4a}$$

where $L_{\text{path}} > 0$ is the total length of the path. Functions for path tangential angle and turn rate are also constructed:

$$\gamma_g : [0, L_{\text{path}}] \to \text{S} \,, \text{ and} \tag{4b}$$

$$r_g : [0, L_{\text{path}}] \to \mathbb{R} \,, \tag{4c}$$

respectively. These functions are subscripted by $(\cdot)_g$ to indicate that they are based on the path geometry.

4.3 Adding temporal information

After obtaining an arc-length parametrized path we add temporal information by assuming a constant surge velocity u_{nom}, a sway velocity v of zero, and piecewise constant yaw rate r. The nominal surge velocity is determined by $u_{\text{nom}} = \frac{L_{\text{path}}}{t_{\max}}$, where $t_{\max} > 0$ is the tunable time to complete the trajectory, which is valid on $t \in [0, t_{\max}]$. The distance traveled will be $L(t) = u_{\text{nom}} \cdot t$, and the states will then have trajectories

$$\begin{bmatrix} x_w(t) & y_w(t) \end{bmatrix}^\top = \boldsymbol{p}_g(L(t)) \tag{5a}$$

$$\psi_w(t) = \gamma_g(L(t)) \tag{5b}$$

$$u_w(t) = u_{\text{nom}} \tag{5c}$$

$$v_w(t) = 0 \tag{5d}$$

$$r_w(t) = r_g(L(t)) \,. \tag{5e}$$

The input trajectory is set to the constant values

$$\tau_{X,w}(t) = \tau_{X,ss}, \quad \tau_{N,w}(t) = 0 \tag{6}$$

where $\tau_{X,ss} \in \mathbb{R}$ is calculated as the steady-state value required to maintain nominal forward velocity u_{nom}. The trajectories are subscripted by $(\cdot)_w$ to indicate that they will be used for warm-starting the OCP in Step 3.

The resulting trajectory is not dynamically feasible according to (1) but will be used as an initial guess for the OCP solver, described in the next section. The trajectory is collected in the following vectors:

$$\boldsymbol{x}_w(t) = \begin{bmatrix} x_w(t) \\ y_w(t) \\ \psi_w(t) \\ u_w(t) \\ v_w(t) \\ r_w(t) \end{bmatrix} \quad \boldsymbol{u}_w(t) = \begin{bmatrix} \tau_{X,w}(t) \\ \tau_{N,w}(t) \end{bmatrix} \quad \forall\, t \in [0, t_{\max}] \,. \tag{7}$$

The goal of the method described in this paper is to find a trajectory of states and inputs that minimizes a cost functional $J(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$:

$$J(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) = \int_0^{t_{\max}} F(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \, \mathrm{d}\tau \,, \tag{8}$$

which is dependent on a cost-to-go function $F(\boldsymbol{x}, \boldsymbol{u})$. This function may be selected to find e.g. the trajectory that minimizes energy usage. The initial guess for the cost trajectory $J_w(\cdot)$ at time t is determined by

$$J_w(t) = \int_0^t F(\boldsymbol{x}_w(\tau), \boldsymbol{u}_w(\tau)) \, \mathrm{d}\tau \,. \tag{9}$$

5. STEP 3: OPTIMAL CONTROL

Optimal control is used to make feasible and optimize the trajectory provided by Step 2. An OCP is formulated as

$$\min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \int_0^{t_{\max}} F(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \, \mathrm{d}\tau \tag{10a}$$

subject to

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \;\forall t \in [0, t_{\max}] \tag{10b}$$

$$\boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \boldsymbol{0} \;\forall t \in [0, t_{\max}] \tag{10c}$$

$$\boldsymbol{e}(\boldsymbol{x}(0), \boldsymbol{x}(t_{\max})) = \boldsymbol{0} \,. \tag{10d}$$

The solution of this OCP gives a trajectory of states $\boldsymbol{x}(\cdot)$ and inputs $\boldsymbol{u}(\cdot)$ that minimizes (8).

5.1 Cost functional

The cost functional described in (8) is dependent on the cost-to-go function $F(\boldsymbol{x}, \boldsymbol{u})$. This function may be adjusted and structured according to the desired sense of optimality. Our aim is a trajectory which is optimized for energy usage, as well as performing readily observable maneuvers, as is required by International Regulations for Preventing

Collisions at Sea (COLREGs) Rule 8. This results in a two-part cost-to-go function:

$$F(\boldsymbol{x}, \boldsymbol{u}) = K_e F_e(\boldsymbol{x}, \boldsymbol{u}) + K_t F_t(\boldsymbol{x}) \,, \tag{11}$$

with tuning parameters $K_e, K_t > 0$. The first term penalizes energy usage and describes work done by the actuators:

$$F_e(\boldsymbol{x}, \boldsymbol{u}) = |u \cdot \tau_X| + |r \cdot \tau_N| \,. \tag{12}$$

The second term is a disproportionate penalization on turn-rate r, which prefers readily observable turns performed with high turn-rate. The function has the form

$$F_t(\boldsymbol{x}) = \left(a_t r^2 + (1 - e^{-\frac{r^2}{b_t}}) \right) \frac{1}{F_{t,max}} \,, \tag{13}$$

where

$$F_{t,max} = a_t r_{\max}^2 + (1 - e^{-\frac{r_{\max}^2}{b_t}}) \,, \tag{14}$$

and $r_{\max} > 0$ is the ASV's maximum yaw rate. The tuning parameters $a_t > 0$ and $b_t > 0$ shape the penalization to prefer higher or lower turn-rate, which is an idea obtained from (Eriksen and Breivik, 2017).

5.2 Obstacles

Obstacles are encoded as elliptic inequalities in (10c). The basis for one elliptic obstacle is

$$\left(\frac{x - x_c}{x_a} \right)^2 + \left(\frac{y - y_c}{y_a} \right)^2 \geq 1 \,, \tag{15}$$

where x_c and y_c describe the ellipse center and x_a and y_a describe the sizes of the two elliptic axes. The ellipses are rotated by α, which is the angle between the global x-axis and the direction of x_a. The resulting inequality becomes

$$\begin{aligned} g_o(x, y, x_c, y_c, x_a, y_a, \alpha) = \\ - \log \Bigg[\left(\frac{(x - x_c)\cos\alpha + (y - y_c)\sin\alpha}{x_a} \right)^2 \\ + \left(\frac{-(x - x_c)\sin\alpha + (y - y_c)\cos\alpha}{y_a} \right)^2 + \epsilon \Bigg] \\ + \log(1 + \epsilon) \leq 0 \,, \end{aligned} \tag{16}$$

where a small value $\varepsilon > 0$ is added to deal with feasibility issues as $x \to x_c$ and $y \to y_c$, and the logarithmic function is used to reduce numerical sizes, without changing the inequality. The same function is used in (Bitar et al., 2019).

5.3 NLP transcription

A multiple-shooting approach is used to transcribe the OCP into an NLP:

$$\min_{\boldsymbol{w}} \phi(\boldsymbol{w}) \tag{17a}$$

subject to

$$\boldsymbol{g}_{lb} \leq \boldsymbol{g}(\boldsymbol{w}) \leq \boldsymbol{g}_{ub} \tag{17b}$$

$$\boldsymbol{w}_{lb} \leq \boldsymbol{w} \leq \boldsymbol{w}_{ub} \,. \tag{17c}$$

The dynamics are discretized into N_{ocp} steps in time, with step length $h = {}^{t_{\max}}/_{N_{ocp}}$. The decision variables $\boldsymbol{w}$ consist of the state variables $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$, $k = 0, 1, \ldots, N_{ocp}$, the accumulated costs $J_k = J(t_k)$, $k = 0, 1, \ldots, N_{ocp}$, where

$$J(t) = \int_0^t F(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \, \mathrm{d}\tau \,, \tag{18}$$

and the control inputs $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$, $k = 0, 1, \ldots, N_{ocp} - 1$:

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{z}_0^\top & \boldsymbol{u}_0^\top & \boldsymbol{z}_1^\top & \ldots & \boldsymbol{u}_{N_{ocp}-1}^\top & \boldsymbol{z}_{N_{ocp}}^\top \end{bmatrix}^\top , \tag{19}$$

where $\boldsymbol{z}_k = [\boldsymbol{x}_k^\top, J_k]^\top$.

The cost function (17a) approximates (10a) and is

$$\phi(\boldsymbol{w}) = J_{N_{ocp}} \,. \tag{20}$$

The constraints (17b) are used to satisfy shooting constraints, as well as the collision avoidance constraints. For the shooting constraints, we construct a discrete representation of the dynamics (10b) as well as the integral (18) using a RK4 scheme with K_{ocp} steps. We define the discrete version of (10b) augmented with the time derivative of (18) as

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}(\boldsymbol{z}_k, \boldsymbol{u}_k) \,, \tag{21}$$

and construct the shooting constraints

$$\boldsymbol{g}_s(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{z}_1 - \boldsymbol{F}(\boldsymbol{z}_0, \boldsymbol{u}_0) \\ \vdots \\ \boldsymbol{z}_{N_{ocp}} - \boldsymbol{F}(\boldsymbol{z}_{N_{ocp}-1}, \boldsymbol{u}_{N_{ocp}-1}) \end{bmatrix} , \tag{22}$$

with associated lower and upper bounds

$$\boldsymbol{g}_{s,lb} = \boldsymbol{g}_{s,ub} = \boldsymbol{0}_{(n+1)\cdot N_{ocp}} \,. \tag{23}$$

For obstacles $i = 1, 2, \ldots, N_o$, we avoid collisions by satisfying the inequality constraint

$$g_o(x_k, y_k, x_{c,i}, y_{c,i}, a_i, b_i, \alpha_i) \leq 0 \,, \tag{24}$$

where $x_k = x(t_k)$ and $y_k = y(t_k)$ for $k = 1, 2, \ldots, N_{ocp}$. We create a vector for all our obstacles in a single time step:

$$\bar{\boldsymbol{g}}_o(\boldsymbol{x}_k) = \begin{bmatrix} g_o(x_k, y_k, x_{c,1}, y_{c,1}, a_1, b_1, \alpha_1) \\ g_o(x_k, y_k, x_{c,2}, y_{c,2}, a_2, b_2, \alpha_2) \\ \vdots \\ g_o(x_k, y_k, x_{c,N_o}, y_{c,N_o}, a_{N_o}, b_{N_o}, \alpha_{N_o}) \end{bmatrix} . \tag{25}$$

Obstacle constraints for all time steps are gathered in

$$\boldsymbol{g}_o(\boldsymbol{w}) = \begin{bmatrix} \bar{\boldsymbol{g}}_o(\boldsymbol{x}_0) \\ \bar{\boldsymbol{g}}_o(\boldsymbol{x}_1) \\ \vdots \\ \bar{\boldsymbol{g}}_o(\boldsymbol{x}_{N_{ocp}-1}) \end{bmatrix} \tag{26}$$

with associated lower and upper bounds

$$\boldsymbol{g}_{o,lb} = -\boldsymbol{\infty}_{N_o \cdot N_{ocp}} \quad \text{and} \quad \boldsymbol{g}_{o,ub} = \boldsymbol{0}_{N_o \cdot N_{ocp}} \,. \tag{27}$$

The nonlinear inequality constraints (17b) are completed as

$$\boldsymbol{g}_{lb} = \begin{bmatrix} \boldsymbol{g}_{s,lb} \\ \boldsymbol{g}_{o,lb} \end{bmatrix} , \; \boldsymbol{g}(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{g}_s(\boldsymbol{w}) \\ \boldsymbol{g}_o(\boldsymbol{w}) \end{bmatrix} , \; \boldsymbol{g}_{ub} = \begin{bmatrix} \boldsymbol{g}_{s,ub} \\ \boldsymbol{g}_{o,ub} \end{bmatrix} . \tag{28}$$

The decision variable bounds (17c) are used to satisfy constant state and input constraints, as well as boundary conditions (10d). The bounds are

$$\boldsymbol{w}_{lb}^\top = \begin{bmatrix} \boldsymbol{z}_{s,lb}^\top & \boldsymbol{u}_{lb}^\top & \boldsymbol{z}_{lb}^\top & \boldsymbol{u}_{lb}^\top & \ldots & \boldsymbol{u}_{lb}^\top & \boldsymbol{z}_{f,lb}^\top \end{bmatrix} \tag{29a}$$

$$\boldsymbol{w}_{ub}^\top = \begin{bmatrix} \boldsymbol{z}_{s,ub}^\top & \boldsymbol{u}_{ub}^\top & \boldsymbol{z}_{ub}^\top & \boldsymbol{u}_{ub}^\top & \ldots & \boldsymbol{u}_{ub}^\top & \boldsymbol{z}_{f,ub}^\top \end{bmatrix} , \tag{29b}$$

Table 1. Algorithm step explanation.

Step	Parametrized by	Dynamic feasibility	Optimality
1	Length	None, piecewise linear	Shortest piecewise linear path
2	Time	Discontinuous yaw rate r	None
3	Time	Adheres to (1)	Energy and COLREGs Rule 8

where

$$\boldsymbol{z}_{s,lb} = \begin{bmatrix} x_s & y_s & \psi_{lb} & u_{r,s} & 0 & 0 & 0 \end{bmatrix}^\top \tag{30a}$$

$$\boldsymbol{z}_{s,ub} = \begin{bmatrix} x_s & y_s & \psi_{ub} & u_{r,s} & 0 & 0 & 0 \end{bmatrix}^\top \tag{30b}$$

$$\boldsymbol{z}_{f,lb} = \begin{bmatrix} x_f & y_f & \psi_{lb} & u_{r,lb} & 0 & 0 & 0 \end{bmatrix}^\top \tag{30c}$$

$$\boldsymbol{z}_{f,ub} = \begin{bmatrix} x_f & y_f & \psi_{ub} & u_{r,ub} & 0 & 0 & \infty \end{bmatrix}^\top \tag{30d}$$

$$\boldsymbol{z}_{lb} = \begin{bmatrix} x_{lb} & y_{lb} & \psi_{lb} & u_{r,lb} & v_{lb} & r_{lb} & 0 \end{bmatrix}^\top \tag{30e}$$

$$\boldsymbol{z}_{ub} = \begin{bmatrix} x_{ub} & y_{ub} & \psi_{ub} & u_{r,ub} & v_{ub} & r_{ub} & \infty \end{bmatrix}^\top \tag{30f}$$

$$\boldsymbol{u}_{lb} = \begin{bmatrix} X_{lb} & N_{lb} \end{bmatrix}^\top \tag{30g}$$

$$\boldsymbol{u}_{ub} = \begin{bmatrix} X_{ub} & N_{ub} \end{bmatrix}^\top, \tag{30h}$$

and where values subscripted with $(\cdot)_s$ represent initial conditions, $(\cdot)_f$ the final conditions, and $(\cdot)_{lb}$ and $(\cdot)_{ub}$ represent lower and upper bounds, respectively.

5.4 Initial guess and solver

The trajectories $\boldsymbol{x}_w(\cdot)$, $\boldsymbol{u}_w(\cdot)$ and $J_w(\cdot)$ from Section 5.4 are used as an initial guess to warm-start the NLP. These trajectories are sampled at the time steps t_k, $k = 0, \ldots, N_{ocp}$ using interpolation, and shaped into the form of the decision vector $\boldsymbol{w}$ (19), providing the initial guess $\boldsymbol{w}_0$.

The NLP as defined by (17) is solved by the interior-point method Ipopt (Wächter and Biegler, 2005) using Casadi (Andersson et al., 2018) for Matlab.

5.5 Algorithm summary

The pipelined algorithm is summarized by the steps in Table 1, where the properties of each step are highlighted in terms of parametrization, feasibility according to (1) and optimality.

While Step 1 gives the shortest piecewise linear path, it is parametrized by length, and will not be dynamically feasible for warm-starting the OCP in Step 3. Step 2 connects the waypoints with circle arcs and adds artificial dynamics, which moves us closer to a dynamically feasible trajectory. However, we lose the optimality of the shortest path with this modification, and the yaw rate is discontinuous, which is not possible according to (1). This trajectory is usable as an initial guess for Step 3, which converges to a trajectory that adheres to (1), and adds optimality according to (10a).

6. SIMULATION SCENARIOS AND RESULTS

The scenario selected for testing our planning method is Sjernarøy north of Stavanger, Norway, near 59.25°N and 5.83°E. A map of this scenario is shown in Figure 3. The scenario has many possible routes between the start and goal positions, including routes that go outside the islands, and the narrow passage between the islands. The narrow passage is the shortest path, and one could claim that in the absence of disturbances, this shortest path is also the most energy efficient. However, since the problem of finding this path is non-convex and resembles an integer problem, the OCP alone would struggle to find the shortest path. We use the algorithm parameters presented in Table 2.

Table 2. Parameter values.

Param.	Val.	Param.	Val.
Δd	50 [m]	$t_{\max}$	2200 [s]
N_{ocp}	1000	K_{ocp}	1
K_e	$3.5 \cdot 10^{-4}$ [J^{-1}]	K_t	800
a_t	112 [s^2/rad^2]	b_t	$6.25 \cdot 10^{-5}$ [rad^2/s^2]
R_{acc}	10 [m]	$R_{\text{turn,min}}$	24.5 [m]
$r_{\max}$	40 [°/s]		

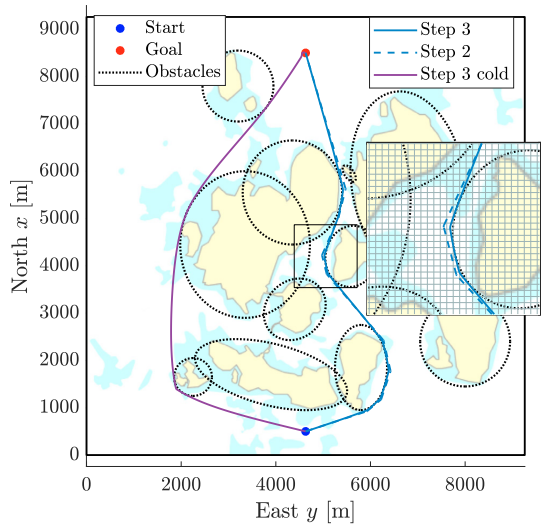


Fig. 3. Map showing the scenario used for planning, with multiple elliptical obstacle boundaries surrounding the small islands. Trajectories after steps 2 and 3 are plotted. A cold-started solution is also included.

To benchmark our planning algorithm, we apply it to the scenario illustrated in Figure 3 in Matlab on a laptop with an Intel Core i7-7700HQ processor. For comparison, we also apply the OCP to the same scenario without an initial guess, i.e. cold starting Step 3. Solutions from these two methods will be dynamically feasible trajectories with different routings to reach the goal position. We use metrics of total cost and run times to compare the algorithms. These metrics will also be applied to the trajectory after Step 2. This trajectory is not dynamically feasible according to (1) but can tell us how the smoothed A* trajectory performs without optimization.

The resulting trajectories are plotted on top of the scenario in Figure 3. We see that the initial guess goes through the narrow passage between the islands and that the warm-

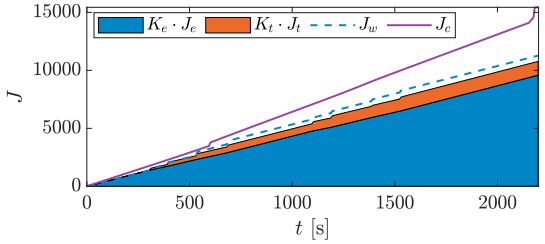


Fig. 4. Cost functional development along both the optimized trajectory and the initial guess. The optimized trajectory shows the cost split into contributions from energy optimization and observable maneuvers. Also, the cost of the cold-started OCP is denoted J_c.

started OCP finds a solution along the same route. As expected, the cold-started OCP goes outside the passage and finds a longer solution. A zoomed inset in Figure 3 shows how the OCP is able to produce readily observable maneuvers by making sharp turns around the obstacle boundaries. The inset also includes the grid used by Step 1.

Figure 4 shows us the cost functional develops along the trajectories of the warm-started OCP ($K_e \cdot J_e + K_t \cdot J_t$), the initial guess ($J_w$) and the cold-started OCP (J_c). Table 3 shows the results at $t = t_{\max}$ for the three methods. We see the scaled total cost as calculated by (10a) and (11), as well as the energy cost calculated by (12). An improvement of 30 % is obtained by warm-starting the OCP compared to cold starting it, explained by the shorter route selection. The warm-started OCP is also able to improve on the dynamically infeasible initial guess by 4 %.

Table 3 also shows the run times of the three methods. Since the initial guess alone does not perform any iterative optimization, it has the lowest run time. The warm-started method spends 27 s in total to find an optimized solution to the path planning problem, including 21 s spent solving the OCP. This is an improvement of 84 % compared to the cold-started OCP which spends approximately three minutes. The run-time cost of obtaining a feasible trajectory via optimal control is significant compared to performing A* and dynamic generation alone.

The state trajectories for the initial guess and warm-started OCP are shown in Figure 5. From the heading angle plot, we see that ψ performs jumps of more than 30°, which is a clear indication of intent to other vessels, even in situations with restricted visibility (Cockcroft and Lameijer, 2004). This is further observed in the yaw rate state r, where instead of having long turns with low yaw rate magnitude, we have abrupt turns with high-valued r. This is shown more clearly in Figure 6, which zooms in on a selected time interval.

7. CONCLUSION

We have developed and demonstrated a pipelined trajectory planning algorithm that exploits the speed and global properties of an A* search with the optimality of an OCP solver. The results from Section 6 show that using the initial guess provided by a smoothed A* path in an OCP significantly improves both run time and optimality compared to a cold-started OCP alone. Performing

Table 3. Scenario results.

	Warm started Step 3	Cold started Step 3	Step 2
Feasible	Yes	Yes	No
Scaled total cost (J)	$\mathbf{1.08 \cdot 10^4}$	$1.54 \cdot 10^4$	$1.13 \cdot 10^4$
Unscaled energy cost (J_e)	$\mathbf{2.74 \cdot 10^7}$ **[J]**	$3.94 \cdot 10^7$ [J]	$2.84 \cdot 10^7$ [J]
Total run time	26.7 [s]	174 [s]	**5.7 [s]**
Step 1 run time	3.4 [s]	-	3.4 [s]
Step 2 run time	2.2 [s]	-	2.2 [s]
Step 3 run time	21.1 [s]	174 [s]	-
Step 3 iterations	**58**	549	-

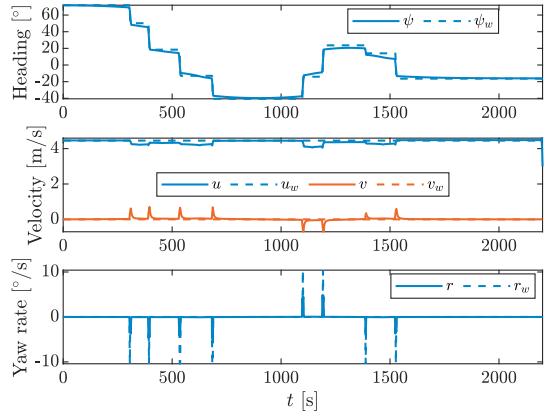


Fig. 5. State values for heading, velocities and yaw rate for both the optimized trajectory and the initial guess.

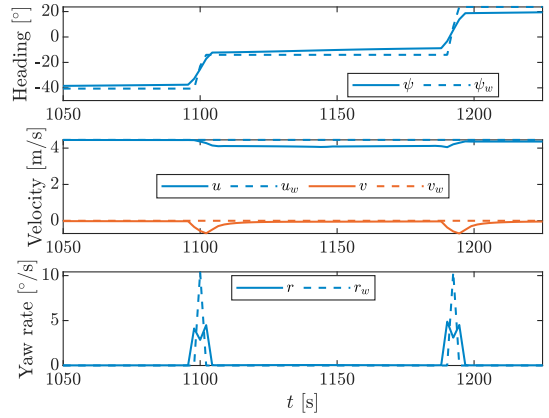


Fig. 6. Zoomed-in section of Figure 5.

optimization on the A* path significantly increases run time but will find a feasible locally optimal trajectory, as opposed to A* alone.

Qualitatively, the developed method is complete in terms of the *shortest path*, since this is the geometric objective of the A* implementation. This is dependent on the discretization of the map, since using larger grid spacing to reduce run time removes narrow passages from the solution space. Using a different discretization scheme such as

e.g. Voronoi diagrams may guarantee a complete solution space. The developed method is also locally optimal in the sense of the provided objective, which is a combination of energy consumption and readily observable maneuvers in our case. The optimality is provided by the implemented OCP which alone is not able to find the global optimum, demonstrated by the cold-started result in Figure 3. However, the OCP warm-started by the shortest path found by the A* method is at least locally optimal and may be close to the global optimum, since, in the absence of disturbances, the shortest path is also the one that requires the least energy. In addition to improving optimality of the A* result, the OCP adds feasibility, unlike the A* consideration which is purely geometric. Using this warm-starting scheme is that the OCP will lock into one routing alternative. Depending on the desired sense of optimality, this may not be the desired solution, which is a disadvantage to some use cases.

The algorithm presented here has been used in a hybrid collision avoidance architecture in (Eriksen et al., 2019), where it is extended to include disturbances in the form of ocean currents.

Further work on this topic includes:

- Implementing a more general obstacle representation to handle a wider range of map representations. E.g. the obstacle representation in (Zhang et al., 2018) handles convex polygons as smooth inequality conditions.
- Improvements on the map discretization scheme are also desirable to reduce computational time of the A* algorithm while preserving completeness of the solution space.
- Additionally, an OCP representation that is parametrized by straight lines between waypoints in combination with full-state dynamics may be advantageous to inherently produce COLREGs-compliant trajectories.

ACKNOWLEDGEMENTS

This work is funded by the Research Council of Norway and Innovation Norway with project number 269116. The work is also supported by the Centres of Excellence funding scheme with project number 223254.

REFERENCES

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Bitar, G., Breivik, M., and Lekkas, A.M. (2018). Energy-optimized path planning for autonomous ferries. In *Proc. of the 11th IFAC CAMS, Opatija, Croatia*, 389–394.

Bitar, G., Eriksen, B.-O.H., Lekkas, A.M., and Breivik, M. (2019). Energy-optimized hybrid collision avoidance for ASVs. In *Proc. of the 17th ECC, Naples, Italy.*

Cockcroft, A.N. and Lameijer, J.N.F. (2004). *A Guide to the Collision Avoidance Rules.* Elsevier Butterworth-Heinemann.

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010). Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5), 485–501.

Eriksen, B.-O.H., Bitar, G., Breivik, M., and Lekkas, A.M. (2019). Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17. arXiv:1907.00198 [eess.SY]. Submitted to *Frontiers in Robotics and AI.*

Eriksen, B.-O.H. and Breivik, M. (2017). MPC-based mid-level collision avoidance for ASVs using nonlinear programming. In *Proc. of the IEEE CCTA, Mauna Lani, HI, USA*, 766–772.

Fossen, T.I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control.* Wiley-Blackwell.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Jallal, C. (2018). Rolls-Royce and Finferries demonstrate world's first fully autonomous ferry. *Maritime Digitalisation & Communications.* URL `https://www.marinemec.com/news/view,rollsroyce-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry_56102.htm`. Accessed 2019-04-11.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 566–580.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.

Loe, Ø.A.G. (2008). *Collision Avoidance for Unmanned Surface Vehicles.* Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Ross, I.M. and Karpenko, M. (2012). A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2), 182–197.

Wächter, A. and Biegler, L.T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106, 25–57.

Zhang, X., Liniger, A., Sakai, A., and Borrelli, F. (2018). Autonomous parking using optimization-based collision avoidance. In *Proc. of the IEEE CDC, Miami Beach, FL, USA*, 4327–4332.

Paper D Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17

Published paper by B.-O. H. Eriksen, **G. Bitar**, M. Breivik, and A. M. Lekkas. "Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17". In: *Frontiers in Robotics and AI* 7 (2020). DOI: 10.3389/frobt.2020.00011. arXiv: 1907.00198 [eess.SY].

Bibliography entry [11].

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 11 February 2020
doi: 10.3389/frobt.2020.00011

Hybrid Collision Avoidance for ASVs Compliant With COLREGs Rules 8 and 13–17

Bjørn-Olav H. Eriksen, Glenn Bitar, Morten Breivik and Anastasios M. Lekkas*

Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway

This paper presents a three-layered hybrid collision avoidance (COLAV) system for autonomous surface vehicles, compliant with rules 8 and 13–17 of the International Regulations for Preventing Collisions at Sea (COLREGs). The COLAV system consists of a high-level planner producing an energy-optimized trajectory, a model-predictive-control-based mid-level COLAV algorithm considering moving obstacles and the COLREGs, and the branching-course model predictive control algorithm for short-term COLAV handling emergency situations in accordance with the COLREGs. Previously developed algorithms by the authors are used for the high-level planner and short-term COLAV, while we in this paper further develop the mid-level algorithm to make it comply with COLREGs rules 13–17. This includes developing a state machine for classifying obstacle vessels using a combination of the geometrical situation, the distance and time to the closest point of approach (CPA) and a new CPA-like measure. The performance of the hybrid COLAV system is tested through numerical simulations for three scenarios representing a range of different challenges, including multi-obstacle situations with multiple simultaneously active COLREGs rules, and also obstacles ignoring the COLREGs. The COLAV system avoids collision in all the scenarios, and follows the energy-optimized trajectory when the obstacles do not interfere with it.

Keywords: hybrid collision avoidance, autonomous surface vehicle (ASV), COLREGs, COLREGs compliant, model predictive control (MPC), energy-optimized control

OPEN ACCESS

Edited by:
Marco Bibuli,
Italian National Research Council, Italy

Reviewed by:
Jingfu Jin,
General Motors, United States
Rafal Szlapczynski,
Gdansk University of Technology, Poland

****Correspondence:***
Bjørn-Olav H. Eriksen
bjorn-olav.holtung.eriksen@ntnu.no

Specialty section:
This article was submitted to Robotic Control Systems, a section of the journal Frontiers in Robotics and AI

Received: *28 May 2019*
Accepted: *20 January 2020*
Published: *11 February 2020*

Citation:
Eriksen B-OH, Bitar G, Breivik M and Lekkas AM (2020) Hybrid Collision Avoidance for ASVs Compliant With COLREGs Rules 8 and 13–17. Front. Robot. AI 7:11. doi: 10.3389/frobt.2020.00011

1. INTRODUCTION

Motivated by the potential for reduced costs and increased safety, the maritime industry is rapidly moving toward autonomous operations. Following groundbreaking advances in the automotive industry, many sectors within the maritime industry are considering the benefits of autonomy, which includes more environmentally friendly operations. For instance, the agricultural chemical company *Yara* together with the maritime technology supplier *Kongsberg Maritime* are developing the electrical autonomous container vessel *Yara Birkeland*, which aims to replace 40,000 yearly truck journeys in urban eastern Norway[1]. Another example is the world's first autonomous car ferry, *Falco*, developed by *Rolls-Royce* (recently bought by Kongsberg Maritime) and *Finferries*. In

[1]https://www.wsj.com/articles/norway-takes-lead-in-race-to-build-autonomous-cargo-ships-1500721202 (accessed May 22, 2019).

2018, *Falco* navigated autonomously between two ports in Finland[2]. Reports state that in excess of 75 % of maritime accidents are due to human errors (Chauvin, 2011; Levander, 2017), indicating that there is also a potential for increased safety in addition to the economical and environmental benefits.

An obvious prerequisite for autonomous ship operations is the development of robust and well-functioning COLAV systems. In addition to generating collision-free maneuvers, a COLAV system must adhere to the "rules of the road" of the oceans, i.e., the COLREGs (Cockcroft and Lameijer, 2004). These rules are written for human ship operators and include qualitative requirements on how to perform safe and readily observable maneuvers. Part B of the COLREGs concern steering and sailing, and includes the following rules that are the most relevant to a motion control system:

Rule 8 Requires maneuvers to be readily observable and to be done in ample time.

Rules 13–15 Describe the maneuvers to perform in cases of overtaking, head-on and crossing situations. Participants in crossing situations are defined by the terms *give-way* and *stand-on* vessels.

Rule 16 Requires that a give-way vessel must take early and substantial action to keep clear of the stand-on vessel.

Rule 17 Consists of two main parts. The first part requires a stand-on vessel to maintain its course and speed, while the second part allows/requires[3] a stand-on vessel to take action to avoid collision if the give-way vessel is not taking action.

Since the rules are written for humans, with few quantitative figures, a challenge for autonomous operation is to quantify them into behaviors that can be executed algorithmically. The focus of the work in this paper is to do that, and to design a hybrid COLAV system that performs motion planning and generates maneuvers in compliance to rules 8 and 13–17 of the COLREGs.

A number of COLAV approaches considering the COLREGs have been proposed in the past. This includes algorithms using simulation-based model predictive control (Hagen et al., 2018), velocity obstacles (Kuwata et al., 2014), rule-based repairing A* (Campbell et al., 2014), and interval programming (Benjamin et al., 2006). All these approaches are single-layer approaches, where one algorithm solves the complete COLAV problem.

Another approach to the COLAV problem is to use a hybrid architecture, where the task of planning an obstacle-free path or trajectory, complying with the COLREGs and ultimately performing safe maneuvers is divided into layers in a control hierarchy. The idea of hybrid architectures is to divide the subtasks of the COLAV problem into multiple algorithms, exploiting their complementary strengths. This also has the side effect of making it easier for human operators or supervisors to understand the system. Most single-layer algorithms use sample-based approaches that consider a finite number of discrete control inputs, as opposed to conventional gradient-based search algorithms. The reason for this is that many gradient-based algorithms are not sufficiently numerically robust, not allowing a COLAV system to solely rely on such an algorithm. This issue can be handled in hybrid architectures, constrained by the bottom-level algorithm being numerically robust and able to handle extraordinary situations where the other algorithms fail. Hence, hybrid architectures also allows using gradient-based algorithms, which are able to solve problems with large search spaces more efficiently than sample-based algorithms. The works by Loe (2008) and Švec et al. (2013) are examples of two-layered hybrid COLAV architectures. The top layers perform trajectory planning among static obstacles, while the bottom layers perform moving obstacle avoidance in compliance with COLREGs rules 13–16. Casalino et al. (2009) presents a three-layered hybrid COLAV system where the top layer also performs trajectory planning amongst static obstacles. The middle layer avoids moving obstacles, while the bottom layer implements safety functions for handling cases where the two

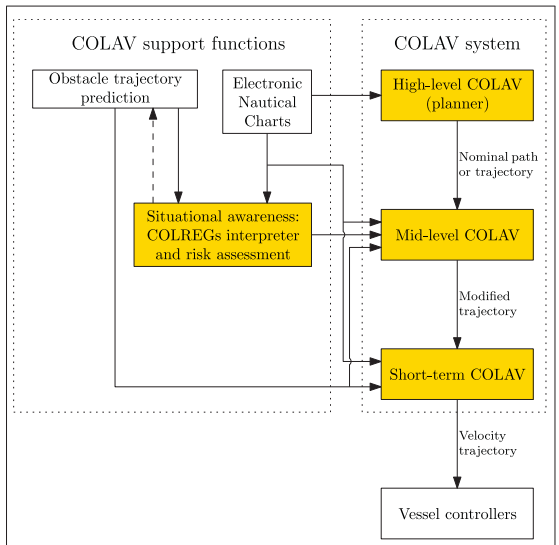


FIGURE 1 | Hybrid architecture with three COLAV layers, where the highlighted functions mark the areas of interest in this article. The COLAV system consists of a high-level planner, a mid-level COLAV algorithm and a short-term COLAV algorithm. The COLAV system is supported by data from electronic nautical charts, represented in a suitable manner for the algorithms, as well as situational awareness functions that track and predict obstacles, interpret the COLREGs and perform risk assessment.

[2]https://www.marinemec.com/news/view,rollsroyce-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry_56102.htm (accessed April 11, 2019).

[3]Rule 17 *allows* the stand-on vessel to maneuver when it becomes apparent that the give-way vessel does maneuver to avoid collision. If the vessels are so close that the give-way vessel cannot avoid collision by itself, Rule 17 *requires* the stand-on vessel to maneuver.

other layers fail. This approach does, however, not consider the COLREGs.

Figure 1 shows a three-layered hybrid COLAV system for an autonomous surface vehicle (ASV). The authors have previously worked extensively on different components of this architecture. Examples include high-level COLAV algorithms (Bitar et al., 2018, 2019b), a mid-level algorithm (Eriksen and Breivik, 2017b; Bitar et al., 2019a), short-term algorithms (Eriksen et al., 2018, 2019; Eriksen and Breivik, 2019) and the development of high-performance vessel controllers (Eriksen and Breivik, 2017a, 2018).

In this paper, we demonstrate the three-layered hybrid COLAV shown in **Figure 1** by combining and extending the COLAV algorithms developed in Eriksen and Breivik (2017b, 2019), Bitar et al. (2019a,b), Eriksen et al. (2019). The high-level planner has a long temporal horizon, and finds an energy-optimized nominal trajectory from an initial to a goal position. It considers static obstacles, which may include bathymetric constraints. Since the high-level planner only considers static information, it is intended to be run offline, but it can also be run online, for instance if new static obstacles are detected. The mid-level algorithm attempts to follow the nominal trajectory, while performing COLAV of static and moving obstacles in compliance with COLREGs rules 8, 13–16, and the first part of Rule 17. The mid-level algorithm is run periodically with a shorter temporal horizon than the high-level algorithm, and produces a modified trajectory which is passed to the short-term layer. Both the high-level and mid-level algorithms use gradient-based optimization. The short-term algorithm attempts to follow the modified trajectory, while it in compliance with the second part of Rule 17 handles situations where obstacles ignore the COLREGs. This algorithm also handles other emergency situations, and uses sample-based optimization to achieve a high level of robustness, ensuring safe operation if the mid-level algorithm fails to find a solution. The following list summarizes our contributions:

- The high-level planner from Bitar et al. (2019b) is modified to include the mathematical model of the *Telemetron* ASV in Bitar et al. (2019a), including ocean currents.
- The development of a state-machine-based COLREGs interpretation scheme.
- The mid-level COLAV from Bitar et al. (2019a) is modified to include rules 13–16 and the first part of Rule 17.
- The branching-course model predictive control (BC-MPC) algorithm for short-term COLAV is modified to reduce oscillatory behavior in turns.
- The three-layered COLAV system is verified in simulations and shown to be compliant with rules 8 and 13–17.

The rest of the paper has the following structure: The mathematical model of the ASV *Telemetron* is described in section 2. The high-level planner, mid-level and short-term COLAV algorithms are described in sections 3–5, respectively. In section 6 we present and discuss the simulation scenarios and results, and we conclude the paper in section 7.

FIGURE 2 | The Telemetron ASV, designed for both manned and unmanned operations. Courtesy of Maritime Robotics.

2. ASV MODELING

The vessel of interest in this article is the Telemetron ASV, which is owned and operated by the Norwegian company Maritime Robotics and shown in **Figure 2**. The Telemetron ASV is a high-speed dual-use vessel propelled by a steerable outboard engine, capable of speeds up to 18 m/s.

Eriksen and Breivik (2017a) presents a model of the Telemetron ASV, which is extended to include ocean currents in Bitar et al. (2019a). The model has the form

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}(\psi)\boldsymbol{x}_r + \begin{bmatrix} \boldsymbol{V}_c^\top & 0 \end{bmatrix}^\top \\ \boldsymbol{M}(\boldsymbol{x}_r)\dot{\boldsymbol{x}}_r + \boldsymbol{\sigma}(\boldsymbol{x}_r) &= \boldsymbol{\tau}\,, \end{aligned} \tag{1}$$

where $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times S$ is the vessel pose and $\boldsymbol{V}_c = [V_x, V_y]^\top$ describes the ocean current, both in the Earth-fixed North-East-Down frame $\{n\}$. The vector $\boldsymbol{x}_r = [u_r, r]^\top \in \mathbb{X}_r \subset \mathbb{R}^2$ is the vessel velocity under the assumption of zero relative sway motion (Bitar et al., 2019a), where the set $\mathbb{X}_r$ describes the vessel-feasible steady-state velocities where (1) is valid. The transformation matrix $\boldsymbol{R}(\psi)$ is given by the heading $\psi \in S$ as

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix}, \tag{2}$$

while $r \in \mathbb{R}$ describes the vessel yaw-rate. The matrix $\boldsymbol{M}(\boldsymbol{x}_r)$ is a state-dependent inertia matrix, while $\boldsymbol{\sigma}(\boldsymbol{x}_r)$ and $\boldsymbol{\tau} = [\tau_m, \tau_\delta]^\top \in \mathbb{U} \subset \mathbb{R}^2$ describe the vessel damping and control input, respectively. The set $\mathbb{U}$ describes the control inputs where (1) is valid.

In this work, we assume that the ocean current $\boldsymbol{V}_c$ is constant and known. For practical applications, the ocean current can be measured via appropriate instrumentation, estimated via sensor fusion methods, or predicted based on e.g., tide tables or sensor networks, such as the European marine observation and data network[4].

[4]http://www.emodnet.eu/ (accessed December 11, 2019).

3. HIGH-LEVEL PLANNER

To plan the ASV's nominal trajectory, we use a high-level trajectory planner developed in Bitar et al. (2019b). This trajectory planner uses the ASV model described in section 2 to generate an energy-optimized trajectory between the start and goal positions. The planning algorithm combines an A* implementation and an optimal control problem (OCP) solver to generate a feasible and optimized trajectory.

The high-level planning algorithm consists of three steps: First the A* implementation finds the shortest piecewise linear path between the start and goal position. Secondly, artificial temporal information is added to the path, converting it to a trajectory of states and inputs. Finally, the trajectory is used as an initial guess for an OCP solver, which finds a locally energy-optimized trajectory near the shortest path. All steps account for static obstacles in the form of elliptical boundaries.

3.1. Static Obstacles

The elliptical boundaries are described with the inequality:

$$\left(\frac{x-x_c}{x_a}\right)^2+\left(\frac{y-y_c}{y_a}\right)^2 \geq 1\,, \tag{3}$$

where x_c and y_c is the ellipsis center, and $x_a, y_a > 0$ are the ellipsis major and minor axes, respectively. To allow for angled obstacles, the ellipses are rotated clockwise by an angle α. We add a small constant $\epsilon > 0$ to each side of the inequality, and take the logarithm to arrive at the following obstacle representation:

$$h_o(x,y,x_c,y_c,x_a,y_a,\alpha) = -\log\left[\left(\frac{(x-x_c)\cos\alpha+(y-y_c)\sin\alpha}{x_a}\right)^2 + \left(\frac{-(x-x_c)\sin\alpha+(y-y_c)\cos\alpha}{y_a}\right)^2+\epsilon\right]+\log(1+\epsilon)\leq 0\,. \tag{4}$$

The logarithm operation is applied to reduce the numerical range of the inequality, which helps with numerical stability of the subsequently described solver, and the constant ϵ is included to avoid singularities when (4) is evaluated for $(x,y)\to(x_c,y_c)$ (Bitar et al., 2019a).

Modeling static obstacles as ellipses poses a challenge for handling obstacles of various shapes, from e.g., electronic nautical charts (ENCs). Generic obstacle shapes can be approximated as a set of elliptical obstacles (Wu, 2019), although this may require a large number of constraints for complex environments. Alternatively, the obstacle modeling can be modified to allow for generic shapes. Zhang et al. (2018) present an interesting solution to handle polygon-shaped obstacles by introducing a signed distance function in the optimization problem. Unfortunately, this approach introduces a large number of slack variables and constraints, limiting feasibility for more than a few static obstacles.

3.2. Trajectory Generation and Optimization

From a scenario consisting of static obstacles, as mentioned in section 3.1, we find the piecewise linear shortest path by performing an A* search on a uniformly decomposed grid. The resulting path is converted to a time-parameterized full-state trajectory by assuming a constant forward velocity, and connecting the shortest path with straight segments and circle arcs. The constant forward velocity is

$$u_{\text{nom}} = \frac{L_{\text{path}}}{t_{\max}}\,, \tag{5}$$

where L_{path} is the length of the connected path, and $t_{\max}$ is the maximum allowed time to complete the trajectory. This full-state trajectory is then used as an initial guess to solve the OCP that gives the energy-optimized trajectory:

$$\min_{\boldsymbol{z}(\cdot),\boldsymbol{\tau}(\cdot)} \int_0^{t_{\max}} F_{\text{hi}}(\boldsymbol{z}(t),\boldsymbol{\tau}(t))\mathrm{d}t \tag{6a}$$

subject to

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{f}(\boldsymbol{z}(t),\boldsymbol{\tau}(t))\;\forall t\in[0,t_{\max}] \tag{6b}$$

$$\boldsymbol{h}_{\text{hi}}(\boldsymbol{z}(t),\boldsymbol{\tau}(t))\leq \boldsymbol{0}\;\forall t\in[0,t_{\max}] \tag{6c}$$

$$\boldsymbol{e}_{\text{hi}}(\boldsymbol{z}(0),\boldsymbol{z}(t_{\max})) = \boldsymbol{0}\,. \tag{6d}$$

The solution of this OCP is a trajectory of states $\boldsymbol{z}(\cdot)$ and inputs $\boldsymbol{\tau}(\cdot)$ that minimizes the cost functional in (6a). The ASV model from section 2 is rewritten as $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z},\boldsymbol{\tau})$, where $\boldsymbol{z} = [\boldsymbol{\eta}^\top, \boldsymbol{x}_r^\top]^\top$ and $\boldsymbol{f}(\boldsymbol{z},\boldsymbol{\tau})$ represents (1).

The cost functional (6a) is chosen to minimize energy. The cost-to-go function is

$$F_{\text{hi}}(\boldsymbol{z},\boldsymbol{\tau}) = K_e F_e(\boldsymbol{z},\boldsymbol{\tau}) + K_\delta \tau_\delta^2, \tag{7}$$

with tuning parameters $K_e, K_\delta > 0$. The first term consists of a function that is proportional to mechanical work performed by the ASV:

$$F_e(\boldsymbol{z},\boldsymbol{\tau}) = |\underbrace{n(\tau_m)^2\cdot\cos\delta(\tau_\delta)}_{\propto \text{ surge force}}\cdot u_r| + |\underbrace{n(\tau_m)^2\cdot\sin\delta(\tau_\delta)\cdot L_m}_{\propto \text{ yaw moment}}\cdot r|\,. \tag{8}$$

The function $n:\mathbb{R}^+\to\mathbb{R}^+$ maps the control input τ_m to propeller angular velocity. The function $\delta:\mathbb{R}\to$ S maps the control input τ_δ to outboard motor angle. The second term in (8) is a quadratic cost to yaw control, included to avoid issues with singularity when solving the OCP.

The inequality constraints (6c) observe state boundaries as well as the static obstacles as represented in section 3.1. The boundary conditions (6d) denote initial and final constraints, i.e., start and end states.

A detailed description of the transcription of the OCP (6) to a non-linear program (NLP) using multiple shooting with N_{hi} shooting intervals is found in Bitar et al. (2019b).

4. MID-LEVEL COLAV

The mid-level algorithm, initially presented in Eriksen and Breivik (2017b) and further developed in Bitar et al. (2019a), is a model predictive control (MPC)-based algorithm intended for long-term COLAV. The algorithm utilizes gradient-based

optimization, and takes both static and moving obstacles into account while attempting to follow an energy-optimized nominal trajectory from the high-level planner. The algorithm produces maneuvers complying with Rule 8 of the COLREGs, which requires maneuvers to be made in ample time and be readily observable for other vessels. The optimization problem is formulated as a NLP, which gives flexibility in designing the optimization problem.

In this section, the algorithm is extended to also consider COLREGs rules 13–16 and the first part of Rule 17.

4.1. The International Regulations for Preventing Collisions at Sea (COLREGs)

The COLREGs consists of a total of 37 rules and is divided into five parts (Cockcroft and Lameijer, 2004), where part B (rules 4–19) contains relevant rules on the conduct of vessels in proximity of each other. The most relevant rules for designing COLAV systems in part B are rules 8 and 13–17:

Rule 8 **Action to avoid collision.** This rule states that actions taken to avoid collision should be large enough to be readily observable of other ships, implying that series of small alternations in speed and/or course should not be applied. The rule also recommends that course changes should be prioritized over speed changes if there is enough free space available, and that maneuvers must be made in ample time.

Rule 13 **Overtaking.** This rule states that a vessel is overtaking another if it approaches the other vessel with a course more than 22.5° abaft her beam. The overtaking vessel has to stay clear of the overtaken vessel, but there is no statements on which side of the vessel one should pass.

Rule 14 **Head on.** When two power-driven vessels approach each other on reciprocal, or nearly reciprocal, courses, they are in a head-on situation. In such a situation, both vessels should change their course to starboard, passing each other port-to-port, as shown in **Figure 3A**. This rule states no explicit definition on what should be considered to be reciprocal, or nearly reciprocal, courses, but court decisions indicate head-on situations exist for opposing courses $\pm 6°$. Notice that the rule does not include sailing vessels, which are covered by Rule 12.

Rule 15 **Crossing.** When two vessels approach each other such that the situation is not a head on or an overtaking, it is a crossing situation. The vessel with the other one to her starboard side is deemed the give-way vessel, while the other vessel is deemed the stand-on vessel. As shown in **Figure 3B**, the give-way vessel should maneuver to avoid collision, preferably by passing behind the stand-on vessel, while the stand-on vessel should keep her speed and course.

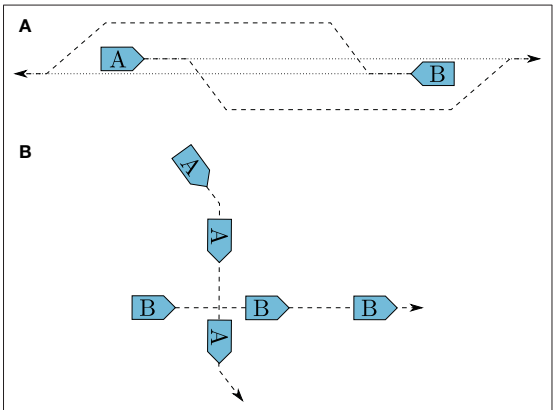


FIGURE 3 | Illustration of head-on **(A)** and crossing **(B)** situations, and how they should be resolved.

Rule 16 **Action by the give-way vessel.** Every vessel which is required to keep out of the way of another vessel should take early and large enough action to safely avoid collision.

Rule 17 **Action by the stand-on vessel.** This rule requires that a stand-on vessel should keep its current speed and course. The stand-on vessel may, however, maneuver to avoid collision if it becomes apparent that the give-way vessel is not taking appropriate actions to avoid collision. Furthermore, if the stand-on vessel finds itself so close to the obstacle that collision can not be avoided by the give-way vessel alone, the stand-on vessel should take such action which best aids to avoid collision. In a crossing situation, the stand-on vessel should avoid maneuvering to port, since this could lead to a collision if the give-way vessel maneuvers to starboard.

In the hybrid architecture illustrated in **Figure 1**, the mid-level algorithm is given the task of strictly enforcing COLREGs rules 13–16 and the stand-on requirement of Rule 17, while also complying with Rule 8.

In addition, we want the mid-level algorithm to comply with the first part of Rule 17, by not maneuvering to avoid collision in crossing situations if the ownship is the stand-on vessel. The hybrid COLAV system is inherently capable of adhering to the remaining requirement of Rule 17, where the stand-on vessel is allowed or required to maneuver, by having different prediction horizons and safety margins in the mid-level and short-term layers. The BC-MPC algorithm does not have any limitations of not maneuvering in stand-on situations, and will hence maneuver in stand-on situations if we come sufficiently close to the obstacle.

The mid-level algorithm as presented in Bitar et al. (2019a) only complies with Rule 8. Further in this section, we therefore present improvements to the mid-level algorithm to make it

comply with rules 13–16 and the stand-on requirement of Rule 17.

4.2. COLREGs Interpretation

A commonly used concept for interpreting obstacles in COLAV algorithms is to assign a spatial region to obstacles, which the ownship should not enter. This approach is commonly referred to as a *domain-based* approach. Specially designed ship domains are commonly used for interpreting the COLREGs in COLAV algorithms, where the required clearance to an obstacle is significantly larger if the maneuver violates the COLREGs (Szlapczynski and Szlapczynska, 2017; Eriksen et al., 2019). This approach is attractive since it continuously captures multiple COLREGs rules, and does not require logic or discrete decisions. However, such an approach does not strictly enforce the COLREGs rules, since it will allow maneuvers violating the rules if they are large enough. In addition, a ship-domain approach will not be able to strictly enforce the stand-on requirement of Rule 17, since a domain-based approach will avoid collision with all obstacles. One could ignore obstacles with give-way obligations, but this would require an explicit COLREGs interpretation which conflicts with domain-based approaches' core idea of implicit COLREGs interpretation. Therefore, we pursue an alternative approach to handling the COLREGs in the mid-level algorithm.

To simplify the COLREGs interpretation task, we look at the situation from a static perspective, assuming that the current COLREGs situations are valid throughout the entire prediction horizon of the mid-level algorithm. In reality, the COLREGs situations may, however, change during the prediction horizon depending on both the ownship's and obstacles future trajectory. For instance, an obstacle approaching from head on, but far enough away to not be considered as a danger may be put in a safe state. Hence, the mid-level algorithm will (for the current iteration) act like no COLREGs rule applies to this vessel for the entire prediction horizon, while the obstacle may get close enough during the prediction horizon to be considered as a head-on situation. An MPC scheme of only implementing a small part of the prediction horizon will reduce the implications of this, since the situation is reassessed each time mid-level algorithm is run, which justifies the assumption of considering the COLREGs from a static perspective. Investigating the possibilities for dynamically predicted future COLREGs situations as part of the MPC prediction will be considered as future work.

4.2.1. State Machine

We propose to utilize a state machine in order to decide which COLREGs rule is active with respect to each obstacle in the vicinity of the ownship. The state machine contains the states:

SF Safe state. This implies that the COLREGs do not enforce any rule with respect to this obstacle.

OT Overtaking state. This implies that COLREGs Rule 13 applies with respect to this obstacle. The state machine does not discriminate on whether the ownship is overtaking another vessel or is being overtaken, but this can be done by looking at which vessel has the higher speed (Tam and Bucknall, 2010).

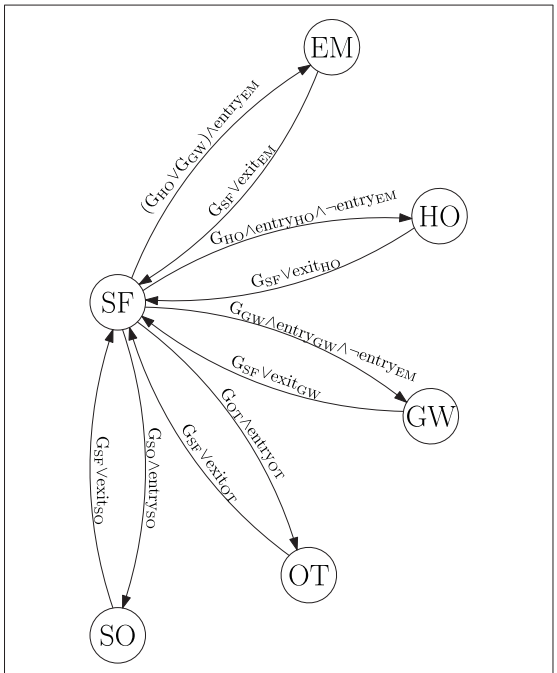


FIGURE 4 | COLREGs state machine. The abbreviations "G_{SF}," "G_{SO}," "G_{OT}," "G_{GW}," and "G_{HO}" denote geometrical situations, while "$entry_{xx}$" and "$exit_{xx}$" denote additional state-dependent entry and exit criterias.

HO Head-on state. This implies that COLREGs Rule 14 applies with respect to this obstacle.

GW Give-way state. This implies that COLREGs Rule 15 applies with respect to this obstacle, and the ownship has to give way.

SO Stand-on state. This implies that COLREGs Rule 15 applies with respect to this obstacle, and the ownship has to stand on.

EM Emergency state. This implies that the obstacle is so close and/or behaves unpredictably, such that special considerations must be made.

As shown in **Figure 4**, all transitions have to go either from or to the safe state.

This implies that when the state machine decides that a COLREGs (or emergency) situation exists with respect to an obstacle, it will not allow switching to another state without the situation being considered as safe first. One could argue that it should be able to transition between specific states, like e.g., from head-on, give-way and overtaking to emergency. This is an interesting topic, which should receive attention in the future. To control the transitions between the different states, we combine the time to and distance at the CPA, a CPA-like measure of the

time until a critical point and a geometrical interpretation of the situation.

4.2.2. Entry and Exit Criteria

CPA is a common concept in maritime risk assessment. Given the current speed and course of the ownship and an obstacle, CPA describes the time to the point where the two vessels are the closest, and the distance to the obstacle at this point. Given the position and velocity vector of the ownship $\boldsymbol{p}, \boldsymbol{v}$ and an obstacle $\boldsymbol{p}_o, \boldsymbol{v}_o$, the time to CPA is calculated as (Kufoalor et al., 2018)

$$t_{\text{CPA}} = \begin{cases} 0 & \text{if } \|\boldsymbol{v} - \boldsymbol{v}_0\|_2 \leq \epsilon \\ \frac{(\boldsymbol{p}-\boldsymbol{p}_o)\cdot(\boldsymbol{v}-\boldsymbol{v}_o)}{\|\boldsymbol{v}-\boldsymbol{v}_o\|_2^2} & \text{else,} \end{cases} \tag{9}$$

where $\epsilon > 0$ is a small constant in order to avoid division by zero in the case where the relative velocity between the ownship and obstacle is zero. Given t_{CPA}, we calculate the distance between the vessels at CPA as

$$d_{\text{CPA}} = \|(\boldsymbol{p} + t_{\text{CPA}}\boldsymbol{v}) - (\boldsymbol{p}_o + t_{\text{CPA}}\boldsymbol{v}_o)\|_2. \tag{10}$$

While the CPA is the point where the distance to an obstacle is at its minimum, the critical point is where the distance to an obstacle crosses underneath a critical distance d_{crit}. This critical distance describes a minimum obstacle distance that the mid-level algorithm is designed for. The time to the critical point t_{crit} can be calculated by solving the equation

$$\|(\boldsymbol{p} + t_{\text{crit}}\boldsymbol{v}) - (\boldsymbol{p}_o + t_{\text{crit}}\boldsymbol{v}_o)\|_2 = d_{\text{crit}}\,. \tag{11}$$

In the cases where the distance between the ships does not fall below d_{crit}, t_{crit} is undefined. Otherwise, there are generally two solutions. The interesting solution is the one with the lowest t_{crit} value, as this is when the obstacle enters the d_{crit} boundary.

The state-machine entry criteria in **Figure 4** are defined as

$$\begin{aligned} \text{entry}_i &= \begin{cases} \text{true} & \text{if } d_{\text{CPA}} < \overline{d}^{i,\text{enter}}_{\text{CPA}} \wedge t_{\text{CPA}} \in [\underline{t}^{i,\text{enter}}_{\text{CPA}}, \overline{t}^{i,\text{enter}}_{\text{CPA}}], \\ \text{false} & \text{otherwise} \end{cases} \\ &\forall i \in \{\text{SO,OT,GW,HO}\} \\ \text{entry}_{\text{EM}} &= \begin{cases} \text{true} & \text{if } t_{\text{crit}} < \overline{t}^{\text{EM,enter}}_{\text{crit}} \wedge t_{\text{CPA}} > 0 \\ \text{false} & \text{otherwise,} \end{cases} \end{aligned} \tag{12}$$

where $\overline{d}^{i,\text{enter}}_{\text{CPA}}$, $\underline{t}^{i,\text{enter}}_{\text{CPA}}$, and $\overline{t}^{i,\text{enter}}_{\text{CPA}}$ for $i \in \{\text{SO,OT,GW,HO}\}$ are tuning parameters denoting thresholds on d_{CPA} and t_{CPA} in order to satisfy the entry criteria for the stand-on, overtaking, give-way and head-on states. The tuning parameter $\overline{t}^{\text{EM,enter}}_{\text{crit}}$ denotes an upper limit on t_{crit} in order to enter the emergency state. The idea behind the stand-on, overtaking, give-way and head-on entry criterias are that in order for the obstacle to represent a risk, both t_{CPA} and d_{CPA} need to be within some tunable thresholds. Situations with a very low d_{CPA}, but with a high t_{CPA}, will not trigger the entry criteria, since the situations will not occur in the near future. Similarly, if t_{CPA} is within the thresholds, but d_{CPA} is large, this indicates a safe passing where risk of collision does not exist. The lower bound on t_{CPA} will typically be selected as zero, and is useful to distinguish between obstacles moving toward of away from the ownship. For the emergency state, the entry criteria is based on the critical point, at which we are so close that the mid-level algorithm may struggle with providing meaningful maneuvers. In addition to t_{crit} being under the threshold $\overline{t}^{\text{EM,enter}}_{\text{crit}}$, we require that t_{CPA} is positive, indicating that we are getting closer to the obstacle. Currently, we only allow entering the emergency state if the situation is a geometrical give-way or head-on, since an overtaking situation represents a smaller danger and has less requirement for special consideration.

The state-machine exit criterias in **Figure 4** are defined as

$$\begin{aligned} \text{exit}_i &= \begin{cases} \text{true} & \text{if } d_{\text{CPA}} \geq \underline{d}^{i,\text{exit}}_{\text{CPA}} \vee t_{\text{CPA}} \notin [\underline{t}^{i,\text{exit}}_{\text{CPA}}, \overline{t}^{i,\text{exit}}_{\text{CPA}}], \\ \text{false} & \text{otherwise} \end{cases} \\ &\forall i \in \{\text{SO,OT,GW,HO}\} \\ \text{exit}_{\text{EM}} &= \begin{cases} \text{true} & \text{if } t_{\text{crit}} \geq \underline{t}^{\text{EM,exit}}_{\text{crit}} \vee t_{\text{CPA}} \leq 0 \\ \text{false} & \text{otherwise,} \end{cases} \end{aligned} \tag{13}$$

where $\underline{d}^{i,\text{exit}}_{\text{CPA}}$, $\underline{t}^{i,\text{exit}}_{\text{CPA}}$, and $\overline{t}^{i,\text{exit}}_{\text{CPA}}$ for $i \in \{\text{SO,OT,GW,HO}\}$ are tuning parameters denoting thresholds on d_{CPA} and t_{CPA} in order to satisfy the exit criteria for the stand-on, overtaking, give-way and head-on states. The exit criteria for the emergency state is satisfied if t_{crit} is larger than the tuning parameter $\overline{t}^{\text{EM,enter}}_{\text{exit}}$, or t_{CPA} is negative, implying that the obstacle is moving further away from the ownship. Note that the exit criterias are obtained by negating the entry criterias, but with other thresholds in order to implement hysteresis to avoid shattering. In general, we allow for different tuning parameters for the different states, but in our simulations we see that selecting the same tuning parameters for all states provides good results. Therefore, we define:

$$\begin{aligned} \overline{d}^{i,\text{enter}}_{\text{CPA}} &= \overline{d}^{\text{enter}}_{\text{CPA}} \\ \underline{t}^{i,\text{enter}}_{\text{CPA}} &= \underline{t}^{\text{enter}}_{\text{CPA}} \\ \overline{t}^{i,\text{enter}}_{\text{CPA}} &= \overline{t}^{\text{enter}}_{\text{CPA}}, \end{aligned} \tag{14}$$

and

$$\begin{aligned} \underline{d}^{i,\text{exit}}_{\text{CPA}} &= \underline{d}^{\text{exit}}_{\text{CPA}} \\ \underline{t}^{i,\text{exit}}_{\text{CPA}} &= \underline{t}^{\text{exit}}_{\text{CPA}} \\ \overline{t}^{i,\text{exit}}_{\text{CPA}} &= \overline{t}^{\text{exit}}_{\text{CPA}} \end{aligned} \tag{15}$$

for all $i \in \{\text{SO,OT,GW,HO}\}$.

4.2.3. Geometrical Situation Interpretation

Tam and Bucknall (2010) present a geometrical interpretation scheme for deciding COLREGs situations based on the relative position, bearing and course of the obstacle with respect to the ownship. We base our geometrical interpretation on a slightly modified version of this scheme, where we include the sign of t_{CPA} to distinguish between situations where the obstacle moves closer toward or farther away from the ownship. The geometrical interpretation is shown in **Figure 5**, where the geometrical situation is obtained by finding which region the obstacle position and course resides in.

Notice that the head-on region is larger than the threshold of $\pm 6°$ as described by the COLREGs. The reason for this is that

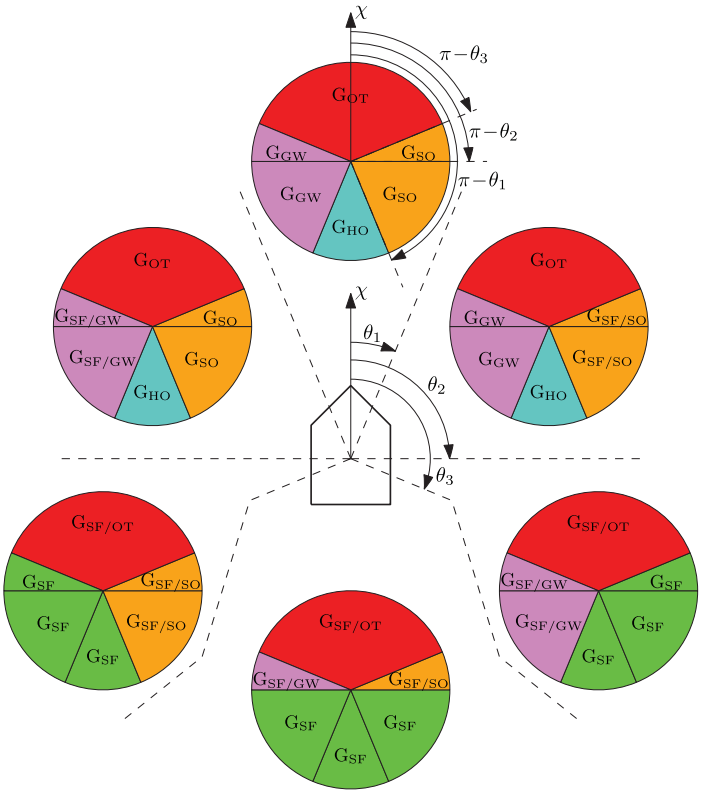


FIGURE 5 | Illustration of the geometrical COLREGs interpretation, where the ownship course is denoted as χ and $\theta_1, \theta_2, \theta_3$ denote symmetrical regions given as [22.5, 90, 112.5°] offset from ahead. The circles illustrate obstacles in different relative bearing regions, and have a fixed orientation with respect to the ownship. The geometrical situations are color-coded and denoted as $G_i, i \in$ {SF, SO, OT, GW, HO} for safe, stand-on, overtaking, give-way and head-on situations, respectively. When two situations are given, like e.g., $G_{SF/SO}$, we use the former (SF) if $t_{CPA} < 0$ and the latter (SO) if $t_{CPA} \geq 0$, analogous to the obstacle moving away or toward the ownship. To decide the geometrical situation, we first find which relative bearing region the obstacle resides in, before finding which obstacle region the obstacle's course resides in. The figure is inspired by Tam and Bucknall (2010).

Tam and Bucknall recommend using a larger region of 22.5° in order to increase the robustness of the geometrical COLREGs interpretation scheme.

4.3. Interface to the High-Level Planner

The high-level planner produces an energy-optimized nominal trajectory for the ownship to follow. However, since the high-level planner does not consider moving obstacles, the speed is the only time-relevant factor of the desired trajectory. In a case where the ownship for some reason, e.g., avoiding moving obstacles, lag behind the nominal trajectory, following the nominal trajectory in absolute time would cause a speed increase in order to catch up with it. Therefore, the mid-level algorithm performs *relative trajectory tracking*, where it tracks the nominal trajectory with a time offset $t_b \in \mathbb{R}$. This results in a relative nominal trajectory for the mid-level algorithm:

$$\bar{\boldsymbol{p}}_d(t) = \boldsymbol{p}_d(t + t_b), \tag{16}$$

where $\boldsymbol{p}_d = [N_d(t), E_d(t)]^\top$ is the nominal trajectory from the high-level planner. The time offset t_b is calculated each time the mid-level algorithm is run by solving a separate optimization problem, and is selected such that $\bar{\boldsymbol{p}}_d(t_0)$ is the point on the nominal trajectory closest to the ownship. See Bitar et al. (2019a) for a detailed description of this concept.

4.4. Optimization Problem Formulation

The mid-level algorithm is formalized as an OCP:

$$\min_{\boldsymbol{\eta}(\cdot), \boldsymbol{x}_r(\cdot)} \quad \phi(\boldsymbol{\eta}(\cdot), \boldsymbol{x}_r(\cdot)) \tag{17a}$$

subject to

$$\dot{\boldsymbol{\eta}}(t) = \boldsymbol{R}(\psi(t))\boldsymbol{x}_r(t) + \begin{bmatrix} \boldsymbol{V}_c \\ 0 \end{bmatrix} \; \forall t \in [t_0, t_0 + T_h] \tag{17b}$$

$$\boldsymbol{h}_{\text{mid}}(\boldsymbol{\eta}(t), \boldsymbol{x}_r(t), t) \leq \boldsymbol{0} \; \forall t \in [t_0, t_0 + T_h] \tag{17c}$$

$$\boldsymbol{e}_{\text{mid}}(\boldsymbol{\eta}(t_0)) = \boldsymbol{0}, \tag{17d}$$

where $T_h > 0$ is the prediction horizon, $\phi(\cdot,\cdot)$ is the objective functional, (17b) contains a kinematic vessel model, (17c) contains inequality constraints and (17d) contains boundary constraints.

Analytical solutions of OCPs are in general not possible to find. A more common approach is to transcribe the OCP to an NLP, and solve that using a gradient optimization scheme. In our case, we transcribe (17) into an NLP with N_p samples using multiple shooting, where the vessel model is discretized using 4th order Runge Kutta and the cost functional is discretized using forward Euler. The resulting NLP is given as

$$\begin{aligned} &\min_{\boldsymbol{w},\boldsymbol{\omega},\boldsymbol{\mu},\boldsymbol{\xi}} \; \phi_p(\boldsymbol{w},\boldsymbol{\omega},\boldsymbol{\mu}) + \phi_c(\boldsymbol{w}) + \phi_{\text{COLREGs}}(\boldsymbol{w}) + \phi_\xi(\boldsymbol{\xi}) \\ &\text{subject to} \\ &\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) = \boldsymbol{0} \\ &\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{\xi}) \leq \boldsymbol{0} \\ &\bar{\boldsymbol{h}}_k(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \bar{\boldsymbol{p}}_{d,k}) \leq \boldsymbol{0} \; \forall k \in \{1, \ldots, N_p\} \\ &\boldsymbol{\xi} \geq \boldsymbol{0}, \end{aligned} \tag{18}$$

where $\boldsymbol{w} = [\boldsymbol{\eta}_0^\top, \boldsymbol{x}_{r,0}^\top, \ldots, \boldsymbol{\eta}_{N_p-1}^\top, \boldsymbol{x}_{r,N_p-1}^\top, \boldsymbol{\eta}_{N_p}^\top]^\top \in \mathbb{R}^{5N_p+3}$ is a vector of $5N_p + 3$ decision variables and $\bar{\boldsymbol{p}}_{d,1:N_p} = [\bar{\boldsymbol{p}}_{d,1}, \bar{\boldsymbol{p}}_{d,2}, \ldots, \bar{\boldsymbol{p}}_{d,N_p}]$ is a sequence of desired positions. The vectors $\boldsymbol{\omega} \in \mathbb{R}^{2N_p}$, $\boldsymbol{\mu} \in \mathbb{R}^{2N_p}$ and $\boldsymbol{\xi} \in \mathbb{R}^{MN_p}$ contain slack variables, where M is the number of moving obstacles to be included in the constraints.

The vector $\boldsymbol{g}(\boldsymbol{w}, \boldsymbol{\eta}(t_0)) \in \mathbb{R}^{3N_p+3}$ contains shooting and boundary constraints, while $\boldsymbol{h}(\boldsymbol{w}) \in \mathbb{R}^{(M+D+4)N_p}$, where D is the number of static obstacles, contain inequality constraints ensuring COLAV and steady-state vessel velocity feasibility. The vectors $\bar{\boldsymbol{h}}_k(\boldsymbol{\eta}_k, \boldsymbol{\omega}_k, \boldsymbol{\mu}_k, \bar{\boldsymbol{p}}_{d,k}) \in \mathbb{R}^6$, $k \in \{1, N_p\}$ contain constraints on the slack variables $\boldsymbol{\omega}$ and $\boldsymbol{\mu}$.

In the following subsections, we describe the terms in (18) in more detail.

4.4.1. Objective Function

The objective function contains four functions, where $\phi_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$ introduces cost on deviating from the relative nominal trajectory $\bar{\boldsymbol{p}}_d(t)$, $\phi_c(\boldsymbol{w})$ introduces cost on using control input, $\phi_{\text{COLREGs}}(\boldsymbol{w})$ is a COLREGs-specific function and $\phi_\xi(\boldsymbol{\xi})$ introduces slack variable cost.

To avoid that the NLP changes behavior when moving away from the nominal trajectory, we wish to have linear growth in the position error function $\phi_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$. This is achieved by instead of using quadratic terms in the position error function, we use the Huber loss function which is quadratic around the origin and resembles the absolute value function above a given threshold $\sigma > 0$:

$$H(\rho) = \begin{cases} \frac{1}{2}\rho^2 & |\rho| \leq \sigma \\ \sigma(|\rho| - \frac{1}{2}\sigma) & |\rho| > \sigma \, . \end{cases} \tag{19}$$

The Huber loss function has a discontinuous gradient, making it slightly complicated to implement in gradient-based optimization problems. It can, however, be implemented in a continuous fashion by utilizing lifting, where slack variables are introduced to create a problem of a higher dimensionality which is easier to solve. Using this technique, $\bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu})$ is defined as

$$\bar{\phi}_p(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}) = K_p \sum_{k=1}^{N_p} \sigma \boldsymbol{1}^\top \boldsymbol{\omega}_k + \frac{1}{2}\boldsymbol{\mu}_k^\top \boldsymbol{\mu}_k, \tag{20}$$

where $K_p > 0$ is a tuning parameter, and $\boldsymbol{\omega}_k \in \mathbb{R}^2$ and $\boldsymbol{\mu}_k \in \mathbb{R}^2$ are slack variables constrained by

$$\bar{\boldsymbol{h}}_k(\boldsymbol{w}, \boldsymbol{\omega}, \boldsymbol{\mu}, \bar{\boldsymbol{p}}_{d,k}) = \begin{bmatrix} \boldsymbol{v_k} + \boldsymbol{\mu_k} + \boldsymbol{p}_k - \bar{\boldsymbol{p}}_{d,k} \\ \boldsymbol{v_k} + \boldsymbol{\mu_k} - (\boldsymbol{p}_k - \bar{\boldsymbol{p}}_{d,k}) \\ -\boldsymbol{\omega}_k \end{bmatrix} \leq \boldsymbol{0} \quad \forall k \in \{1, \ldots, N_p\}, \tag{21}$$

where $\boldsymbol{p}_k$ is the predicted vessel position at time step k, i.e., $\boldsymbol{\eta}_k = [\boldsymbol{p}_k^\top, \psi_k]^\top$. See Bitar et al. (2019a) for more details.

Rule 8 of the COLREGs requires that maneuvers are readily observable for other vessels, implying that speed and course changes should have a sufficiently large magnitude, and not be performed as a sequence of small changes. In order to enforce this in the optimization problem, the control cost function $\phi_c(\boldsymbol{w})$ introduces a non-linear cost on the change in speed and course, which makes the algorithm favor readily observable maneuvers. The function is defined as

$$\phi_c(\boldsymbol{w}) = \sum_{k=0}^{N_p-1} K_{\dot{U}} q_{\dot{U}}(\dot{U}_k) + K_{\dot{\chi}} q_{\dot{\chi}}(\dot{\chi}_k), \tag{22}$$

where $K_{\dot{U}}, K_{\dot{\chi}} > 0$ are tuning parameters, while $q_{\dot{U}}(\dot{U}_k)$ and $q_{\dot{\chi}}(\dot{\chi}_k)$ are the non-linear cost functions. Notice that neither the speed over ground (SOG) U nor the course χ are elements of the search space, but they can be computed as $U = \sqrt{u^2 + v^2}$ and $\chi = \psi + \arcsin\frac{v}{U}$. Their derivatives are then calculated by finite differencing. See Eriksen and Breivik (2017a) and Bitar et al. (2019a) for more details on the control cost function.

The $\phi_{\text{COLREGs}}(\boldsymbol{w})$ function introduces a COLREGs-specific cost with respect to obstacles based on the rule currently applicable as defined by the state machine. We hence tailor the NLP to the current situation. The function is defined as

$$\phi_{\text{COLREGs}}(\boldsymbol{w}) = \sum_{k=1}^{N_p} \Bigg[\sum_{i \in \mathcal{O}_{\text{HO}}} K_{\text{HO}} V_{\text{HO},i,k}(\boldsymbol{p}_k) + \sum_{i \in \mathcal{O}_{\text{GW}}} K_{\text{GW}} V_{\text{GW},i,k}(\boldsymbol{p}_k) + \sum_{i \in \mathcal{O}_{\text{SO}}} K_{\text{SO}} V_{\text{SO},i,k}(\boldsymbol{w}) + \sum_{i \in \mathcal{O}_{\text{EM}}} K_{\text{EM}} V_{\text{EM},i,k}(\boldsymbol{w}) \Bigg], \tag{23}$$

where $\mathcal{O}_{\text{HO}}, \mathcal{O}_{\text{GW}}, \mathcal{O}_{\text{SO}}$, and $\mathcal{O}_{\text{EM}}$ contain obstacles which are in the head-on, give-way, stand-on and emergency states, respectively, and $K_{\text{HO}}, K_{\text{GW}}, K_{\text{SO}}, K_{\text{EM}} > 0$ are tuning parameters. The functions $V_{\text{HO},i,k}(\boldsymbol{p}_k)$, $V_{\text{GW},i,k}(\boldsymbol{p}_k)$, $V_{\text{SO},k}(\boldsymbol{w})$, and $V_{\text{EM},k}(\boldsymbol{w})$ describe functions capturing head-on, give-way, stand-on and emergency behavior with respect to obstacle i, respectively. Notice that the head-on and give-way functions vary with both the obstacle number and time step number, which is due to the functions depending on the given obstacles position and course at time step k.

For head-on situations, we define a potential function with a positive value on the obstacle's starboard side, and a negative value on its port side. When used in the objective function, this will favor trajectories passing a head-on obstacle on its port side, in compliance with Rule 14 of the COLREGs. In addition, the potential function has an attenuation term, reducing the impact of the function when far away from an obstacle:

$$V_{\text{HO},i,k}(\boldsymbol{p}) = \frac{\tanh\left(\alpha_{x,\text{HO}}(x_{0,\text{HO}} - x^{\{i,k\}})\right)}{2} \tanh(\alpha_{y,\text{HO}} y^{\{i,k\}}) \in (-1, 1), \tag{24}$$

where $\alpha_{x,\text{HO}}, \alpha_{y,\text{HO}} > 0$ are tuning parameters controlling the steepness of the head-on potential function and $\bar{x}_{0,\text{HO}} > 0$ is a tuning parameter controlling the influence of the attenuating potential. The coordinate $(x^{\{i,k\}}, y^{\{i,k\}})$ is $\boldsymbol{p}$ given in obstacles i's course-fixed frame (in which the x-axis points along the obstacle's course) at time step k, computed as

$$\begin{bmatrix} x^{\{i,k\}} \\ y^{\{i,k\}} \end{bmatrix} = \boldsymbol{R}(\chi_{i,k})^{\top} \left(\boldsymbol{p} - \boldsymbol{p}_{o,k,i}\right), \tag{25}$$

where $\boldsymbol{p}_{o,k,i}$ and $\chi_{i,k}$ are the position and course of obstacle i at time step k. The head-on potential function with parameters $\alpha_{x,\text{HO}} = 1/500, \alpha_{y,\text{HO}} = 1/400$ and $x_{0,\text{HO}} = 1,000$ m is shown in **Figure 6A**.

For give-way situations, we define a similar potential function, but rotated such that the function is positive in front of an obstacle and negative behind it. This will favor trajectories passing behind an obstacle, as desirable with respect to Rule 15 when a give-way obligation is active. The give-way potential function is defined as

$$V_{\text{GW},i,k}(\boldsymbol{p}) = \frac{\tanh\left(\alpha_{y,\text{GW}}(y^{\{i,k\}} - y_{0,\text{GW}})\right)}{2} \tanh(\alpha_{x,\text{GW}} x^{\{i,k\}}) \in (-1, 1), \tag{26}$$

where $\alpha_{x,\text{GW}}, \alpha_{y,\text{GW}} > 0$ control the steepness of the give-way potential function and $\bar{y}_{0,\text{GW}} < 0$ control the attenuation on the port side of an obstacle. The give-way potential function with parameters $\alpha_{x,\text{GW}} = 1/400, \alpha_{y,\text{GW}} = 1/500$ and $y_{0,\text{GW}} = -500$ m is shown in **Figure 6B**.

In stand-on situations, we want the mid-level algorithm to disregard the obstacle and keep the current speed and course

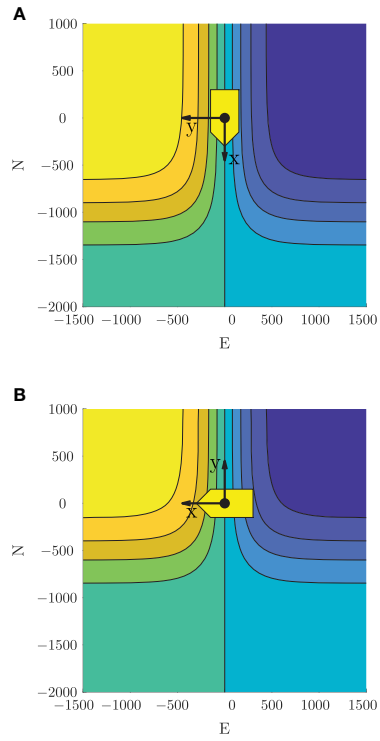


FIGURE 6 | Potential functions ensuring passing on the correct side in head-on and give-way situations. Yellow indicates a positive value, blue indicates a negative value, while the yellow patch and axis cross show the obstacle location and course-fixed coordinate system. Used in a minimization scheme, this will favor starboard maneuvers in head-on situations, and passing behind obstacles in give-way situations. Note that the obstacle here has zero sideslip, resulting in the heading and course pointing in the same direction. **(A)** Head-on potential function. **(B)** Give-way potential function.

in order to comply with the first part of Rule 17. One could simply constrain the algorithm to not maneuver, but this would be perilous in situations where the ownship simultaneously finds itself in a head-on or give-way situation. In such a situation it would be of extra importance to choose readily observable maneuvers, and we therefore design the stand-on cost with the same terms as used in the control cost (22) to amplify the effect:

$$V_{\text{SO},k}(\boldsymbol{w}) = K_{\dot{U}} q_{\dot{U}}(\dot{U}_k) + K_{\dot{\chi}} q_{\dot{\chi}}(\dot{\chi}_k). \tag{27}$$

If an obstacle is in an emergency state, the obstacle is disregarded in the mid-level algorithm and left for the short-term algorithm to handle. In such a situation, it is important that the mid-level algorithm behaves predictable, and we therefore use the same cost function as for stand-on situations:

$$V_{\text{EM},k}(\boldsymbol{w}) = V_{\text{SO},k}(\boldsymbol{w}). \tag{28}$$

The slack variable $\boldsymbol{\xi}$ is used in a homotopy scheme, which we introduce to avoid getting trapped in local minima around moving obstacles. The homotopy scheme is described in further detail in section 4.5. The homotopy cost function $\phi_\xi(\boldsymbol{\xi})$ introduces slack cost on $\boldsymbol{\xi}$:

$$\phi_\xi(\boldsymbol{\xi}) = K_\xi \mathbf{1}^\top \boldsymbol{\xi}, \tag{29}$$

where $K_\xi > 0$ is iteratively increased as part of the homotopy scheme.

4.5. Obstacle Handling and Steady-State Feasibility

The inequality constraint $\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{\xi}) \leq \mathbf{0}$ ensures COLAV and steady-state feasibility with respect to actuator limitations.

Static obstacles are handled similarly as in the high-level algorithm, with (4) representing an elliptical obstacle with center (x_c, y_c), angle α and major and minor axes x_a and y_a, respectively. The constraint (4) needs to be enforced at each time step. Hence, for the i-th static obstacle, we define the constraint

$$\boldsymbol{h}_{s_i}(\boldsymbol{w}) = \begin{bmatrix} h_o(x_1, y_1, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \\ h_o(x_2, y_2, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \\ \vdots \\ h_o(x_{N_p}, y_{N_p}, x_{c,i}, y_{c,i}, x_{a,i}, y_{a,i}, \alpha_i) \end{bmatrix} \leq \mathbf{0}. \tag{30}$$

Moving obstacles are handled in a similar fashion, but letting the ellipsis center position and angle be time varying. Obstacles in stand-on situations should, however, not be included in the constraints, since the mid-level algorithm is supposed to stand on in such situations. Moreover, if an obstacle has entered an emergency state, the obstacle is so close and behaving unpredictably that the mid-level algorithm should disregard it and leave it for the short-term layer. Hence, for the i-th moving obstacle not in a stand-on or an emergency situation, we define the constraint

$$\boldsymbol{h}_{m_i}(\boldsymbol{w}) = \begin{bmatrix} h_o(x_1, y_1, x_{c,i,1}, y_{c,i,1}, x_{a,i}, y_{a,i}, \alpha_{i,1}) \\ \vdots \\ h_o(x_{N_p}, y_{N_p}, x_{c,i,N_p}, y_{c,i,N_p}, x_{a,i}, y_{a,i}, \alpha_{i,N_p}) \end{bmatrix} \leq \mathbf{0}, \tag{31}$$

where $x_{c,i,k}, y_{c,i,k}$, and $\alpha_{i,k}$ denote the position and course of the i-th moving obstacle at time step k.

Given D static obstacles and M obstacles not in stand-on or emergency situations, we define the constraint

$$\boldsymbol{h}_o(\boldsymbol{w}, \boldsymbol{\xi}) = \begin{bmatrix} \boldsymbol{h}_{s_1}(\boldsymbol{w}) \\ \vdots \\ \boldsymbol{h}_{s_D}(\boldsymbol{w}) \\ \boldsymbol{h}_{m_1}(\boldsymbol{w}) \\ \vdots \\ \boldsymbol{h}_{m_M}(\boldsymbol{w}) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\xi} \end{bmatrix}, \tag{32}$$

where we include slack variables $\boldsymbol{\xi} \geq \mathbf{0}$ on the moving obstacle constraints as part of a homotopy scheme. The reason for using homotopy is that NLP solvers in general only finds local minima, and can have issues with moving an initial guess "through" obstacles. Normally, this is not an issue, but for the mid-level algorithm the optimal solution can change drastically from one iteration to another. This can for instance happen if an obstacle enters a head-on or give-way state, where the solution can be trapped on the wrong side of an obstacle. In general, homotopy describes introducing an extra parameter which is iteratively adjusted in order to iteratively move a local solution toward a global solution (Deuflhard, 2011). In our homotopy scheme, we introduce slack variables on the moving obstacle constraints, which will allow solutions to travel through obstacles at the cost of a homotopy cost (29) scaled by the homotopy parameter K_ξ. Initially, this is selected as a low value to have a high amount of slack on the moving obstacles, while it is iteratively increased toward $K_\xi \to \infty$, which results in $\boldsymbol{\xi} = \mathbf{0}$ and hence no slack on moving obstacles. Currently, we only introduce slack on moving obstacles, but slack should also be introduced to static obstacles if they are small enough for the algorithm to be able to pass on both sides, like e.g., rocks, navigational marks, etc.

Similarly as in Eriksen and Breivik (2017b) and Bitar et al. (2019a), we ensure steady-state feasible trajectories at each time step through a constraint $\boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_{r,k}) \leq \mathbf{0} \in \mathbb{R}^4$, which captures the state constraint $\boldsymbol{x}_r \in \mathcal{X}_r$ at time step k. To ensure steady-state feasibility for the entire prediction horizon, we define the constraint

$$\boldsymbol{h}_{\boldsymbol{x}_r}(\boldsymbol{w}) = \begin{bmatrix} \boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_{r,0}) \\ \boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_{r,1}) \\ \vdots \\ \boldsymbol{h}_{\boldsymbol{x}_{r,k}}(\boldsymbol{x}_{r,N_p-1}) \end{bmatrix} \leq \mathbf{0}. \tag{33}$$

Finally, the inequality constraints are combined as.

$$\boldsymbol{h}(\boldsymbol{w}, \boldsymbol{\xi}) = \begin{bmatrix} \boldsymbol{h}_o(\boldsymbol{w}, \boldsymbol{\xi}) \\ \boldsymbol{h}_{\boldsymbol{x}_r}(\boldsymbol{w}) \end{bmatrix} \in \mathbb{R}^{(M+D+4)N_p}. \tag{34}$$

5. SHORT-TERM COLAV

For the short-term layer, the branching-course model predictive control (BC-MPC) algorithm is used, which is a sample-based MPC algorithm intended for short-term ASV COLAV. The BC-MPC algorithm was initially developed in Eriksen et al. (2019), extended to also consider static obstacles in Eriksen and Breivik (2019) and is experimentally validated in several full-scale experiments using a radar-based system for detecting and tracking obstacles. The algorithm complies with COLREGs rules 8, 13, and the second part of Rule 17, while favoring maneuvers complying with the maneuvering aspects of rules 14 and 15. Notice that Rule 17 allows a ship to ignore the maneuvering aspects of rules 14 and 15 in situations where the give-way vessel does not maneuver. The obstacle clearance will be larger if the algorithm ignores the maneuvering aspects of rules 14 and 15, like e.g., passing in front of an obstacle in a crossing situation where the ownship is the give-way vessel. Moving obstacles are in general handled by the mid-level algorithm, making this applicable only in emergency situations and for

obstacles detected so late that the mid-level algorithm is unable to avoid them.

The algorithm constructs a search space consisting of a finite number of trajectories, which each contain a sequence of maneuvers. The maneuvers are constructed using a dynamic model of the ownship and a set of acceleration motion primitives, resulting in feasible trajectories being specified to the vessel controller. For each maneuver, a discrete set of SOG and course accelerations are created as

$$\begin{aligned} \dot{\boldsymbol{U}}_{\text{samples}} &= \left\{\dot{U}_1, \dot{U}_2, \ldots, \dot{U}_{N_U}\right\} \\ \ddot{\boldsymbol{\chi}}_{\text{samples}} &= \left\{\ddot{\chi}_1, \ddot{\chi}_2, \ldots, \ddot{\chi}_{N_\chi}\right\}, \end{aligned} \tag{35}$$

where $\dot{U}_i, i \in [1, N_U]$ and $\ddot{\chi}_i, i \in [1, N_\chi]$ denote $N_U \in \mathbb{N}$ and $N_\chi \in \mathbb{N}$ vessel-feasible speed and course accelerations. Given the acceleration samples (35) and motion primitives for each maneuver in a trajectory, we create a set of desired SOG and course trajectories $\mathcal{U}_d$. These trajectories have continuous acceleration, and is designed in an open-loop fashion by using the current reference tracked by the vessel controller for initialization, rather than the current vessel SOG and course. The reason for this is that the reference to the vessel controller should be continuous in order to avoid jumps in the actuator commands. To include feedback in the trajectory prediction, a set of feedback-corrected SOG and course trajectories $\bar{\mathcal{U}}_d$ is predicted using a simplified error model of the vessel and vessel controller. Finally, the feedback-corrected SOG and course trajectories are used to compute a set of feedback-corrected pose trajectories:

$$\bar{\mathcal{H}} = \left\{\bar{\boldsymbol{\eta}}(\cdot) \middle| (\bar{U}(\cdot), \bar{\chi}(\cdot)) \in \bar{\mathcal{U}}\right\}, \tag{36}$$

where $\bar{\boldsymbol{\eta}}(\cdot)$ denotes a kinematic simulation procedure that given SOG and course trajectories, $\bar{U}(\cdot)$ and $\bar{\chi}(\cdot)$, in $\bar{\mathcal{U}}_d$ computes the vessel pose. See Eriksen and Breivik (2019) and Eriksen et al. (2019) for more details on the trajectory generation procedure.

In order to converge toward the trajectory specified by the mid-level algorithm, a desired acceleration is computed based on a line-of-sight guidance scheme. In Eriksen and Breivik (2019) and Eriksen et al. (2019), the samples closest to the desired acceleration in (35) are replaced with the desired acceleration, given that this is vessel-feasible. A problem with this, is that when operating at high speeds, the possible acceleration may not be symmetric, resulting in that zero acceleration (hence keeping a constant speed and course), may not be part of the search space. This can cause undesirable behavior, since the BC-MPC algorithm will be unable to keep the speed and course constant, which can cause oscillatory behavior. In this paper, we therefore propose to move the acceleration samples closest to zero, and adding the desired acceleration as a separate sample, given that it is vessel feasible. This will make sure that keeping a constant speed and course, as well as a trajectory converging toward the desired trajectory is included in the search space.

Given the predicted trajectories, the algorithm finds the optimal desired SOG and course trajectory for the vessel controller $\boldsymbol{u}_d^*(\cdot) = [U_d(\cdot)^*, \chi_d(\cdot)^*]$ as

$$\boldsymbol{u}_d^*(\cdot) = \underset{(\bar{\boldsymbol{\eta}}_k(\cdot), \boldsymbol{u}_{d,k}(\cdot)) \in (\bar{\mathcal{H}}, \mathcal{U}_d)}{\operatorname{argmin}} G(\bar{\boldsymbol{\eta}}_k(\cdot), \boldsymbol{u}_{d,k}(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot)), \tag{37}$$

where the objective function is given as

$$\begin{aligned} G(\bar{\boldsymbol{\eta}}(\cdot), \boldsymbol{u}_d(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot)) = \; & w_{\text{al}} \text{align}(\bar{\boldsymbol{\eta}}(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot)) \\ & + w_{\text{av,m}} \text{avoid}_{\text{m}}(\bar{\boldsymbol{\eta}}(\cdot)) + w_{\text{av,s}} \text{avoid}_{\text{s}}(\bar{\boldsymbol{\eta}}(\cdot)) \\ & + w_{\text{t},U} \text{tran}_U(\boldsymbol{u}_d(\cdot)) + w_{\text{t},\chi} \text{tran}_\chi(\boldsymbol{u}_d(\cdot)). \end{aligned} \tag{38}$$

The variables $w_{\text{al}}, w_{\text{av,m}}, w_{\text{av,s}}, w_{\text{t},U}, w_{\text{t},\chi} > 0$ are tuning parameters, while $\text{align}(\bar{\boldsymbol{\eta}}(\cdot); \boldsymbol{p}_d^{\text{mid}}(\cdot))$ measures the alignment between a candidate trajectory $\bar{\boldsymbol{\eta}}(\cdot)$ and the desired trajectory from the mid-level algorithm $\boldsymbol{p}_d^{\text{mid}}(\cdot)$. The function $\text{avoid}_{\text{m}}(\bar{\boldsymbol{\eta}}(\cdot))$ ensures COLAV of moving obstacles by penalizing trajectories close to obstacles, using a non-symmetric obstacle ship domain designed with the COLREGs in mind. The function $\text{avoid}_{\text{s}}(\bar{\boldsymbol{\eta}}(\cdot))$ ensures COLAV of static obstacles by introducing an occupancy grid, while $\text{tran}_U(\boldsymbol{u}_d(\cdot))$ and $\text{tran}_\chi(\boldsymbol{u}_d(\cdot))$ introduces transitional costs to avoid shattering. The transitional terms penalize deviations from the planned trajectory of the previous iteration, unless changing to the trajectory corresponding by the desired acceleration. See Eriksen and Breivik (2019) and Eriksen et al. (2019) for more details and descriptions of the terms.

6. SIMULATION RESULTS

The hybrid COLAV system is verified through simulations, which are present in this section. The simulations include ocean current and both static and moving obstacles. We include moving obstacles both acting in compliance with the COLREGs, and violating the COLREGs.

6.1. Simulation Setup

The simulations are performed in MATLAB on a computer with an 2.8 GHz Intel Core i7 processor running macOS Mojave, using CasADi (Andersson et al., 2019) and IPOPT (Wächter and Biegler, 2005) for implementing the high-level and mid-level algorithms. The simulator is built upon the mathematical model of the Telemetron ASV described in section 2, and the model-based speed and course controller in Eriksen and Breivik (2018) is used as the vessel controller.

The parameters of the high-level algorithm are listed in **Table 1**. The number of prediction steps N_{hi} is chosen to achieve a time step length $h = {}^{t_{\max}}/_{N_{\text{hi}}} < 1.5$ s, which seems to be a good compromise between capturing the relevant system dynamics and having a feasible computational requirement.

The mid-level algorithm is implemented using the parameters in **Table 2**.

The slack variable cost K_ξ has five elements, implying that we use five steps in our homotopy scheme. The mid-level NLP is initially warm started with the solution from the previous iteration, while each step in the homotopy scheme is warm started with the solution from the previous step of the homotopy scheme, converging toward the solution without slack on the constraints. To reduce the computational load and increase the predictability of the mid-level algorithm, we utilize six steps of each planned mid-level trajectory, only running the mid-level algorithm every 60 s. This implies that six steps of the predicted

TABLE 1 | Tuning parameters for the high-level algorithm.

Param.	Value	Comment
$t_{\max}$		Maximum trajectory time
Scenario 1	1420 s	
Scenario 2	1420 s	
Scenario 3	725 s	
N_{hi}	1000	Number of prediction steps
K_Θ	1.0 s^3/m	Energy penalty gain
K_δ	1.0	Quadratic yaw control penalty gain
L_m	4.0 m	Length between control origin and outboard motor

TABLE 2 | Tuning parameters for the mid-level algorithm.

Param.	Value	Comment
$\overline{d}_{\text{CPA}}^{\text{enter}}$	900 m	State machine d_{CPA} entry criteria
$\underline{d}_{\text{CPA}}^{\text{exit}}$	2000 m	State machine d_{CPA} exit criteria
$[\underline{t}_{\text{CPA}}^{\text{enter}}, \overline{t}_{\text{CPA}}^{\text{enter}}]$	[0, 270] s	State machine t_{CPA} entry criteria
$[\underline{t}_{\text{CPA}}^{\text{exit}}, \overline{t}_{\text{CPA}}^{\text{exit}}]$	[-20, 290] s	State machine t_{CPA} exit criteria
$\overline{t}_{\text{crit}}^{\text{EM,enter}}$	20 s	Emergency state t_{crit} entry criteria
$\underline{t}_{\text{crit}}^{\text{EM,exit}}$	25 s	Emergency state t_{crit} exit criteria
h	10 s	Step size
N_p	36	Number of prediction steps
K_p	0.02	Position error scaling
σ	1	Huber loss function threshold
$K_{\dot{U}}$	0.3	SOG-derivative penalty term scaling
$K_{\dot{\chi}}$	2.5	Course-derivative penalty term scaling
K_{HO}	40	Head-on potential function scaling
$[\alpha_{x,\text{HO}}, \alpha_{y,\text{HO}}]$	$[1/500, 1/400]$	Head-on potential function steepness parameters
$x_{0,\text{HO}}$	1000 m	Head-on potential function attenuation parameter
K_{GW}	40	Give-way potential function scaling
$[\alpha_{x,\text{GW}}, \alpha_{y,\text{GW}}]$	$[1/400, 1/500]$	Give-way potential function steepness parameters
$y_{0,\text{GW}}$	−500 m	Give-way potential function attenuation parameter
K_{SO}	3	Stand-on function scaling
K_{EM}	3	Emergency function scaling
K_ξ	[0.1, 1, 10, 100, ∞]	Iterative slack variable cost
x_a	600 m	Moving obstacle ellipsis major axis size
y_a	225 m	Moving obstacle ellipsis minor axis size

solution will be implemented before computing a new solution, which further implies that the state machine is also only run every 60 s. If the mid-level algorithm fails in finding a feasible solution, the algorithm will re-use the solution from the last iteration. This may for instance happen if the algorithm tries to compute a solution while being inside a moving obstacle ellipse, which sometimes can be the case when an obstacle is exiting an emergency or stand-on state. The BC-MPC algorithm is run every 5 s, with parameters as described in Eriksen and Breivik (2019). An update rate of 5 s is considered sufficient due the typically large maneuvering margins at sea. It is also worth noting that the detection and tracking system can represent a significant time delay, especially for radar-based systems (Eriksen et al., 2019). For confined and congested areas the BC-MPC algorithm may need to be run at a higher rate, which also imposes requirements for high-bandwidth obstacle estimates. Static obstacles are padded with a safety margin of 150 m for the high-level and mid-level algorithms, while the BC-MPC algorithm uses a safety margin of 100 m for static obstacles. The reason for having a smaller static obstacle safety margin for the BC-MPC algorithm is that it tends to struggle with following trajectories on the static obstacle boundaries. The BC-MPC algorithm would hence not be able to follow the nominal trajectory if the static obstacle safety margin was the same as for the mid-level and high-level algorithms.

The simulations are performed without any noise on the obstacle estimates, providing the algorithms with exact information about the obstacles position, course, and speed. The BC-MPC algorithm has previously been shown to perform well with noisy and uncertain obstacle estimates in full-scale experiments using radar-based detection and tracking of obstacles (Eriksen and Breivik, 2019; Eriksen et al., 2019). The mid-level algorithm is likely to have a larger requirement to low noise levels on the obstacle estimates, since the state machine in the mid-level algorithm depends on logic and discrete switching. However, the algorithm is also run less frequently, reducing the required bandwidth of the obstacle estimates, possibly allowing using smoothing or tracking filters with a lower process noise if necessary. It may also be feasible to make the mid-level algorithm depend on data from the automatic identification system, which typically have much lower noise levels than radar-based tracking systems, while being subject to robustness issues (Harati-Mokhtari et al., 2007).

We present three scenarios, which demonstrate different important properties of the hybrid COLAV system:

Scenario 1 This scenario contains two static obstacles, and four moving obstacles of which all comply with the COLREGs. The moving obstacles demonstrate stand-on, give-way and head-on situations.

Scenario 2 This scenario contains one static and five moving obstacles. The moving obstacles demonstrate stand-on with an obstacle ignoring the COLREGs, an overtaking and a simultaneous head-on, give-way and stand-on situation with obstacles complying with the COLREGs.

Scenario 3 This scenario contains two moving obstacles, which suddenly perform dangerous maneuvers close to the ownship, displaying the use of the emergency state.

6.2. Scenario 1

Scenario 1 contains two static obstacles, four moving obstacles, an ocean current of $[-2, 0]^\top$ m/s and is shown in **Figure 7**. The high-level planner plans a nominal trajectory between the

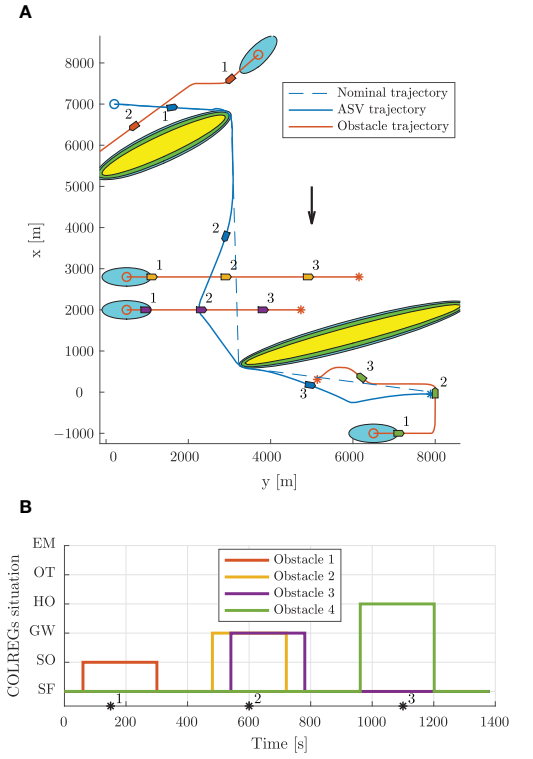


FIGURE 7 | Scenario 1: trajectory and COLREGs interpretation. The text marks denote the time steps [150, 600, 1100] s. **(A)** Trajectory plot. The initial position of the ownship and obstacles are shown with circles, with the blue ellipses illustrating the moving obstacle ellipse size. The vessel patches, which are overexaggerated for visualization, mark the ownship and obstacle poses at given time stamps. The static obstacles are shown in yellow, with the BC-MPC and mid-level safety margins enclosed around. The black arrow indicates the ocean current direction. **(B)** Output from the state machine for each obstacle. The asterisks mark time stamps, and the colors correspond to the obstacle patch colors in the trajectory plot.

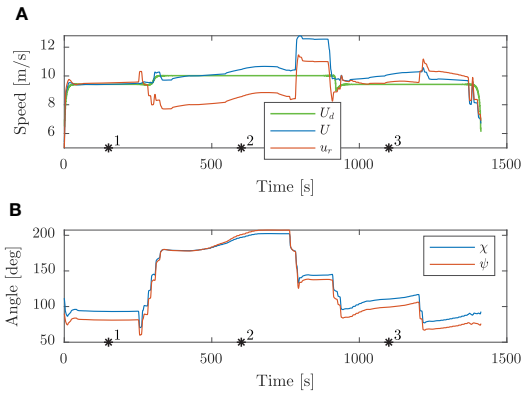


FIGURE 8 | Scenario 1: speed and angular trajectories. The asterisks mark the same time samples as in **Figure 7**. **(A)** Speed trajectories. **(B)** Angular trajectories.

initial and goal positions at $[7000, 200]^\top$ m and $[0, 7900]^\top$ m, respectively. The first obstacle is in a stand-on situation, where it is required to maneuver in order to avoid collision with the ownship, which is required to stand on. As shown in **Figure 7B**, the first obstacle is quickly considered as a stand-on situation, at which the mid-level algorithm disregards the obstacle and continues with the current speed and course. Following this, the obstacle maneuvers in accordance to the COLREGs, and we avoid collision. After the first static obstacle, we encounter two crossing vessels where the ownship is deemed the give-way vessel. In accordance with the COLREGs, we maneuver to starboard in order to pass behind both obstacles. Notice that the second give-way obstacle is detected as a give-way situation later than the first, since the entry criteria in the state machine includes the time to CPA, which is higher for the second give-way obstacle. After avoiding the two give-way obstacles, we converge toward the nominal trajectory and encounter a head-on situation. This is correctly identified by the state machine as head on, and we maneuver to starboard in order to avoid collision. Notice that even though the obstacle maneuvers, we keep the obstacle in the head-on state until we have passed it.

Figure 8 shows the speed and angular trajectories during Scenario 1, where the desired speed is calculated as the nominal speed at the closest point on the nominal trajectory given the ownship position. From this, we see that the mid-level and BC-MPC algorithms manage to track the desired nominal speed before and after the first static obstacle, where no obstacles require maneuvering away from the nominal trajectory. Notice that when encountering the two crossing obstacles, the mid-level algorithm chooses to slowly change the course, which is due to the attenuation of the give-way potential function and the large distance between the vessels. It would be better to make a clear course change, which is a subject of tuning. After passing the two crossing obstacles, the mid-level algorithm increases the speed in order to get back to the nominal trajectory, which is due to the algorithm attempting to keep the speed projected on the nominal trajectory equal as the desired nominal speed. Furthermore, notice that the mid-level algorithm actively controls the relative surge speed in order achieve the desired SOG, which is clearly seen when passing the first static obstacle.

6.3. Scenario 2

Scenario 2, shown in **Figure 9**, is more complex than Scenario 1, with a total of five moving obstacles, and has an ocean current of $[-1, 1]^\top$ m/s. The high-level planner plans a nominal trajectory between the initial and goal positions at $[200, 200]^\top$ m and $[5500, 7000]^\top$ m, respectively. The first obstacle is a crossing vessel, which similarly as in Scenario 1 is deemed to give way for the ownship, which should keep the current speed and course. However, in this scenario, the obstacle violates the COLREGs by

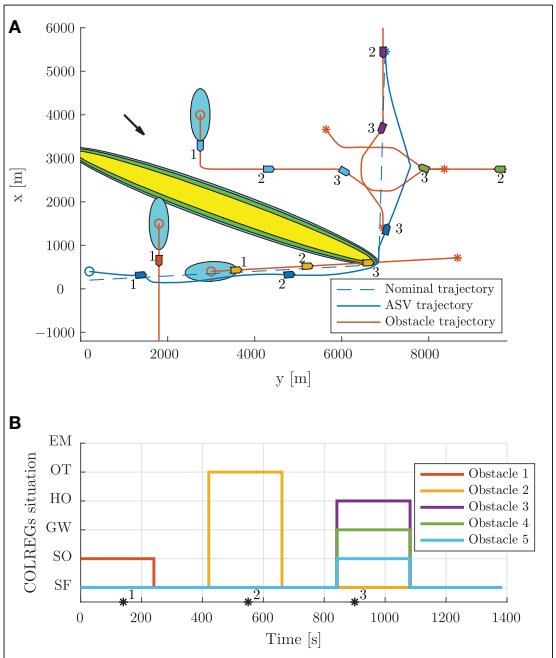


FIGURE 9 | Scenario 2: trajectory and COLREGs interpretation. The text marks denote the time steps [140, 550, 900] s. **(A)** Trajectory plot. The initial position of the ownship and obstacles are shown with circles, with the blue ellipses illustrating the moving obstacle ellipse size. The vessel patches, which are overexaggerated for visualization, mark the ownship and obstacle poses at given time stamps. The static obstacles are shown in yellow, with the BC-MPC and mid-level safety margins enclosed around. The black arrow indicates the ocean current direction. **(B)** Output from the state machine for each obstacle. The asterisks mark time stamps, and the colors correspond to the obstacle patch colors in the trajectory plot.

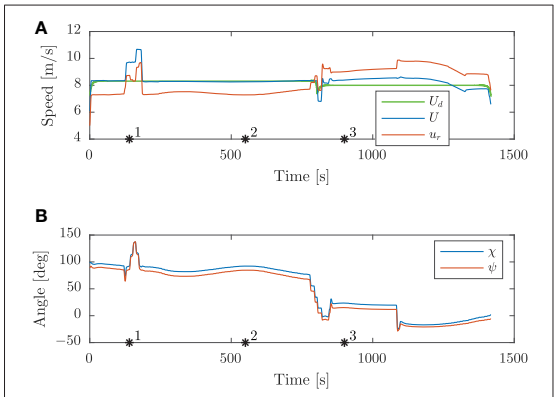


FIGURE 10 | Scenario 2: speed and angular trajectories. The asterisks mark the same time samples as in **Figure 9**. **(A)** Speed trajectories. **(B)** Angular trajectories.

not maneuvering in order to avoid collision. Therefore, the BC-MPC algorithm maneuvers to avoid collision when the obstacle gets so close that the safety margins of the BC-MPC algorithms is violated. The BC-MPC algorithm maneuvers to port, as advised by COLREGs Rule 17 for crossing situations where the stand-on vessel has to maneuver, and safely avoid the first obstacle. The second obstacle is overtaken by the ownship, and correctly considered as an overtaking situation by the state machine. For such an situation, there is no requirement on how the ownship should maneuver, except keeping clear from the overtaken vessel. After passing the second obstacle, we encounter a complex situation with simultaneous head-on, give-way and stand-on obligations. In this situation, each vessel, including the ownship, finds itself in a situation where a head-on and a give-way situation require starboard maneuvers, while a stand-on situation requires the vessel to keep the current speed and course. However, head-on and give-way obligations should be prioritized higher than stand-on situations, and the situation is quite easily solved by each vessel maneuvering to starboard and passing behind the vessel crossing from starboard. The mid-level algorithm solves this situation with the desirable behavior, and converges toward the nominal trajectory after the situation is resolved. As shown in **Figure 9B**, the state machine interprets the situations correctly.

From the speed trajectory in **Figure 10** it is clear that the mid-level algorithm follows the desired nominal speed also when overtaking the second obstacle.

6.4. Scenario 3

Scenario 3, shown in **Figure 11**, contains two moving obstacles on parallel courses with the ownship, and has an ocean current of $[-1, 1]^\top$ m/s. The high-level planner plans a nominal trajectory between the initial and goal positions at $[500, 500]^\top$ m and $[3328, 5399]^\top$ m, respectively, which results in a straight line trajectory with a course angle of 60°. The first obstacle travels at a higher speed than the ownship, while the second one travels at a lower speed and will be overtaken by the ownship. Since the obstacles are on parallel paths with the obstacle, the time to CPA is sufficiently high such that the obstacles are in the safe state, even though the vessels are quite close. However, both obstacles make sudden maneuvers to port dangerously close to the ownship and enters on a crossing course with the ownship. With respect to the COLREGs, the ownship is required to give way to both obstacles since they are crossing from the ownship's starboard side. One can, however, argue that the maneuvers displayed by the obstacles are dangerous and displays poor seamanship, such that the ownship should not be held accountable if a collision occurred. Nevertheless, the hybrid COLAV system manages to avoid both obstacles. As seen in **Figure 11B**, the first obstacle is sufficiently far away from the ownship to be considered as a give-way situation when the state machine interprets the situation, and the mid-level algorithm plans a trajectory passing behind the first obstacle. The second obstacle maneuvers to port even closer to the ownship, resulting in the distance to the critical point being within the threshold

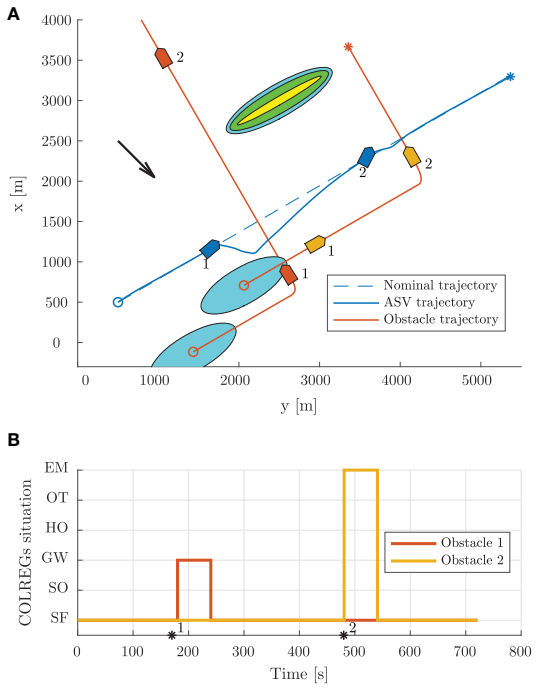


FIGURE 11 | Scenario 3: trajectory and COLREGs interpretation. The text marks denote the time steps [170, 480] s. **(A)** Trajectory plot. The initial position of the ownship and obstacles are shown with circles, with the blue ellipses illustrating the moving obstacle ellipse size. The vessel patches, which are overexaggerated for visualization, mark the ownship and obstacle poses at given time stamps. The static obstacles are shown in yellow, with the BC-MPC and mid-level safety margins enclosed around. The black arrow indicates the ocean current direction. **(B)** Output from the state machine for each obstacle. The asterisks mark time stamps, and the colors correspond to the obstacle patch colors in the trajectory plot.

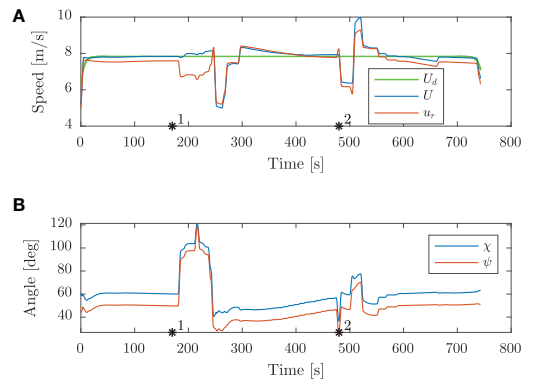


FIGURE 12 | Scenario 3: speed and angular trajectories. The asterisks mark the same time samples as in **Figure 11**. **(A)** Speed trajectories. **(B)** Angular trajectories.

TABLE 3 | Minimum distance to static and moving obstacles for the simulation scenarios.

Scenario	Minimum distance to static obstacles (m)	Minimum distance to moving obstacle number (m)				
		1	2	3	4	5
Scenario 1	93.7	634.3	596.3	522.7	726.8	–
Scenario 2	118.2	185.5	228.3	1,097.2	575.6	842.3
Scenario 3	1123.8	326.4	106.6	–	–	–

for entering the emergency situation when the state machine interprets the situation. In this situation, the mid-level algorithm disregards the obstacle and leaves it to the BC-MPC algorithm to avoid collision.

As seen in **Figure 12**, the mid-level algorithm both reduces the speed and changes the course to avoid the first obstacle. When approaching the second obstacle, the BC-MPC algorithm initiates a speed reduction, and after some time also maneuver to starboard in order to pass behind the obstacle and resolve the situation.

6.5. Simulation Summary

The simulation results show that the hybrid COLAV system is able to handle a wide range of situations, while also behaving in an energy-optimal way when moving obstacles are not interfering with the ownship trajectory. **Table 3** shows the minimum distance to static and moving obstacles for the scenarios.

The minimum distance to static obstacles is in Scenario 1 below the safety region size of the BC-MPC algorithm, which is intentional and caused by the algorithm using a smooth penalty function for interpreting static obstacles. The penalty function value increases linearly when moving further into the safety region, see Eriksen and Breivik (2019) for more details. The minimum distance to moving obstacles is a bit difficult to interpret, since the obstacle ship domains are non-circular, implying that the required clearance depends on relative position of the ownship with respect to the moving obstacles. However, we see that we have a larger clearance in head-on, give-way and stand-on situations where the obstacles comply with the COLREGs, and do not perform dangerous maneuvers (as in Scenario 3), compared to overtaking situations. The reason for this is that when overtaking (obstacle 2 in Scenario 2), we pass the obstacle on a parallel course, resulting in the minor axis of the moving obstacle ellipsis indicating the required clearance. Furthermore, we see that obstacle 1 in Scenario 2, which ignores its give-way obligation, comes significantly closer than other crossing obstacles except for those in Scenario 3. The reason for this is that the BC-MPC algorithm, which handles this situation, has a lower clearance requirement than the mid-level algorithm, which still should be considered as safe. In Scenario 3, the two obstacles display poor seamanship, and behave dangerously. Obstacle 1 is handled by the mid-level algorithm and passed

with a clearance lower than the major axis of the mid-level algorithm, which is caused by the BC-MPC algorithm "cutting the corner." The clearance should still be considered safe since we are behind the obstacle, and the clearance requirements of the BC-MPC algorithm is enforced. Obstacle 2, which is placed in the emergency state and handled by the BC-MPC algorithm, is passed with a clearance of only 106.6 m. This is lower than the clearance to Obstacle 1 in Scenario 2 (which violated its stand-on requirement), and is due to the BC-MPC algorithm having a non-symmetric obstacle ship domain function allowing for a smaller clearance when passing behind an obstacle than in front.

For the three scenarios, the high-level planner used an average of 67 s with a maximum of 93 s to compute the solution. Since the high-level planner is intended to be run off-line, this is well within reasonable limits. The mid-level algorithm used 0.60 s on average, and a maximum of 2.1 s, which we consider to be computationally feasible since the mid-level algorithm only is run every 60 s. The BC-MPC algorithm used 0.29 s on average, and a maximum of 0.63 s, which we also consider to be real-time feasible when the BC-MPC algorithm is run every 5 s. The BC-MPC algorithm is highly parallelizable, which could reduce the BC-MPC runtime by a large magnitude if required. The mid- and high-level algorithms may not return solutions as they are non-convex optimization problems, but the BC-MPC algorithm makes the hybrid COLAV system real-time feasible since it always will find a (potentially sub-optimal) solution.

7. CONCLUSION

In this paper, we have presented a three-layered hybrid COLAV system, compliant with COLREGs rules 8 and 13–17. As part of this, we have further developed the MPC-based mid-level COLAV algorithm in Eriksen and Breivik (2017b) and Bitar et al. (2019a) to comply with COLREGs rules 13–16 and parts of Rule 17, which includes developing a state machine for COLREGs interpretation. The hybrid COLAV system has a well-defined division of labor, including an inherent understanding of COLREGs Rule 17, where the mid-level algorithm obeys stand-on situations, while the BC-MPC algorithm handles situations where give-way vessels do not maneuver.

The hybrid COLAV system is verified through simulations, where we in three scenarios challenge the system with a number of different situations. The scenarios include multi-obstacle situations with multiple simultaneously active COLREGs rules, and situations where obstacles violate the COLREGs. Collision is avoided in all the scenarios, and we show that the ownship follows an energy-optimized trajectory generated by the high-level planner when moving obstacles do not interfere with this trajectory.

For further work, we suggest to:

- Investigate if using situation-dependent entry and exit criteria parameters in the state machine improves the performance.
- Expand the state machine with the possibility of transitioning from head-on, give-way and overtaking states to the emergency state for situations where obstacles behave dangerously or hostile.
- Develop a methodology for deciding tuning parameters.
- Perform simulations with noisy obstacle estimates to investigate how the state machine and mid-level algorithm respond to this.
- Explore the possibilities for integrating the COLREGs interpretation in the mid-level NLP, relaxing the assumption of the current COLREGs situation being valid for the entire prediction horizon.
- Investigate the possibility of including static obstacles from e.g., ENCs in the high- and mid-level algorithms.
- Simulate scenarios where multiple vessels running the hybrid COLAV system interact with each other.
- Validate the hybrid COLAV system in full-scale experiments.

DATA AVAILABILITY STATEMENT

The datasets generated for this study are available on request to the corresponding author.

AUTHOR CONTRIBUTIONS

The work in this article was the result of a collaboration between B-OE and GB, supervised by MB and AL. The contributions to the mid-level and short-term algorithms are made by B-OE, while the COLREGs interpreter was developed in collaboration between GB and B-OE. GB has implemented the high-level planner, and prepared this for integration with the developed simulator. B-OE has taken lead on writing the paper, in collaboration with GB. MB and AL have provided valuable feedback in the writing process.

FUNDING

This work was supported by the Research Council of Norway through project number 269116, project number 244116, as well as the Centers of Excellence funding scheme with project number 223254.

REFERENCES

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. (2019). CasADi: a software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* 11, 1–36. doi: 10.1007/s12532-018-0139-4

Benjamin, M. R., Leonard, J. J., Curcio, J. A., and Newman, P. M. (2006). A method for protocol-based collision avoidance between autonomous marine surface craft. *J. Field Robot.* 23, 333–346. doi: 10.1002/rob.20121

Bitar, G., Breivik, M., and Lekkas, A. M. (2018). "Energy-optimized path planning for autonomous ferries," in *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)* (Opatija), 389–394.

Bitar, G., Eriksen, B.-O. H., Lekkas, A. M., and Breivik, M. (2019a). "Energy-optimized hybrid collision avoidance for ASVs," in *Proceedings of the 17th IEEE European Control Conference (ECC)* (Naples), 2522–2529.

Bitar, G., Vestad, V. N., Lekkas, A. M., and Breivik, M. (2019b). "Warm-started optimized trajectory planning for ASVs," in *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)* (Daejeon).

Campbell, S., Abu-Tair, M., and Naeem, W. (2014). An automatic COLREGs-compliant obstacle avoidance system for an unmanned surface vehicle. *Proc. Inst. Mech. Eng. M J. Eng. Marit. Environ.* 228, 108–121. doi: 10.1177/1475090213498229

Casalino, G., Turetta, A., and Simetti, E. (2009). "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields," in *Proceedings of the 2009 IEEE OCEANS-EUROPE Conference* (Bremen).

Chauvin, C. (2011). Human factors and maritime safety. *J. Navig.* 64, 625–632. doi: 10.1017/S0373463311000142

Cockcroft, A. N., and Lameijer, J. N. F. (2004). *A Guide to the Collision Avoidance Rules*. Oxford, UK: Elsevier.

Deuflhard, P. (2011). *Newton Methods for Nonlinear Problems*. Berlin: Springer.

Eriksen, B.-O. H., and Breivik, M. (2017a). *Modeling, Identification and Control of High-Speed ASVs: Theory and Experiments*. Cham: Springer International Publishing, 407–431.

Eriksen, B.-O. H., and Breivik, M. (2017b). "MPC-based mid-level collision avoidance for ASVs using nonlinear programming," in *Proceedings of the 1st IEEE Conference on Control Technology and Applications (CCTA)* (Kohala Coast, HI), 766–772.

Eriksen, B.-O. H., and Breivik, M. (2018). "A model-based speed and course controller for high-speed ASVs," in *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)* (Opatija), 317–322.

Eriksen, B.-O. H., and Breivik, M. (2019). Short-term ASV collision avoidance with static and moving obstacles. *Model. Identif. Control* 40, 177–187. doi: 10.4173/mic.2019.3.4

Eriksen, B.-O. H., Breivik, M., Wilthil, E. F., Flåten, A. L., and Brekke, E. F. (2019). The branching-course model predictive control algorithm for maritime collision avoidance. *J. Field Robot.* 36, 1222–1249. doi: 10.1002/rob.21900

Eriksen, B.-O. H., Wilthil, E. F., Flåten, A. L., Brekke, E. F., and Breivik, M. (2018). "Radar-based maritime collision avoidance using dynamic window," in *Proceedings of the 2018 IEEE Aerospace Conference* (Big Sky, MT), 1–9.

Hagen, I. B., Kufoalor, D. K. M., Brekke, E., and Johansen, T. A. (2018). "MPC-based collision avoidance strategy for existing marine vessel guidance systems," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD), 7618–7623.

Harati-Mokhtari, A., Wall, A., Brooks, P., and Wang, J. (2007). Automatic identification system (AIS): data reliability and human error implications. *J. Navig.* 60, 373–389. doi: 10.1017/S0373463307004298

Kufoalor, D. K. M., Brekke, E. F., and Johansen, T. A. (2018). "Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method," in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid), 2402–2409.

Kuwata, Y., Wolf, M. T., Zarzhitsky, D., and Huntsberger, T. L. (2014). Safe maritime autonomous navigation with COLREGS, using velocity obstacles. *IEEE J. Ocean. Eng.* 39, 110–119. doi: 10.1109/joe.2013.2254214

Levander, O. (2017). Autonomous ships on the high seas. *IEEE Spectr.* 54, 26–31. doi: 10.1109/MSPEC.2017.7833502

Loe, Ø. A. G. (2008). *Collision avoidance for unmanned surface vehicles* (Master's thesis), Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Švec, P., Shah, B. C., Bertaska, I. R., Alvarez, J., Sinisterra, A. J., von Ellenrieder, K., et al. (2013). "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Tokyo), 3871–3878.

Szlapczynski, R., and Szlapczynska, J. (2017). Review of ship safety domains: models and applications. *Ocean Eng.* 145, 277–289. doi: 10.1016/j.oceaneng.2017.09.020

Tam, C., and Bucknall, R. (2010). Collision risk assessment for ships. *J. Mar. Sci. Technol.* 15, 257–270. doi: 10.1007/s00773-010-0089-7

Wächter, A., and Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* 106, 25–57. doi: 10.1007/s10107-004-0559-y

Wu, D. (2019). *Proactive maritime collision avoidance based on historical AIS data* (Master's thesis), Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Zhang, X., Liniger, A., Sakai, A., and Borrelli, F. (2018). "Autonomous parking using optimization-based collision avoidance," in *Proceedings of the 57th IEEE Conference on Decision and Control (CDC)* (Miami, FL), 4327–4332.

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Paper E Trajectory planning and control for automatic docking of ASVs with full-scale experiments

Postprint by **G. Bitar**, A. B. Martinsen, A. M. Lekkas, and M. Breivik. "Trajectory planning and control for automatic docking of ASVs with full-scale experiments". In: *Proceedings of the 1st Virtual IFAC World Congress*. 2020. arXiv: 2004.07793 [eess.SY].

Bibliography entry [12].

Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments ⋆

Glenn Bitar, Andreas B. Martinsen, Anastasios M. Lekkas and Morten Breivik

Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), O. S. Bragstads plass 2D, NO-7491 Trondheim, Norway
E-mails: {glenn.bitar,andreas.b.martinsen, anastasios.lekkas}@ntnu.no, morten.breivik@ieee.org

Abstract: We propose a method for performing automatic docking of a small autonomous surface vehicle (ASV) by interconnecting an optimization-based trajectory planner with a dynamic positioning (DP) controller for trajectory tracking. The trajectory planner provides collision-free trajectories by considering a map with static obstacles, and produces feasible trajectories through inclusion of a mathematical model of the ASV and its actuators. The DP controller tracks the time-parametrized position, velocity and acceleration produced by the trajectory planner using proportional-integral-derivative feedback with velocity and acceleration feed forward. The method's performance is tested on a small ASV in confined waters in Trondheim, Norway. The ASV performs collision-free docking maneuvers with respect to static obstacles when tracking the generated reference trajectories and achieves successful docking.

Keywords: Autonomous surface vehicles, automatic docking, model predictive control, optimal control, path planning.

1. INTRODUCTION

Autonomous surface vehicles (ASVs) constitute a topic of significant research and commercial attention and effort. Motivating factors are economy, flexibility, safety and environmental advantages. Technology developments in this field are rapid, and the use cases are many, e.g. mapping of the ocean floor, military applications such as surveillance, and transportation. In addition, the relatively low cost of smaller ASVs enables novel concepts, for example autonomous urban passenger ferries that are an alternative to bridges in a city landscape.

To achieve autonomy in transportation operations the following phases must be automated:

- Undocking – moving from the quay in a confined harbor area to open waters,
- transit – crossing a canal or large body of water towards the destination harbor,
- docking – moving from open waters towards the docking position along the quay in a harbor area.

Since this paper focuses on the docking phase, we provide a background on automatic docking methods. The number of reported existing methods is limited in research literature and in commercial applications. Methods for docking of autonomous underwater vehicles (AUVs) have been introduced by e.g. Rae and Smith (1992), Teo et al. (2015) and Hong et al. (2003), but they are of limited value for use with surface vessels in a confined harbor area, due to the lack of consideration of nearby obstacles. Tran and Im (2012) propose a method for docking of a large ship based on artificial neural networks to control the ship's thrusters, which has shown promising simulation results. However, this method does not include the harbor layout for collision avoidance. Mizuno et al. (2015) propose an optimization-based approach taking into account known disturbances. An optimal nominal path is generated once, and a lower-level model predictive controller (MPC) attempts to follow it. This method also does not include the harbor layout for collision avoidance, and it is not very realistic to assume known disturbances in such dynamic settings. Commercial demonstrations of automatic docking have been performed by Wärtsilä [1] and Rolls-Royce [2] (now Kongsberg Maritime). Details about the methods used in these approaches are unavailable to the public.

The docking method from (Martinsen et al., 2019) is a nonlinear model predictive controller (NMPC) that takes into account vessel dynamics in the form of its dynamic model, as well as collision avoidance by planning trajec-

⋆ This work is supported by the Research Council of Norway through the project number 269116 as well as through the Centres of Excellence funding scheme with project number 223254.

[1] Wärtsilä press release: `https://www.wartsila.com/twentyfour7/innovation/look-ma-no-hands-auto-docking-ferry-successfully-tested-in-norway`.

[2] Rolls-Royce press release: `https://www.rolls-royce.com/media/press-releases/2018/03-12-2018-rr-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry.aspx`.

Fig. 1. The experimental autonomous urban passenger ferry *milliAmpere*, developed at NTNU, moored near Brattøra in Trondheim, Norway.

tories within a convex set, based on the harbor layout. Advantages of that approach include explicit handling of static obstacles, planning of dynamically feasible trajectories, and flexible behavior shaping via the nonlinear cost function. The method does not handle moving obstacles or account for external unknown disturbances. Additionally, due to the non-convex shape of the optimal control problem (OCP), guarantees on run time or feasibility are not provided. In this paper, we build on (Martinsen et al., 2019) and add the following contributions:

- Instead of running the trajectory planner as an MPC controller by using the inputs directly, the state trajectory is sent to a trajectory-tracking dynamic positioning (DP) controller to account for disturbances and unmodeled dynamics.
- The thruster model is adjusted to improve run times and convexity properties.
- The cost function is adjusted to deal with the wrap-around problem in the heading variable, and to avoid quadratic costs on large position deviations.
- Slack variables are added to deal with feasibility issues that arise when implementing the method in a real-world scenario.
- Although not detailed in this paper, we have implemented an algorithm that dynamically updates the convex set which represents the static obstacles. The set is updated based on the vessel's current position in the map, which allows us to use convex constraints in a non-convex map.

By modifying the method from (Martinsen et al., 2019), it is shown to produce collision-free and successful maneuvers in full-scale experiments on the experimental autonomous urban passenger ferry *milliAmpere*, seen in Figure 1, in Trondheim, Norway. Although the method is implemented to solve the docking problem on an autonomous urban passenger ferry, this is a generic approach that is suitable also for other use cases and vessel types.

The rest of this paper is structured as follows: We introduce the experimental platform *milliAmpere* in Section 2. The trajectory planner used for generating docking trajectories is presented in Section 3, along with the trajectory-tracking DP controller. Section 4 presents the experimental results, and we conclude the paper in Section 5. We present the mathematical models used in the paper in Appendix A.

Table 1. *milliAmpere* specifications.

Dimensions	5 m by 2.8 m symmetric footprint
Position and heading reference system	Vector VS330 dual GNSS with RTK capabilities
Thrusters	Two azimuth thrusters on the center line, 1.8 m aft and fore of center
Existing control modules	Trajectory-tracking DP controller and thrust allocation system

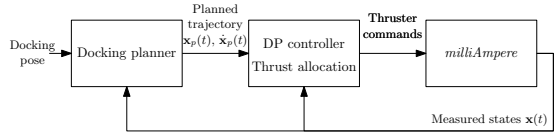


Fig. 2. Block diagram of the docking system setup. The DP controller and thrust allocation are existing functions on *milliAmpere*.

2. THE *MILLIAMPERE* AUTONOMOUS FERRY PLATFORM

For the sea trials, we used the experimental autonomous urban passenger ferry *milliAmpere*, depicted in Figure 1 and with specifications as listed in Table 1. Developed at the Norwegian University of Science and Technology (NTNU) since 2017, *milliAmpere* has been an experimental platform where many students have contributed with control systems as well as hardware solutions. A larger version is being designed and built by the research group Autoferry. [3] Small passenger ferries for urban water transport is a novel concept which is being made economically feasible due to increased availability and advances in both sensor systems and autonomous technology. Such a solution is anticipated to make areas that are separated by waterways more accessible at a lower cost and with less interfering infrastructure than building a bridge.

For simulation purposes, we have used a surge-decoupled three-degree-of-freedom model, along with dynamic models for azimuth angles and propeller speeds of the thrusters. Separate models are also used for planning and trajectory tracking. Since we are using three different models in the work described in this paper, we place the model information in Appendix A to improve readability. Parameters and information about the model identification process are available in (Pedersen, 2019).

3. TRAJECTORY PLANNING AND CONTROL

The trajectory planner is an OCP that takes into account the vessel dynamics via a mathematical model, as well as the harbor layout by including a map as constraints. The OCP is a modified version of the one developed in (Martinsen et al., 2019). In our case, the OCP runs at a set rate and provides pose, velocity and acceleration trajectories for an existing trajectory-following DP controller, as illustrated in Figure 2.

The OCP is described by the following equations:

[3] Autoferry website: `https://www.ntnu.edu/autoferry`.

$$\min_{\boldsymbol{x}_p(\cdot),\boldsymbol{u}_p(\cdot),\boldsymbol{s}(\cdot)} \int_{t_0}^{t_0+T} \Big(F(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) + \boldsymbol{k}_s^\top \boldsymbol{s}(t) \Big) \, \mathrm{d}t \quad \text{(1a)}$$

subject to

$$\dot{\boldsymbol{x}}_p(t) = \boldsymbol{f}(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) \qquad \forall t \in [t_0, t_0+T] \quad \text{(1b)}$$

$$\boldsymbol{h}(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) - \boldsymbol{s}(t) \leq \boldsymbol{0} \qquad \forall t \in [t_0, t_0+T] \quad \text{(1c)}$$

$$\boldsymbol{s}(t) \geq \boldsymbol{0} \qquad \forall t \in [t_0, t_0+T] \quad \text{(1d)}$$

$$\boldsymbol{x}_p(t_0) = \boldsymbol{x}(t_0). \quad \text{(1e)}$$

The planned states are denoted $\boldsymbol{x}_p = [\boldsymbol{\eta}_p^\top, \boldsymbol{\nu}_p^\top]^\top$, where $\boldsymbol{\eta}_p = [x_p, y_p, \psi_p]^\top$ is the Earth-fixed pose, and $\boldsymbol{\nu}_p = [u_p, v_p, r_p]^\top$ is the body-fixed velocity vector. The kinematic relationship between the pose and velocity vectors is detailed in Appendix A. The goal of the OCP is to arrive at the constant state vector $\boldsymbol{x}_d = [\boldsymbol{\eta}_d^\top, \boldsymbol{0}_3^\top]^\top$ while avoiding collisions, where $\boldsymbol{\eta}_d = [x_d, y_d, \psi_d]^\top$ is referred to as the *docking pose*. The vector $\boldsymbol{x}(t_0)$ is the vessel's measured state at time t_0. The planning horizon is $T = 120\,\mathrm{s}$.

The input vector $\boldsymbol{u}_p = [f_{x1}, f_{y1}, f_{x2}, f_{y2}]^\top$ is used to denote the forces decomposed in surge and sway of *milliAmpere*'s two actuators, where f_{x1} represents a force in surge direction from thruster 1, f_{y2} represents a force in sway direction from thruster 2, etc. Details on the mapping from this input to control forces are found in Appendix A. The cost functional and constraints are elaborated upon in the following subsections. The OCP is discretized using direct collocation and solved as a nonlinear program (NLP) with 60 control intervals.

3.1 Cost functional

The cost functional (1a) operates on the trajectories of the states $\boldsymbol{x}_p(\cdot)$, inputs $\boldsymbol{u}_p(\cdot)$ and slack variables $\boldsymbol{s}(\cdot)$. It consists of a cost-to-go function $F(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t))$, as well as a cost-to-go on the slack variables $\boldsymbol{k}_s^\top \boldsymbol{s}(t)$ with the elements of $\boldsymbol{k}_s$ having values large enough (1.0×10^3) so that the slack variables are active only when the problem otherwise would be infeasible.

The cost-to-go function is

$$\begin{aligned} F(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) = \\ H\left(\begin{bmatrix} x_p(t) - x_d \\ y_p(t) - y_d \end{bmatrix} \right) + \\ 20\,(1 - \cos(\psi_p(t) - \psi_d)) + \\ 10\, v_p(t)^2 + 10\, r_p(t)^2 + \\ \boldsymbol{u}_p(t)^\top \boldsymbol{u}_p(t) \,/\, m_{11}^2 \,, \end{aligned} \quad (2)$$

where the terms are costs on position error, heading error, quadratic sway velocity and yaw rate, and quadratic input, respectively. The parameter m_{11} is the system inertia in surge, detailed in Appendix A. The terms are scaled so that the cost function becomes dimensionless. The pseudo-Huber function

$$H(\boldsymbol{a}) = \delta^2 \left(\sqrt{1 + \frac{\boldsymbol{a}^\top \boldsymbol{a}}{\delta^2}} - 1 \right) \quad (3)$$

with $\delta = 10\,\mathrm{m}$ provides a quadratic penalty when the quadrature position errors are low and linear when they are high.

The resulting cost functional encourages the planned trajectories to converge to the docking pose $\boldsymbol{\eta}_d$ with zero velocity, while penalizing sway and yaw rates, as well as the input forces. Including the docking pose in the cost functional instead of as terminal constraints allows us to use the planner far away from the dock, when the docking pose is outside the reach of the planning horizon T. Additionally, if the operator selects a docking pose that is in violation of the collision constraints, the planner will accept it and find a feasible pose close to the docking pose.

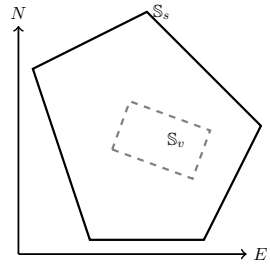


Fig. 3. Spatial constraints illustration.

3.2 Vessel model

Equation (1b) is a simplified model of the vessel dynamics. A diagonalized version of the surge-decoupled model in (Pedersen, 2019) is used, with details found in Appendix A. The kinematic and kinetic models are concatenated to

$$\dot{\boldsymbol{x}}_p = \boldsymbol{f}(\boldsymbol{x}_p, \boldsymbol{u}_p) = \begin{bmatrix} \mathbf{R}(\psi_p)\boldsymbol{\nu}_p \\ (\mathbf{S}\,\mathbf{M}_p)^{-1}(-\mathbf{C}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p - \mathbf{D}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p + \boldsymbol{\tau}_p(\boldsymbol{u}_p)) \end{bmatrix}, \quad (4)$$

where the time argument is omitted for notational brevity. This equation is included as dynamic constraints in the OCP.

3.3 Inequality constraints

The inequality constraints (1c) encode collision avoidance criteria as well as state and input limitations. These constraints are softened by using slack variables and linear slack costs that keep the optimization problem feasible should disturbances push the vessel outside of these boundaries.

To avoid collisions, we specify a set $\mathbb{S}_v \subset \mathbb{R}^2$ representing the footprint of the vessel, as well as a permissible convex set $\mathbb{S}_s \subset \mathbb{R}^2$. The collision avoidance constraint is to ensure $\mathbb{S}_v \subset \mathbb{S}_s$, which can be controlled by checking that the vertices of $\mathbb{S}_v$ are within $\mathbb{S}_s$, as illustrated in Figure 3. Since $\mathbb{S}_s$ is a convex polyhedron, we can describe it as

$$\mathbb{S}_s = \left\{ \boldsymbol{p} \in \mathbb{R}^2 \mid \mathbf{A}_s \boldsymbol{p} \leq \boldsymbol{b}_s \right\}, \quad (5)$$

where $\mathbf{A}_s \in \mathbb{R}^{k \times 2}$ and $\boldsymbol{b}_s \in \mathbb{R}^k$ and k is the number of vertices in the convex set. This results in the collision avoidance constraint being equivalent to

$$\mathbf{A}_s \left(\mathbf{R}_2(\psi_p(t))\boldsymbol{v} + \begin{bmatrix} x_p(t) \\ y_p(t) \end{bmatrix} \right) \leq \boldsymbol{b}_s \qquad \forall\, \boldsymbol{v} \in \mathrm{Vertex}(\mathbb{S}_v)\,. \quad (6)$$

The rotation matrix $\mathbf{R}_2(\psi_p(t))$ is equal to the upper-left $\mathbb{R}^{2\times 2}$ of (A.3) in Appendix A. The set $\mathbb{S}_s$ is generated

regularly and consists of the eight edges made up of landmasses in the map that are closest to the vessel in order to form a convex set. Including more edges increases the accuracy of the inequality constraints, but negatively affects run time, and we have found eight to be a good compromise.

The thrusters on *milliAmpere* are each limited in the amount of thrust they are able to produce, so we place limits on the norms of each individual thruster output:

$$f_{xi}(t)^2 + f_{yi}(t)^2 \leq f_{\max}^2, \quad i \in \{1, 2\}\,. \tag{7}$$

There are also limits on the states $\boldsymbol{x}_p$, i.e.

$$\boldsymbol{x}_{lb} \leq \boldsymbol{x}_p(t) \leq \boldsymbol{x}_{ub}\,, \tag{8}$$

which ensure that the OCP does not plan trajectories with out-of-bounds velocities. The limits are only in effect for the velocities in surge and sway ($\pm 1.0\,\mathrm{m\,s^{-1}}$) and the yaw rate ($\pm 5\,^\circ\,\mathrm{s^{-1}}$).

As noted, all these constraints are softened with slack variables to ensure feasibility when e.g. a disturbance pushes the vessel's state outside the velocity limits or the collision avoidance criterion. The constraints are gathered in a single vector, giving the inequality constraint vector in (1c).

3.4 Trajectory-tracking DP controller

The planned state trajectory and its derivative from the solution of (1) are used as reference values for a trajectory-tracking DP controller, which was already implemented on *milliAmpere* before we added the trajectory planner. There are several reasons for preferring this approach instead of directly using the thruster commands from the solution of (1):

- The planner does not account for drift, disturbances or modeling errors, while the tracking controller does so through feedback.
- While the planner is iteration-based with no formal performance guarantees, the tracking controller provides a robust bottom layer that acts also as a safety measure.
- The sampling rate of the planner is too low to stabilize the vessel on its own.

The tracking controller is based on proportional-integral-derivative (PID) action with feed-forward terms from both velocity and acceleration:

$$\boldsymbol{\tau}_c(t) = \boldsymbol{\tau}_{\mathrm{ff}}(t) + \boldsymbol{\tau}_{\mathrm{fb}}(t)\,. \tag{9}$$

The feed-forward term is

$$\boldsymbol{\tau}_{\mathrm{ff}}(t) = \mathbf{M}_p \dot{\boldsymbol{\nu}}_p(t) + \mathbf{D}_p(\boldsymbol{\nu}_p(t))\boldsymbol{\nu}_p(t)\,, \tag{10}$$

with details in Appendix A. An issue with this feed-forward term is that it doesn't include coupling Coriolis or damping effects, which may degrade its performance. This discrepancy is left for the feedback to handle. The PID feedback is

$$\boldsymbol{\tau}_{\mathrm{fb}}(t) = -\mathbf{R}(\psi(t))^\top \cdot \left(\mathbf{K}_p \tilde{\boldsymbol{\eta}}(t) + \int_0^t \mathbf{K}_i \tilde{\boldsymbol{\eta}}(\tau)\,\mathrm{d}\tau + \mathbf{K}_d \dot{\tilde{\boldsymbol{\eta}}}(t) \right), \tag{11}$$

with $\tilde{\boldsymbol{\eta}}(t) = \boldsymbol{\eta}(t) - \boldsymbol{\eta}_p(t)$. The controller gains are $\mathbf{K}_p = \mathrm{diag}\{100, 100, 200\}$, $\mathbf{K}_i = \mathrm{diag}\{10, 10, 20\}$ and $\mathbf{K}_d = \mathrm{diag}\{1000, 1000, 1500\}$ with units that transform the respective elements to force and moment units. The integrator term in (11) has an anti-windup condition, limiting its contribution to $\pm[150\,\mathrm{N}, 150\,\mathrm{N}, 200\,\mathrm{N\,m}]^\top$.

The control command $\boldsymbol{\tau}_c(t)$ is sent to *milliAmpere*'s thrust allocation system, detailed in (Torben et al., 2019), which sends commanded actuator azimuth angles and propeller speeds to the actuators.

3.5 Design tradeoffs

In designing the docking system, it has been necessary to compromise between optimality, performance and robustness. One of the compromises was to separate trajectory planning and motion control. While it would be possible to run the trajectory planner as an MPC and use the inputs from its solution directly, separation gives several advantages:

- A PID controller accounts for steady-state disturbances and corrects for modeling errors, as opposed to the MPC approach.
- Using a high-rate feedback controller allows us to run the planner at a low rate, even though the vessel's dynamics are quite fast. The planner has run-times between 0.3 and 0.7 s, which would make it difficult to stabilize the vessel.
- Having a trajectory-tracking controller as the bottom control layer makes the docking system more robust to situations where the solver fails to find a feasible solution.

Choosing this hybrid structure, where we separate planning from motion control, we have achieved flexibility in the trajectory planner, disturbance rejection through feedback, and robustness to failures in the planning level.

4. EXPERIMENTAL RESULTS

Experiments were performed with the *milliAmpere* passenger ferry in confined waters in Trondheim, Norway on October 18, 2019. The weather conditions were calm with winds of $2\,\mathrm{m\,s^{-1}}$ to $3\,\mathrm{m\,s^{-1}}$ and rare gusts of $5\,\mathrm{m\,s^{-1}}$. The vessel is highly susceptible to wind disturbances, due to its large cross-sectional area above water and low underwater profile. The confined waters protect against waves and currents, however, the shallow depth of *milliAmpere*'s thrusters causes the thrust wake to disturb the hull when operating close to quay, as is the case in the final docking stage.

To test the docking system, we piloted the ferry to an initial pose around 40 m away from the docking pose, and activated the docking system once we came to a standstill. The trajectory planner then calculated state trajectories at a rate of 0.1 Hz towards the docking pose. A higher rate caused frequent resetting of the error between the planned and measured poses, limiting the effect of the feedback controller (11). A lower rate would limit the trajectory planner's ability to take into account new information. Since the trajectory planner calculates a safe trajectory towards the docking pose, a rate of 0.1 Hz is a well-functioning compromise. Before every run of the planner, an algorithm quickly calculated a new convex area

$\mathbb{S}_s$ based on the vessel's current position, which served as collision-avoidance constraints in the OCP (5). State measurements, the planned trajectory and its derivative were fed to the DP controller at a rate of 10 Hz. This is sufficient for motion control, since the vessel's dynamics are much slower.

A bird's eye view of the resulting trajectory is seen in Figure 4, with full-state trajectories in Figure 8. As is seen in Figure 4, *milliAmpere* is able to safely navigate to the docking pose by the help of the docking system.

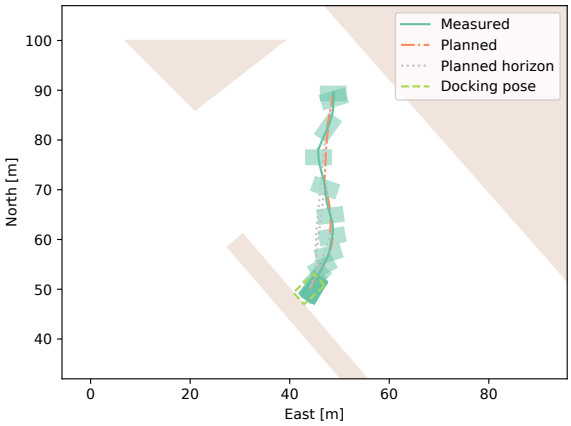


Fig. 4. Overview of *milliAmpere*'s trajectory during a docking experiment. The vessel's pose is depicted at 5 s intervals with green rectangles. The measured position is drawn in solid green, while the active planned reference is in dash-dotted orange. The dotted gray lines show the trajectory planner's reference for the entirety of each planning horizon, also after a new solution is calculated. The docking pose is marked with a rectangular bright green dashed outline.

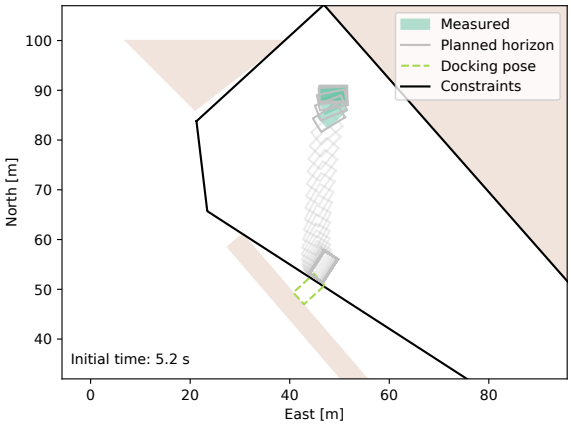


Fig. 5. Planned and measured positions from the first step of the planner. In 2 s intervals, the plot shows the entire planned pose trajectory as gray outlines, and the first 10 s of the measured pose as green rectangles. The black solid polyhedron shows the current convex area that represents the spatial constraints from (6).

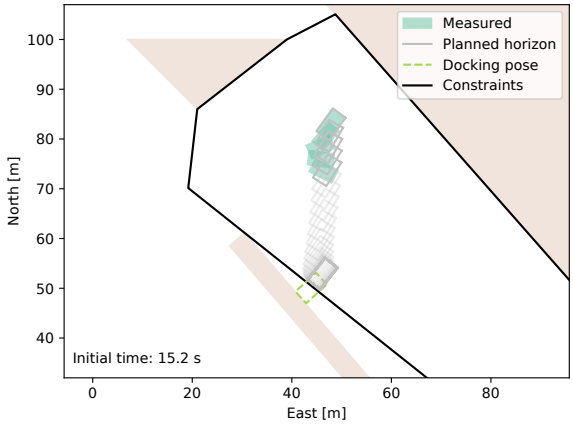


Fig. 6. As Figure 5, but at the second planning step. In this step we see that the tracking controller struggles to follow the heading commands. We believe this is due to *milliAmpere*'s lack of natural stability in heading, as well as due to poor performance of the DP controller at high velocities.

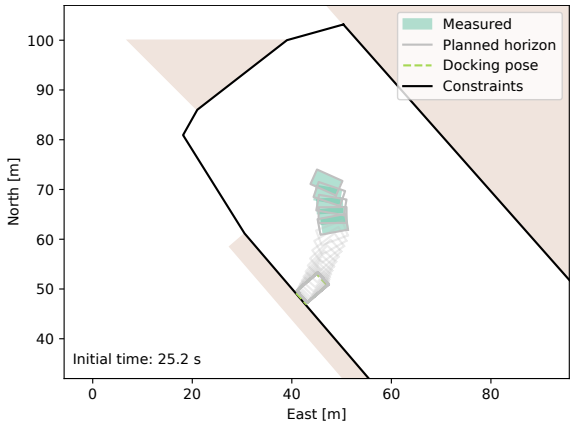


Fig. 7. As Figure 5, but at the third planning step. At this slow speed, the tracking controller is able to follow the planned trajectory well.

The trajectory is collision-free and slows down nicely when approaching the quay. In the course of the experiment there were 13 re-planning steps. Figures 5 through 7 show the planned trajectories at the first, second and third steps, respectively. The figures also show the convex area that the trajectory planner uses for spatial constraints. Due to how the convex-set algorithm works, the first step does not include the docking pose in its permissible set, so the trajectory planner generates a trajectory towards the edge of its constraints. The vessel is able to closely follow this trajectory until the second step. Here we see that the vessel's heading angle is failing to track the planned one. We believe this is due to *milliAmpere*'s lack of stability in heading, and due to poor tuning of the DP controller, which fails to handle tracking of heading and yaw rate at high speeds. In the third step, the trajectory planner is able to plan all the way towards the docking pose, and

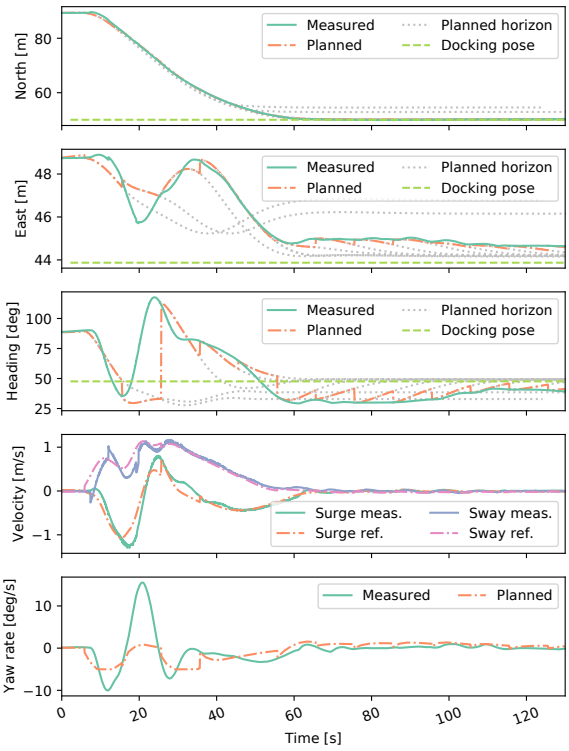


Fig. 8. Measured pose and velocity states during the docking experiments, along with reference trajectories and docking pose. As in Figure 4, we include the full horizon of the planned trajectories in dotted gray lines.

the vessel is able to track the commanded trajectory well, since the speed has decreased.

Figure 8 shows the state trajectories for pose and velocities over time. It can be seen that the planned trajectories are tracked tightly for the linear positions and velocities. A notable observation is that the first two plans do not converge to the docking pose, due to the convex area not including the docking pose. This is corrected as the vessel approaches the harbor. The heading angle and yaw rate are not converging as well as the linear velocities, especially at high speeds, as seen from the figure. Additionally, due to the periodic resetting of the planned trajectory to the current vessel state, integration is slow in the DP controller, causing steady-state disturbance rejection to be poor towards the end of the trajectory.

From Figure 9, we see that the solution times of the trajectory planner are in the 0.3 s to 0.7 s range, which is fast enough to be considered real-time feasible when run at a period of 10 s. These results are repeatable when docking from and to the same pose, and similar results are also seen when docking from other locations.

5. CONCLUSIONS AND FUTURE WORK

We have demonstrated the capabilities for docking an ASV using an OCP-based trajectory planner in combination

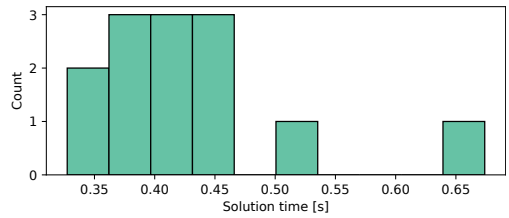


Fig. 9. Histogram of solution times of the trajectory planner.

with a DP controller. The solution is tested experimentally in confined waters in Trondheim, Norway, and produces safe maneuvers. The maneuvers avoid collision with static obstacles and complete the docking phase, ending up in a position adjacent to the dock, ready to moor. We have shown that the combination of an OCP-based trajectory planner and a tracking controller is suitable for the docking problem. The method is also general, requiring only a geographic map of sufficient resolution of the harbor environment. This map may be known a priori, but may also be adjusted with exteroceptive sensors, enabling extensions to the method with camera, lidar and radar systems, e.g. using simultaneous localization and mapping techniques.

The experiments have uncovered several possibilities for improvement which are points for future work. A main conclusion is that although we are able to combine a trajectory planner with an existing tracking controller, the tracking controller must be well-designed and tuned for the combination to function satisfactorily. The following points are considered as future work:

- Improve the tuning of the existing DP controller in order to better track the reference trajectory.
- Include coupling effects in the feed-forward term of the DP controller.
- Investigate other trajectory-tracking controllers.
- Adjust the cost function so that the trajectory planner produces maneuvers that are more consistent with a harbor pilot's experience with docking.
- Adjust the trajectory planner to produce more conservative trajectories.
- Develop a disturbance estimator that can provide the trajectory planner with valuable information.

Future work also includes integrating the docking system in a control structure that handles all the phases of a ferry transport. The next item in our research is to integrate systems for the undocking and transit phases. For the undocking phase, the approach presented in this paper is well suited. For the transit phase, we look to integrate a version of the method from (Bitar et al., 2019), which can bring the vessel to a location suitable for docking.

REFERENCES

Bitar, G., Vestad, V.N., Lekkas, A.M., and Breivik, M. (2019). Warm-started optimized trajectory planning for ASVs. In *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*, 308–314. arXiv:1907.02696 [eess.SY].

Hong, Y.H., Kim, J.Y., Oh, J.H., Lee, P.M., Jeon, B.H., and Oh, K.H. (2003). Development of the homing and docking algorithm for AUV. In *Proceedings of*

the 13th International Offshore and Polar Engineering Conference, Honolulu, Hawaii, USA.

Martinsen, A.B., Lekkas, A.M., and Gros, S. (2019). Autonomous docking using direct optimal control. In *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*, 97–102. arXiv:1910.11625 [eess.SY].

Mizuno, N., Uchida, Y., and Okazaki, T. (2015). Quasi real-time optimal control scheme for automatic berthing. In *Proceedings of the 10th IFAC MCMC, Copenhagen, Denmark*, 305–312.

Pedersen, A.A. (2019). *Optimization Based System Identification for the milliAmpere Ferry.* Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. URL `http://hdl.handle.net/11250/2625699`.

Rae, G.J.S. and Smith, S.M. (1992). A fuzzy rule based docking procedure for autonomous underwater vehicles. In *Proceedings of OCEANS, Newport, RI, USA.*

Teo, K., Goh, B., and Chai, O.K. (2015). Fuzzy docking guidance using augmented navigation system on an AUV. *IEEE Journal of Oceanic Engineering*, 40(2), 349–361.

Torben, T.R., Brodtkorb, A.H., and Sørensen, A.J. (2019). Control allocation for double-ended ferries with full-scale experimental results. In *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*, 45–50.

Tran, V.L. and Im, N. (2012). A study on ship automatic berthing with assistance of auxiliary devices. *International Journal of Naval Architecture and Ocean Engineering*, 4(3), 199–210.

Appendix A. MATHEMATICAL VESSEL MODELS

In this work, we have used three separate models for the *milliAmpere* vessel, respectively for simulation, planning in the OCP, and for trajectory tracking with the DP controller. All of them are based on the surge-decoupled three-degree-of-freedom model from (Pedersen, 2019). The models use the state vector

$$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{\eta}^\top & \boldsymbol{\nu}^\top\end{bmatrix}^\top, \tag{A.1}$$

with $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times S$ being the pose states position north and east of an origin, and heading angle (yaw), respectively. The velocity vector $\boldsymbol{\nu} = [u, v, r]^\top$ contains body-fixed surge velocity, sway velocity and yaw rate, respectively. The kinematic relationship between the pose and velocity is

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu}\,, \tag{A.2}$$

where

$$\mathbf{R}(\psi) = \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix} \tag{A.3}$$

is the kinematic rotation matrix. The kinetic equations that describe the propagation of $\boldsymbol{\nu}$ are different for the three applications.

A.1 Simulation model

When we simulated the approach prior to running full-scale experiments, we used the surge-decoupled three-degree-of-freedom model from (Pedersen, 2019). That model has the form

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}(\boldsymbol{\alpha}, \boldsymbol{n})\,, \tag{A.4}$$

where $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the positive definite system inertia matrix, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the skew symmetric Coriolis and centripetal matrix, and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the positive definite damping matrix. The force vector $\boldsymbol{\tau} \in \mathbb{R}^3$ is a function of the thrusters' azimuth angles $\boldsymbol{\alpha} = [\alpha_1, \alpha_2]^\top$ and their propeller speeds $\boldsymbol{n} = [n_1, n_2]^\top$. These values are modeled dynamically based on commanded values, with details in (Pedersen, 2019).

A.2 Planning model

For the dynamic constraints in the OCP, we use a simplified version of (A.4):

$$\mathbf{S}\,\mathbf{M}_p\dot{\boldsymbol{\nu}}_p + \mathbf{C}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p + \mathbf{D}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p = \boldsymbol{\tau}_p(\boldsymbol{u}_p)\,, \tag{A.5}$$

where $\mathbf{M}_p$, $\mathbf{C}_p$ and $\mathbf{D}_p$ are diagonalized versions of $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{D}$ from (A.4), respectively. The matrices are

$$\mathbf{M}_p = \operatorname{diag}\{m_{11}, m_{22}, m_{33}\} > 0\,, \tag{A.6}$$

$$\mathbf{C}_p(\boldsymbol{\nu}_p) = \begin{bmatrix}0 & 0 & -m_{22}v_p\\ 0 & 0 & m_{11}u_p\\ m_{22}v_p & -m_{11}u_p & 0\end{bmatrix} \tag{A.7}$$

and

$$\mathbf{D}_p(\boldsymbol{\nu}_p) = \operatorname{diag}\{d_{11}(u_p), d_{22}(v_p), d_{33}(r_p)\} > 0\,, \tag{A.8}$$

where

$$d_{11}(u_p) = -X_u - X_{|u|u}|u_p| - X_{u^3}u_p^2 \tag{A.9a}$$

$$d_{22}(v_p) = -Y_v - Y_{|v|v}|v_p| - Y_{v^3}v_p^2 \tag{A.9b}$$

$$d_{33}(r_p) = -N_r - N_{|r|r}|r_p|\,. \tag{A.9c}$$

The coefficient matrix

$$\mathbf{S} = \operatorname{diag}\{2.5, 2.5, 5.0\} \tag{A.10}$$

is factored into (A.5) to amplify the inertia, making the planned trajectories more sluggish.

The dynamic thruster model from (Pedersen, 2019) is excluded from the OCP in order to keep the run times down. Instead, forces from *milliAmpere*'s two thrusters are decomposed in the surge and sway directions, and used directly as inputs:

$$\boldsymbol{u}_p = \begin{bmatrix}f_{x1} & f_{y1} & f_{x2} & f_{y2}\end{bmatrix}^\top, \tag{A.11}$$

where f_{x1} represents a force in surge direction from thruster 1, f_{y2} represents a force in sway direction from thruster 2, etc. This is mapped to forces and moments in surge, sway and yaw by the function

$$\boldsymbol{\tau}_p(\boldsymbol{u}_p) = \begin{bmatrix}1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 0 & l_1 & 0 & l_2\end{bmatrix}\boldsymbol{u}_p\,. \tag{A.12}$$

The parameters $l_1, l_2 \in \mathbb{R}$ are the distances from the vessel's origin to its thrusters.

A.3 Tracking controller model

For the feed-forward terms in the DP controller, we also use a simplified version of the simulation model (A.4):

$$\mathbf{M}_p\boldsymbol{\nu}_p + \mathbf{D}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p = \boldsymbol{\tau}_{\text{ff}}\,. \tag{A.13}$$

The DP controller was originally developed for station keeping, and does not contain the $\mathbf{C}$ matrix. Otherwise, the matrix values in (A.13) are equal to those in the planning model (A.5).

Paper F Two-stage optimized trajectory planning for ASVs under polygonal obstacle constraints: theory and experiments

Published paper by **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". In: *IEEE Access* 8 (2020), pp. 199953–199969. DOI: 10.1109/ACCESS.2020.3035256.

Bibliography entry [13].

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 15, 2020, accepted October 23, 2020, date of publication November 2, 2020,
date of current version November 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035256

Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments

GLENN BITAR, ANDREAS B. MARTINSEN, ANASTASIOS M. LEKKAS, (Member, IEEE), AND MORTEN BREIVIK, (Member, IEEE)

Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway

Corresponding author: Glenn Bitar (glennbitar@outlook.com)

This work was supported in part by the Research Council of Norway under Project 269116, and in part by the Centres of Excellence Funding Scheme under Project 223254.

ABSTRACT We propose a method for energy-optimized trajectory planning for autonomous surface vehicles (ASVs), which can handle arbitrary polygonal maps as obstacle constraints. The method comprises two stages: The first is a hybrid A* search that finds a dynamically feasible trajectory in a polygonal map on a discretized configuration space using optimal motion primitives. The second stage uses the resulting hybrid A* trajectory as an initial guess to an optimal control problem (OCP) solver. In addition to providing the OCP with a warm start, we use the initial guess to create convex regions encoded as halfspace descriptions, which converts the inherent nonconvex obstacle constraints into a convex and smooth representation. The OCP uses this representation in order to optimize the initial guess within a collision-free corridor. The OCP solves the trajectory planning problem in continuous state space. Our approach solves two challenges related to optimization-based trajectory planning: The need for a dynamically feasible initial guess that can guide the solver away from undesirable local optima and the ability to represent arbitrary obstacle shapes as smooth constraints. The method can take into account external disturbances such as wind or ocean currents. We compare our method to two similar trajectory planning methods in simulation and have found significant computation time improvements. Additionally, we have validated the method in full-scale experiments in the Trondheim harbor area.

INDEX TERMS Autonomous vehicles, collision avoidance, marine vehicles, motion planning, polygonal collision-avoidance constraints, trajectory optimization, trajectory planning.

I. INTRODUCTION

In marine applications, we see efforts to increase the level of autonomy in research, defense, and commercial applications. Motivated by benefits to costs, safety, and environmental impact, many actors consider using autonomous vessels in their operations. In 2018, both Wärtsilä and Rolls-Royce Marine (acquired by Kongsberg Maritime) demonstrated autonomous capabilities with the ferries *Folgefonn* and *Falco*, respectively.[1] Both tests included automatic transit and docking. Another example of commercial use of maritime autonomous technology is when the Japanese shipping company NYK completed the world's first maritime autonomous surface ship trial in 2019.[2]

An essential part of an autonomous marine system is path and trajectory planning, where the goal is to plan how the vessel will move from its start location to the goal location. Path planning finds a sequence of collision-free configurations without temporal constraints, while trajectory planning adds temporal constraints, often via a time-parametrized state trajectory. Our interests lie within energy-optimized operations, and since energy consumption is highly sensitive to velocity, we focus on trajectory planning.

The associate editor coordinating the review of this manuscript and approving it for publication was Heng Wang.

[1] https://www.maritime-executive.com/article/rolls-royce-and-wartsila-in-close-race-with-autonomous-ferries (accessed September 14, 2020).

[2] https://www.nyk.com/english/news/2019/20190930_01.html (accessed August 31, 2020).

A. BACKGROUND AND RELEVANT WORK

Maritime agencies and research institutions actively research autonomous technology for, e.g., underwater operations for ocean mapping and monitoring [1], and autonomous transportation, focusing on the international regulations for preventing collisions at sea (COLREGs) [2]. Seto [3] gives an overview of autonomous technologies for maritime systems, and Pendleton *et al.* [4] give an overview of autonomy in vehicles in general. Pendleton Path and trajectory planning is a crucial technology for enabling autonomy at sea.

In robotics, there are numerous methods developed for path and trajectory planning. A general introduction to path planning is written by LaValle [5], who looks at the topic from the perspective of computer science while introducing widespread notation and nomenclature. Wolek and Woolsey [6] give an overview of model-based approaches to path planning for ground, surface, underwater, and air vehicles. We can coarsely divide planning methods into *roadmap methods* that explore points in the configuration space that, when connected, build a path between start and goal, and *optimization-based methods* that produce connected paths or trajectories using analytical or approximate optimization. Some advantages of roadmap methods include quickly finding the global solution of a path planning problem, and they allow for flexible obstacle representations, e.g., polygonal constraints. On the other hand, roadmap methods are discrete and are not generally able to find an optimal path or trajectory in a continuous domain. Optimization-based methods are often slower and subject to finding local optima. However, they naturally search in the continuous domain. Additionally, gradient-based methods for solving optimization problems require continuously differentiable representations of constraints, restricting how we can represent obstacles.

A simple example of roadmap methods is the A* search algorithm [7]. A* is a graph search algorithm commonly used as a path planner by discretizing a continuous map, often into a uniform, rectangular grid. A* quickly provides a piecewise linear path from start to goal. A more involved roadmap method comes from Candeloro *et al.* [8], where the authors discretize a map using a Voronoi diagram, subsequently refining and smoothing the result to give a curvature-continuous path. These methods are fast, but lack dynamic feasibility[3], and can only be optimal in terms of the employed map discretization. Roadmap methods also include sampling-based methods. These methods explore random points to build a roadmap between start and goal. Examples include the probabilistic roadmap [9], as well as rapidly-exploring random trees [10] and variations of those. Sampling-based methods are shown to be useful for planning in high-dimensional configuration spaces, where combinatorial roadmap methods often run into the so-called *curse of dimensionality* [11].

Model-based optimization-based methods are researched in automotive, aerial, and marine applications to create dynamically feasible paths or trajectories. Optimization-based methods are sometimes used to refine the result of a roadmap search or used as the primary tool to plan a trajectory. In [12]–[14], the authors present optimization-based trajectory planning methods that use smooth representations of rectangles and ellipses to approximate the obstacle map. This type of representation makes the optimization problem feasible to solve using gradient-based methods. However, there is an impractical tradeoff between the representation accuracy and number of constraints in the optimization problem. Additionally, these shapes may not be generic enough to represent detailed obstacle maps. By reformulating the obstacle avoidance constraint and introducing auxiliary optimization variables, Zhang *et al.* [15] have developed an alternative method for representing obstacles. This method allows the encoding of arbitrary convex polygons as smooth optimization constraints by introducing auxiliary optimization variables. The method works well for a low number of obstacles, but the optimization problem grows significantly with the number of obstacles and the number of polygon edges, to the point where it is not feasible to use it for marine applications with detailed obstacle maps. Bergman *et al.* [16] propose to bypass the inherent non-convexity of static obstacle avoidance by calculating a series of convex polytopes where their vehicle is allowed to move. The method gives smooth, convex obstacle avoidance constraints for their optimization-based planner, but lacks consideration of environmental disturbances. An optimization-based trajectory planning method for autonomous driving developed by Chen *et al.* [17] can represent polygonal obstacle constraints. Their method is based on linear quadratic control with an iterative optimization solver. A prerequisite for their method is an initial dynamically feasible trajectory in order to perform the optimization. However, their method does not provide a way of determining such a trajectory. This issue is common with optimization-based methods, and without an initial guess, they are prone to locking into solutions that represent undesirable local optima, i.e., solutions that may be far away from the globally optimal solution, as demonstrated in, e.g., [14]. In that example, the optimization-based planner finds a poor solution in the absence of a helpful initial guess. Depending on the objective function, finding a good initial guess to warm-start an optimization-based planner can be straightforward. In the case of finding a minimum-distance path, simple roadmap-based methods may quickly find paths in the discrete domain that lie close to the optimal solution in the continuous domain. Optimization-based methods can use this type of path as an initial guess. For energy-based objective functions, for instance, or when introducing dynamic constraints, creating feasible trajectories to use as initial guesses is more challenging, and suggests alternative approaches.

[3]We use the term "dynamically feasible" to indicate that a trajectory satisfies dynamic constraints in the form of model-based differential equations. A path that consists of a smoothed roadmap is usually feasible in terms of specified a turning radius. This turning radius is dependent on vessel speed, and the path is thus not *dynamically* feasible since it is not based on a model that includes speed.

Zhang *et al.* [15] propose using the hybrid A* algorithm [18] to find such a trajectory for an optimization-based solver. Their application is autonomous parking of a car, described with a dynamical model, and using a cost function that blends minimum time and control effort. A simplified dynamical model and cost function is used in the hybrid A* search stage, and the search solution is used as an initial guess for the optimization-based planner. The method does not take into account external disturbances. Bergman *et al.* [16] have developed a receding-horizon optimization-based planner warm-started by using a graph search method. The graph search method works on a lattice of a marine vessel's discretized state space with optimal state transitions. To facilitate motion in confined harbor areas, the authors use a cost function that blends distance to obstacles, minimum time, control effort, and control smoothness. Zhang *et al.* [19] and Meng *et al.* [20] propose optimization-based trajectory planning methods for autonomous driving that utilize roadmap methods to generate nominal trajectories for geometrical paths and subsequently use optimization to improve them. In both papers, speed profiles are handled separately from the geometrical path planning.

B. CONTRIBUTIONS

We have developed a method that plans energy-optimized trajectories in an environment defined by polygonal obstacles for an autonomous surface vehicle (ASV) under the influence of external disturbances. Our method is based on continuous optimal control, and the optimal control problem (OCP) solver is warm-started by the solution of a hybrid A* search algorithm. The method's proposed use case is to plan an ASV voyage's transit stage before the voyage starts. The method handles only static obstacles, and is thus suitable for use as the top layer in a hybrid collision avoidance scheme, as proposed in [2], [21], [22]. Figure 1 shows a high-level block diagram of the trajectory planning method. The main differences between our method and the planner described in [16] are that we use a hybrid A* search to calculate the initial guess, which allows us to account for estimated external disturbances, such as wind. Additionally, we use an alternative method to calculate the convex envelopes in preparation for the trajectory optimization stage. Like the method in [15], we also use hybrid A* to generate an initial guess before optimizing. However, we propose an alternative obstacle representation, which scales more efficiently with the number of polygons and polygon edges in the obstacle map, in terms of the number of optimization variables. Our method shares similarities with [23] as well, where the workspace is decomposed into triangular cells to account for static, polygonal obstacles, and an optimization-based search finds sub-trajectories in each of the triangular cells. However, that method does not include an initial guess to warm-start the OCP solver.

Our contributions are as follows:

- We have extended the hybrid A* search developed by Dolgov *et al.* [18] to the ASV application by using an energy-based cost function that depends on the velocity relative to external disturbances such as wind.
- We use a trajectory of pose, velocity, and force from the hybrid A* solution as an initial guess to a general OCP solver.
- In the OCP, we utilize a sequence of convex polygons to generate a state corridor in a nonconvex obstacle map. This representation of obstacles causes the OCP's obstacle avoidance constraint to be convex, rather than nonconvex. Additionally, it allows us to easily use polygonal obstacle maps in the gradient-based OCP solver, which is generally hard due to their piecewise linear and nonconvex nature.
- We have compared our method to similar trajectory planning methods and found significant improvements in terms of run time, with equivalent energy use.
- We have performed full-scale experiments that have validated our method based on the experimental vessel's capability to track the resulting trajectory.

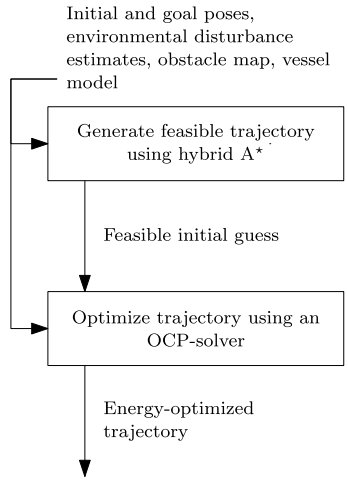


FIGURE 1. A block diagram of the high-level functionality of our proposed trajectory planning method.

C. OUTLINE

In Section II we cover preliminary information about notation and vessel modeling. Sections III and IV present the development of our trajectory planning method. In Section III, we describe the hybrid A* method that generates the initial guess. The section covers the generation of motion primitives, two different search heuristics, and the search algorithm itself. In Section IV, we present the OCP, how we convert the obstacle map to a sequence of convex polygons, the transcription of the OCP to a nonlinear program (NLP), and how we solve the NLP Section V contains simulations and comparisons to other trajectory planning methods. The results are compared in quantitative measures of planning time and energy-usage when tracking the trajectories. In Section VI, we present results from full-scale experiments, which serve as

validation of the method and show how well the experimental vessel can track the produced trajectories. Section VII gives concluding remarks.

II. PRELIMINARIES

A. NOTATION

From LaValle [5], we have widely used notation related to *path* planning. As opposed to trajectories, a path places no temporal constraints on the following vehicle. Except for this, the two topics of planning paths and trajectories are similar. We let $\mathcal{W} := \mathbb{R}^2$ denote the world that contains our vessel and obstacles. The union of obstacles is $\mathcal{O} \subset \mathcal{W}$. The free workspace is defined to be $\mathcal{W}_{\text{free}} := \mathcal{W} \setminus \mathcal{O}$.

Our vessel lives in $\mathcal{W}$, but its configuration is better described in the configuration space

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & \psi \end{bmatrix}^\top \in \mathcal{C} := \mathbb{R}^2 \times S. \tag{1}$$

Here, x and y are the vessel's position coordinates North and East of some origin, respectively, and ψ is its heading angle relative to North. The position coordinates refer to the vessel's center of gravity, which is at its centroid. The vector $\boldsymbol{\eta}$ is referred to as the vessel's *pose*. We denote its footprint in the workspace as a set of points $\mathcal{A}(\boldsymbol{\eta}) \subset \mathcal{W}$, which defines the vessel's shape. The set of noncolliding configurations is thus

$$\mathcal{C}_{\text{free}} := \{\boldsymbol{\eta} \in \mathcal{C} \mid \mathcal{A}(\boldsymbol{\eta}) \cap \mathcal{O} = \emptyset\}. \tag{2}$$

Most path planning algorithms operate on a discretized version of the configuration space, denoted by $\mathcal{C}_d \subset \mathcal{C}$. In our work we uniformly discretize the configuration space on a grid with resolution

$$\boldsymbol{r}_{\mathcal{C}} := \begin{bmatrix} r_p & r_p & r_h \end{bmatrix}^\top, \tag{3}$$

where $r_p > 0$ is the positional resolution and $r_h > 0$ is the angular heading resolution. Similarly, the discrete free configuration space is denoted $\mathcal{C}_{d,\text{free}}$. While points in the continuous configuration space are denoted by $\boldsymbol{\eta}$, we use a tilde for points in the discrete configuration space: $\tilde{\boldsymbol{\eta}}$. The mapping from $\mathcal{C}$ to $\mathcal{C}_d$ is denoted $\text{KEY} : \mathcal{C} \mapsto \mathcal{C}_d$ and is done by rounding $\boldsymbol{\eta}$ to its closest multiple of $\boldsymbol{r}_{\mathcal{C}}$.

The formal goal of path planning is to find a continuous path, entirely in $\mathcal{C}_{\text{free}}$, from a start pose $\boldsymbol{\eta}_0 \in \mathcal{C}_{\text{free}}$ to a goal pose $\boldsymbol{\eta}_f \in \mathcal{C}_{\text{free}}$. In discrete algorithms, the paths are often piecewise linear, with connections on $\mathcal{C}_{d,\text{free}}$. Generally, this problem has many solutions, however, we usually also associate the problem with a definition of an *optimal* path, e.g., the shortest. In trajectory planning, the goal is similar, but we have additional kinodynamic constraints to satisfy, e.g., a set of time-parametrized differential equations. Section II-B introduces such constraints in the form of a mathematical vessel model.

B. ASV MODELING

Our ASV is modeled as a surge-decoupled three-degree-of-freedom displacement vessel, with the state vector

$$\boldsymbol{x} := \begin{bmatrix} \boldsymbol{\eta}^\top & \boldsymbol{\nu}^\top \end{bmatrix}^\top \in \mathcal{X} := \mathcal{C} \times \mathbb{R}^3 \tag{4}$$

with $\boldsymbol{\eta}$ being the pose described in (1), and $\boldsymbol{\nu} := [u, v, r]^\top$ the body-fixed velocity vector, where u is the surge velocity, v sway velocity and r yaw rate. The state space is denoted $\mathcal{X}$. The kinematic relationship between the pose and velocity is described by

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu}, \tag{5}$$

where

$$\mathbf{R}(\psi) := \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

The kinetics of the ASV is described by

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}. \tag{7}$$

This notation is widely used for vessel models in the maritime control literature [24]. Here, $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the positive definite system inertia matrix, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the skew-symmetrix Coriolis and centripetal matrix, and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the positive definite damping matrix. The force vector $\boldsymbol{\tau} = [X, Y, N]^\top \in \mathcal{T} \subset \mathbb{R}^3$ are the control forces produced by the ASV's actuators in surge, sway and yaw, respectively, where $\mathcal{T}$ denotes the space of valid inputs. These are in turn governed by dynamical models of the actuators. For simulation purposes we include those models, but for planning and control we have simplified the model to let $\boldsymbol{\tau}$ be directly controllable. The environmental forces $\boldsymbol{\tau}_{\text{env}}$ can come from wind, ocean current and waves. We have only modeled wind effects for our experimental vessel, and the environmental forces are a function of relative wind velocity:

$$\boldsymbol{\tau}_{\text{env}} = \boldsymbol{\tau}_{\text{env}}(\psi, \boldsymbol{\nu}, \boldsymbol{V}_w), \tag{8}$$

where $\boldsymbol{V}_w \in \mathbb{R}^2$ is the wind velocity in North and East components. Matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{D}$, along with the actuator models, as well as a wind model are defined in [25].

The model is concatenated to

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{V}_w) \\ &:= \begin{bmatrix} \mathbf{R}(\psi)\boldsymbol{\nu} \\ \mathbf{M}^{-1}\big[-\big(\mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu})\big)\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}(\psi, \boldsymbol{\nu}, \boldsymbol{V}_w)\big] \end{bmatrix} \end{aligned} \tag{9}$$

for ease of reference when discussing OCPs later in the paper.

III. STAGE 1: GENERATING A DYNAMICALLY FEASIBLE INITIAL GUESS

As we mention in the introduction, our trajectory planning method comprises two stages. The entire method, its subcomponents, and their interconnections are illustrated in Figure 2. Each subcomponent will be described in this section and the next. Stage 1 of our method is to find a dynamically feasible trajectory using the hybrid A* search.

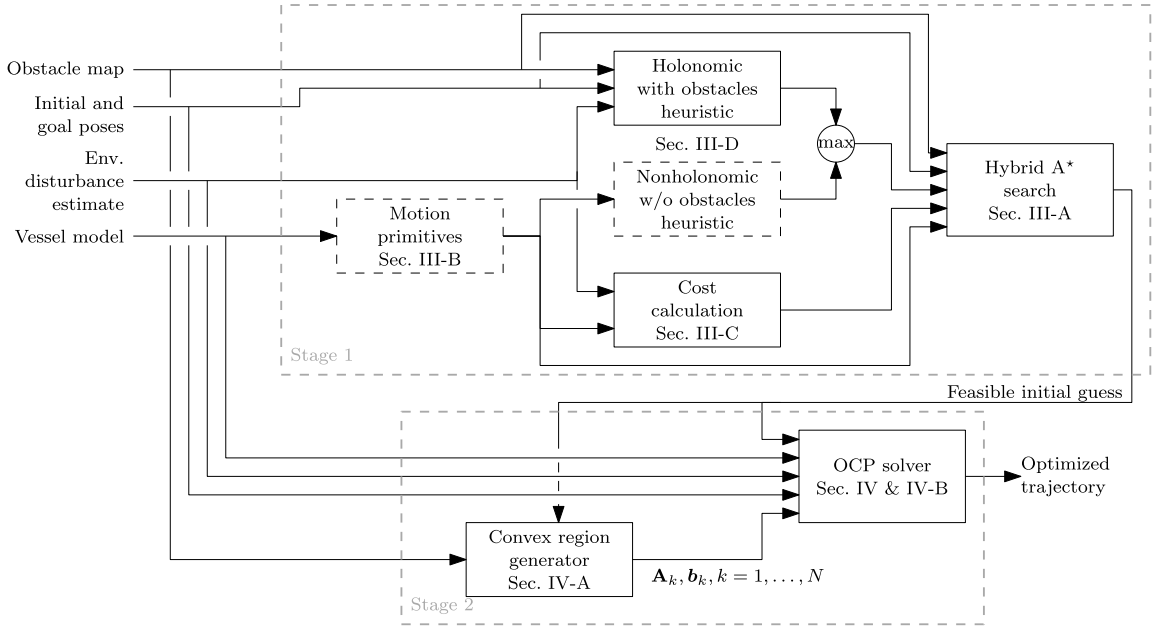


FIGURE 2. Block diagram of the trajectory planning method. Stage 1 refers to the generation of the initial guess, described in Section III, and Stage 2 refers to the trajectory optimization from Section IV.

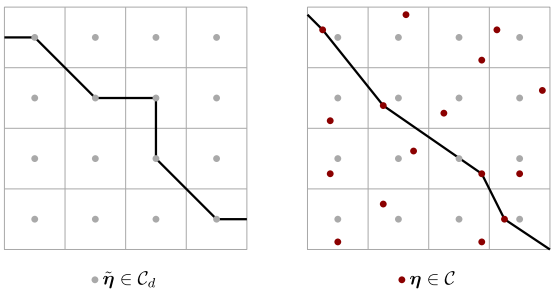


FIGURE 3. Comparison of traditional A* search space to the hybrid A* search space in a two-dimensional grid. To the left is the commonly found eight-connected uniform grid associated with A*, where states are associated with grid cell centers. To the right is the search space of hybrid A*, where states can lie anywhere in the cells.

A. HYBRID A*

Dolgov *et al.* [18] developed the hybrid A* algorithm to plan paths for autonomous cars. Hybrid A* is a variant of the well-known A* algorithm that captures continuous-state data in discrete search nodes. The search space is discretized, but a continuous state is associated with each discrete node, as illustrated in Figure 3. An advantage of the hybrid A* search space is that it does not require the connections between two states in different nodes to be exact, which allows us to be flexible when using motion primitives in the discrete search. A disadvantage is that the optimality from traditional A* is no longer strictly guaranteed due to the merging of continuous and discrete states.

Algorithm 1 is pseudocode for the hybrid A* search. Like an A* search, it uses a priority queue to keep track of the open set. In Algorithm 1, that functionality is maintained by the PUSH and POP functions. PUSH adds a key with a priority value to the open set O, while POP removes and returns the key with the lowest associated priority. The mappings STATE, COST, and PARENT keep track of continuous states, cost values, and parents associated with discrete keys $\tilde{\boldsymbol{\eta}} \in \mathcal{C}_d$. The mappings are updated as the search progresses. The function PRIMITIVES returns a set of motion primitives, COLLISION checks whether there is a collision, and HEURISTICS returns heuristic cost estimates. These functions are further described in sections III-B, III-D, and III-E, respectively.

B. MOTION PRIMITIVES

In the hybrid A* search algorithm, new configurations are discovered by propagating motion primitives from an existing configuration. A motion primitive is a dynamically feasible state trajectory between two configurations in $\mathcal{C}$. Dynamic feasibility, as discussed in Section II-A, is inherently satisfied by using motion primitives with trajectories that satisfy (9). While we search in $\mathcal{C}$, the trajectories are in $\mathcal{X}$, which means that to feasibly connect two configurations with state trajectories in $\mathcal{X}$, they must start and end with the same velocities.

Motion primitives with varying lengths and turn angles are precomputed using an OCP. During the search, the primitives are translated and rotated to fit with the originating configuration. The motion primitives used in our results are shown in Figure 4.

Algorithm 1 Hybrid A* Search Pseudocode

```
function HYBRID A*( η_0, η_f, V_w, O )
    η̃_0 ← KEY(η_0), η̃_f ← KEY(η_f)
    O ← ∅, C ← ∅
    PUSH(O, η̃_0, 0)
    STATE(η̃_0) ← η_0, COST(η̃_0) ← 0
    while O ≠ ∅ do
        η̃ ← POP(O)
        C ← C ∪ {η̃}
        if η̃ = η̃_f then
            return sequence from η̃_0 to η̃_f
        η ← STATE(η̃)
        for all P, c, η_n ∈ PRIMITIVES(η, V_w) do
            η̃_n ← KEY(η_n)
            if COLLISION(P, O) or η̃_n ∈ C then
                continue
            f ← COST(η̃) + c
            if η̃_n ∉ C ∪ O then
                COST(η̃_n) ← ∞
            if f < COST(η̃_n) then
                COST(η̃_n) ← f
                PARENT(η̃_n) ← η̃
                STATE(η̃_n) ← η_n
                O ← O \ {η̃_n}
                h ← f + HEURISTICS(η_n, η_f, V_w)
                PUSH(O, η̃_n, h)
    return error, no path found
```

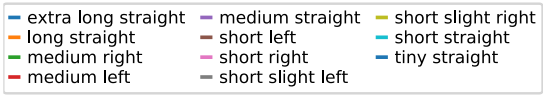


FIGURE 4. Motion primitives used in our results.

The OCP used to generate motion primitives is

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{\tau}(\cdot)} \int_0^{t_f} F(\boldsymbol{x}(\tau),\boldsymbol{\tau}(\tau))\mathrm{d}\tau \tag{10a}$$

$$\text{subject to } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{\tau}(t),\mathbf{0}_2) \quad t \in [0,t_f] \tag{10b}$$

$$\boldsymbol{x}_{lb} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{ub} \quad t \in [0,t_f] \tag{10c}$$

$$\boldsymbol{\tau}_{lb} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}_{ub} \quad t \in [0,t_f] \tag{10d}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{10e}$$

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f. \tag{10f}$$

The OCP is equal for every primitive, except for the final time t_f, the state bounds (10c) and the final condition (10f), all of which depend on the motion primitive length $L > 0$ and direction angle χ. The vessel is assumed to travel with a nominal speed U_{nom}, which in our results is $1.5\,\text{m}\,\text{s}^{-1}$. For a specific primitive defined by (L,χ), the parameters of (10) are

$$t_f = L/U_{\text{nom}} \tag{11a}$$

$$\boldsymbol{x}_{lb} = \begin{bmatrix} \min(0, L\cos\chi) \\ \min(0, L\sin\chi) \\ \min(0,\chi) \\ u_{lb} \\ -v_{ub} \\ -r_{ub} \end{bmatrix} \tag{11b}$$

$$\boldsymbol{x}_{ub} = \begin{bmatrix} \max(0, L\cos\chi) \\ \max(0, L\sin\chi) \\ \max(0,\chi) \\ u_{ub} \\ v_{ub} \\ r_{ub} \end{bmatrix} \tag{11c}$$

$$\boldsymbol{\tau}_{lb} = \begin{bmatrix} X_{lb} & -Y_{ub} & -N_{ub} \end{bmatrix}^\top \tag{11d}$$

$$\boldsymbol{\tau}_{ub} = \begin{bmatrix} X_{ub} & Y_{ub} & N_{ub} \end{bmatrix}^\top \tag{11e}$$

$$\boldsymbol{x}_0 = \begin{bmatrix} 0 & 0 & 0 & U_{\text{nom}} & 0 & 0 \end{bmatrix}^\top \tag{11f}$$

$$\boldsymbol{x}_f = \begin{bmatrix} L\cos\chi & L\sin\chi & \chi & U_{\text{nom}} & 0 & 0 \end{bmatrix}^\top. \tag{11g}$$

The values u_{lb}, u_{ub}, v_{ub} and r_{ub} are velocity bounds, and X_{lb}, X_{ub}, Y_{ub}, and N_{ub} are bounds on surge force, sway force, and yaw moment, respectively. Table 1 specifies the parameters (L,χ) in our results, and Table 2 gives the boundary values.

TABLE 1. Motion primitive parameters.

Name	Length L [m]	Turn angle χ [°]
extra long straight	200	0
long straight	100	0
medium right	50	30
medium left	50	−30
medium straight	50	0
short right	25	30
short left	25	−30
short slight right	25	15
short slight left	25	−15
short straight	25	0
tiny straight	10	0

The OCP (10) contains a cost-to-go function:

$$F(\boldsymbol{x},\boldsymbol{\tau}) = \overbrace{|\boldsymbol{v}|^\top \cdot |\boldsymbol{\tau}|}^{\text{energy}} + 1000\left((v/v_{ub})^2 + (r/r_{ub})^2\right) + 100\left((X/X_{ub})^2 + (Y/Y_{ub})^2 + (N/N_{ub})^2\right). \tag{12}$$

The cost's main contributor is energy spent but includes quadratic costs on velocity states and input forces. Without

TABLE 2. OCP boundary values for motion primitives.

U_{nom}	1.5	m s^{-1}
u_{lb}	0	m s^{-1}
u_{ub}	2.5	m s^{-1}
v_{ub}	1.5	m s^{-1}
r_{ub}	5	$^\circ\,\text{s}^{-1}$
X_{lb}	−1000	N
X_{ub}	1000	N
Y_{ub}	1000	N
N_{ub}	1800	N m

these quadratic costs, the OCP becomes significantly harder to solve. The pure energy part of the cost function makes up ∼95% of the straight motion primitives' costs and ∼80% of the costs in turns.

The choice of length and direction parameters L and χ of the primitives are tightly connected to the resolutions defined in (3). At least one of the motion primitives must have a length longer than the diagonal of the grid cells defined by the positional resolution r_p in order to be guaranteed to traverse from one cell to another. We use a positional resolution of $r_p = 10\,\text{m}$, so we need at least one primitive longer than $\sqrt{2} \cdot 10\,\text{m} \approx 14.14\,\text{m}$. Additionally, one of the primitives should have a length equal to r_p, so that the search does not "jump over" the goal cell. It will also ease the discrete search if the motion primitives' direction angles are multiples of the angular resolution r_h. The primitives in Table 1 include these important properties.

The positional resolution greatly affects the performance of the hybrid A* search. A smaller resolution r_p makes the search space denser, which increases the computational load and time to find a solution, but improves the accuracy of the search.

The OCPs are transcribed to NLPs using direct collocation, and then solved using an interior point algorithm [26] offline prior to performing any search. The details of the transcription and solving are the same as in the main OCP-stage of our planning method – those details are found in Section IV-B.

In Algorithm 1, motion primitives from a configuration $\boldsymbol{\eta} \in \mathcal{C}$ are returned by the function PRIMITIVES. This function returns a sequence of geometrical paths $\mathcal{P} \in \mathcal{W}$, the cost of the maneuver c whose calculation is described in Section III-C, and the new neighboring state $\boldsymbol{\eta}_n \in \mathcal{C}$. The cost is dependent on the wind velocity $\boldsymbol{V}_w$. A mathematical description of the function is

$$\text{PRIMITIVES} : \mathcal{C} \times \mathbb{R}^2 \mapsto [\mathcal{W} \times \mathbb{R}^+ \times \mathcal{C}]_{1,\dots,M}, \qquad (13)$$

where M is the number of motion primitives.

C. COST FUNCTION

While the OCP that generates the motion primitives uses the generic cost-to-go function (12), these OCPs are solved offline and have no information about environmental disturbances. Therefore, we need an alternative method to quickly calculate the energy usage of each maneuver online, when the disturbances are known or estimated. For calculating energy exerted to overcome environmental disturbances, we use the definition of mechanical work:

$$W_r = \int_0^{t_f} |\boldsymbol{\tau}_r|^\top \cdot |\boldsymbol{\nu}_r|\,\mathrm{d}t, \qquad (14)$$

where we use the absolute values since there is no energy regeneration in the ASV's propulsion system. In this calculation, the subscript $(\cdot)_r$ denotes *relative* values, e.g., the force needed to overcome relative wind velocity. The work required to move through the wind is

$$W_{\text{wind}} = \int_0^{t_f} |\boldsymbol{\tau}_w|^\top \cdot \left| \boldsymbol{\nu} - \mathbf{R}(\psi)^\top \begin{bmatrix} \boldsymbol{V}_w \\ 0 \end{bmatrix} \right| \mathrm{d}t, \qquad (15)$$

where $\boldsymbol{\tau}_w$ is the force needed to overcome wind effects, calculated with our wind model. We have assumed zero ocean currents for moving through the water since the vessel we are working with has a very shallow and flat hull. Additionally, we do not have access to accurate information about ocean currents in our test areas. The work required to move through the water is then

$$W_{\text{water}} = \int_0^{t_f} |\mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu}|^\top \cdot |\boldsymbol{\nu}|\,\mathrm{d}t. \qquad (16)$$

The total energy cost $c = W_{\text{wind}} + W_{\text{water}}$ is calculated by propagating the integrands of (15) and (16) over the discretized solution trajectories from (10) with the appropriate wind velocity. This relative energy formulation is inspired by [27].

D. COLLISION CHECKING

For each solution trajectory generated by (10), the position state trajectories $x(\cdot)$ and $y(\cdot)$ make up the vessel's geometrical footprint in $\mathcal{W}$. After translating and rotating a motion primitive, the geometrical footprint is checked for overlap with $\mathcal{O}$, and a collision is reported if that is the case. The geometrical footprint is diluted by a clearance radius r_c to account for the shape of the vessel and additional clearance to keep a proper distance from obstacles. The clearance radius and footprint are illustrated in Figure 5. Our vessel is rectangular with a shape of 5 m by 2.8 m, and we use a clearance

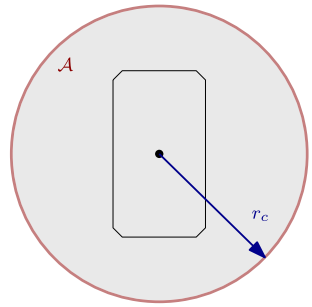


FIGURE 5. Vessel shape along with the clearance radius r_c which defines the footprint $\mathcal{A}$ used for collision checking.

radius of $r_c = 10\,\text{m}$. The COLLISION function in Algorithm 1 performs the collision checking:

$$\text{COLLISION} : \mathcal{W} \times \mathcal{W} \mapsto \{\text{true}, \text{false}\}\,. \tag{17}$$

E. SEARCH HEURISTICS

To guide the hybrid A* search, we use heuristic cost functions. These are functions that estimate the remaining cost from a node in $\mathcal{C}_d$ to the goal node. The search will prioritize exploring nodes with the lowest estimated total cost. In a traditional A* search, using *admissible* heuristic functions, i.e., functions that never overestimate the true cost, maintains a Dijkstra search's optimality guarantee. However, hybrid A* does not have any optimality guarantees due to the merging of continuous states in discrete "bins," so the heuristic functions' admissibility is not as important.

Similar to [18], we combine two different heuristic functions. We employ a *holonomic with obstacles* heuristic that guides the search towards the two-dimensional cheapest path, and a *nonholonomic without obstacles* heuristic that avoids trajectories that the ASV cannot feasibly follow. Their designs are described in the following, and they are combined using the maximum of the two heuristics.

The description of the HEURISTICS function from Algorithm 1 is

$$\text{HEURISTICS} : \mathcal{C} \times \mathcal{C} \times \mathbb{R}^2 \mapsto \mathbb{R}^+, \tag{18}$$

where the function maps the current state $\boldsymbol{\eta}$, the goal state $\boldsymbol{\eta}_f$, and the wind velocity $\boldsymbol{V}_w$ to a positive scalar.

1) HOLONOMIC WITH OBSTACLES

The holonomic with obstacles heuristic uses a simple model that can move in any direction without the nonholonomic constraint of moving along the vessel's heading angle. It considers the obstacle map $\mathcal{O}$ and assigns costs to nodes using a breadth-first search on a two-dimensional grid with resolution r_p. Instead of the standard eight-connected graph illustrated in Figure 3, we use a 16-connected graph, as seen in Figure 6, to allow more movement angles. We use the same cost function described in Section III-C, which results in a mapping from every node in $\mathcal{C}_{d,\text{free}}$ to a positive scalar that estimates the remaining cost to navigate to the goal node. Figure 7 shows an example of the mapping near a harbor.

The 16-connected breadth-first search is limiting since it biases towards paths with the same directions as the graph

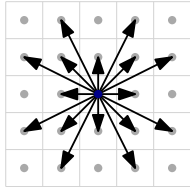


FIGURE 6. 16-connected graph. In this connectivity scheme, edges are added to all nodes two layers from the center node, unless the travel direction already exists in an inner layer.

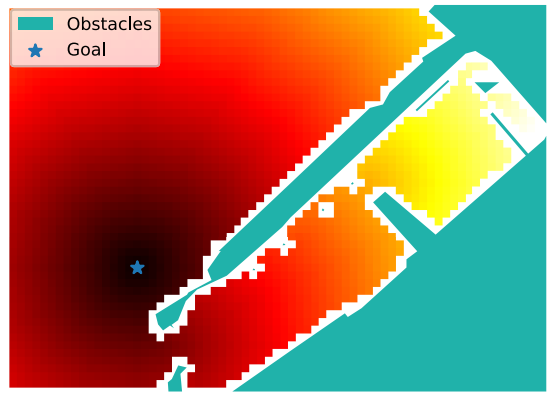


FIGURE 7. Example of the holonomic with obstacles heuristic function on a map. Brighter squares are more costly.

connectivity in Figure 6, i.e., on ∼22° increments. Without disturbances, the error between the real cost function and the heuristic averages 1.8% in an obstacle-free map of 1 km by 1 km.

Alternative heuristics include the fast marching method, which can calculate a cost function in the presence of obstacles without bias to particular directions. Standard implementations of the fast marching method [28] do not support the inclusion of a directional component in the cost function, on which we rely. Implementations of the fast marching method subject to a vector field are available [29], [30]. Furthermore, graph searches with simplified models can function as guiding heuristics, demonstrated in [15].

Since the calculation of our holonomic-with-obstacle heuristic requires information about the goal location and disturbances, the mapping has to be calculated online.

2) NONHOLONOMIC WITHOUT OBSTACLES

The dual to the holonomic with obstacles heuristic is one that considers nonholonomic movements without obstacles. This heuristic places high costs on nodes that lead to trajectories the ASV cannot feasibly follow. It utilizes the motion primitives from Section III-B and performs a breadth-first hybrid A* search from every node in a limited, rectangular, collision-free grid around the origin of $\mathcal{C}_d$. This results in a mapping from the included nodes in $\mathcal{C}_d$ to a positive scalar and is precomputed offline. The mapping is translated and rotated to the desired goal node when used in the search. Figure 8 shows the heuristic mapping for different initial heading angles.

Since the environmental disturbances are unknown at the time of precomputation, we cannot say anything about the effects these disturbances have on the cost. However, this heuristic is only active in the final part of the search, and we argue that the energy-optimality criterion is less critical in this stage. Additionally, the optimization stage described

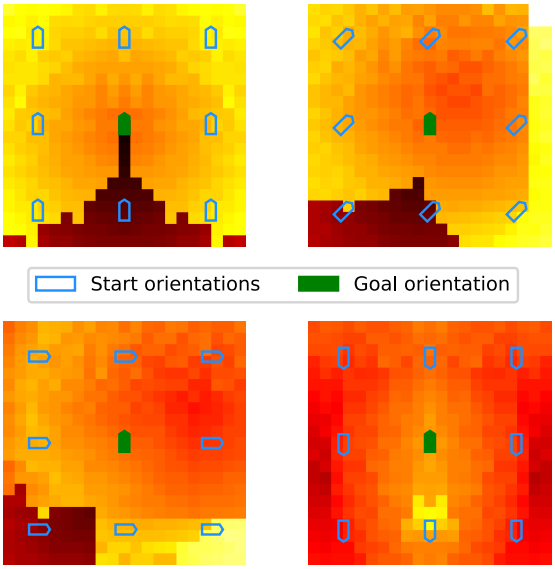


FIGURE 8. Plot of the nonholonomic without obstacles heuristic function for initial heading angles 0°, 45°, 90° and 180°. Brighter squares are more costly.

in Section IV locally optimizes the trajectory accounting for known or estimated disturbances.

F. SEARCH OUTPUT

A search is completed when the goal node is discovered by a motion primitive. The result is a chain of nodes from the goal node towards the start node by following their parents. This chain is reversed, and the resulting sequence of motion primitives are concatenated into forming the solution trajectories

$$\boldsymbol{x}^\star : [0, t_f^\star] \mapsto \mathcal{X} \tag{19a}$$
$$\boldsymbol{\tau}^\star : [0, t_f^\star] \mapsto \mathcal{T}, \tag{19b}$$

which are valid on the time interval $[0, t_f^\star]$, where $t_f^\star$ is the sum of the motion primitive durations. In practice, these mappings are a discrete sequence of points in the state and input spaces ($\mathcal{X}$ and $\mathcal{T}$), interpolated to form time-continuous trajectories. The points' density depends on the number of shooting intervals used when solving (10).

To summarize Stage 1, it consists of a hybrid A* search guided by two heuristics, propagating motion primitives that lead from the start pose to the desired end pose. Since the trajectory so far consists of only the motion primitive maneuvers, it must be improved to find an optimized trajectory in the continuous search space.

IV. STAGE 2: TRAJECTORY OPTIMIZATION

The second stage of the trajectory planner is to solve an OCP that describes the trajectory planning problem. Stage 2 in Figure 2 shows the subcomponents of this trajectory optimization. The OCP is similar to (10) in Section III-B. The initial and final conditions are different, we include external disturbances, and we have added obstacle avoidance constraints. Additionally, the final time is a free optimization variable. We restate the OCP, including the stated changes:

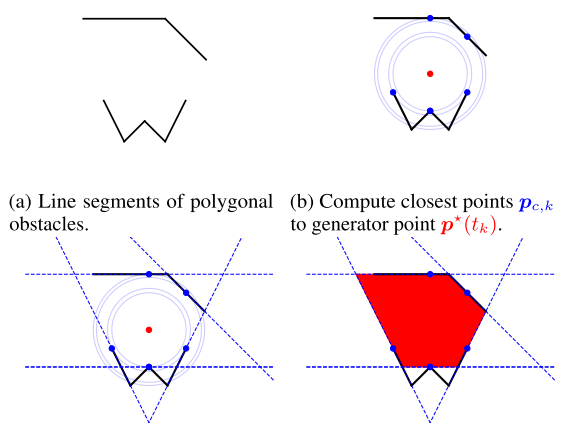
$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{\tau}(\cdot),t_f} \int_0^{t_f} F(\boldsymbol{x}(\tau), \boldsymbol{\tau}(\tau))\mathrm{d}\tau \tag{20a}$$
$$\text{subject to } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\tau}(t), \boldsymbol{V}_w) \quad t \in [0, t_f] \tag{20b}$$
$$\boldsymbol{x}_{lb} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{ub} \quad t \in [0, t_f] \tag{20c}$$
$$\boldsymbol{\tau}_{lb} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}_{ub} \quad t \in [0, t_f] \tag{20d}$$
$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{20e}$$
$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f \tag{20f}$$
$$\mathbf{A}_k \cdot \begin{bmatrix} x(t_k) & y(t_k) \end{bmatrix}^\top \le \boldsymbol{b}_k - r_c \quad k = 1, \ldots, N \tag{20g}$$
$$0 \le t_f \le t_f^\star. \tag{20h}$$

The initial and final conditions are replaced with the initial and final desired pose, with zero velocities. The cost-to-go function is the same, as are the velocity and force bounds. Equation (20g) encodes obstacle avoidance constraints, which will be described in Section IV-A. Since the final time is a free variable, we place bounds on it in (20h). The transcription and solution process is described in Section IV-B.

A. CONVEX COLLISION AVOIDANCE CONSTRAINTS

The OCP contains obstacle avoidance constraints in the form of halfspaces in the matrix-vector form (20g). The halfspaces are defined for the points in time $t_k, k = 1, \ldots, N$, where N is the number of shooting intervals used in the transcription of (20). With $h = t_f/N$ being the shooting interval duration, we have $t_k = h{\cdot}k$. The convex regions that define the obstacle avoidance constraints are generated along the solution of the hybrid A* trajectory, i.e., the initial guess. The positional part of the state trajectory $\boldsymbol{x}^\star(\cdot)$ from (19a) is denoted $\boldsymbol{p}^\star(\cdot) = [x^\star(\cdot), y^\star(\cdot)]^\top$. For the points in time $t_k, k = 1, \ldots, N$, the parameters $\mathbf{A}_k \in \mathbb{R}^{m_k \times 2}$ and $\boldsymbol{b}_k \in \mathbb{R}^{m_k}$ are generated based on the obstacle map $\mathcal{O}$ with $\boldsymbol{p}^\star(t_k)$ being the *generator points*.

To create the convex region constraints, we use an algorithm that calculates an inner approximation of the obstacle map based on the polygons' edges in that map. The process is summarized as follows: Given a generator point $\boldsymbol{p}^\star(t_k)$, grow a circle centered at $\boldsymbol{p}^\star(t_k)$ until it reaches a point $\boldsymbol{p}_{c,k}$ where it touches an obstacle, and then create a constraint tangent to the expansion circle at the point at $\boldsymbol{p}_{c,k}$. Continue growing and create constraints until no further growth is possible. The process is illustrated in Figure 9. The parameters $\mathbf{A}_k$ and $\boldsymbol{b}_k$



(a) Line segments of polygonal obstacles.

(b) Compute closest points $\boldsymbol{p}_{c,k}$ to generator point $\boldsymbol{p}^\star(t_k)$.

(c) Find normal lines to the line segments from $\boldsymbol{p}^\star(t_k)$ to $\boldsymbol{p}_{c,k}$, that pass through $\boldsymbol{p}_{c,k}$.

(d) Final convex inner approximation.

FIGURE 9. Illustration of how to compute the convex spatial constraints.

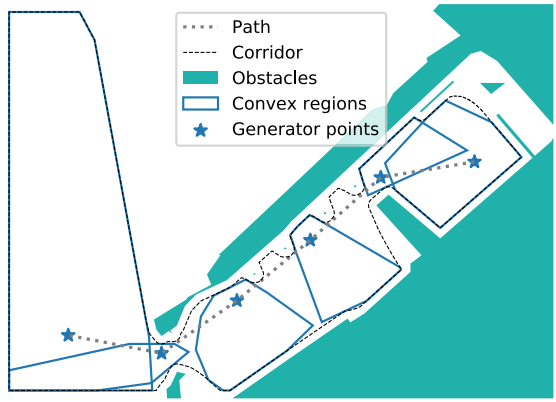


FIGURE 10. Example of convex regions along an arbitrary path with generator points spaced by 100 m. In the OCP the spacing would be ∼1.5 m, causing dense overlapping, resulting in a corridor as depicted in the figure.

are defined by

$$\underbrace{\begin{bmatrix} \frac{(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))^\top}{||(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))||_2} \\ \frac{(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))^\top}{||(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))||_2} \\ \vdots \\ \frac{(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))^\top}{||(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))||_2} \end{bmatrix}}_{\mathbf{A}_k} \boldsymbol{p} \leq \underbrace{\begin{bmatrix} \frac{(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))^\top \boldsymbol{p}_{c,k,1}}{||(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))||_2} \\ \frac{(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))^\top \boldsymbol{p}_{c,k,2}}{||(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))||_2} \\ \vdots \\ \frac{(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))^\top \boldsymbol{p}_{c,k,m_k}}{||(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))||_2} \end{bmatrix}}_{\boldsymbol{b}_k}. \tag{21}$$

A point $\boldsymbol{p} \in \mathbb{R}^2$ is inside the convex region if the inequality constraints are satisfied, which is (20g) in the OCP. The number of halfspaces that make up a specific region is denoted $m_k, k = 1, \ldots, N$, and has an upper limit, in our case 12. The unit dimension of this inequality is distance, and a subtraction of the right-hand side of (21) shrinks the convex regions, implicitly increasing the clearance by, e.g., r_c, which is the clearance radius from Figure 5, used in (20g).

Figure 10 shows an example of convex regions using an arbitrary path as the basis for generator points. For each point in time t_k, the OCP may freely adjust the ASV's position inside the respective convex region. With dense overlapping, this allows the ASV to travel inside a corridor along the initial guess.

The convex regions constrain only a discrete set of points in the state trajectory $(\boldsymbol{x}(t_k), k = 1, \ldots, N)$. This limitation means that the points in between can violate the collision avoidance constraints. However, the vessel's dynamics restrict the trajectory's velocity, thus limiting the movement in a neighborhood around $\boldsymbol{x}(t_k)$. Having a short shooting interval duration h gives satisfactory collision avoidance behavior. In our results, we use a density of $h \approx 1$ s.

B. TRANSCRIPTION AND SOLVER

To solve the continuous OCP (20), we discretize it into an NLP. We use direct collocation with three Legendre collocation points per shooting interval to discretize the dynamics (20b). Both the state and input trajectories are encoded as polynomials over N shooting intervals. In our results, the number of shooting intervals is determined by the estimated final time $t_f^\star$ from the hybrid A* results in Section III-F. An initial shooting interval duration of $h^\star = 1$ s determines $N = \lfloor t_f^\star / h^\star \rfloor + 1$, while since the final time t_f is a free variable with upper bound $t_f^\star$, the actual shooting interval duration can be shorter. The cost function is determined by propagating the quadrature integral (20a) along the state and input polynomials. The resulting NLP is

$$\min_{\boldsymbol{w}} \phi(\boldsymbol{w}) \tag{22a}$$
$$\text{subject to } \boldsymbol{w}_{lb} \leq \boldsymbol{w} \leq \boldsymbol{w}_{ub} \tag{22b}$$
$$\boldsymbol{g}_{lb} \leq \boldsymbol{g}(\boldsymbol{w}) \leq \boldsymbol{g}_{ub}. \tag{22c}$$

The decision variables $\boldsymbol{w}$ include states and inputs at all collocation points, and the final time t_f. The bounds (22b) are box bounds on all the decision variables and encode the state and input constraints (20c) through (20f), and (20h). The function $\boldsymbol{g}$ and its bounds in (22c) encode the dynamics (20b) in addition to the obstacle avoidance constraints (20g).

The NLP is solved using the interior point algorithm "Ipopt" by Wächter and Biegler [26]. Since the initial guess provided by the hybrid A* algorithm results in minimal violations of the constraints, the initial value of the auxiliary boundary parameter μ in Ipopt is set quite low to 1×10^{-6}, compared to its default value of 1×10^{-1}. This reduction causes fast convergence of the solution.

Solving (22) provides the optimal decision variables $\boldsymbol{w}^\diamond$. These are converted to optimal trajectories

$$\boldsymbol{x}^\diamond : [0, t_f^\diamond] \mapsto \mathcal{X} \tag{23a}$$
$$\boldsymbol{\tau}^\diamond : [0, t_f^\diamond] \mapsto \mathcal{T}, \tag{23b}$$

where $t_f^\diamond$ is the optimal final time. Accurate interpolation of the discrete values returned from the solver is achieved by using the polynomial definition of the state and input trajectories.

C. METHOD SUMMARY

Figure 2 illustrates how all the subcomponents of our method are connected. Stage 1 performs a discrete search with continuous states using the hybrid A* algorithm guided by two heuristics and propagating the states with motion primitives. This results in a dynamically feasible initial guess for an energy-optimized trajectory between the start and goal poses. The resulting trajectory consists of a sequence of the motion primitives from Section III-B, limiting the search space to only those maneuvers. Therefore the trajectory cannot be optimal with respect to our cost functional. Stage 2 is a trajectory optimization step that uses the initial guess for two purposes: 1) To provide a sequence of convex and smooth polygonal constraints that represent a collision-free corridor from start to goal, and 2) to warm-start the OCP solver. The convex polygonal constraints are constructed with the process shown in Figure 9 and allow the OCP solver to handle the inherently nonconvex obstacle avoidance problem easily. Combined, this gives us a fast solution to (20), which is a locally optimal and dynamically feasible trajectory between the start and goal poses.

V. SIMULATION RESULTS

In this section, we describe the simulation and control setup used to evaluate our planning method and present the evaluation itself. We evaluate our method by performing planning and simulation in various scenarios and wind conditions and comparing our planner to other methods.

A. SIMULATOR AND CONTROL SYSTEM

The different trajectory planning methods are tested in a software-in-the-loop vessel simulator. The simulator comprises dynamic models of the vessel, its actuators, and its control systems. The vessel model is described in Section II-B, and the simulator performs Runge-Kutta 4 integration to propagate the differential equations. Additionally, the actuators' propeller and azimuth dynamics are simulated, whose models are available in [25].

The vessel's control system for trajectory tracking is divided into two layers, as seen in Figure 11: A trajectory-tracking dynamic positioning (DP) controller and a thrust allocation algorithm. The DP controller consists of a PID feedback term and a model-based feed-forward term for velocity and acceleration. Its details are available in [31, Section 3.4]. The controller sends the desired force

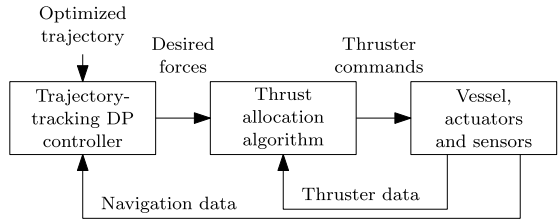


FIGURE 11. Vessel control system architecture.

output to the thrust allocation algorithm, which in turn sends thruster commands to the vessel's actuators. This thrust allocation algorithm is described in [32].

For evaluation, energy use is measured by integrating the simulated power output, similar to the energy-part of (12):

$$E = \int_{t_0}^{t_f} |\boldsymbol{v}(t)|^\top \cdot |\boldsymbol{\tau}(t)| \, \mathrm{d}t. \tag{24}$$

B. EVALUATING THE EFFECT OF INCLUDING DISTURBANCE INFORMATION

One of the goals while developing the method was the ability to include known or estimated disturbance effects in both planning stages. To magnify the effects of wind on planning, we have designed a scenario where the starting point and goal are far apart, and the vessel is under the influence of crosswinds. Figure 12 shows the scenario where the plan is to sail from south to north.

The scenario is planned twice. Once when no wind information is included in the search, assuming that the wind velocity is $\boldsymbol{V}_w = [0, 0]$ when it is, in fact, $\boldsymbol{V}_w = [0, 3]\,\mathrm{m\,s^{-1}}$, and once using the correct wind velocity. The warm start solutions from the hybrid A* search differ significantly in the two cases, as shown Figure 12. However, the optimized trajectories of the two plans are nearly identical. Additionally, the power outputs from the simulated trackings are not that different – the total energy use for the two scenarios are 170 W h when not accounting for wind in the planning, compared to 164 W h when including wind information, a mere 3.5% improvement, attributed mainly to a difference in heading during transit.

In practice, models of how wind affects a ship are uncertain. For such a low improvement, it might not be beneficial to include wind effects when planning a long-term trajectory. Adding this information may worsen the result if the wind model or wind velocity estimates are erroneous. Including environmental disturbances may be more appropriate for other types of vessels or other types of disturbances, such as waves and ocean currents.

C. COMPARISONS TO OTHER TRAJECTORY PLANNING METHODS

Our method is compared to two other trajectory planning methods by planning a trajectory in the same scenario with all three methods. The two other planning methods are a

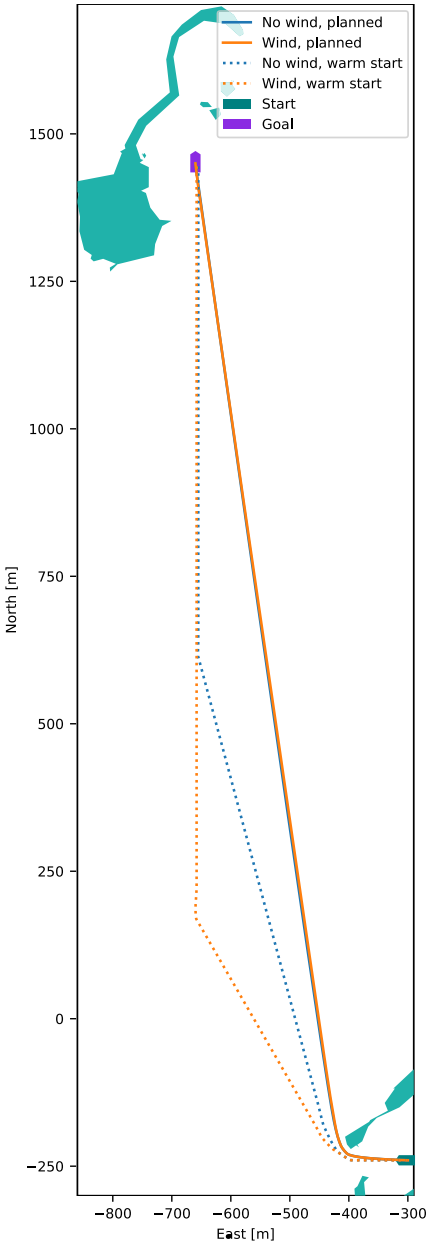


FIGURE 12. Comparison of trajectories planned without and with knowledge of simulated wind conditions. Described in Section V-B.

warm-started optimization scheme developed in [14], labeled C1 in the plots, and an optimal control-based complete cell decomposition method from [23] labeled C2. Our method is labeled TP. These two methods are selected for comparison because they are both optimization-based methods. C1 is similar in terms of the warm-starting methodology, and C2 is interesting because of the map discretization's completeness.

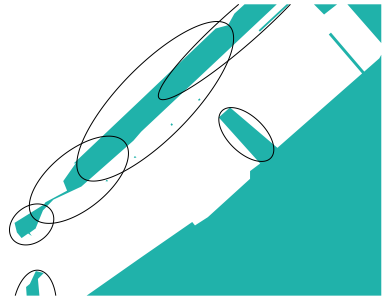


FIGURE 13. Elliptic obstacles that approximately match our map. Used when planning with the C1 method in Section V-C1.

1) C1: ALTERNATIVE WARM-STARTED OPTIMIZATION METHOD

The method developed in [14] uses a similar approach to our method. The main difference is how the warm start is generated and how the obstacles are represented in the optimization stage. C1 uses a standard A* search on an 8-connected uniformly discretized grid to search for the *shortest* path. That search results in a piecewise linear path which is converted to a trajectory by smoothing the connections with circular arcs and adding artificial dynamic information. The trajectory is not dynamically feasible with respect to the ASV's model, but it is used as the initial guess for an OCP solver. The OCP solver represents obstacles as inequalities in the form of ellipses, which are smooth representations, suitable for an optimization problem, but cannot accurately represent polygonal maps.

To compare TP to C1, we adjust the cost-to-go function in [14] to be equivalent to (12). Additionally, we have created elliptic obstacles to approximately match the polygonal obstacles which define our map, seen in Figure 13. We plan and simulate with zero wind, and with the initial and final poses, as shown in Figure 14. From the figure, we see that the resulting trajectories differ only slightly, mainly due to the different obstacle constraints. In the simulation, the trajectories give equal energy consumption, both at 52 W h. Figure 15 shows significant positional tracking error at the start and end of the transit, for both TP and C1. The vessel and its control system cannot track the acceleration that happens from and to a standstill. The models used in trajectory planning do not consider actuator dynamics or the control system, which probably is the cause of these errors. The positional tracking errors are comparable between TP and C1 for the remainder of the transit, with an error of 1 m around the turn and negligible error for the straights. Similar deviations are also evident in the heading, due to the coupling between linear and angular velocities. The positional error in Figure 15 is calculated as $||[x(t), y(t)] - [x^\diamond(t), y^\diamond(t)]||_2$, while the heading error is $\psi(t) - \psi^\diamond(t)$.

2) C2: COMPLETE OPTIMIZATION-BASED CELL DECOMPOSITION

Martinsen *et al.* [23] have developed an optimization-based trajectory planner that searches for a trajectory by

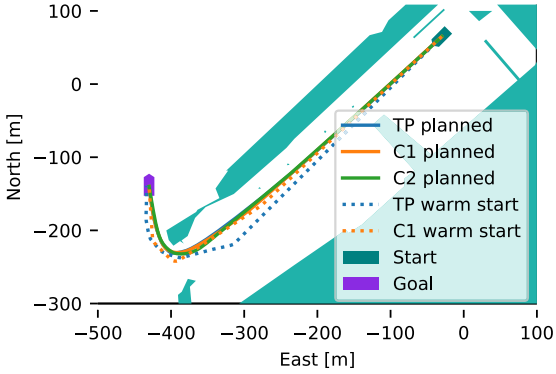


FIGURE 14. Comparison of trajectories planned with different methods from Section V-C. Results from our method are labeled TP, while C1 and C2 denote the other planning methods.

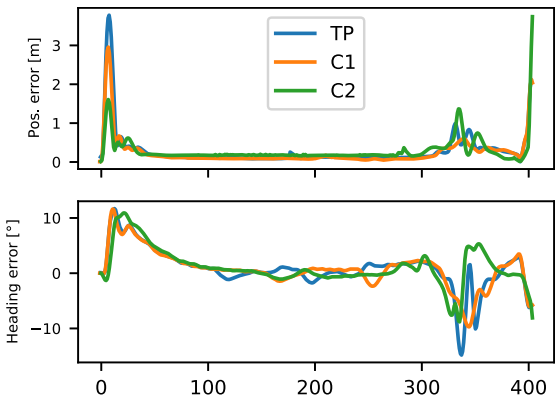


FIGURE 15. Tracking errors from the simulation comparisons. Results from our method are labeled TP, while C1 and C2 denote the other planning methods.

considering sequences of collision-free triangles from a constrained Delaunay triangulation of the workspace. C2 finds a globally optimal trajectory for linear models regardless of the inherent non-convex obstacles due to the cell decomposition by triangulation.

A trajectory with the same initial and final positions, as described in Section V-C1, is generated using a similar cost-to-go function and a simplified dynamic model. Figure 14 shows that also with C2, the trajectory difference is minimal. The differing cost-to-go function and dynamic model may cause the small differences we see. The slight difference may be caused by the differing cost-to-go function and the dynamic model used. As TP and C1, C2 gives an energy consumption of 52 W h. The tracking errors are similar between TP and C2, with better performance at the start of the trajectory for C2, as we see in Figure 15.

D. COMPLEX SCENARIO

The previous planning scenarios have been simple, with obvious routing choices. Figure 16 shows a more complex

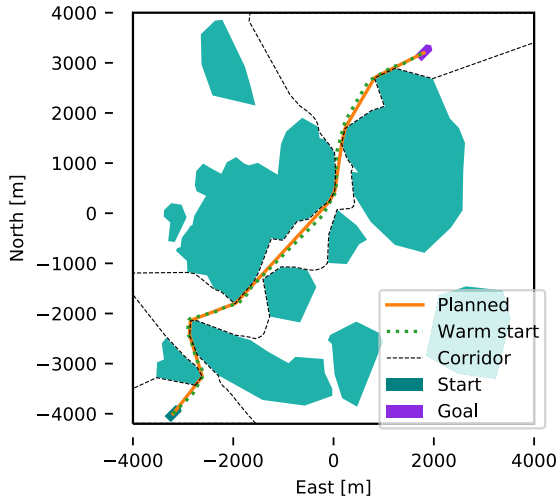


FIGURE 16. Planning in a map of Sjernarøyane, Norway – a more complex scenario with multiple routing options. Described in Section V-D.

scenario with multiple routing options. Our method was able to find the most direct and energy-efficient routing and optimized a trajectory from start to goal in 75 s. The figure also shows the corridor composed of the union of convex regions that allow the OCP to optimize freely. The resulting trajectory is dynamically feasible and adheres to the obstacle clearance constraints.

E. CONCLUSION

Including available wind information when planning a scenario yielded negligible improvements, shown in Section V-B. Only minor differences are found in the state trajectories. We conclude that there is no benefit to energy consumption for our application and vessel model when including wind estimates in trajectory planning. This conclusion is supported by the fact that there will be significant uncertainties in both wind estimates and wind force models.

Compared to two other optimization-based trajectory planning methods in Section V-C, we have shown that our method produced a similar trajectory with equal energy consumption. This similarity verifies that our method can find a desirable optimized trajectory with good energy performance. Significant improvements in runtime are achieved by using our method in this scenario, highlighted in Table 3.

TABLE 3. Performance comparisons for simulated planning scenarios in Section V-C.

Method	Energy usage [W h]	Planning time [s]
TP	52	**31**
C1	52	57
C2	52	785

As we show in Section V-D, our method can find the most reasonable trajectory in a complex routing environment, which is a major challenge when using purely optimization-based trajectory planning methods.

VI. EXPERIMENTAL VALIDATION

To validate that our method will produce collision-free, dynamically feasible trajectories, we have applied it in a full-scale experiment with *milliAmpere*, an experimental autonomous ferry developed at NTNU, depicted in Figure 17. The specifications of *milliAmpere*, its sensors, and control systems are found in Table 4, and it uses the same control setup as described in Section V. We tested the planning method and tracking capabilities in the Trondheim harbor area, using the same scenario as in Section V-C. On the day of testing, we measured a light breeze from North-northeast, but we were shielded by a breakwater for most of the route, causing us to experience almost no wind. We tested planning with zero wind and with the measured wind, finding a difference in measured energy use of 2%, which we deem insignificant given the measurement uncertainties, and thus we only present the results from planning with zero wind. Energy use is measured by integrating power as determined by the voltage and current measured on both the azimuth thrusters' propeller motors:

$$E = \int_{t_0}^{t_f} \left(|I_1(t) \cdot U_1(t)| + |I_2(t) \cdot U_2(t)|\right) \mathrm{d}t, \qquad (25)$$

where I_i and U_i are motor current and voltage, respectively, for $i \in \{1, 2\}$.

FIGURE 17. Picture of the experimental autonomous ferry *milliAmpere*.

TABLE 4. *milliAmpere* specifications.

Dimensions	5 m by 2.8 m symmetric footprint
Position and heading reference system	Vector VS330 dual GNSS with RTK capabilities
Thrusters	Two azimuth thrusters on the center line, 1.8 m aft and fore of center
Existing control modules	Trajectory-tracking DP controller and thrust allocation system

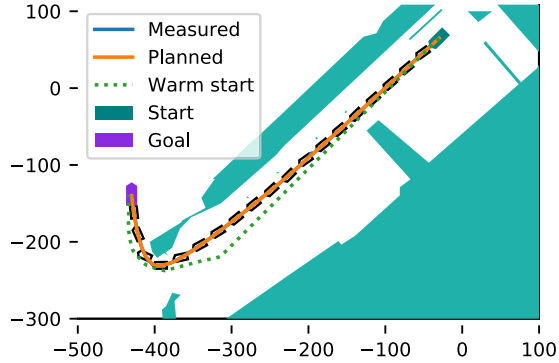


FIGURE 18. Measured and planned trajectories from validation experiment in Section VI.

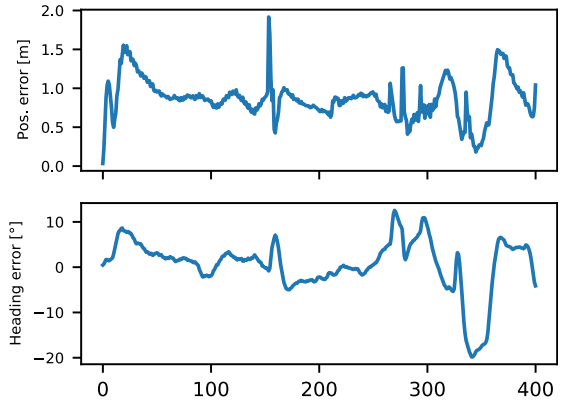


FIGURE 19. Tracking errors from the validation experiment in Section VI.

Figure 18 shows an overview of the scenario, including the measured and planned trajectories, and the warm start. The planned trajectory was naturally equivalent to the one in Section V-C, as nothing in the scenario was changed.

Figure 19 shows tracking errors for the tests. The positional error stayed within 1.5 m, with an exception at 160 s, which stems from a jump in the GNSS measurement experienced during the experiment. The positional tracking error was large at the beginning of the experiment, where *milliAmpere* and its control system struggled to follow the trajectory's acceleration. The unmodeled thruster dynamics and control systems can explain this initial lag. Large positional errors were also induced during the turn near the end of the experiments, which can be explained by the control system's poor tracking performance. Heading errors also occurred during the beginning of the experiment, as well as during the turn. These errors can be attributed to the coupling between linear and angular velocity, which is unaccounted for in the control system. *milliAmpere*'s physical properties, with its shallow and flat hull, make it hard to control its heading.

Figure 20 shows the distances from the measured and planned positions to the nearest obstacle. Of course,

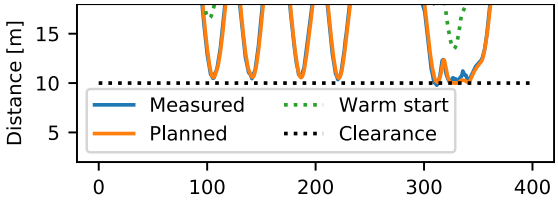


FIGURE 20. Measured and planned obstacle distances from the validation experiments in Section VI.

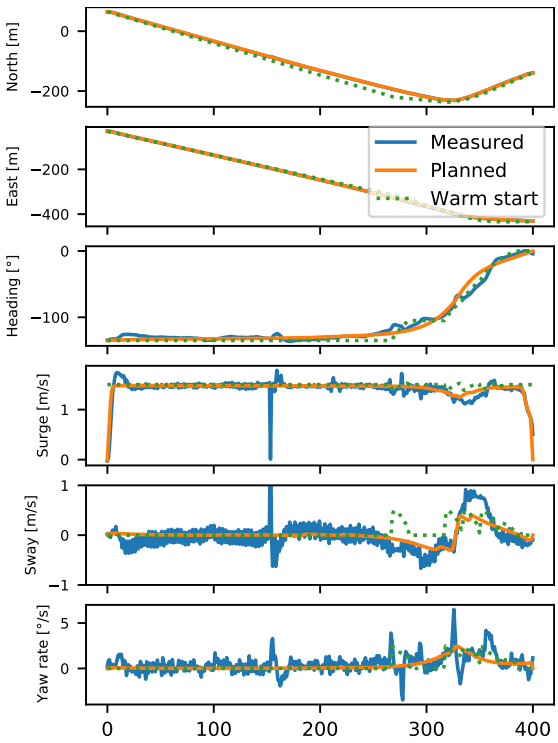


FIGURE 21. Measured and planned state trajectories from the validation experiment in Section VI.

the planned distance is more than the clearance of $r_c = 10\,\text{m}$ away from obstacles at all times. The measured distance was 9.9 m from the closest obstacle at the nearest, which we consider safe.

Figure 21 shows the measured and planned state trajectories. The referenced jump in GNSS measurement at 160 s is clearly visible in this plot, where the error propagates into the velocity estimates. While the plots show jumps in velocity, no such jump was experienced during the experiment – this is a measurement error. The sway velocity and yaw rate measurements oscillate, making hard to determine tracking error for these states. This oscillation may be due to the ship's natural frequency – it is not filtered out in the onboard navigation system, but we can see that the measurements lie around the planned trajectories.

During the experiment, the measured energy use was 245 W h, which is almost five times the simulated energy estimate. This discrepancy is due to the completely different approaches used to estimate the energy in simulation and experiments.

VII. CONCLUSION

Solving the continuous optimal control problem for trajectory planning is difficult and requires an initial guess close to the globally optimal solution to be a feasible option. Moreover, since the trajectory planning problem is inherently nonconvex, some clever encoding of obstacles is needed to reduce complexity, especially when dealing with polyhedral shapes with discontinuous gradients. We propose a continuous, model-based method for energy-optimized trajectory planning for ASVs that leverages a discrete search's desirable advantages to generate a good initial guess and performs convex encoding of obstacles to achieve collision avoidance. Our method is based on continuous optimal control, and the warm starting is provided by the hybrid A* algorithm. We have compared the method with an optimal control-based complete cell decomposition method with a similar cost function to find comparable performance in terms of optimality and significantly improved computational time. A comparison with a warm-started optimal control-based method from earlier work by us has shown improved performance, in addition to being able to use more general obstacle representations.

There are several areas where we can improve our method in further work:

- The search space can be extended to include surge velocity. This extension would allow the hybrid A* search to look for variations in the speed profile that could benefit energy efficiency.
- Including velocity in the search space will require modifications to the heuristic functions. Additionally, to prevent the search from always choosing the slowest velocity which would be energy optimal, we need to limit the trajectory's maximum duration. This constraint could be introduced by computing a map with the shortest path, similar to the holonomic with obstacles heuristic, and constrain the search from selecting nodes that cannot reach the goal within the time limit with the highest velocity. The fast marching method is appropriate for this distance map.
- The hybrid A* search is currently a naive Python implementation and contributes significantly to computation time. Improvements to this implementation would increase the performance of our method. This issue is also the case for the construction of the NLP.
- In our work, we have included external disturbances in terms of wind velocity. We have found that it does not affect the optimized trajectory or energy consumption in a significant manner. Other environmental effects, such as waves or ocean currents, can have more of an impact on energy consumption and should be explored.

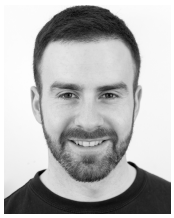
- Including the COLREGs in trajectory planning for marine vessels should also be a priority to further work on this topic.

ACKNOWLEDGMENT

The authors thank Emil H. Thyri at NTNU for his help during the experiments with *milliAmpere*. Glenn Bitar also thanks Marius Thoresen at the Norwegian Defence Research Establishment for helpful discussions during algorithm implementation.

REFERENCES

[1] M. Ludvigsen and A. J. Sørensen, "Towards integrated autonomous underwater operations for ocean mapping and monitoring," *Annu. Rev. Control*, vol. 42, pp. 145–157, Dec. 2016.
[2] B.-O.-H. Eriksen, G. Bitar, M. Breivik, and A. M. Lekkas, "Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17," *Frontiers Robot.*, vol. 7, pp. 1–11, Feb. 2020.
[3] M. L. Seto, *Marine Robot Autonomy*. New York, NY, USA: Springer-Verlag, 2013.
[4] S. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. Eng, D. Rus, and M. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, Feb. 2017.
[5] S. M. LaValle, "Motion planning. Part I: The essentials," *IEEE Robot. Autom. Mag.*, vol. 18, no. 1, pp. 79–89, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5751929, doi: 10.1109/MRA.2011.940276.
[6] A. Wolek and C. A. Woolsey, "Model-based path planning," in *Sensing and Control for Autonomous Vehicles*, T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, Eds. Cham, Switzerland: Springer, 2017, pp. 183–206.
[7] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.
[8] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Eng. Pract.*, vol. 61, pp. 41–54, Apr. 2017.
[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Dec. 1996.
[10] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path plannning," Iowa State Univ., Ames, IA, USA, Tech. Rep., 1998.
[11] R. Bellman, *Dynamic Programming*. New York, NY, USA: Courier Dover, 2003.
[12] Q. Gong, L. R. Lewis, and I. M. Ross, "Pseudospectral motion planning for autonomous vehicles," *J. Guid., Control, Dyn.*, vol. 32, no. 3, pp. 1039–1045, May 2009.
[13] G. Bitar, M. Breivik, and A. M. Lekkas, "Energy-optimized path planning for autonomous ferries," in *Proc. 11th IFAC CAMS*, 2018, pp. 389–394.
[14] G. Bitar, V. N. Vestad, A. M. Lekkas, and M. Breivik, "Warm-started optimized trajectory planning for ASVs," in *Proc. IFAC Conf. Control Appl. Mar. Syst., Robot. Veh. (IFAC CAMS)*, 2019, vol. 52, no. 21, pp. 308–314.
[15] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *Proc. IEEE Conf. Decision Control (CDC)*, Dec. 2018, pp. 4327–4332.
[16] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill, "An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments," 2020, *arXiv:2005.02674*. [Online]. Available: https://arxiv.org/abs/2005.02674
[17] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative LQR," *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 244–254, Jun. 2019, doi: 10.1109/TIV.2019.2904385.
[18] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," Amer. Assoc. Artif. Intell., Tech. Rep. 1001.48105, 2008.
[19] Y. Zhang, H. Chen, S. L. Waslander, J. Gong, G. Xiong, T. Yang, and K. Liu, "Hybrid trajectory planning for autonomous driving in highly constrained environments," *IEEE Access*, vol. 6, pp. 32800–32819, 2018.
[20] Y. Meng, Y. Wu, Q. Gu, and L. Liu, "A decoupled trajectory planning framework based on the integration of lattice searching and convex optimization," *IEEE Access*, vol. 7, pp. 130530–130551, 2019.
[21] G. Casalino, A. Turetta, and E. Simetti, "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields," in *Proc. Oceans-Europe*, May 2009, pp. 1–8.
[22] P. Svec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. von Ellenrieder, M. Dhanak, and S. K. Gupta, "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 3871–3878.
[23] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Optimal model-based trajectory planning with static polygonal constraints," 2020, *arXiv:2010.14428*. [Online]. Available: https://arxiv.org/abs/2010.14428
[24] T. I. Fossen, *Handbook Mar. Craft Hydrodynamics Motion Control*. Hoboken, NJ, USA: Wiley, 2011.
[25] A. A. Pedersen, "Optimization based system identification for the milliAmpere ferry," M.S. thesis, Norwegian Univ. Sci. Technol., Trondheim, Norway, 2019. [Online]. Available: http://hdl.handle.net/11250/2625699
[26] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2005, doi: 10.1007/s10107-004-0559-y.
[27] T. Lee, H. Chung, and H. Myung, "Multi-resolution path planning for marine surface vehicle considering environmental effects," in *Proc. OCEANS IEEE*, Jun. 2011, pp. 1–9.
[28] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci. USA*, vol. 93, no. 4, pp. 1591–1595, 1996.
[29] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 331–341, Apr. 2007.
[30] S. Garrido, L. Moreno, F. Martín, and D. Álvarez, "Fast marching subjected to a vector field–path planning method for mars rovers," *Expert Syst. Appl.*, vol. 78, pp. 334–346, Jul. 2017.
[31] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, "Trajectory planning and control for automatic docking of ASVs with full-scale experiments," in *Proc. 1st Virtual IFAC World Congress*, 2020, pp. 1–5.
[32] T. R. Torben, A. H. Brodtkorb, and A. J. Sørensen, "Control allocation for double-ended ferries with full-scale experimental results," in *Proc. 12th IFAC Conf. Control Appl. Mar. Syst., Robot. Veh. (IFAC CAMS)*, 2019, pp. 556–563.



GLENN BITAR received the B.S. degree in computer science and industrial automation from the Telemark University College, Porsgrunn, Norway, in 2015, and the M.S. degree in cybernetics and robotics from the Department of Engineering Cybernetics, NTNU, Trondheim, Norway, in 2017, where he is currently pursuing the Ph.D. degree.

His research interests include energy-optimized autonomous motion control for ships, automatic docking, and optimization-based path and trajectory planning.

ANDREAS B. MARTINSEN received the M.S. degree in engineering cybernetics from NTNU, Trondheim, Norway, in 2018, where he is currently pursuing the Ph.D. degree in engineering cybernetics.

His research interests include reinforcement learning, optimal control, and machine learning, with a focus on marine and maritime applications.

ANASTASIOS M. LEKKAS (Member, IEEE) received the M.S. degree in mechanical and aeronautical engineering from the University of Patras, Greece, in 2008, and the Ph.D. degree in engineering cybernetics from NTNU, Trondheim, Norway, in 2014.

From 2015 to 2017, he has worked as a Science Officer at JPI OCEANS, Brussels, Belgium. He is currently an Associate Professor of autonomous systems with the Department of Engineering Cybernetics, NTNU, where he is also affiliated with the Centre for Autonomous Marine Operations and Systems. He participates in projects focusing on the autonomy of marine vehicles (Autoferry, SEAVENTION) and is the Project Manager of the EXAIGON Project, where the main goal is to develop explainable AI methods for safety- and business-critical applications. His research interest includes merging artificial intelligence with control engineering in order to develop cyber-physical systems with increased autonomy.

MORTEN BREIVIK (Member, IEEE) received the M.S. and Ph.D. degrees in engineering cybernetics from the Department of Engineering Cybernetics, NTNU, Trondheim, Norway, in 2003 and 2010, respectively.

He is currently the Head of the Department of Engineering Cybernetics, NTNU. He is also an Associate Researcher with the Centre for Autonomous Marine Operations and Systems. He has previously worked as an Assistant Professor and a Researcher with NTNU, a Scientific Advisor for Maritime Robotics, and a Principal Engineer and the Department Manager in applied cybernetics at Kongsberg Maritime. His research interests include nonlinear and adaptive motion control of unmanned vehicles in general and in autonomous ships in particular. He is also a member of the Norwegian Board of Technology.

Paper G Optimization-based automatic docking and berthing of ASVs using exteroceptive sensors: theory and experiments

Published paper by A. B. Martinsen, **G. Bitar**, A. M. Lekkas, and S. Gros. "Optimization-based automatic docking and berthing of ASVs using exteroceptive sensors: theory and experiments". In: *IEEE Access* 8 (2020), pp. 204974–204986. DOI: 10.1109/ACCESS.2020.3037171.

Bibliography entry [14].

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 30, 2020, accepted November 8, 2020, date of publication November 11, 2020,
date of current version November 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3037171

Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments

ANDREAS B. MARTINSEN[1], GLENN BITAR[1,2], ANASTASIOS M. LEKKAS[1,2], AND SÉBASTIEN GROS[1]

[1]Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
[2]Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway

Corresponding author: Andreas B. Martinsen (andreas.b.martinsen@ntnu.no)

This work was supported in part by the Research Council of Norway under Project 269116, and in part by the Centers of Excellence funding Scheme under Project 223254.

ABSTRACT Docking of autonomous surface vehicles (ASVs) involves intricate maneuvering at low speeds under the influence of unknown environmental forces, and is often a challenging operation even for experienced helmsmen. In this paper, we propose an optimization-based trajectory planner for performing automatic docking of a small ASV. The approach formulates the docking objective as a nonlinear optimal control problem, which is used to plan collision-free trajectories. Compared to recent works, the main contributions are the inclusion of a map of the harbor and additional measurements from range sensors, such as LIDAR and ultrasonic distance sensors, to account for map inaccuracies as well as unmapped objects, such as moored vessels. To use the map and sensor data, a set generation method is developed, which in real-time computes a safe operating region, this is then used to ensure the planned trajectory is safe. To track the planned trajectory, a trajectory-tracking dynamic positioning controller is used. The performance of the method is tested experimentally on a small ASV in confined waters in Trondheim, Norway. The experiments demonstrate that the proposed method is able to perform collision-free docking maneuvers with respect to static obstacles, and achieves successful docking.

INDEX TERMS Autonomous surface vehicles, berthing, collision avoidance, docking, marine vehicles, motion planning, optimal control, trajectory optimization.

I. INTRODUCTION

Autonomy, and autonomous systems is a rapidly growing area of interest in a wide variety of industries. This includes the maritime industry, where autonomous surface vehicles (ASVs) have been proposed for surveying and mapping, surveillance, and transportation, to name a few application areas. With the motivation factors including lower costs, higher availability and flexibility, better safety and reliability, and reduced environmental impact. In the case of transportation, autonomous operations can be roughly split into the following three phases.

- Undocking – moving from the quay in a confined harbor area to open waters,
- transit – crossing a body of water towards the destination harbor,
- docking – moving from open waters towards the docking position along the quay in a harbor area.

In this paper we will focus on docking of a vessel in a confined harbor area. While the method we propose in this paper is able to solve both the docking and undocking problem, the focus is on docking, as it is the most challenging of the two and requires very precise movements [1] when performing the final controlled collision with the quay.

The problem of automatic docking and berthing is an important part of performing autonomous transportation, and hence the problem has seen a lot of interest, with a variety of solutions. However, due to the complexity of performing docking, most of the existing methods rely on simplifying the docking problem, this has lead to a lack of experimental results. The traditional approach for docking large under-actuated vessels, requires the use of tug boats, as support vessels, in order to push and pull the vessel to

The associate editor coordinating the review of this manuscript and approving it for publication was Xiwang Dong.

perform the docking maneuver. This has lead to research into synchronizing the movement of multiple tugboats, in order to perform the desired maneuvers [2]–[5]. With many newer vessels being fully actuated, or even over-actuated, research has shifted to seeking methods for automatically performing docking without the use of additional support vessels. One such approach to solving the docking problem is via fuzzy control. The control system is then based on fuzzy logic, and its behaviour changes based on a set of predetermined rules [6]–[8]. An other approach for docking, that has seen a lot of interest, is the use of Artificial Neural Networks (ANNs) [2], [5], [9]–[15]. For these approaches, an ANN is used as a function approximator for the policy, and is tasked with learning to imitate pre-recorded docking trajectories, and hence learning how to perform the docking maneuvers. More recent learning-based methods have expanded on this by using advances in Deep Learning (DL) [16]–[18]. Additional approaches include docking using a rule-based expert system [19], docking by target tracking [20], and docking using artificial potential fields [21]. Within industry, several companies have developed methods for automatic docking [22]–[24], however details about the different approaches remain sparse. The most promising approaches however, rely on optimization-based planning [25]–[34], where trajectories are planned using convex optimization. These methods are often preferable, as they allow for explicitly including dynamics and constraints when planning a trajectory.

When developing automatic docking systems to facilitate berthing of vessels, it is important to have accurate and reliable positioning systems in place. This is required in order accurately determine the position of the vessel hull relative to the berth. While this is possible to do using satellite positioning systems, it requires high precision satellite positioning and the position of the berth must be well known, which may not always be the case. In order to overcome these problems, the use of quay-mounted laser or radar ranging systems [35]–[37] is often used in larger ports, in order to independently identify the position and velocity of the vessel relative to the quay.

The docking method from [32] is a nonlinear model predictive controller (NMPC) that takes into account vessel dynamics in the form of its dynamic model, as well as collision avoidance by planning trajectories within a convex set, based on the harbor layout. Advantages of this approach include the explicit handling of static obstacles, the planning of dynamically feasible trajectories, and a flexible behavior shaping via the nonlinear cost function. The method does not handle moving obstacles or account for external unknown disturbances. Additionally, due to the non-convexity of the optimal control problem, guarantees on run time or feasibility are not provided. In this paper, we build on [32], [33] and propose a novel algorithm for dynamically creating a convex safety set, based on a map of the environment. We also show how this method can be combined with sensor data from on board sensors such as LIDAR pointclouds, and ultrasonic distance sensors, in order to account for missing or inaccurate map data. This allows to plan and perform docking maneuvers in harbors without the need for land-based sensor systems, even if the harbor layout changes. We also propose some modifications to the cost function, in order to generate more efficient docking trajectories. Finally, we validate the method in full-scale experiments on the experimental autonomous urban passenger ferry *milliAmpere*, seen in Figure 1, and show how the proposed approach is able to successfully plan and perform safe and collision free docking maneuvers in confined waters in Trondheim, Norway. The contributions of this work can be split into two main categories, namely methodology, and implementation.

FIGURE 1. Experimental platform milliAmpere.

1) Methodology builds on the work in [32], with the following improvements:
   - A set generation method for identifying a safe operating region in real-time (Section IV-A).
   - Improvements to the cost function, which give more refined docking maneuvers (Section III-B).
2) Implementation builds on the work in [33], with the following improvements:
   - Addition of exteroceptive sensor data to account for map inaccuracies as well as unmapped objects (Section IV-B).
   - Improved interplay between the tracking controller and planner for better tracking performance (Section V-A).
   - Improvements to the cost function, which gives better tracking performance (Section III-B).

To the authors' best knowledge, this is the first work to demonstrate fully automatic docking using only on-board sensors and map data, and use this data to plan a safe and feasible trajectory in real-time.

II. VESSEL MODEL

This section presents the kinematic, dynamic and thruster models of an ASV, which have to be taken into account when planning a safe and feasible docking trajectory.

A. KINEMATICS AND DYNAMICS

When modeling vessels for the purpose of autonomous docking, we consider only the vessel movement on the ocean

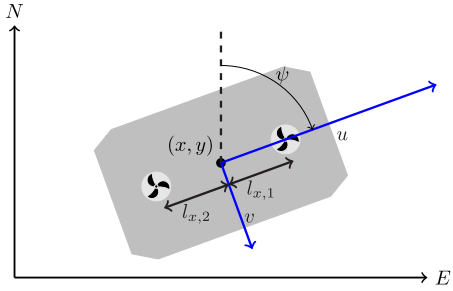


FIGURE 2. 3-DOF vessel centered at $p_c = [x, y]^\top$, with surge velocity u, sway velocity v, heading ψ in a North-East-Down (NED) reference frame.

surface, neglecting the roll, pitch and heave motions. The mathematical model used to describe the system can then be kept reasonably simple as it is limited to the planar position and orientation of the vessel. Given $\mathbb{R}$ as the set of real numbers, $\mathbb{S} = [0, 2\pi]$ as the set of angles, and $SO(n) = \{\boldsymbol{R} | \boldsymbol{R} \in \mathbb{R}^{n\times n}, \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}, \det(\boldsymbol{R}) = 1\}$ as the special orthogonal group in n dimensions, the motion of a surface vessel can be represented by the pose vector $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times \mathbb{S}$, and velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$. Here, $\boldsymbol{p}_c = [x, y]^\top$ describe the Cartesian position in the Earth-fixed reference frame, ψ is yaw angle, (u, v) is the body fixed linear velocities, and r is the yaw rate, an illustration is given in Figure 2. Using the notation in [38] we can describe a 3-DOF vessel model as follows

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\psi)\boldsymbol{\nu}, \tag{1}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \tag{2}$$

where $\boldsymbol{M} \in \mathbb{R}^{3\times3}$, $\boldsymbol{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{\tau} \in \mathbb{R}^3$ and $\boldsymbol{J}(\psi) \in SO(3)$ are the inertia matrix, Coriolis matrix, dampening matrix, control forces and moments, and transformation matrix, respectively. The transformation matrix $\boldsymbol{J}(\psi)$ is given by

$$\boldsymbol{J}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

and represent the rotation from the body frame to the Earth-fixed reference frame. For detailed information about the *milliAmpere* model parameters used in this paper, the reader is referred to [33].

B. THRUST CONFIGURATION

The control surfaces of the vessel are specified by the thrust configuration matrix $\boldsymbol{T} \in \mathbb{R}^{3,n_{\text{thrusters}}}$, which maps the thrust $\boldsymbol{f} \in \mathbb{R}^{n_{\text{thrusters}}}$ from each thruster into the surge, sway and yaw forces and moments in the body frame of the vessel

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{f}. \tag{4}$$

Each column $\boldsymbol{T}_i$ in $\boldsymbol{T}$ gives the configuration of the forces and moments of a thruster i as follows:

$$\boldsymbol{T}_i f_i = \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{y,i} \cdot l_{x,i} - F_{x,i} \cdot l_{y,i} \end{bmatrix}, \tag{5}$$

where $F_{x,i}$ and $F_{y,i}$ are the forces in the body frame, and $l_{x,i}$ and $l_{y,i}$ is the position of the thruster in the body frame. Given a desired force vector $\boldsymbol{\tau}$, finding the individual thruster forces that produce it, is called the thrust allocation problem. While there are numerous ways of solving the thrust allocation problem [39], in order to account for the thrust configuration and individual thruster constraints, we want to include the thrust allocation as part of the optimization and planning for performing the docking operations. For the *milliAmpere* vessel, illustrated in Figure 2, there are two thrusters mounted along the center line of the vessel, giving the following control force and moments:

$$\boldsymbol{\tau} = \begin{bmatrix} F_{x,1} \\ F_{y,1} \\ F_{y,1} \cdot l_{x,1} \end{bmatrix} + \begin{bmatrix} F_{x,2} \\ F_{y,2} \\ F_{y,2} \cdot l_{x,2} \end{bmatrix}. \tag{6}$$

III. TRAJECTORY PLANNING AND CONTROL

Automatic docking is a complex problem, which includes planning and performing maneuvers to control a vessel to a desired orientation and position, while adhering to spatial constraints in order to avoid collisions. In order to perform the docking, we expand upon [32], [33], where we use a docking trajectory planner, constructed as an Optimal Control Problem (OCP). This allows the planner to take into account the vessel dynamics in terms of a mathematical model, as well as the harbor layout in terms of a map of landmasses. Additionally we include the use of ranging sensors, namely ultrasonic distance sensors and LIDAR, in order to account for obstacles not present in the map of the harbor layout.

A. OBSTACLE AVOIDANCE

Given a desired position x_d, y_d and a desired heading ψ_d, we define the docking problem as maneuvering a vessel as close as possible to the desired docking pose $\boldsymbol{\eta}_d = [x_d, y_d, \psi_d]^\top$, without running the vessel into obstacles, i.e. adhering to spatial constraints. As proposed in [32], the safe operation of the vessel can be formalised in terms of the vessel boundary $\mathbb{S}_v$ being contained within a convex inner approximation of the surrounding obstacles $\mathbb{S}_s$, called the spatial constraints, see Figure 3. Since the safe operating region, given in terms of the spatial constraints $\mathbb{S}_s$, is given as a convex set, the region can be defined in terms of a set of linear inequality constraints:

$$\mathbb{S}_s = \{\boldsymbol{p} \in \mathbb{R}^2 \mid \boldsymbol{A}_s\boldsymbol{p} \leq \boldsymbol{b}_s\},$$

where the matrix $\boldsymbol{A}_s \in \mathbb{R}^{n_{\text{constraints}},2}$ and vector $\boldsymbol{b}_s \in \mathbb{R}^{n_{\text{constraints}}}$ define the linear inequality constraints. Furthermore, we can note that if both the vessel set $\mathbb{S}_v$ and spatial constraint $\mathbb{S}_s$ are convex, then the vessel is contained within the spatial constraints so long as all the vertices of the vessel boundary are contained within the spatial constraints. This is useful as it allows us to simplify the safety constraints to the following:

$$\mathbb{S}_v \subseteq \mathbb{S}_s \iff \boldsymbol{A}_s\boldsymbol{p}_i^n \leq \boldsymbol{b}_s \quad \forall \boldsymbol{p}_i^n \in \text{Vertex}(\mathbb{S}_v), \tag{7}$$

where $\boldsymbol{p}_i^n$ denotes the position of vertex i in the North-East-Down (NED) reference frame. Since the vertices of the

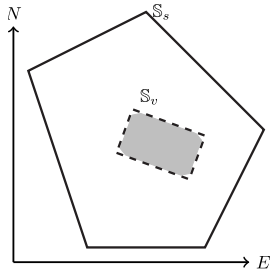


FIGURE 3. The gray convex polytope illustrates the vessel, whereas $\mathbb{S}_v$ is the vessel boundary, and $\mathbb{S}_s$ are the spatial constraints. The vessel will always lie within the spatial constraints $\mathbb{S}_s$ as long as all the vertices of $\mathbb{S}_v$ lie within the spatial constraints.

vessel boundary are given in the body frame of the vessel, we need to transform them from the body frame to the NED frame, giving the following nonlinear constraints:

$$\boldsymbol{A}_s \left(\boldsymbol{R}(\psi) \boldsymbol{p}_i^b + \boldsymbol{p}_c \right) \leq \boldsymbol{b}_s \quad \forall \boldsymbol{p}_i^b \in \text{Vertex}(\mathbb{S}_v), \tag{8}$$

where $\boldsymbol{p}_i^b$ denotes the position of vertex i in the body frame, $\boldsymbol{p}_c = [x, y]^\top$ is the vessel position and $\boldsymbol{R}$ is the rotation from the body frame to NED.

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{9}$$

The constraints in (8) can be directly implemented as inequality constraints in an optimization problem, and ensures the vessel is contained within a safe region given by the spatial constraints.

While this constraint is easily implemented in a nonlinear programming (NLP) problem, the constraint is not convex. As a result, a feasible solution of the problem is guaranteed to satisfy the spatial constraints, but we cannot guarantee that the NLP will converge to a global optimum.

B. OPTIMAL CONTROL PROBLEM

In order to plan a safe trajectory for the vessel, we formulate the problem as the following continuous time nonlinear optimal control problem:

$$\min_{\boldsymbol{x}_p(\cdot), \boldsymbol{u}_p(\cdot), \boldsymbol{s}(\cdot)} \int_{t_0}^{t_0+T} \left(l(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) + \boldsymbol{k}_s^\top \boldsymbol{s}(t) \right) dt \tag{10a}$$

$$\text{s.t. } \dot{\boldsymbol{x}}_p(t) = \boldsymbol{f}(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) \quad \forall t \in [t_0, t_0 + T] \tag{10b}$$

$$\boldsymbol{h}(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) - \boldsymbol{s}(t) \leq \boldsymbol{0} \quad \forall t \in [t_0, t_0 + T] \tag{10c}$$

$$\boldsymbol{s}(t) \geq \boldsymbol{0} \quad \forall t \in [t_0, t_0 + T] \tag{10d}$$

$$\boldsymbol{x}_p(t_0) = \boldsymbol{x}(t_0), \tag{10e}$$

where $\boldsymbol{x}_p(t) = [\boldsymbol{\eta}_p, \boldsymbol{\nu}_p]^\top$ is the planned state trajectory of the vessel, $\boldsymbol{u}(t) = [F_{x,1}, F_{y,1}, F_{x,2}, F_{y,1}]^\top$, are the planned thruster forces, and $\boldsymbol{s}(t)$ are slack variables. The constraint relaxation (10c), is added in order to allow the planner to plan a trajectory from a possibly infeasible initial pose. This is useful, as it allows for re-planning the docking trajectory in a model predictive control (MPC) like fashion, and use low-level controllers to follow the planned trajectory.

The cost function (10a) includes a slack variable cost, and a stage cost. For the slack variable cost $\boldsymbol{k}_s^\top \boldsymbol{s}(t)$, the gain $\boldsymbol{k}_s$ is chosen large enough such that the slack variables are active only when the non-relaxed constraints are infeasible. The following stage cost was chosen:

$$l(\boldsymbol{x}_p(t), \boldsymbol{u}_p(t)) = q_{x,y} \cdot c_{x,y}(\boldsymbol{\eta}_p(t)) + q_\psi \cdot c_\psi(\boldsymbol{\eta}_p(t)) + \boldsymbol{\nu}_p(t)^\top \boldsymbol{Q} \boldsymbol{\nu}_p(t) + \boldsymbol{u}_p(t)^\top \boldsymbol{R} \boldsymbol{u}_p(t). \tag{11}$$

The velocity and control actions are penalized with a quadratic penalty, with weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. The position cost $c_{x,y}(\boldsymbol{\eta}_p(t))$ is chosen as a pseudo-Huber function, penalizing the difference between the current pose and the docking pose $\boldsymbol{\eta}_d$, and is given as follows:

$$c_{x,y}(\boldsymbol{\eta}_p) = \delta^2 \left(\sqrt{1 + \frac{(x_p(t) - x_d)^2 + (y_p(t) - y_d)^2}{\delta^2}} - 1 \right). \tag{12}$$

Using a pseudo-Huber cost, provides a quadratic penalty when the quadrature position error is low and linear when the position error is high. This helps with numerical stability, as well as performance when large position errors are observed [40], [41]. For the heading cost function $c_\psi(\boldsymbol{\eta}_p(t))$, the following was chosen:

$$c_\psi(\boldsymbol{\eta}_p(t)) = \frac{1 - \cos(\psi_p(t) - \psi_d))}{2} e^{-\frac{(x_p(t) - x_d)^2 + (y_p(t) - y_d)^2}{2\delta^2}}. \tag{13}$$

The first factor of the heading cost function is the cost of the heading error, in a form that avoids the wraparound of the heading. The second part is a Gaussian function, which discounts the heading error when far away from the docking pose. This cost function has the effect of having the planner chose an efficient heading when far away from the dock, and then gradually rotate the vessel towards the desired heading when closing in on the dock. It is worth noting that the cost works similarly to a terminal cost, but is phased in more gradually than a genuine terminal cost. This has the benefit of making the OCP less sensitive to the length of the prediction horizon.

The first constraint (10b) ensures that the planned trajectory satisfies the kinematic and dynamic models of the vessel described by (1) and (2). The inequality constraint (10c) consists of the spatial constraints described in (8), as well as constraints on the maximum and minimum velocity, angular velocity, and thruster forces. The constraint in (10d) ensures that the slack variables are positive, and (10e) sets the initial state to that of the vessel.

In order to implement the continuous time problem given in (10) we need to transcribe the problem into the standard form.

$$\begin{aligned} &\min_{\boldsymbol{w}} \ \phi(\boldsymbol{w}) \\ &\text{s.t. } \boldsymbol{g}(\boldsymbol{w}) = 0 \\ &\quad \boldsymbol{h}(\boldsymbol{w}) \leq 0 \end{aligned}$$

This can be done in multiple ways, however the two main classes of methods are *sequential methods*, such as direct single shooting [42], and *simultaneous methods* such as direct multiple shooting [43], and direct collocation [44]. For this approach we chose to use direct collocation, in where implicit numerical integration of the ordinary differential equation (ODE) constraints (10b), as well as the objective function (10a), is performed as part of the nonlinear optimization. For the implementation of the docking planner, we defined a planning horizon of $T = 120s$, with $N = 60$ shooting intervals, and degree $d = 3$ Legendre polynomials. This was chosen as it gave a good trade-off in terms of horizon length, integration accuracy and computational complexity.

IV. AUTOMATIC CONSTRAINT GENERATION

Given the OCP formulation in the previous section, we are faced with the problem of finding a convex set:

$$\mathbb{S}_s = \{\boldsymbol{p} \in \mathbb{R}^2 \mid \boldsymbol{A}_s\boldsymbol{p} \leq \boldsymbol{b}_s\},$$

within which the vessel must be contained. The set $\mathbb{S}_s$ must be created in a way such that it does not intersect with any constraints stemming from the environmental obstacles. Ideally we would also like the set to be as large as possible, in order to not unnecessarily restrict the vessel movement. While a number of methods for performing constraint generation already exists [45]–[48], they can often be computationally expensive. In this section, we propose a method similar to [45], [46], where a constraint set is generated as a vessel-centered convex inner approximation of the environmental constraints. We will also show how we can easily and efficiently compute this set from known map data, as well as from range sensor such as LIDAR and ultrasonic distance sensors.

A. COMPUTING A CONVEX INNER APPROXIMATION

In this section, we show how to compute the safe set as a vessel-centered convex inner approximation of the environmental constraints, when the obstacles are represented by line segments making up obstacle polygons. Intuitively the method can be summarized as follows:

1) Given a center point $\boldsymbol{p}_c$, grow an ellipse centered at $\boldsymbol{p}_c$, until it touches an environmental constraint (Section IV-A1).
2) Create a linear constraint tangent to the expansion ellipse at the contact point between the ellipse and the environmental constraint (Section IV-A2).
3) Continue growing and creating linear constraints until no further growth is possible, and combine the linear constraints into the final convex inner approximation of the environment (Section IV-A3)

In the next subsections we will show how to quickly and efficiently perform the steps above in order to end up with a simple closed form solution to generating the convex inner approximation.

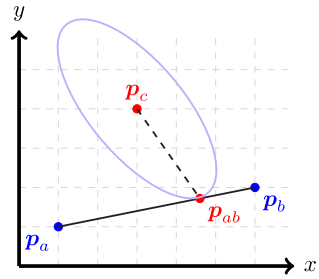


FIGURE 4. The contact point p_{ab} is given by the shortest non-Euclidean distance from point p_c to line segment (p_a, p_b).

1) COMPUTING THE CONTACT POINT BETWEEN ELLIPSE AND ENVIRONMENTAL CONSTRAINT

Given a line segment $(\boldsymbol{p}_a, \boldsymbol{p}_b)$ making up the environmental constraints, we want to find the contact point $\boldsymbol{p}_{ab}$ between the line segment and the expansion ellipse centered at $\boldsymbol{p}_c$, this is illustrated in Figure 4. We can formulate this as an optimization problem which minimizes the distance between $\boldsymbol{p}_c$ and the parametric line segment:

$$\boldsymbol{p}_{ab}(\omega) = \boldsymbol{p}_a(1-\omega) + \boldsymbol{p}_b\omega, \tag{14}$$

where $\omega \in [0, 1]$. For the optimization problem we wish to find the parameter ω that minimizes a non-Euclidean distance from $\boldsymbol{p}_{ab}(\omega)$ to $\boldsymbol{p}_c$, giving the following:

$$\min_{\omega} f(\omega) = \left(\boldsymbol{p}_{ab}(\omega) - \boldsymbol{p}_c\right)^\top \boldsymbol{\Sigma} \left(\boldsymbol{p}_{ab}(\omega) - \boldsymbol{p}_c\right) \tag{15}$$

$$s.t.\ 0 \leq \omega \leq 1, \tag{16}$$

where $\boldsymbol{\Sigma}$ is a positive definite symmetric projection matrix, defining the expansion ellipse. Choosing $\boldsymbol{\Sigma} = \boldsymbol{I}$ gives the Euclidean distance, while choosing $\boldsymbol{\Sigma}$ as a different positive definite symmetric matrix, allows prioritizing a direction, when minimizing the distance. For the unconstrained optimization problem, the necessary conditions give:

$$\begin{aligned}\frac{\mathrm{d}f(\omega)}{\mathrm{d}\omega} &= (\boldsymbol{p}_a(1-\omega) + \boldsymbol{p}_b\omega - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}(\boldsymbol{p}_b - \boldsymbol{p}_a) = 0 \\ \Rightarrow \omega &= -\frac{(\boldsymbol{p}_a - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}(\boldsymbol{p}_b - \boldsymbol{p}_a)}{(\boldsymbol{p}_b - \boldsymbol{p}_a)^\top \boldsymbol{\Sigma}(\boldsymbol{p}_b - \boldsymbol{p}_a)}.\end{aligned} \tag{17}$$

This gives the parameterized variable ω for the unconstrained problem. The constrained $\omega \in [0, 1]$, due to the simplicity of the constraints and convexity of the optimization problem, can be found by simply clipping ω between 0 and 1:

$$\omega = \mathrm{clip}\left(-\frac{(\boldsymbol{p}_a - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}(\boldsymbol{p}_b - \boldsymbol{p}_a)}{(\boldsymbol{p}_b - \boldsymbol{p}_a)^\top \boldsymbol{\Sigma}(\boldsymbol{p}_b - \boldsymbol{p}_a)}, 0, 1\right) \tag{18}$$

The closest point $\boldsymbol{p}_{ab}$, constrained to being on the line segment $(\boldsymbol{p}_a, \boldsymbol{p}_b)$ is then given by inserting the parameter w from (18) into (14). We should note that this gives a closed form solution for the contact point $\boldsymbol{p}_{ab}$, which means it can be efficiently computed.

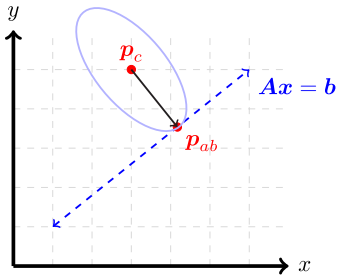


FIGURE 5. Finding the line $Ax = b$ tangent to the expansion ellipse at p_{ab}, i.e. normal line to the vector $\Sigma(p_{ab} - p_c)$ passing through p_{ab}.

2) FINDING THE TANGENT LINE TO THE EXPANSION ELLIPSE

Given the closest point $\boldsymbol{p}_{ab}$ on a line segment, the next step is to find the line $\boldsymbol{Ax} = \boldsymbol{b}$ which is tangent to the expansion ellipse, and hence normal to the ellipse gradient $\boldsymbol{\Sigma}(\boldsymbol{p}_{ab} - \boldsymbol{p}_c)$, as illustrated in Figure 5. Given the expansion ellipse gradient for a point $\boldsymbol{p}_{ab}$ as:

$$\boldsymbol{\Sigma}(\boldsymbol{p}_{ab} - \boldsymbol{p}_c) \tag{19}$$

we can note that any nonzero vector $[x, y]^\top$ is orthogonal to the expansion ellipse, if the inner product is zero.

$$(\boldsymbol{p}_{ab} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \begin{bmatrix} x \\ y \end{bmatrix} = 0 \;\Rightarrow \text{Orthogonal.} \tag{20}$$

Using (20), the tangent line of the expansion ellipse, passing through the point $[x_0, y_0]^\top$ is given as the following.

$$(\boldsymbol{p}_{ab} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \right) = 0 \tag{21}$$

Since we want the tangent line of the expansion ellipse to pass through the point $\boldsymbol{p}_{ab}$, the tangent line is given by all points $[x, y]^\top$ which fulfill the following equality.

$$\underbrace{(\boldsymbol{p}_{ab} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{(\boldsymbol{p}_{ab} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab}}_{\boldsymbol{b}} \tag{22}$$

3) LINEAR INEQUALITY CONSTRAINT GENERATION

In order to generate the convex constraints around a point $\boldsymbol{p}_c$, our proposed method is based on finding the tangent line to the expansion ellipse (see Section IV-A2) from the point $\boldsymbol{p}_c$ to the closest point $\boldsymbol{p}_{ab,i}$ (see Section IV-A1) on each line segment $i \in 1, 2, \ldots N$, making up environmental constraints. By stacking the tangent lines we get a half-space representation of the convex inner approximation as the following linear inequality constraints.

$$\underbrace{\begin{bmatrix} (\boldsymbol{p}_{ab,1} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \\ (\boldsymbol{p}_{ab,2} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \\ \vdots \\ (\boldsymbol{p}_{ab,N} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{p} \leq \underbrace{\begin{bmatrix} (\boldsymbol{p}_{ab,1} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab,1} \\ (\boldsymbol{p}_{ab,2} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab,2} \\ \vdots \\ (\boldsymbol{p}_{ab,N} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab,N} \end{bmatrix}}_{\boldsymbol{b}} \tag{23}$$

We can note that since (18) is piecewise linear and smooth, the constraints given above are continuous with respect to the center point $\boldsymbol{p}_c$. This is a useful property, as this means that the shape of the convex inner approximation will change continuously with the center point $\boldsymbol{p}_c$.

While (23) gives a set of linear inequalities that may be used directly in an OCP, it is worth noting that the rows of the constraint may contain a large variance in magnitude. Since the inequalities are linear however, we can multiply each row of the inequalities by a normalizing factor. One such convenient normalizing factor is:

$$\frac{1}{||(\boldsymbol{p}_{ab,i} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \tag{24}$$

The reason this normalizing factor is convenient, is the fact that the resulting matrix $\boldsymbol{A}$ becomes dimensionless with every row having unit length, and hence the the constraint will have the same units as $\boldsymbol{p}$. This has several benefits, including that $\boldsymbol{Ap} - \boldsymbol{b}$ will have a physical meaning in terms of distance until the constraint is reached. This allows for adding a back-off or margin in order to shrink the constraints, and make them more conservative. An other benefit is when using a slack variabels in order to relax the constraints in the OCP, the slack variable will have a physical meaning. Using this normalization factor we get the linear inequality constraints given below.

$$\underbrace{\begin{bmatrix} \frac{(\boldsymbol{p}_{ab,1} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}}{||(\boldsymbol{p}_{ab,1} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \\ \frac{(\boldsymbol{p}_{ab,2} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}}{||(\boldsymbol{p}_{ab,2} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \\ \vdots \\ \frac{(\boldsymbol{p}_{ab,N} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma}}{||(\boldsymbol{p}_{ab,N} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{p} \leq \underbrace{\begin{bmatrix} \frac{(\boldsymbol{p}_{ab,1} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab,1}}{||(\boldsymbol{p}_{ab,1} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \\ \frac{(\boldsymbol{p}_{ab,2} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab,2}}{||(\boldsymbol{p}_{ab,2} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \\ \vdots \\ \frac{(\boldsymbol{p}_{ab,N} - \boldsymbol{p}_c)^\top \boldsymbol{\Sigma} \boldsymbol{p}_{ab,N}}{||(\boldsymbol{p}_{ab,N} - \boldsymbol{p}_c)\boldsymbol{\Sigma}||_2} \end{bmatrix}}_{\boldsymbol{b}} \tag{25}$$

4) REMOVING REDUNDANT CONSTRAINTS

The inequality constraints in (25) can be further reduced to $M \leq N$ constraints, by removing redundant constraints (Figure 6). Given a system of linear inequalities,

$$\boldsymbol{Ap} \leq \boldsymbol{b} \tag{26}$$

a given row $\boldsymbol{A}_k, \boldsymbol{b}_k$ can be identified as being a redundant constraint by solving the following Linear Programming (LP) problem:

$$\begin{aligned} &\min_{\boldsymbol{p}} \; y_k = \boldsymbol{b}_k - \boldsymbol{A}_k \boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}_i \boldsymbol{p} \leq \boldsymbol{b}_i \quad \forall i \neq k, \end{aligned} \tag{27}$$

and checking if the above problem has a solution $\boldsymbol{p}$ for which $y_k \geq 0$ [49]. For large numbers of constraints, solving an LP to check if each constraint is redundant is however very inefficient. Fortunately, a number of other approaches exists, [50]. For our approach, we used the qhull library [51], which efficiently perform constraint reduction for large numbers of

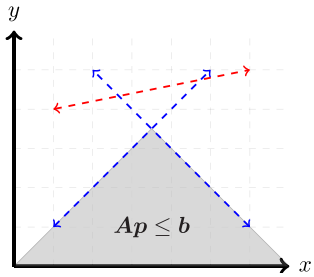
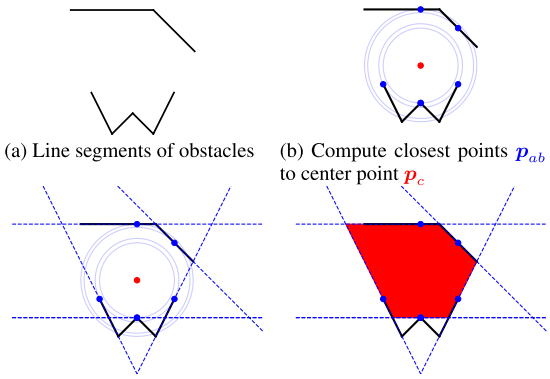


FIGURE 6. Redundant constraints are constraints that don't make up the support of the intersecting half-plane $\{p \in \mathbb{R}^2 \mid Ap \leq b\}$.



(a) Line segments of obstacles

(b) Compute closest points p_{ab} to center point p_c

(c) Find tangent lines to the expansion ellipse at each point p_{ab}

(d) Final convex inner approximation

FIGURE 7. Illustration of how to compute the convex spatial constraints.

constraints. It does this by first computing the dual points of the constraints as:

$$\boldsymbol{d}_i = \frac{\boldsymbol{A}_i}{\boldsymbol{b}_i - \boldsymbol{A}_i \boldsymbol{p}}, \tag{28}$$

where $\boldsymbol{p}$ is an interior point of the constraints. Then computes the convex hull of the dual points as:

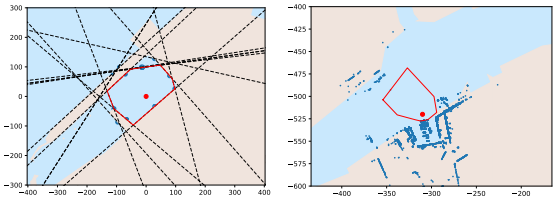
$$\mathcal{D} = \text{Conv}(\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots \boldsymbol{d}_N\}). \tag{29}$$

The support of the constraints are then given by the rows i for which the dual points $\boldsymbol{d}_i$ are extreme points of the convex hull $\mathcal{D}$, and redundant for dual points which are not extreme points of $\mathcal{D}$.

Given a set of line segments making up the boundary of obstacles, we can use the method above to first compute the closest points to all obstacles, then find the tangent line to the expansion ellipse, and generate the final constraint as a set of linear inequality constraints. This procedure is illustrated in Figure 7.

B. COMPUTING SPATIAL CONSTRAINTS FROM MAP AND SENSOR DATA

In order to compute the spatial constraints for the docking problem, it is possible to use a known map of the environment. Given a map where landmasses are represented as polygons, we can use the proposed constraint generation method with the line segments making up the edges of the polygons. This will give a convex inner approximation which can be used in the docking planner. This is shown in Figure 8a, where we compute the safe region of water, not intersecting polygonal land masses around a certain point.



(a) Set generation from map data, where blue dots are the Closest points p_{ab} on the line segments of the land masses.

(b) Set generation from map and LIDAR data, where blue dots represent the point cloud from the LIDAR sensor.

FIGURE 8. Set generation is performed by growing a region (red polygon) of water around p_c (red dot), that does not intersect with land.

In many real world applications however, relying only on map data may not be sufficient, as the maps may be inaccurate, out of date, or missing information. In order to compensate for this we propose using additional sensory information, in order to account for inaccurate map information. For our proposed constraint generation method, point cloud data–such as that generated by LIDAR, radar, or other types of proximity sensors–can be easily incorporated. This can be done by directly using the points in the point cloud as the close points $\boldsymbol{p}_{ab}$. An example of a map augmented with LIDAR data can be seen in Figure 8b. For sensor data such as short range ultrasonic distance measurements, where the sensors are configured as in Figure 9, we can approximate the constraints seen by the sensors as the line segment between the measurement of each sensor. This line segment can then be added to the constraint set similarly to the map data. Using redundant and various exteroceptive sensors is beneficial, as additional sensor may be used to improve coverage and avoid blind spots, which will improve the accuracy of spatial constraints.

When computing the constraint set, it is also possible to use the projection matrix $\boldsymbol{\Sigma}$ in order to create an axis for which we prioritize the set generation. In the case of a vessel, it it can be useful to prioritize constraint generation in the longitudinal direction of the vessel's body frame. This can be done by using the following projection matrix:

$$\boldsymbol{\Sigma} = \boldsymbol{R}(\psi) \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \boldsymbol{R}(\psi)^\top, \tag{30}$$

where choosing $\sigma_x < \sigma_y$ will prioritize set expansion in the direction of the vessel heading. For docking or set point tracking, it is alternatively possible to expand the spatial constraints in the direction of the dock or set point, as this is the direction that we ideally want the vessel to travel.

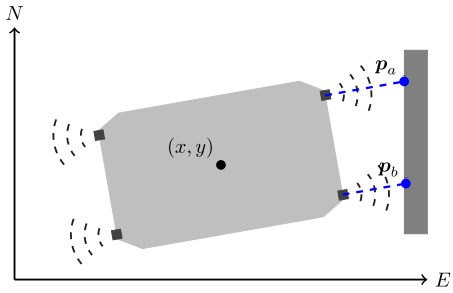


FIGURE 9. Ultrasonic distance sensors attached to the front and rear of the vessel, where the line segments (p_a, p_b) can be added to the spatial constraints.

C. CONSIDERATIONS WHEN USING THE CONVEX SET GENERATION FOR PLANNING

The set generation method we have detailed above, is a computationally efficient way of computing a good inner approximation of a set of non-convex constraints. The goal of the method is not to maximize the area of the convex inner approximation, as this is in general a computationally expensive, and is not guaranteed to create a set which expands in a desired direction. The goal of the method is rather to create an inner approximation with a preferred expansion direction, in our case this is controlled by the expansion ellipse, which is efficient to compute even when handling large numbers of constraints.

When using the constraints from the constraint generation method in the docking planner, it is useful to be able to guarantee recursive feasibility of the planner when it is run in an MPC like fashion. By updating the constraints frequently, and only choosing a new constraint set if it remains feasible for the previous iteration, recursive feasibility of the planner can be guaranteed. We can note that the set generation will always remain feasible for the point $\boldsymbol{p}_c$, however since we are considering all the vessel vertices when planning a safe trajectory, we can not guarantee that the constraint generation method will results in a feasible constraints for all the vessel vertices.

In order to use the linear constraints in an optimization problem, it is practical to have a fixed number of constraints, such that the optimization problem does not need to be transcribed, and built each time a new number of constraints change. In order to do this we use the full constraints generated by the constraint generation method described above, and create a reduced constraint as the K closest constraints in terms of the distance $\boldsymbol{p}_{ab} - \boldsymbol{p}_c$. This reduced constraint, will then be an outer approximation of the full constraints.

V. EXPERIMENTS

A. EXPERIMENTAL PLATFORM

For the sea trials, we used the experimental autonomous urban passenger ferry *milliAmpere*, as shown in Figure 1 with the specifications listed in Table 1, and the planning parameters gien in Appendix. The *milliAmpere* platform has been in constant development at the Norwegian University of Science and Technology (NTNU) since 2017, and has served as a platform for testing and developing autonomous technology, including software, sensor arrays, as well as hardware solutions. A larger version is currently being designed and built by the research group Autoferry,[1] and is planned to operate as an on-demand ferry in the Trondheim harbor.

TABLE 1. ***milliAmpere*** **specifications.**

Dimensions	5m by 2.8m symmetric footprint
Position and heading reference system	Vector VS330 dual GNSS with Real-time kinematic (RTK) capabilities
Thrusters	Two azimuth thrusters on the center line, 1.8m aft and fore of center
Existing control modules	Trajectory tracking DP controller and thrust allocation system
Obstacle detection	Velodyne Puck VLP-16 LIDAR Sensor, and two front mounted ultrasonic range sensors (rear mounted ultrasonic sensor were not available).

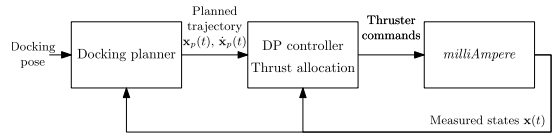


FIGURE 10. Block diagram of the docking system setup. The DP controller and thrust allocation are existing functions on *milliAmpere*.

For the experiments, the docking planner (10) was run with a sampling rate of 0.1Hz, with the output of the docking planner being used as the reference trajectory for a Dynamic Positioning (DP) tracking controller, which was already implemented on *milliAmpere*. The DP tracking controller was a proportional-integral-derivative (PID) controller, with velocity and acceleration feed-forward. Based on the forces and torque computed by the DP controller, the force and angles of the azimuth thrusters were calculated by an optimization-based control allocation scheme [52]. The block diagram in Figure 10 illustrates how the planner, DP controller, and actuators were connected.

Instead of using the DP controller with control allocation, it would have been possible to implement the docking planner (10) as a Nonlinear Model Predictive Control (NMPC) scheme, where the thruster forces computed by the planner are directly used as setpoints for the vessel thrusters. There are however several practical reasons why we chose not to do this:

- The planner does not account for drift, disturbances or modeling errors, while the tracking controller does so through feedback.

[1] Autoferry website: https://www.ntnu.edu/autoferry.

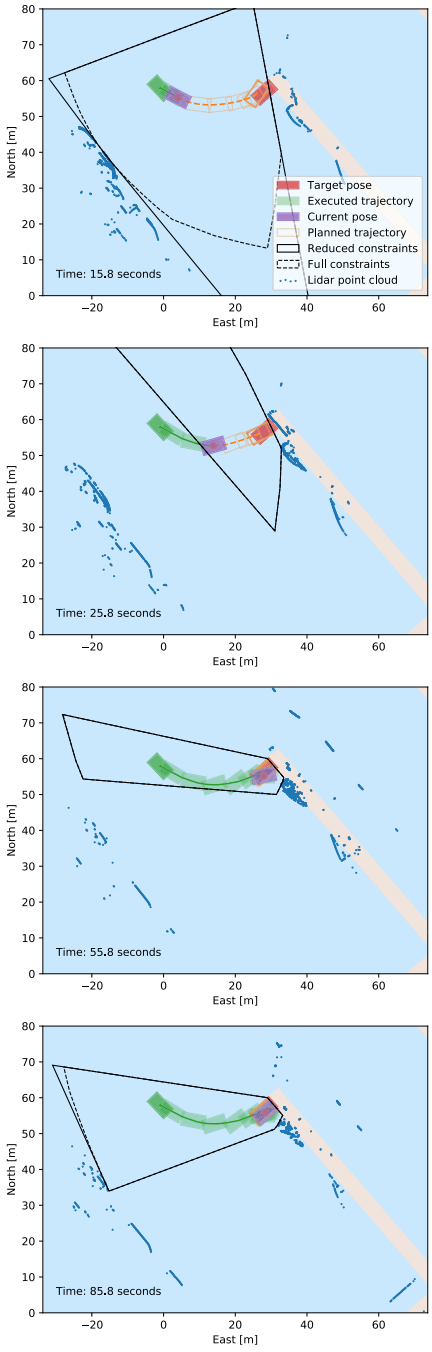


FIGURE 11. Visualisation of the docking motion during the experiments on September 7th 2020 (E1).

- While the planner is iteration-based with no formal performance guarantees, the tracking controller provides a robust bottom layer that acts also as a safety measure.

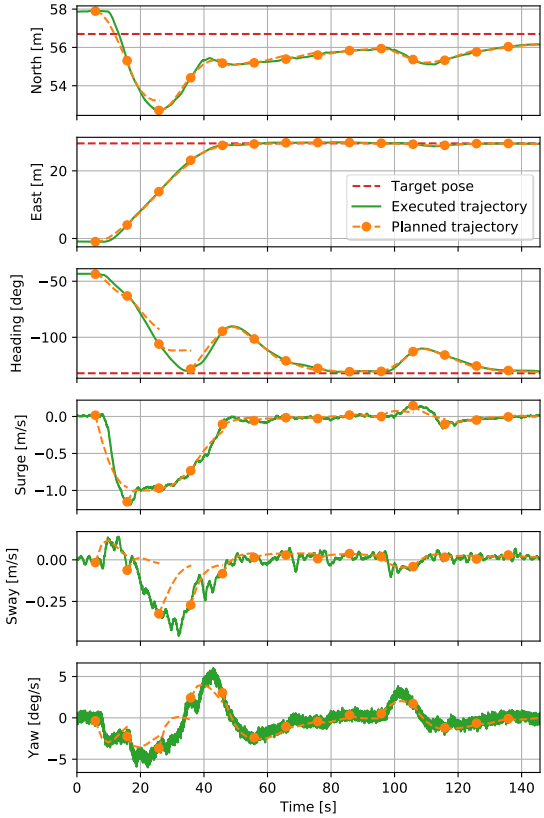


FIGURE 12. Planned and executed trajectory during the experiments on September 7th 2020 (E1).

- Due to the computational demand of solving the planning problem, it is difficult to achieve a sampling rate that is fast enough to stabilize the vessel through feedback from the planner.
- Using this multi-layer architecture separates the planner from the vessel control systems, allowing the planner to easily be retrofitted onto the existing implementation.

Choosing such a multi-layered structure, where planning and motion control are separated, provides flexibility in the trajectory planner, disturbance rejection through feedback, robustness to failures in the planning level, and ease of implementation on platforms with existing motion control systems.

B. RESULTS

Experiments[2] were performed with the *milliAmpere* platform in confined waters in Trondheim, Norway on September 7th (E1) and 11th (E2), 2020. The weather conditions were calm with winds between $1 m/s$ and $3\ m/s$. The results from E1 are shown in Figure 11 and 12, and the results from the E2 are shown in Figure 13 and 14, with photos from the experiments in Figure 15. It should be noted that the ultrasonic distance

[2]Video of experiments is available at: https://youtu.be/AyaWlJvI6K8

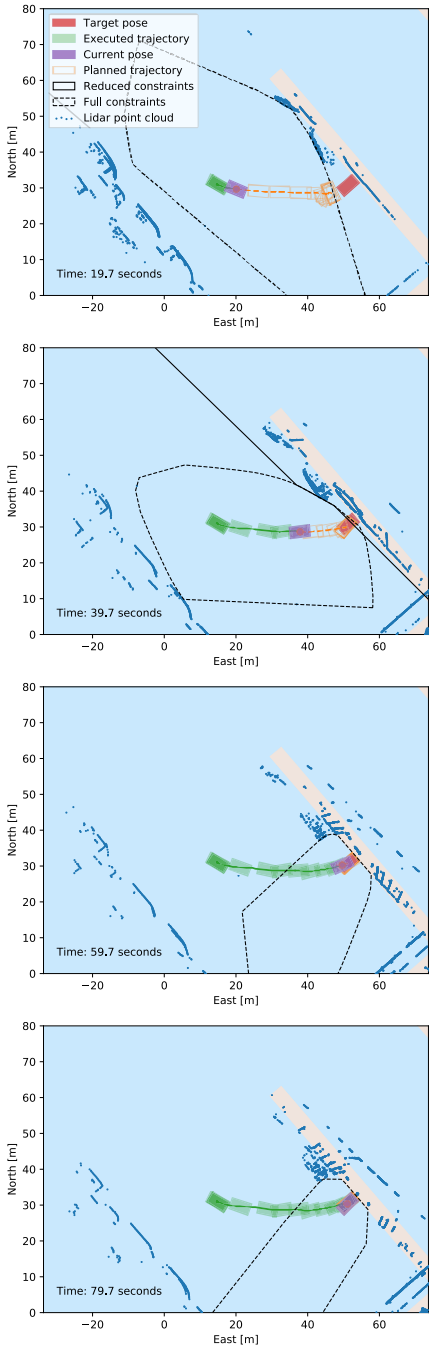


FIGURE 13. Visualisation of the docking motion during the experiments on September 11th 2020 (E2).

measurements were not used due to technical problems with the sensors at the time, meaning that only lidar and mapping data was used for computing the spatial constraints during the tests.

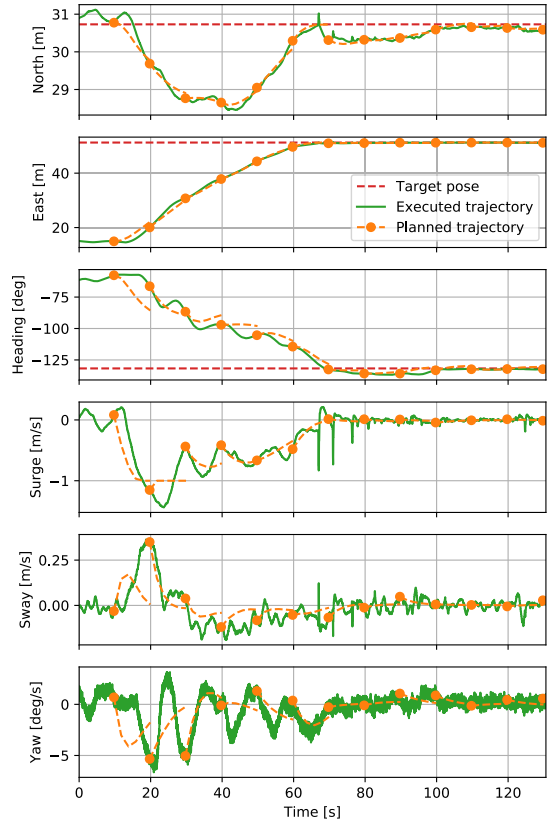


FIGURE 14. Planned and executed trajectory during the experiments on September 11th 2020 (E2).

The experiments E1 were performed at the end of a floating dock, while E2 at the center of the floating dock, due to to space availability on each day of the experiments. The difference in final docking pose had an influence on the complexity of the constraint sets, which can be seen in Figure 11 and 13. The *full constraints* pertain to the full set of constraints generated by the constraint generation method (25), while the *reduced constraints* were chosen as the 8 closest constraints eventually used in the optimization problem. This was done since the optimization problem needs a fixed number of constraints, and 8 gives a good balance between accuracy and computational cost when solving the OCP. For E1, shown in Figure 11, we see that the reduced constraints are much closer to the full constraints, compared to E2 Figure 13. This indicates that more potential obstacles were present in E2. The results show that the proposed constraint generation method is able to construct a good convex inner approximation of the free region, within which the vessel is allowed to operate, and that by choosing the number of constraints to reduce the full constraints to, we can achieve a good balance in terms of computation and constraint accuracy. We should however note the unexpected set of constraints at 25.8 seconds, in Figure 11, which was caused by rain

FIGURE 15. The *milliAmpere* while performing the docking maneuver, including closing in on the dock, and the final docking pose.

during the experiment, leading to the LIDAR misclassifying a raindrop as a potential obstacle. This can be avoided by better filtering the incoming LIDAR data before feeding them to the set generation algorithm. It is also worth noting the LIDAR point clouds in Figure 11 and 13, are not capturing the dock itself when the vessel is close to the final docking pose. This is due to limitations in the vertical transmitting angle of the LIDAR causing the dock to end up in the LIDAR blind spot. This problem can be solved by using the ultrasonic distance sensors at close range, adding redundant LIDAR sensors to improve coverage, or in this case relying on the map data, as the ultrasonic distance sensors were unavailable.

Figures 11 and 13 also show how the LIDAR helps in detecting unmapped static obstacles, in this case mostly docked vessels, especially as the milliAmpere gets closer to them. Accounting for these surrounding obstacles is of critical importance when planning and tracking a safe docking trajectory, and would not have been possible if only map data were used.

The results indicate that the planned trajectory does not violate any of the spatial constraints, and ensures that the vessel does not collide with surrounding obstacles. Also, the generated trajectory is intuitive for a docking operation, as the vessel initially moves in the surge direction towards the dock with a reasonably high velocity, then it slows down as it gets closer to the final docking pose, and finally rotates in order to reach the desired docking heading. Figures 12 and 14, indicate that the final trajectory mostly converges to the desired docking pose, with one exception being the North direction in Figure 12. This discrepancy was due to the docking pose overlapping with the dock, as can be seen in Figure 11, and hence the desired docking pose is not possible to be reached without violating the spatial constraints.

In Figure 12 and 14, we see the executed trajectory given in green, and the planned trajectory given in orange, where every 10 seconds the start of a replanned trajectory is marked with a dot. The observed discontinuity in the planned trajectory is due to the replanned trajectory being initialized to the state of the vessel at the time of the replanning. For both experiments we see that the tracking performance is very good. The tracking performance is highly reliant on not only the performance of the underlying DP controller and thrust allocation algorithm, but also the accuracy of the model used in the optimization-based planner. The most notable discrepancies in the tracking performance are found in the heading. We believe these are due to the the inherent heading instability of the vessel, as well as unmodeled thruster dynamics, which may cause a slight delay between desired and produced thrust.

VI. CONCLUSION

We have presented a method for planning and performing docking maneuvers in a confined harbor. The method utilizes map data, which is known in advance, as well as sensor data gathered in real time, to iteratively and safely plan a trajectory that brings the vessel to a desired docking pose. To perform the docking maneuvers given by the planner, we used an existing trajectory tracking DP controller, which added robustness to disturbances, and helped demonstrate that the planner is easy to retrofit on an existing platform. In order to validate the proposed control scheme, we conducted full-scale experiments in a confined harbor area with the *milliAmpere* ferry developed at NTNU, and demonstrated how the proposed method is able to plan and well as execute safe and collision-free docking maneuvers. To the best of our knowledge, there's no existing published work that involves field trials of docking operations for autonomous surface vehicles using only exteroceptive sensors.

For future work, we would like to look at the possibility of integrating additional sensors, such as radar and cameras, in order to generate an even better view of the environment. Additionally, we would like to look at ways of filtering the sensor data in order to get more reliable sensor readings. We would also like to integrate the docking system in a control structure that handles transportation phases autonomously. Since our proposed approach is able to handle the docking and undocking phase, what remains to achieve a fully autonomous operation with mission objective "Navigate from Dock A to Dock B", is the development of control and planning strategies for handling the transit phase of the journey. This development will include additional work on collision avoidance, situational awareness, and planning methods that comply with the maritime navigation rules.

APPENDIX
LIST OF PARAMETER VALUES

A list of parameter values is given in Table 2, while details on the vessel model can be found in [33].

TABLE 2. List of parameters.

Symbol	Value	Description
$q_{x,y}$	1	Huber penalty weight
q_ψ	20	Heading penalty weight
$\boldsymbol{Q}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 100 \end{bmatrix}$	Velocity weight matrix
$\boldsymbol{R}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Thruster force weight matrix
δ	10	Huber penalty slope
$\boldsymbol{\Sigma}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	Expansion ellipse weights
T	120s	OCP prediction horizon
N	60	OCP shooting intervals
d	3	OCP Legendre polynomial degree
K	8	Reduced constraint size

ACKNOWLEDGMENT

The authors would like to thank Emil Hjelseth Thyri for his help during the experiments.

REFERENCES

[1] E. Murdoch, C. Clarke, I. W. Dand, and B. Glover, *A Master's Guide to Berthing*. Livingston, U.K.: Witherby & Company, 2004.

[2] K. Hasegawa and T. Fukutomi, "On harbor maneuvering and neural control system for berthing with tug operation," in *Proc. 3rd Int. Conf. Manoeuvring Control Mar. Craft*, 1994, pp. 197–210.

[3] V. P. Bui, H. Kawai, Y. B. Kim, and K. S. Lee, "A ship berthing system design with four tug boats," *J. Mech. Sci. Technol.*, vol. 25, no. 5, pp. 1257–1264, May 2011.

[4] P. Van Bui and Y. B. Kim, "Development of constrained control allocation for ship berthing by using autonomous tugboats," *Int. J. Control, Autom. Syst.*, vol. 9, no. 6, pp. 1203–1208, Dec. 2011.

[5] V. L. Tran and N. Im, "A study on ship automatic berthing with assistance of auxiliary devices," *Int. J. Nav. Archit. Ocean Eng.*, vol. 4, no. 3, pp. 199–210, Sep. 2012.

[6] G. J. S. Rae, S. M. Smith, D. T. Anderson, and A. M. Shein, "A fuzzy rule based docking procedure for two moving autonomous underwater vehicles," in *Proc. Amer. Control Conf.*, Jun. 1993, pp. 580–584.

[7] K. Teo, B. Goh, and O. K. Chai, "Fuzzy docking guidance using augmented navigation system on an AUV," *IEEE J. Ocean. Eng.*, vol. 40, no. 2, pp. 349–361, Apr. 2015.

[8] V.-S. Nguyen and N.-K. Im, "Automatic ship berthing based on fuzzy logic," *Int. J. FUZZY Log. Intell. Syst.*, vol. 19, no. 3, pp. 163–171, Sep. 2019.

[9] H. Yamato, "Automatic berthing by the neural controller," in *Proc. 9th Ship Control Syst. Symp.*, vol. 3, 1990, pp. 3183–3201.

[10] K. Hasegawa, "Automatic berthing control system using network and knowledgebase," *J. Soc. Naval Architects Jpn.*, vol. 220, pp. 135–143, 1993.

[11] Y. Zhang, G. E. Hearn, and P. Sen, "A multivariable neural controller for automatic ship berthing," *IEEE Control Syst.*, vol. 17, no. 4, pp. 31–45, Aug. 1997.

[12] N. Im and K. Hasegawa, "A Srudy on automatic ship berthing using parallel neural controller," *J. Kansai Soc. Naval Architects, Jpn.*, vol. 2001, no. 236, pp. 65–70, 2001.

[13] Y. A. Ahmed and K. Hasegawa, "Automatic ship berthing using artificial neural network trained by consistent teaching data using nonlinear programming method," *Eng. Appl. Artif. Intell.*, vol. 26, no. 10, pp. 2287–2304, Nov. 2013.

[14] N.-K. Im and V.-S. Nguyen, "Artificial neural network controller for automatic ship berthing using head-up coordinate system," *Int. J. Nav. Archit. Ocean Eng.*, vol. 10, no. 3, pp. 235–249, May 2018.

[15] V.-S. Nguyen, V.-C. Do, and N.-K. Im, "Development of automatic ship berthing system using artificial neural network and distance measurement system," *Int. J. FUZZY Log. Intell. Syst.*, vol. 18, no. 1, pp. 41–49, Mar. 2018.

[16] D. Lee, S.-J. Lee, and S. Yu-Jeong, "Application of recent developments in deep learning to ann-based automatic berthing systems," *Int. J. Eng. Technol. Innov.*, vol. 10, no. 1, p. 75, 2019.

[17] N. Mizuno and R. Kuboshima, "Implementation and evaluation of nonlinear optimal feedback control for ship's automatic berthing by recurrent neural network," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 91–96, 2019.

[18] E.-L. H. Rørvik, "Automatic docking of an autonomous surface vessel," M.S. thesis, Dept. Eng. Cybern., Norwegian Univ. Sci. Technol. (NTNU), Trondheim, Norway, 2020. [Online]. Available: https://hdl.handle.net/11250/2656724

[19] H. Yamato, T. Koyama, and T. Nakagawa, "Automatic berthing using the expert system," *IFAC Proc. Volumes*, vol. 25, no. 3, pp. 173–184, Apr. 1992.

[20] M. Breivik and J.-E. Loberg, "A virtual target-based underway docking procedure for unmanned surface vehicles," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 13630–13635, Jan. 2011.

[21] J. Woo and N. Kim, "Vector field based guidance method for docking of an unmanned surface vehicle," in *Proc. 12th ISOPE Pacific/Asia Offshore Mech. Symp.* Mountain View, CA, USA: International Society of Offshore and Polar Engineers, 2016, pp. 1–6.

[22] A. Farnsworth, *Auto-docking ferry successfully tested in Norway*. Helsinki, Finland: Wärtsilä, 2018. [Online]. Available: https://www.wartsila.com/twentyfour7/innovation/look-ma-no-hands-auto-d ocking-ferry-successfully-tested-in-norway

[23] Wärtsilä, Helsinki, Finland. (2018). *Rolls-Royce and Finferries Demonstrate World's First Fully Autonomous Ferry*. Rolls-Royce Press Release. [Online]. Available: https://www.rolls-royce.com/media/press-releases/2018/03-12-2018-rr-and -finferries-demonstrate-worlds-first-fully-autonomous-ferry.aspx

[24] Kongsberg. (2020). *Automatic Ferry Enters Regular Service Following World-First Crossing With Passengers Onboard*. Kongsberg Press Release. [Online]. Available: https://www.kongsberg.com/maritime/about-us/news-and-media/news-archive /2020/first-adaptive-transit-on-bastofosen-vi/

[25] K. Shouji, K. Ohtsu, and S. Mizoguchi, "An automatic berthing study by optimal control techniques," *IFAC Proc. Volumes*, vol. 25, no. 3, pp. 185–194, Apr. 1992.

[26] K. Djouani and Y. Hamam, "Minimum time-energy trajectory planning for automatic ship berthing," *IEEE J. Ocean. Eng.*, vol. 20, no. 1, pp. 4–12, 1995.

[27] K. Ohtsu, K. Shoji, and T. Okazaki, "Minimum-time maneuvering of a ship, with wind disturbances," *Control Eng. Pract.*, vol. 4, no. 3, pp. 385–392, Mar. 1996.

[28] T. Okazaki, K. Ohtsu, and N. Mizuno, "A study of minimum time berthing solutions," *IFAC Proc. Volumes*, vol. 33, no. 21, pp. 135–139, Aug. 2000.

[29] N. Mizuno, Y. Uchida, and T. Okazaki, "Quasi real-time optimal control scheme for automatic berthing," *IFAC-PapersOnLine*, vol. 48, no. 16, pp. 305–312, 2015.

[30] M. C. Nielsen, T. A. Johansen, and M. Blanke, "Cooperative rendezvous and docking for underwater robots using model predictive control and dual decomposition," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 14–19.

[31] N. Mizuno, T. Kita, and T. Ishikawa, "A new solving method for nonlinear optimal control problem and its application to automatic berthing problem," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2018, pp. 2183–2188.

[32] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Autonomous docking using direct optimal control," *IFAC-PapersOnLine*, vol. 52, no. 21, pp. 97–102, 2019.

[33] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, "Trajectory planning and control for automatic docking of ASVs with full-scale experiments," 2020, *arXiv:2004.07793*. [Online]. Available: https://arxiv.org/abs/2004.07793

[34] S. Li, J. Liu, R. R. Negenborn, and Q. Wu, "Automatic docking for underactuated ships based on multi-objective nonlinear model predictive control," *IEEE Access*, vol. 8, pp. 70044–70057, 2020.

[35] T. Takai and K. Ohtsu, "Automatic berthing experiments using 'Shiojimaru,"' *The J. Jpn. Inst. Navigat.*, vol. 83, pp. 267–276, Mar. 1990.

[36] L. Gucma, A. Bak, M. Gucma, S. Jankowski, P. Zalewski, and M. Perkovic, "Laser docking system integrated with pilot navigation support system. Background to high precision, fast and reliable vessel docking," in *Proc. 17th Saint Petersburg Int. Conf. Integr. Navigat. Syst. (ICINS)*, 2010, pp. 268–275.
[37] M. Perkovic, M. Gucma, B. Luin, L. Gucma, and T. Brcko, "Accommodating larger container vessels using an integrated laser system for approach and berthing," *Microprocessors Microsyst.*, vol. 52, pp. 106–116, Jul. 2017.
[38] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Hoboken, NJ, USA: Wiley, 2011.
[39] T. A. Johansen and T. I. Fossen, "Control allocation—A survey," *Automatica*, vol. 49, no. 5, pp. 1087–1103, May 2013.
[40] S. Gros and M. Zanon, "Penalty functions for handling large deviation of quadrature states in NMPC," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3848–3860, Aug. 2017.
[41] S. Gros and M. Diehl, "NMPC based on huber penalty functions to handle large deviations of quadrature states," in *Proc. Amer. Control Conf.*, Jun. 2013, pp. 3159–3164.
[42] G. A. Hicks and W. H. Ray, "Approximation methods for optimal control synthesis," *Can. J. Chem. Eng.*, vol. 49, no. 4, pp. 522–528, Aug. 1971.
[43] P. Deuflhard, "A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting," *Numerische Math.*, vol. 22, no. 4, pp. 289–315, Aug. 1974.
[44] T. H. Tsang, D. M. Himmelblau, and T. F. Edgar, "Optimal control via collocation and non-linear programming," *Int. J. Control*, vol. 21, no. 5, pp. 763–768, May 1975.
[45] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*. Cham, Switzerland: Springer, 2015, pp. 109–124.
[46] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017.
[47] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, "CIAO*: MPC-based safe motion planning in predictable dynamic environments," 2020, *arXiv:2001.05449*. [Online]. Available: https://arxiv.org/abs/2001.05449
[48] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill, "An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments," 2020, *arXiv:2005.02674*. [Online]. Available: http://arxiv.org/abs/2005.02674
[49] J. Telgen, "Identifying redundant constraints and implicit equalities in systems of linear constraints," *Manage. Sci.*, vol. 29, no. 10, pp. 1209–1222, Oct. 1983.
[50] S. Paulraj and P. Sumathi, "A comparative study of redundant constraints identification methods in linear programming problems," *Math. Problems Eng.*, vol. 2010, Dec. 2010, Art. no. 723402.
[51] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996.
[52] T. R. Torben, A. H. Brodtkorb, and A. J. Sørensen, "Control allocation for double-ended ferries with full-scale experimental results," in *Proc. 12th IFAC CAMS*, Daejeon, South Korea, 2019, pp. 45–50.

ANDREAS B. MARTINSEN received the M.Sc. degree in engineering cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2018, where he is currently pursuing the Ph.D. degree in engineering cybernetics.

His research interests include reinforcement learning, optimal control, and machine learning, with a focus on marine and maritime applications.

GLENN BITAR received the B.S. degree in computer science and industrial automation from Telemark University College, Porsgrunn, Norway, in 2015, and the M.Sc. degree in cybernetics and robotics from the Department of Engineering Cybernetics, NTNU, Trondheim, Norway, in 2017, where he is currently pursuing the Ph.D. degree in researching energy-optimized autonomous motion control for ships.

ANASTASIOS M. LEKKAS received the M.Sc. degree in mechanical and aeronautical engineering from the University of Patras, Patras, Greece, in 2008, and the Ph.D. degree in engineering cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2014.

He is currently an Associate Professor of autonomous systems with the Department of Engineering Cybernetics, NTNU. He is the Project Manager of the EXAIGON project, where the main goal is to develop explainable AI methods for safety- and business-critical applications. His research interests include merging artificial intelligence with control engineering in order to develop cyber-physical systems with increased autonomy.

SÉBASTIEN GROS received the Ph.D. degree in mechatronics from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2007.

He was part of a Research and Development Group hosted at Strathclyde University, Glasgow, U.K., focusing on wind turbine control. In 2011, he joined the University of KU Leuven, Leuven, Belgium, where his main research focus was on optimal control and fast NMPC for complex mechanical systems. He joined the Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, in 2013, where he became an Associate Professor, in 2017. He is currently a Full Professor with the Norwegian University of Science and Technology, Norway, and an Affiliate Professor with Chalmers. His research interests include numerical methods, real-time optimal control, reinforcement learning, and the optimal control of energy-related applications.

Paper H Three-phase automatic crossing for a passenger ferry with field trials

Preprint by **G. Bitar**, B.-O. H. Eriksen, A. M. Lekkas, and M. Breivik. *Three-phase automatic crossing for a passenger ferry with field trials*. Submitted for publication.

Bibliography entry [15].

Three-Phase Automatic Crossing for a Passenger Ferry With Field Trials

Glenn Bitar, Bjørn-Olav H. Eriksen, Anastasios M. Lekkas and Morten Breivik

***Abstract*— We propose and demonstrate a method for automatic crossing with an autonomous surface vehicle by combining undocking, trajectory planning, transit, and docking capabilities. The proposed method divides the crossing operation into three phases and switches sequentially between previously developed fit-for-purpose planners and controllers. The docking and undocking phases are handled by a model-predictive local planner, while a two-stage energy-optimized global trajectory planner plans the transit phase. Both planners take into account vessel dynamics, and a trajectory-tracking controller tracks the resulting trajectories. We design and implement the phase switching using Unified Modeling Language state diagrams, facilitating architectures beyond what we have implemented in this paper. We demonstrate our method in successful full-scale experiments in the harbor area of Trondheim, Norway.**

I. Introduction

Several types of marine operations are transitioning to higher levels of autonomy. Among them are military surveillance operations using unmanned surface vehicles, mine reconnaissance and scientific surveys with autonomous underwater vehicles [1, 2], and transportation with autonomous surface vehicles (ASVs) [3]. Commercial cases of automatic crossing with ships were demonstrated by Wärtsilä and Rolls-Royce (later acquired by Kongsberg Maritime) in 2018, with limited scope.[1] Both tests included automatic transit and docking. The shipping company NYK performed automatic crossing trials in 2019,[2] utilizing a proprietary action planning system, which required manual verification by an operator before actions were passed to the ship's control system [3].

For automatic crossing operations, a vessel must consider collision avoidance with static and moving obstacles, and during transit, a real-world implementation should comply with the International Regulations for Preventing Collisions at Sea (COLREGs) [4]. Several methods for collision avoidance at sea are available in the literature. There are local, short-term methods such as dynamic window (DW), velocity obstacles (VOs), and the branching-course model-predictive controller (BC-MPC) [5, 6, 7, 8], and global methods, often employing graph-search or optimization-based techniques [9, 10]. Some of these (often partially) take into account the COLREGs, a set of regulations specified by the International Maritime Organization, also called the maritime "rules of the road." Rules 8 and 13–17 of the COLREGs deal with how the maneuvering should be performed generally and in cases of overtaking, head-on, and crossing situations. Studies of COLREGs-compliant collision avoidance methods include [11, 12].

Simplified, we can divide automatic crossing with manned and unmanned surface vessels into three phases:

Undocking: Moving at slow speeds from a docked position along a quay to freer waters.

Transit: Moving at higher speeds towards the destination.

Docking: Moving at slow speeds at the destination to a docked position along a quay.

One approach for handling automatic crossing is to use a single planner and controller for all three phases, thus treating the operation as a single process. Some motion planners can handle both confined and open waters, which make them suitable for the single-process approach [13, 14]. The downside to this approach is the lack of flexibility and modularity. For instance, the transit phase may involve high speeds and may require that the vessel abides by the COLREGs, while the docking and undocking phases occur at low speed and may have tighter tolerances for trajectory tracking. Additionally, the aspects mentioned above of the COLREGs can often be ignored during docking due to the low speeds, absence of other vessels, and short distances involved. Using different planning and controller modules for the three phases is more flexible but requires some architecture to switch between the active modules.

Unified Modeling Language state diagrams (UML-SDs) [15] are a suitable tool for designing and implementing an automatic switching architecture. Using UML-SDs, we can explicitly program the desired behavior during automatic crossing. The UML-SD is an open standard, and free, open-source software is available for implementing these diagrams. UML-SDs allow for hierarchical state machines and support concurrency. Explicitly programming behavior for the automatic architecture requires the design to consider all possible events that may occur during operation or have measures in place that ensure safe operation when encountering unexpected situations. Since accounting for all possible events is impossible, this type of architecture is best suited for manned or supervised applications of automatic crossing so that an operator can take over control if a situation emerges.

This paper presents an architecture for automatic crossing for ASVs that uses two different types of trajectory planners

This work was supported in part by the Research Council of Norway through project number 269116, as well as through the Centres of Excellence funding scheme with project number 223254.

The authors are with the Centre for Autonomous Marine Operations and Systems and the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway. Emails: glennbitar@outlook.com, bjorn-olav.h.eriksen@ieee.org, anastasios.lekkas@ntnu.no, morten.breivik@ieee.org.

[1] https://www.maritime-executive.com/article/rolls-royce-and-wartsila-in-close-race-with-autonomous-ferries (accessed September 14, 2020).

[2] https://www.maritime-executive.com/article/nyk-conducts-live-autonomous-navigation-test-with-pctc (accessed October 12, 2020).

Fig. 1: The experimental autonomous urban passenger ferry *milliAmpere*.

TABLE I: *milliAmpere* specifications.

Dimensions	5 m by 2.8 m rectangular footprint
Position and heading reference system	Vector VS330 dual GNSS with RTK capabilities
Thrusters	Two azimuth thrusters on the center line, 1.8 m aft and fore of center
Existing control modules	Docking planner, transit planner, trajectory-tracking DP controller, and a thrust allocation system

for the docking, transit, and docking phases. The architecture is designed with a UML-SD and implemented using a state machine library on the experimental, autonomous ferry *milliAmpere*, developed at the Norwegian University of Science and Technology. The architecture is tested and shown to work well in a full-scale experiment. We discuss where such an explicit approach would fail and propose alternative approaches that may be more suitable in unsupervised, autonomous implementations. In summary, our contributions are as follows:

- We develop an architecture for automatic dock-to-dock transportation for ASVs that is designed with a UML-SD.
- We demonstrate that this architecture successfully can achieve its goal via a full-scale experiment.
- We discuss scenarios where explicit programming would fail and suggest alternative approaches to come closer to autonomous operations.

The rest of our paper is structured as follows: We present our experimental platform and existing control and planning modules in Section II. The automatic crossing architecture is presented in Section III. Results from our validation experiment are presented in Section IV. Section V contains a discussion about our automatic crossing approach and suggestions for alternative methods that can achieve higher degrees of autonomy. In Section VI, we conclude the paper and present some ideas on how we would like to extend our method.

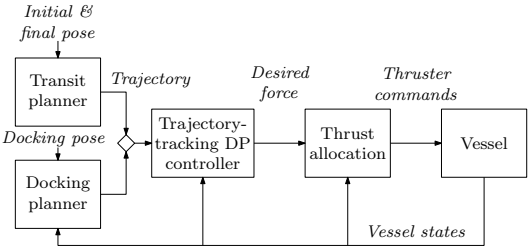


Fig. 2: Control architecture for *milliAmpere*.

II. Experimental platform: milliAmpere

We performed our experiments using the experimental autonomous urban passenger ferry *milliAmpere*, developed at the Norwegian University of Science and Technology (NTNU). Figure 1 shows a picture of *milliAmpere*, and its specifications are listed in Table I. The platform is under continuous development and many students and researchers have contributed with control systems as well as hardware solutions. A larger version is currently being designed and built by the research group Autoferry.[3]

The *milliAmpere* has an existing dynamic positioning (DP) controller capable of tracking trajectories, a thrust allocation algorithm providing thruster commands based on the desired control force, and trajectory planners for both docking and transit operations. Depending on the active phase (undocking, transit, or docking), one of the planners feeds its trajectory to the trajectory-tracking DP controller, which again feeds desired force commands to the thrust allocation algorithm [16]. The control architecture is illustrated in Figure 2.

A. Docking planner

The docking planner was first presented in [17], tested experimentally with global navigation satellite system (GNSS) measurements in [18] and further extended with exteroceptive sensors in [19]. It consists of an optimal control problem (OCP) solver that provides local collision-free trajectories based on static map data as well as obstacles generated by a lidar sensor. The trajectory is replanned periodically to account for errors, and converges on a target docking pose.

B. Transit planner

The transit trajectory planner is a global planner based on a discrete hybrid A* search as well as an OCP solver [10]. It takes into account static obstacles in the form of a map with polygons. It uses the hybrid A* search to find a close-to-optimal dynamically feasible which is subsequently used to generate a sequence of convex obstacle-avoidance constraints and an initial guess to warm-start the OCP solver. The objective function of the OCP solver is minimum energy usage.

The transit planner only takes into account a known and static map of obstacles. Hence, moving and unknown obstacles are not handled when the transit mode is active.

[3]Autoferry website: `https://www.ntnu.edu/autoferry`.

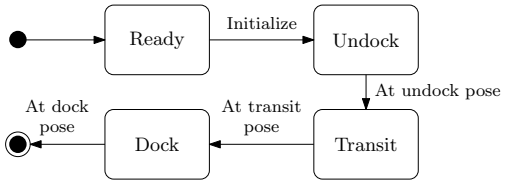


Fig. 3: A UML-SD that describes the simplest possible automatic crossing architecture. There is no error handling or planning in this architecture.

This is of course not a sufficient approach for a practical application, but provides a proof of concept for the sequential control approach in the scope of this paper. One way to handle both static and moving obstacles is to use a three-level hierarchical, or hybrid, collision-avoidance approach described and developed in, e.g., [11, 20, 21], together with exteroceptive sensors such as radar and lidar, and tracking algorithms. With such a hybrid collision-avoidance approach, the trajectory planner would provide a nominal trajectory which would be modified if required to generate safe maneuvers. The trajectory planner described here fits directly into the top-level layer of the architecture described in [11].

III. Automatic crossing architecture

The simplest possible architecture for a three-phase crossing operation is depicted in the UML-SD in Figure 3. In this UML-SD, we assume that the transit trajectory is pre-planned and collision-free. However, in our implementation, the trajectory is planned after initialization, so we require concurrence. For the undocking and docking phases, we use the docking planner described in Section II-A. For safety, we add a state for "safe mode," which is activated by the operator in case of unexpected situations. In addition, for our experiments, we did not have access to two quays far apart, so we add a secondary transit state to emulate a longer transit journey. This also required a reorientation state, to align the vessel with the secondary trajectory. The result is the automatic crossing architecture described by the UML-SD in Figure 4. A description of each state in the UML-SD is shown in Table II. The transition criteria are that the vessel's pose is measured at the planned values while standing still, within a specified tolerance.

IV. Experimental results

We performed the experiments using *milliAmpere* and an implementation of the UML-SD from Figure 4 implemented in ROS and Python with a state-machine library called Smach.[4] The experiments were done near the Brattøra harbor area in Trondheim, Norway on September 30th 2020, with calm weather conditions and little traffic. A map of the area where we performed experiments is provided in Figure 5. The map also shows the different operation poses, which are described by $\boldsymbol{\eta}_{(\cdot)} \in \mathbb{R}^2 \times S$. These poses consist of coordinates North and East of the origin in an Earth-tangential plane,

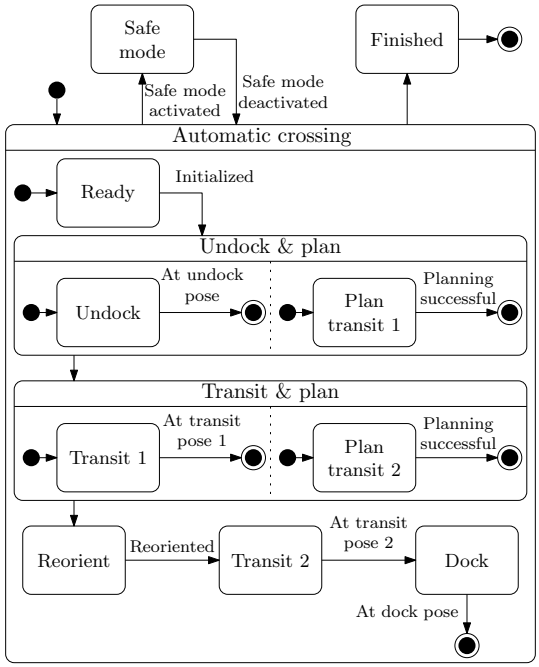


Fig. 4: The UML-SD that describes the automatic crossing architecture implemented in this paper. The diagram uses hierarchical state containers to generalize safe mode state switching from every substate, and orthogonal regions to enable concurrency when planning.

and the vessel's heading angle. The operation starts at $\boldsymbol{\eta}_0$, undocking towards $\boldsymbol{\eta}_u$. The first transit is between $\boldsymbol{\eta}_u$ and $\boldsymbol{\eta}_{t1}$, where the vessel reorients itself to $\boldsymbol{\eta}_r$ before transiting back to $\boldsymbol{\eta}_{t2}$. Finally, the vessel docks at $\boldsymbol{\eta}_d$. The pose descriptions are found in Table III. Each transit leg is approximately $520\,\mathrm{m}$ in length.

Figure 6 shows the pose trajectories of the undocking phase, the two transit phases, and the docking phase. The pose trajectories show that *milliAmpere* manages to undock from the start position to the appointed undock pose $\boldsymbol{\eta}_u$. The first transit phase then starts, where the vessel maneuvers from within the harbor area, to just outside it, at $\boldsymbol{\eta}_{t1}$. After reorienting to $\boldsymbol{\eta}_r$, *milliAmpere* continues to transit towards $\boldsymbol{\eta}_{t2}$, before successfully docking with its back towards the quay at $\boldsymbol{\eta}_d$.

Figure 7 shows the full state trajectories, including pose and velocity, as well as indications at the times t of state switches. We see jumps in velocity at, e.g., $t \approx 170\,\mathrm{s}$ and $t \approx 660\,\mathrm{s}$, which are caused by jumps in GNSS measurements.

[4]Robot Operating System (ROS) is a software framework for robots that enables inter-process messaging, services and data logging. Client libraries exist for the C++, Python, and Lisp programming languages. Smach is an independent but affiliated Python library for building hierarchical state machines. See `https://wiki.ros.org/` and `https://wiki.ros.org/smach` for more information.

TABLE II: Description of states and actions.

State	Actions
Ready	Loitering, ready for crossing
Undock	Use docking planner to move to undock pose
Plan transit 1	Use global transit planner to plan a trajectory from the undock pose to the first transit pose
Transit 1	Use trajectory-tracking DP controller to track the planned trajectory towards the first transit pose
Plan transit 2	Use global transit planner to plan a trajectory from the reorientation pose to the secondary transit pose
Reorient	Use DP controller to change vessel heading to align to the second trajectory
Transit 2	Use trajectory-tracking DP controller to track the planned trajectory towards the secondary transit pose
Dock	Use docking planner to move to docking pose
Safe mode	Set vessel in manual control mode
Finished	Crossing completed, loitering

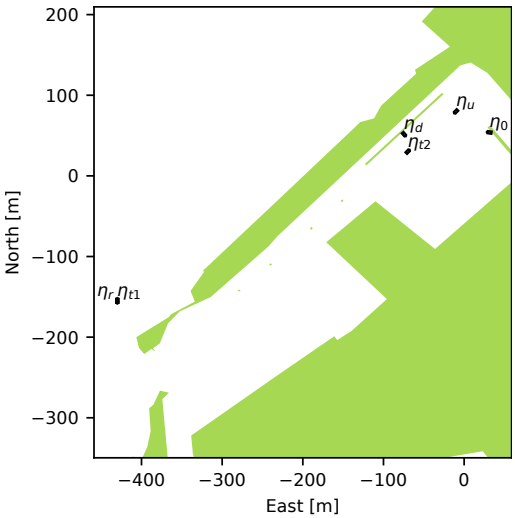


Fig. 5: Map of the experiment area with markers at key positions. The positions are listed in Table III.

The velocity estimates are naively calculated using finite differencing between positional estimates, so these jumps cause large velocity estimation errors.

The experiment shows that this approach to automatic transportation operations works well. Obviously, practical applications require that complex scenarios are accounted for, such as fault handling, collision avoidance, and replanning. For automatic or autonomous operations that are supervised by humans, our approach is sufficiently flexible and simple. In Section V we discuss the limitations of our approach and possible solutions.

V. Limitations of the Automatic Crossing Architecture

In the previous section we showed a proof of concept for an automatic crossing architecture with experimental validation. The experiment showed that the proof of concept works for a simple automatic operation. In this section we discuss some of the limitations and highlight two issues.

TABLE III: Poses involved in the experiment.

Name	Symbol	Value $(\mathrm{m}, \mathrm{m}, ^\circ)$
Start	$\boldsymbol{\eta}_0$	$(24, 31, -80)$
Undock	$\boldsymbol{\eta}_u$	$(80, -10, -135)$
Transit 1	$\boldsymbol{\eta}_{t1}$	$(-155, -430, 0)$
Reorient	$\boldsymbol{\eta}_r$	$(-155, -430, 180)$
Transit 2	$\boldsymbol{\eta}_{t2}$	$(30, -70, 45)$
Dock	$\boldsymbol{\eta}_d$	$(51.71, -74.60, 138.7)$

A. Zero-velocity switching condition

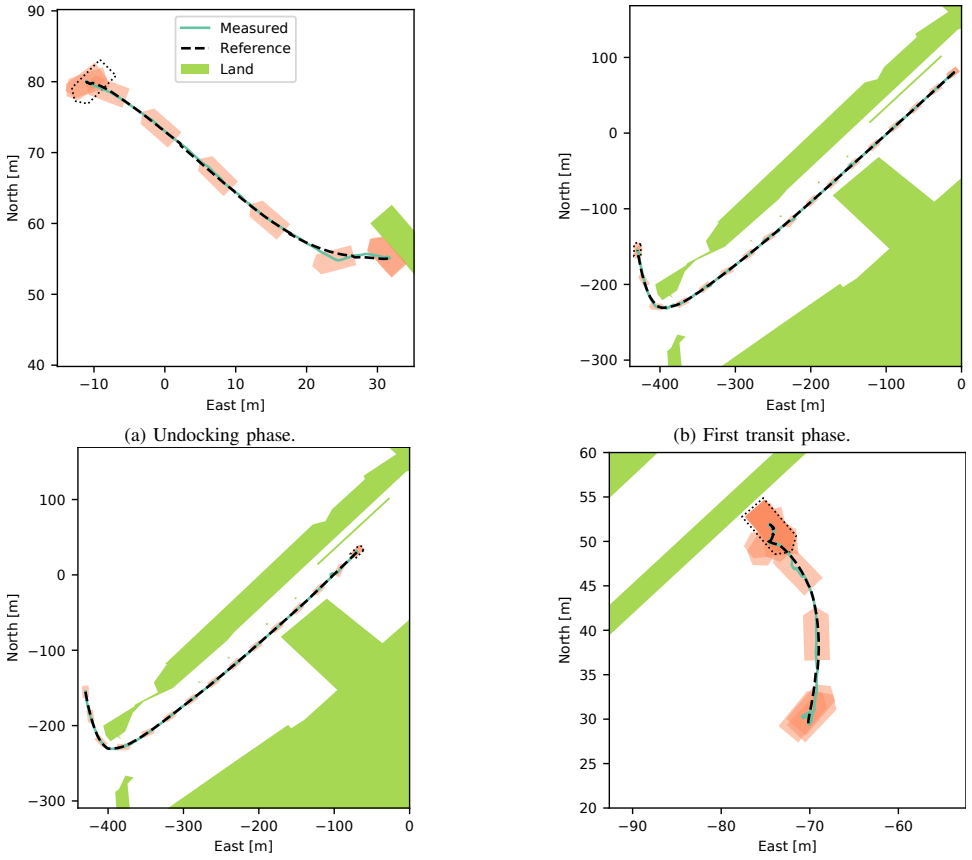
The trajectory planners produce trajectories that arrive at the destination pose with zero velocity. This makes it possible to design switching conditions for the UML-SD that are trivial to reach. The underlying tracking controller can easily converge to the desired poses and slow down to zero velocity, within the provided tolerances. However, this might be too strict a requirement, since it might be unnecessary and unnatural to stop between these phases—a human operator would certainly not make these intermediate stops unless an unexpected situation emerged. Tighter integration between the planning modules would be necessary to achieve switching with non-zero velocity, e.g., by planning an undocking trajectory that ended with the starting velocity of the transit trajectory. This takes away some of the simplicity provided by the modular approach, but is still achievable with the right interface between the modules.

B. Explicit programming of automatic behavior

Using UML-SDs or other types of explicit programming to produce automatic behavior is a simple solution that would work in many cases, especially manned, automatic operations. However, for unsupervised operations, these approaches might not be sufficient. Explicit programming implies that a supervisory agent handles unforeseen incidents, e.g., by a safe mode, as implemented in Figure 4. Otherwise, the design would have to take into account all types of unforeseen incidents, which is impossible. It is possible to take into account the specific incidents we can foresee, by explicitly handling them in the UML-SD. An example of this could be that the undocking pose is occupied by another vessel, making the docking planner unable to reach it. The ASV would in this case loiter while waiting for the space to clear, which would be an undesired behavior. In this case, if the transit planner has successfully finished, a module could provide an intermediate trajectory between the current vessel state and a state along the transit trajectory. A proposed change to our UML-SD that would handle this specific scenario is provided in Figure 8.

There are however many other incidents to take into account, e.g.:

- Blocked switching poses
- Sensor failures
- Actuator failures
- Trajectory planners unable to find solutions



(a) Undocking phase.

(b) First transit phase.

(c) Second transit phase.

(d) Docking phase.

Fig. 6: Maps and trajectories during the main phases of our experiment.

Other methodologies that enable increased autonomy are available in the scientific literature. This includes T-REX from the Monterey Bay Aquarium Research Institute [2], HAL from the Norwegian Defence Research Establishment [22], CARACaS by the US Office of Naval Research [23], GenoM by Laboratory for Analysis and Architecture of Systems [24], and MOOS-IvP from the Massachusetts Institute of Technology [25]. These modules have advanced capabilities such as replanning, negotiating competing goals, and deliberative task-level planning, which would take us closer to unsupervised, autonomous operation.

VI. Conclusion

Crossing with a surface vessel is a process that can be divided into three phases: undocking, transit, and docking. Automatically performing this process requires a control system that can handle all three phases safely. We suggest a modular approach that handles these phases with different fit-for-purpose planning and control modules. We have designed such a control architecture with a UML-SD, and show that it is a simple and effective way to achieve automatic crossing. The experimental results demonstrate that our implementation successfully achieves automatic crossing between two quays. This is done by combining two planning and control modules and by conditionally switching between them. We have also shown that the UML-SD approach is flexible and general enough to include complex situations, but we emphasize that explicit programming cannot take into account unforeseen incidents. For unsupervised autonomous systems, we suggest alternative approaches based on autonomy frameworks.

We wish to expand our architecture to handle moving obstacles and the COLREGs. This expansion can be achieved by integrating the transit control module with a hybrid collision-avoidance system, proposed in, e.g., [11, 20, 21]. Additionally, our implementation's only safety-action is when the operator activates a safety condition, which places the vessel in manual control mode. We want to implement automatic failure detection with flexible safety modes to improve upon this limitation. E.g., if a failure is detected

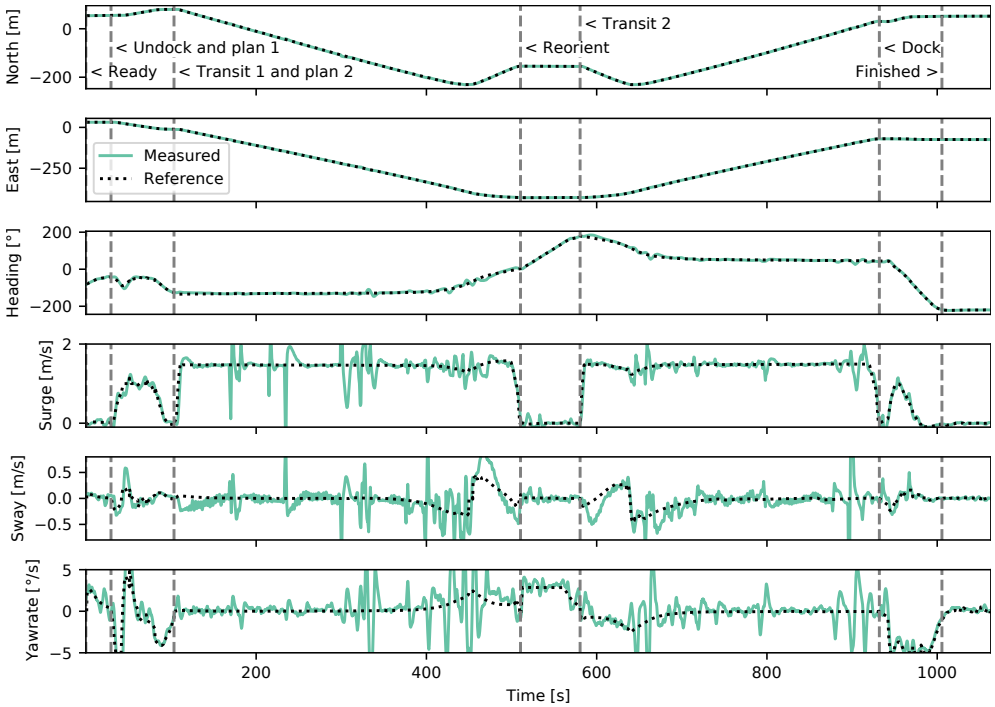


Fig. 7: Complete state trajectory during our experiment. The vertical lines signify the sequential phase changes.

(problems with exteroceptive sensors such as the lidar, for instance), the vessel could initiate loitering and restart the interrupted procedure once the problem is fixed.

Acknowledgment

The authors thank Simen K. Knudsen for his help with the experiments.

References

[1] M. S. Wiig, "Decisional autonomy for the HUGIN autonomous underwater vehicle," in *Proc. SCI-202 Symposium on Intelligent Uninhabited Vehicle Guidance Systems*, 2009.

[2] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, "A deliberative architecture for AUV control," in *Proc. 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA*, 2008.

[3] K. Kutsuna, H. Ando, T. Nakashima, S. Kuwahara, and S. Nakamura, "NYK's approach for autonomous navigation – structure of action planning system and demonstration experiments," *Journal of Physics: Conference Series*, vol. 1357, 2019.

[4] A. N. Cockroft and J. N. F. Lameijer, *A Guide to the Collision Avoidance Rules*, 7th ed. Butterworth-Heinemann, 2011.

[5] B.-O. H. Eriksen, E. F. Wilthil, A. L. Flaten, E. F. Brekke, and M. Breivik, "Radar-based maritime collision avoidance using dynamic window," in *2018 IEEE Aerospace Conference*, 2018.

[6] D. K. M. Kufoalor, E. F. Brekke, and T. A. Johansen, "Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain*, 2018.

[7] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2014.

[8] B.-O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke, "The branching-course model predictive control algorithm for maritime collision avoidance," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1222–1249, 2019.

[9] 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.

[10] 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.

[11] B.-O. H. Eriksen, G. Bitar, M. Breivik, and A. M.

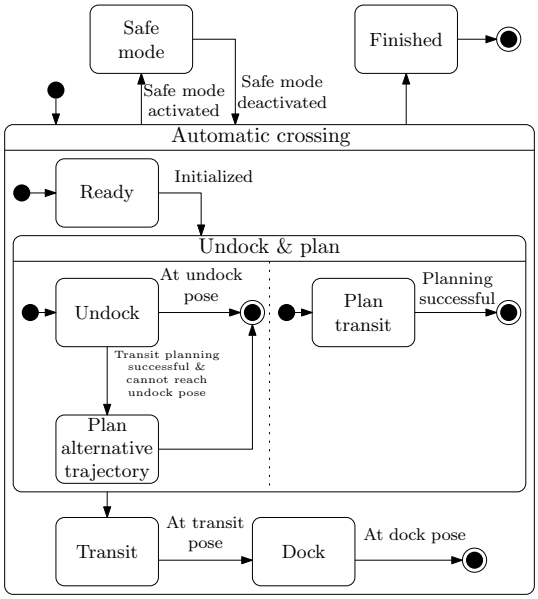


Fig. 8: Alternative UML-SD that plans an alternative trajectory to connect the current vessel state with a point at the transit trajectory if the undock state cannot reach the undock pose.

Lekkas, "Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17," *Frontiers in Robotics and AI*, vol. 7, pp. 1–11, 2020.

[12] K. L. Woerner, "COLREGS-compliant autonomous collision avoidance using multi-objective optimization with interval programming," Master's thesis, Massachusetts Institute of Technology, 2014. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA609415

[13] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill, "An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments," 2020, submitted for publication. arXiv: 2005.02674 [math.OC].

[14] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Optimal model-based trajectory planning with static polygonal constraints," 2020, submitted for publication. arXiv: 2010.14428 [eess.SY].

[15] Object Management Group, *Unified Modeling Language*, 2017, version 2.5.1. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF

[16] T. R. Torben, A. H. Brodtkorb, and A. J. Sørensen, "Control allocation for double-ended ferries with full-scale experimental results," in *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS), Daejeon, South Korea*, 2019.

[17] A. B. Martinsen, A. M. Lekkas, and S. Gros, "Autonomous docking using direct optimal control," in *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS), Daejeon, South Korea*, 2019.

[18] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, "Trajectory planning and control for automatic docking of ASVs with full-scale experiments," in *Proceedings of the 1st Virtual IFAC World Congress*, 2020.

[19] A. B. Martinsen, G. Bitar, A. M. Lekkas, and S. Gros, "Trajectory planning for automatic docking and berthing of ASVs using exteroceptive sensors: theory and experiments," *IEEE Access*, 2020.

[20] G. Casalino, A. Turetta, and E. Simetti, "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields," in *Proceedings of the IEEE Oceans Conference, Bremen, Germany*, 2009, pp. 1–8.

[21] P. Švec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. von Ellenrieder, M. Dhanak, and S. K. Gupta, "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan*, 2013, pp. 3871–3878.

[22] M. S. Wiig, R. A. Løvlid, K. Mathiassen, and T. R. Krogstad, "Decision autonomy for unmanned vehicles," in *Proc. 2016 Information Systems Technology Panel*, ser. 127. NATO Science and Technology Organization, 2016.

[23] T. Huntsberger, H. Aghazarian, A. Castano, G. Woodward, C. Padgett, D. Gaines, and C. Buzzel, "Intelligent autonomy for unmanned sea surface and underwater vehicles," in *Proceedings of the AUVSI North America Conference, San Diego, CA, USA*, 2008.

[24] S. Fleury, M. Herrb, and R. Chatila, "GENOM: A tool for the specification and the implementation of operating modules in a distributed robot architecture," in *Proc. 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems, Grenoble, France*, 1997.

[25] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.

Bibliography

[1] On-Road Automated Driving Committee. *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. SAE Standard J3016 201806. SAE International, 2018. URL: https://www.sae.org/standards/content/j3016_201806/.

[2] Ø. J. Rødseth and M. Vagia. "A taxonomy for autonomy in industrial autonomous mobile robots including autonomous merchant ships". In: *Proceedings of the 3rd International Conference on Maritime Autonomous Surface Ship (ICMASS)*. Ulsan, South Korea, 2020. DOI: 10.1088/1757-899x/929/1/012003.

[3] C. Chauvin. "Human factors and maritime safety". In: *Journal of Navigation* 64.4 (2011), pp. 625–632. DOI: 10.1017/s0373463311000142.

[4] O. Levander. "Autonomous ships on the high seas". In: *IEEE Spectrum* 54.2 (2017), pp. 26–31. DOI: 10.1109/mspec.2017.7833502.

[5] A. N. Cockroft and J. N. F. Lameijer. *A guide to the collision avoidance rules*. 7th ed. Butterworth-Heinemann, 2011. 200 pp. ISBN: 9780080971704.

[6] DNV GL. *Sammenstilling av grunnlagsdata om dagens skipstrafikk og drivstofforbruk. Miljøtiltak for maritim sektor*. Norwegian. Tech. rep. 2014-1667. Klima- og miljødepartementet, 2014.

[7] G. Bitar, A. M. Lekkas, and M. Breivik. *Improvements to warm-started optimized trajectory planning for ASVs*. arXiv: 1908.07311 [eess.SY].

[8] G. Bitar, M. Breivik, and A. M. Lekkas. "Energy-optimized path planning for autonomous ferries". In: *Proceedings of the 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. Opatija, Croatia, 2018, pp. 389–394. DOI: 10.1016/j.ifacol.2018.09.456.

[9] G. Bitar, B.-O. H. Eriksen, A. M. Lekkas, and M. Breivik. "Energy-optimized hybrid collision avoidance for ASVs". In: *Proceedings of the 18th European Control Conference (ECC)*. Naples, Italy, 2019, pp. 2522–2529. DOI: 10.23919/ECC.2019.8795645.

[10] G. Bitar, V. N. Vestad, A. M. Lekkas, and M. Breivik. "Warm-started optimized trajectory planning for ASVs". In: *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. Daejeon, South Korea, 2019, pp. 308–314. DOI: 10.1016/j.ifacol.2019.12.325. arXiv: 1907.02696 [eess.SY].

[11] B.-O. H. Eriksen, G. Bitar, M. Breivik, and A. M. Lekkas. "Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17". In: *Frontiers in Robotics and AI* 7 (2020). DOI: 10.3389/frobt.2020.00011. arXiv: 1907.00198 [eess.SY].

[12] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik. "Trajectory planning and control for automatic docking of ASVs with full-scale experiments". In: *Proceedings of the 1st Virtual IFAC World Congress*. 2020. arXiv: 2004.07793 [eess.SY].

[13] 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". In: *IEEE Access* 8 (2020), pp. 199953–199969. DOI: 10.1109/ACCESS.2020.3035256.

[14] A. B. Martinsen, G. Bitar, A. M. Lekkas, and S. Gros. "Optimization-based automatic docking and berthing of ASVs using exteroceptive sensors: theory and experiments". In: *IEEE Access* 8 (2020), pp. 204974–204986. DOI: 10.1109/ACCESS.2020.3037171.

[15] G. Bitar, B.-O. H. Eriksen, A. M. Lekkas, and M. Breivik. *Three-phase automatic crossing for a passenger ferry with field trials*. Submitted for publication.

[16] A. A. Pedersen. "Optimization based system identification for the milliAmpere ferry". MA thesis. Norwegian University of Science and Technology, 2019. URL: http://hdl.handle.net/11250/2625699.

[17] V. N. Vestad. "Automatic and practical route planning for ships". MA thesis. Norwegian University of Science and Technology, 2019. URL: http://hdl.handle.net/11250/2625697.

[18] E. D. Molven. "Optimal control-based docking for autonomous ferries". MA thesis. Norwegian University of Science and Technology, 2020.

[19] T. R. Torben, A. H. Brodtkorb, and A. J. Sørensen. "Control allocation for double-ended ferries with full-scale experimental results". In: *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)*. Daejeon, South Korea, 2019.

[20] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. Wiley-Blackwell, 2011. ISBN: 978-1-119-99149-6. DOI: 10.1002/9781119994138.

[21] E. Morelli and V. Klein. *Aircraft system identification: theory and practice*. 2nd ed. Sunflyte Enterprises, 2016. ISBN: 9780997430615.

[22] F. Topputo and C. Zhang. "Survey of direct transcription for low-thrust space trajectory optimization with applications". In: *Abstract and Applied Analysis* (2014), pp. 1–15. DOI: 10.1155/2014/851720.

[23] S. M. LaValle. "Motion planning. Part I: The Essentials". In: *IEEE Robotics & Automation Magazine* 18 (1 2011), pp. 79–89. DOI: 10.1109/MRA.2011.940276.

[24] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006. 844 pp. ISBN: 978-0521862059. URL: http://www.worldcat.org/title/planning-algorithms/oclc/905273351.

[25] P. Hart, N. Nilsson, and B. Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[26] M. Candeloro, A. M. Lekkas, and A. J. Sørensen. "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels". In: *Control Engineering Practice* 61 (2017), pp. 41–54. DOI: 10.1016/j.conengprac.2017.01.007.

[27] P. Bhattacharya and M. L. Gavrilova. "Voronoi diagram in optimal path planning". In: *Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*. Glamorgan, UK, 2007. DOI: 10.1109/isvd.2007.43.

[28] P. Bhattacharya and M. Gavrilova. "Roadmap-based path planning - using the Voronoi diagram for a clearance-based shortest path". In: *IEEE Robotics & Automation Magazine* 15.2 (2008), pp. 58–66. DOI: 10.1109/mra.2008.921540.

[29] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1996. ISBN: 9780262133166.

[30] Y. Hu and S. X. Yang. "A knowledge based genetic algorithm for path planning of a mobile robot". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. New Orleans, LA, USA, 2004. DOI: 10.1109/robot.2004.1302402.

[31] C. Lamini, S. Benhlima, and A. Elbekri. "Genetic algorithm based approach for autonomous mobile robot path planning". In: *Procedia Computer Science* 127 (2018), pp. 180–189. DOI: 10.1016/j.procs.2018.01.113.

[32] J. A. Sethian. "A fast marching level set method for monotonically advancing fronts". In: *Proceedings of the National Academy of Sciences* 93.4 (1996), pp. 1591–1595. DOI: 10.1073/pnas.93.4.1591.

[33] S. Garrido, D. Alvarez, and L. E. Moreno. "Marine applications of the fast marching method". In: *Frontiers in Robotics and AI* 7 (2020), pp. 1–12. DOI: 10.3389/frobt.2020.00002.

[34] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580. DOI: 10.1109/70.508439.

[35] S. M. LaValle. *Rapidly-exploring random trees: a new tool for path plannning*. Tech. rep. Iowa State University, 1998. URL: http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf.

[36] R. Geraerts and M. H. Overmars. "A comparative study of probabilistic roadmap planners". In: *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, 2004, pp. 43–57. DOI: 10.1007/978-3-540-45058-0_4.

[37] S. M. LaValle and J. Kuffner Jr. *Rapidly-exploring random trees: progress and prospects*. 2000. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.1387.

[38] L. E. Dubins. "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents". In: *American Journal of Mathematics* 79.3 (1957), p. 497. DOI: 10.2307/2372560.

[39] J. Reeds and L. Shepp. "Optimal paths for a car that goes both forwards and backwards". In: *Pacific Journal of Mathematics* 145.2 (1990), pp. 367–393. DOI: 10.2140/pjm.1990.145.367.

[40] Q. Gong, R. Lewis, and I. M. Ross. "Pseudospectral motion planning for autonomous vehicles". In: *Journal of Guidance, Control, and Dynamics* 32.3 (2009), pp. 1039–1045. DOI: 10.2514/1.39697.

[41] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli. "Autonomous parking using optimization-based collision avoidance". In: *Proceedings of the 57th IEEE Conference on Decision and Control (CDC)*. Miami Beach, FL, USA, 2018, pp. 4327–4332. DOI: 10.1109/cdc.2018.8619433.

[42] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill. *An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments*. Submitted for publication. 2020. arXiv: 2005.02674v1 [math.OC].

[43] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. St. Louis, MO, USA, 1985. DOI: 10.1109/robot.1985.1087247.

[44] Y. Koren and J. Borenstein. "Potential field methods and their inherent limitations for mobile robot navigation". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Sacramento, CA, USA, 1991. DOI: 10.1109/robot.1991.131810.

[45] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. *Practical search techniques in path planning for autonomous driving*. Tech. rep. American Association for Artificial Intelligence, 2008. URL: https://www.aaai.org/Library/Workshops/2008/ws08-10-006.php.

[46] Y. Meng, Y. Wu, Q. Gu, and L. Liu. "A decoupled trajectory planning framework based on the integration of lattice searching and convex optimization". In: *IEEE Access* 7 (2019), pp. 130530–130551. DOI: 10.1109/access.2019.2940271.

[47] D. Fox, W. Burgard, and S. Thrun. "The dynamic window approach to collision avoidance". In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33. DOI: 10.1109/100.580977.

[48] B.-O. H. Eriksen, E. F. Wilthil, A. L. Flåten, E. F. Brekke, and M. Breivik. "Radar-based maritime collision avoidance using dynamic window". In: *Proceedings of the IEEE Aerospace Conference*. Big Sky, MT, USA, 2018. DOI: 10.1109/aero.2018.8396666.

[49] P. Fiorini and Z. Shiller. "Motion planning in dynamic environments using velocity obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772. DOI: 10.1177/027836499801700706.

[50] B.-O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke. "The branching-course model predictive control algorithm for maritime collision avoidance". In: *Journal of Field Robotics* 36.7 (2019), pp. 1222–1249. DOI: 10.1002/rob.21900.

[51] B.-O. H. Eriksen and M. Breivik. "Short-term ASV collision avoidance with static and moving obstacles". In: *Modeling, Identification and Control* 40.3 (2019), pp. 177–187. DOI: 10.4173/mic.2019.3.4.

[52] T. A. Johansen, T. Perez, and A. Cristofaro. "Ship collision avoidance and COLREGS compliance using simulation-based control behavior selection with predictive hazard assessment". In: *IEEE Transactions on Intelligent Transportation Systems* 17.12 (2016), pp. 3407–3422. DOI: 10.1109/tits.2016.2551780.

[53] 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". In: *Journal of Field Robotics* 37.3 (2020), pp. 387–403. DOI: 10.1002/rob.21919.

[54] B.-O. H. Eriksen. "Collision avoidance and motion control for autonomous surface vehicles". PhD thesis. Norwegian University of Science and Technology, 2019. ISBN: 978-82-326-4056-0. URL: http://hdl.handle.net/11250/2616394.

[55] Ø. A. G. Loe. "Collision avoidance for unmanned surface vehicles". MA thesis. Norwegian University of Science and Technology, 2008. URL: http://hdl.handle.net/11250/259696.

[56] P. Švec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. von Ellenrieder, M. Dhanak, and S. K. Gupta. "Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan, 2013, pp. 3871–3878. DOI: 10.1109/iros.2013.6696910.

[57] G. Casalino, A. Turetta, and E. Simetti. "A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields". In: *Proceedings of the IEEE OCEANS Conference*. Bremen, Germany, 2009. DOI: 10.1109/OCEANSE.2009.5278104.

[58] B.-O. H. Eriksen and M. Breivik. "MPC-based mid-level collision avoidance for ASVs using nonlinear programming". In: *Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*. Mauna Lani, HI, USA, 2017, pp. 766–772. DOI: 10.1109/ccta.2017.8062554.

[59] E. Murdoch, C. Clarke, and I. W. Dand. *A master's guide to: berthing, 2nd edition*. Ed. by C. Spencer. The Standard, 2012.

[60] G. J. S. Rae and S. M. Smith. “A fuzzy rule based docking procedure for autonomous underwater vehicles”. In: *Proceedings of the IEEE OCEANS Conference*. Newport, RI, USA, 1992. DOI: 10.1109/oceans.1992.607638.

[61] K. Teo, B. Goh, and O. K. Chai. “Fuzzy docking guidance using augmented navigation system on an AUV”. In: *IEEE Journal of Oceanic Engineering* 40.2 (2015), pp. 349–361. DOI: 10.1109/joe.2014.2312593.

[62] Y.-H. Hong, J.-Y. Kim, J.-h. Oh, P.-M. Lee, B.-H. Jeon, and K.-H. Oh. “Development of the homing and docking algorithm for AUV”. In: *The Thirteenth International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers. Honolulu, Hawaii, USA, 2003. URL: https://www.onepetro.org/conference-paper/ISOPE-I-03-112.

[63] V. L. Tran and N. Im. “A study on ship automatic berthing with assistance of auxiliary devices”. In: *International Journal of Naval Architecture and Ocean Engineering* 4.3 (2012), pp. 199–210. DOI: 10.2478/ijnaoe-2013-0090.

[64] K. Hasegawa and T. Fukutomi. “On harbour manoeuvring and neural control system for berthing with tug operation”. In: *Proceedings of the 3rd IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*. Southampton, UK, 1994, pp. 197–210.

[65] Y. A. Ahmed and K. Hasegawa. “Automatic ship berthing using artificial neural network trained by consistent teaching data using nonlinear programming method”. In: *Engineering Applications of Artificial Intelligence* 26.10 (2013), pp. 2287–2304. DOI: 10.1016/j.engappai.2013.08.009.

[66] V.-S. Nguyen and N.-K. Im. “Automatic ship berthing based on fuzzy logic”. In: *International Journal of Fuzzy Logic and Intelligent Systems* 19.3 (2019), pp. 163–171. DOI: 10.5391/ijfis.2019.19.3.163.

[67] J. Woo and N. Kim. “Vector field based guidance method for docking of an unmanned surface vehicle”. In: *Proceedings of the ISOPE Pacific/Asia Offshore Mechanics Symposium*. Gold Coast, Australia, 2016, pp. 276–281.

[68] K. Djouani and Y. Hamam. “Minimum time-energy trajectory planning for automatic ship berthing”. In: *IEEE Journal of Oceanic Engineering* 20.1 (1995), pp. 4–12. DOI: 10.1109/48.380251.

[69] N. Mizuno, Y. Uchida, and T. Okazaki. "Quasi real-time optimal control scheme for automatic berthing". In: *Proceedings of the 10th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*. Copenhagen, Denmark, 2015, pp. 305–312. DOI: 10.1016/j.ifacol.2015.10.297.

[70] S. Li, J. Liu, R. R. Negenborn, and Q. Wu. "Automatic docking for underactuated ships based on multi-objective nonlinear model predictive control". In: *IEEE Access* 8 (2020), pp. 70044–70057. DOI: 10.1109/access.2020.2984812.

[71] A. B. Martinsen, A. M. Lekkas, and S. Gros. "Autonomous docking using direct optimal control". In: *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles (CAMS)*. Daejeon, South Korea, 2019, pp. 97–102. DOI: 10.1016/j.ifacol.2019.12.290. arXiv: 1910.11625 [eess.SY].

[72] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11 (2018), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[73] A. Wächter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106 (2005), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[74] A. Uttisrud. "Hybrid collision avoidance for autonomous passenger ferries". MA thesis. Norwegian University of Science and Technology, 2020. URL: https://hdl.handle.net/11250/2656722.

[75] E. H. Thyri, E. A. Basso, M. Breivik, K. Y. Pettersen, R. Skjetne, and A. M. Lekkas. "Reactive collision avoidance for ASVs based on control barrier functions". In: *Proceedings of the IEEE Conference on Control Technology and Applications (CCTA)*. Montreal, QC, Canada, 2020, pp. 380–387. DOI: 10.1109/ccta41146.2020.9206340.

[76] I. M. Ross and M. Karpenko. "A review of pseudospectral optimal control: from theory to flight". In: *Annual Reviews in Control* 36.2 (2012), pp. 182–197. DOI: 10.1016/j.arcontrol.2012.09.002.

[77] A. M. Lekkas, A. L. Roald, and M. Breivik. "Online path planning for surface vehicles exposed to unknown ocean currents using pseudospectral optimal control". In: *Proceedings of the 10th IFAC Conference on Control Applications in Marine Systems (CAMS)*. Trondheim, Norway, 2016. DOI: 10.1016/j.ifacol.2016.10.313.

[78] G. Bitar. “Towards the development of autonomous ferries”. MA thesis. Norwegian University of Science and Technology, 2017. URL: http://hdl.handle.net/11250/2465617.

[79] A. L. Roald. “Path planning for vehicle motion control using numerical optimization methods”. MA thesis. Norwegian University of Science and Technology, 2015. URL: http://hdl.handle.net/11250/2352517.

[80] M. Breivik and T. I. Fossen. “Path following for marine surface vessels”. In: *Proceedings of the MTS/IEEE OCEANS Conference*. Kobe, Japan, 2004, pp. 2282–2289. DOI: 10.1109/oceans.2004.1406507.

[81] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger. “Safe maritime autonomous navigation with COLREGS, using velocity obstacles”. In: *IEEE Journal of Oceanic Engineering* 39.1 (2014), pp. 110–119. ISSN: 0364-9059. DOI: 10.1109/JOE.2013.2254214.

[82] M. R. Benjamin, J. J. Leonard, J. A. Curcio, and P. M. Newman. “A method for protocol-based collision avoidance between autonomous marine surface craft”. In: *Journal of Field Robotics* 23.5 (2006), pp. 333–346. DOI: 10.1002/rob.20121.

[83] A. B. Martinsen, A. M. Lekkas, and S. Gros. *Optimal model-based trajectory planning with static polygonal constraints*. Submitted for publication. 2020. arXiv: 2010.14428 [eess.SY].

[84] Object Management Group. *Unified modeling language*. Version 2.5.1. 2017. URL: https://www.omg.org/spec/UML/2.5.1/PDF.

[85] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen. “A deliberative architecture for AUV control”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Pasadena, CA, USA, 2008. DOI: 10.1109/robot.2008.4543343.

[86] M. S. Wiig, R. A. Løvlid, K. Mathiassen, and T. R. Krogstad. “Decision autonomy for unmanned vehicles”. In: *Proceedings of the Information Systems Technology Panel*. 127. NATO Science and Technology Organization. 2016.

[87] T. Huntsberger, H. Aghazarian, A. Castano, G. Woodward, C. Padgett, D. Gaines, and C. Buzzel. “Intelligent autonomy for unmanned sea surface and underwater vehicles”. In: *Proceedings of the AUVSI North America Conference*. San Diego, CA, USA, 2008.

[88] S. Fleury, M. Herrb, and R. Chatila. "GENOM: a tool for the specification and the implementation of operating modules in a distributed robot architecture". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS)*. Grenoble, France, 1997. DOI: 10.1109/iros.1997.655108.

[89] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard. "Nested autonomy for unmanned marine vehicles with MOOS-IvP". In: *Journal of Field Robotics* 27.6 (2010), pp. 834–875. DOI: 10.1002/rob.20370.