

Doctoral thesis

Doctoral theses at NTNU, 2021:164

Andreas Bell Martinsen

Optimization-based Planning and Control for Autonomous Surface Vehicles

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



Norwegian University of
Science and Technology

Andreas Bell Martinsen

Optimization-based Planning and Control for Autonomous Surface Vehicles

Thesis for the Degree of Philosophiae Doctor

Trondheim, September 2021

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



Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

© Andreas Bell Martinsen

ISBN 978-82-326-6068-1 (printed ver.)
ISBN 978-82-326-5438-3 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2021:164

Printed by NTNU Grafisk senter

Summary

With autonomy offering a number of benefits in robotics applications, such as increased safety, better consistency and reliability, reduced environmental impact and higher efficiency, it is not surprising that the topic has seen an increase in interest from both the research community as well as commercial and defence industries. In the maritime sector, autonomy has mostly been limited to autonomous underwater vehicles (AUVs), where the operational conditions allow for only limited or delayed communication, making direct or remote control by humans difficult. In recent years however, the focus has shifted to include autonomous surface vehicles (ASVs), with applications such as surveying and mapping, surveillance, and transportation. In order to deliver on the promises of autonomy for ASVs, one of the challenges that needs to be overcome, is designing robust, efficient and safe control systems, enabling the ASVs to plan their mission, make decisions based on sensory feedback, and command the vehicle control surfaces.

This thesis presents topics related to optimization and control of ASVs. This includes low-level motion control, mid-level local trajectory planning and collision avoidance (COLAV), and high-level global trajectory planning. The main part of the thesis, is a collection of peer-reviewed articles, six journal articles and three conference papers. In addition to the article collection, the initial part of the thesis contains an introduction to the main topics of low-level motion control, mid-level local trajectory planning and COLAV and high-level global trajectory planning. This provides context to the publications, and explains the relationship between the different publications.

In the context of performing autonomous marine operations, one of the first tasks, is to plan a high-level path or trajectory in order to meet the mission objective. This should be done in a way that accounts for geographical data as well as the limitations of the ASV, in order to ensure that the vessel is able to follow the plan without having to worry about colliding into known static

Summary

obstacles. As part of this thesis, we present three papers concerned with planning high-level global trajectories, which in addition to planning collision free trajectories for ASVs, also finds a trajectory which optimizes a performance measure, such as energy, time and distance. The proposed planning methods combine classical combinatorial planning algorithms and convex optimization into a new class of hybrid methods, which improves both the performance of the algorithms and the optimality of the planned trajectory.

Once an ASV is following the high-level global trajectory new obstacles such as other moving vessels and unmapped landmasses may be detected, leaving the initial global trajectory no longer feasible. To solve this problem, a mid-level local trajectory planner is needed, in order re-plan parts of the trajectory such that collisions with the obstacles is avoided. As part of this thesis, we present four papers concerned with planning mid-level local trajectories. Three of these papers focus on the problem of docking and berthing in confined waters, in a way that accounts for the vessel geometry, the harbor layout, and unmapped obstacles from exteroceptive sensors. The fourth paper discusses the problem of risk assessment and COLAV during transit, and proposes a novel approach for representing dynamic obstacles with both measurement and behavioural uncertainty.

Once a trajectory has been planned, we would like to execute the plan by maneuvering the ASV. This process, called motion control, involves controlling the actuators and control surfaces of the vessel in a way that follows a course, path or trajectory. For marine vessels, motion control is complicated by the unpredictable nature of the marine environment, and the complex hydrodynamic interactions, which can vary significantly during operations. As part of this thesis, we present two papers on reinforcement learning (RL)-based motion control for marine vessels, which demonstrate how on-line learning can be used to optimize the performance of the motion control system.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU). The work has been carried out at the Department of Engineering Cybernetics (ITK), with Associate Professor Anastasios Lekkas as my main supervisor, and Professor Sébastien Gros as my co-supervisor.

Acknowledgements

I am very grateful for the support of my supervisors, who have helped me through the PhD. I would first of all like to thank my main supervisor Anastasios Lekkas. Without your early encouragement I would probably not have even considered doing a PhD. You have always been supportive and helpful, and our regular meeting schedule has helped keep me focused with clear goals. I would also like to thank my co-supervisor Sébastien Gros. I have enjoyed our many interesting discussions regarding this work, and I have really appreciated your help in wrapping my head around some of the more technical aspects.

Through out this work, I have been fortunate enough to collaborate with a number of people. I would particularly like to thank Glenn Bitar, I have really enjoyed working together, and appreciated our many discussions. Without your help and determination, much of the experimental work on *milliAmpere* would not have been possible. I would also like to thank DNV (earlier DNV GL), and in particular Tom Arne Pedersen for his help with the *ReVolt* experiments. Despite some initial setbacks, we were able to get things up and running eventually. I would additionally like to thank Jon Arne Glomsrud and Morten Breivik for their contributions and feedback on our collaborations.

I would also like to thank my friends and colleagues at ITK for a great work environment. Having home office for much of the past year made me realize

Preface

how important a good work environment really is. Additionally, I would like to thank my office mates Joakim, Otávio, Mathilde and Erlend for a great office environment, and for keeping me hydrated with plenty of coffee breaks. I would in particular like to thank my deskmate Joakim: Sorry for distracting you with all my questions, but I hope our discussions were mutually beneficial.

Finally, I would like to thank my parents, Leann and Alf Henning, and my siblings Martin and Edwin, thanks for inspiring and encouraging me throughout the years. Without your support this would not have been possible.

May 2021, Trondheim

Andreas

Contents

Summary	iii
Preface	v
Contents	vii
Abbreviations	ix
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	3
1.3 Publications	5
1.4 Outline	7
2 Background	9
2.1 High-level global trajectory planning	10
2.2 Mid-level local trajectory planning	15
2.3 Low-level motion control	20
3 Contributions	23
3.1 Optimization-based trajectory planning in static polygonal environments	24

vii

3.2	Docking and berthing of ASVs	26
3.3	Obstacle representation for risk assessment and collision avoidance	29
3.4	Reinforcement learning-based motion control	30
4	Discussion	33
4.1	Conclusion	33
4.2	Reflections	35
4.3	Future work	38
5	Publications	41
A	Autonomous docking using direct optimal control	43
B	Reinforcement learning-based tracking control of USVs in vary- ing operational conditions	61
C	Combining system identification with reinforcement learning- based MPC	95
D	Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments	113
E	Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments . . .	133
F	Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments	171
G	Optimal Model-Based Trajectory Planning With Static Polyg- onal Constraints	203
H	Two space-time obstacle representations based on ellipsoids and polytopes	235
I	Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments	259
	References	291

Abbreviations

ADP approximate dynamic programming	3, 31, 35, 61, 259
AI artificial intelligence	21, 35, 61, 259
ANN artificial neural network	18, 38, 171
asNMPC advanced-step nonlinear model predictive control	32, 259
ASV autonomous surface vehicle	iii, iv, 2, 3, 5, 9, 13–15, 19, 23, 24, 26, 27, 33–35, 37, 38, 61, 113, 133, 171, 203, 259
AUV autonomous underwater vehicle	iii, 2, 43, 113
CDT constrained Delaunay triangulation	25, 203
COLAV collision avoidance	iii, iv, 5, 9, 15–17, 33–35, 37–39, 235
COLREGs international regulations for preventing collisions at sea	16, 17, 38, 133, 235
CPA closest point of approach	16, 235
dCPA distance at closest point of approach	235
DL deep learning	18
DNN deep neural network	61, 95
DOF degrees of freedom	21, 43, 171, 203, 259
DP dynamic positioning	1, 21, 27, 31, 61, 113, 133, 171, 259
DQN deep Q-network	61
DRL deep reinforcement learning	30, 35, 61

Abbreviations

DT Delaunay triangulation	203
DW dynamic window	16, 235
EKF extended Kalman filter	61
GNSS global navigation satellite system	18, 21, 61, 113, 133, 171
HJB Hamilton–Jacobi–Bellman	61
IAE integral absolute error	61, 259
IFT implicit function theorem	259
KKT Karush–Kuhn–Tucker	95
LICQ linear independence constraint qualification	259
LIDAR light detection and ranging	28, 171
LP linear programming	171
LQG linear–quadratic–Gaussian	21
LQR linear–quadratic regulator	21
MDP markov decision processes	95
ML machine learning	21, 35, 259
MPC model predictive control 3, 17, 21, 27, 31, 32, 39, 43, 95, 113, 171, 259	
NED North-East-Down	43, 61, 171, 203, 259
NLP nonlinear programming	43, 113, 133, 171, 259
NMPC nonlinear model predictive control . 27, 32, 34, 35, 43, 113, 171, 235, 259	
OCP optimal control problem	21, 24, 26, 27, 43, 113, 133, 171, 235, 259
ODE ordinary differential equation	43, 61, 171
PEM prediction error method	95, 259
PID proportional-integral-derivative	20, 21, 113, 133, 171, 259
POA projected obstacle area	235

PRM probabilistic roadmap	12, 203, 235
PSO particle swarm optimization	13, 203
RL reinforcement learning	iv, 3, 21, 30–32, 35, 36, 61, 95, 259
RRT rapidly-exploring random tree.....	12, 133, 203, 235
RTK real-time kinematic	61, 113, 133, 171
SOSC second order sufficient conditions	259
SYSID system identification.....	3, 31, 32, 35, 95, 259
TA thrust allocation	43, 61, 113, 133, 171
tCPA time to closest point of approach.....	235
USV unmanned surface vehicle	2, 31, 32, 43, 61, 259
VO velocity obstacle	16, 235

Abbreviations

1 | Introduction

This chapter contains a brief motivation for the topics covered in this thesis, a summary of the main contributions, an overview of the publications presented in the thesis, and finally a outline of the thesis.

1.1 Motivation

In late 1946, Mr. D. S. Harder of the Ford Motor Company introduced the word *automation*, meaning self-acting, moving or acting on its own. The term was initially used in the context of manufacturing, to describe work that could be done with little to no human intervention. However, automation was quickly adopted outside of manufacturing, and was used to describe a variety of systems, where mechanical, electrical, or computerized actions are used to control a process, and reduce the need for human effort, intelligence and intervention. While the term automation was first introduced in 1946, its history dates back back much further. In ancient times mechanisms such as water floats were used to automatically control water level [1], and during the industrial revolution, mechanisms such as the governor, used to measure and regulate the speed of a machine, helped lay the foundation of the field of automatic control and control theory [2]. With the advent of computers, automation has become more accessible and more powerful, allowing for even more complex processes to be automated. In recent years, this has lead to the rise of *autonomy*, meaning independent or having its own laws, where even the complex decision making is being left to computers, allowing the system to not only act on its own, but also be self-governing, and operating without human intervention.

In the maritime industry, research into ship automation started in the early 1870s, with the German Navy conducting experiments on automatic steering. This was done by using electric motors connected to the rudder of torpedo

Introduction

boats, and controlled by relays connected to the magnetic needle of a compass [3]. This early work saw only limited success, but research into the problem continued, and going into the 1920s, work by Nicolas Minorsky on the automatic steering problem [4] helped lay the foundation of control theory with his formal discussions on the topic [5]. In 1922 automatic steering became commercially available with the invention of the *gyropilot*, a heading autopilot produced by the Sperry Corporation. The *gyropilot* was initially installed on the cargo and passenger ship *Munargo*, and within 10 years, more than 400 *gyropilots* were in service [3]. Since then, marine vessels have become increasingly automated, with most commercial vessels having some form of autopilot, and many newer vessels having advanced motion control systems, such as dynamic positioning (DP), allowing vessels position to be controlled with a high degree of accuracy and maneuverability.

Autonomy in the maritime industry, has mostly been limited to autonomous underwater vehicles (AUVs), where the operational conditions allow for only limited or delayed communication, making direct or remote control by humans difficult. For AUV operations, the mission and objectives are typically specified by a human operator ahead of time, which the vehicle then performs autonomously once it has been launched. While autonomy has historically been used to get around the problem of limited and delayed communication, autonomy also offers a number of other benefits, such as increased safety, better consistency and reliability, reduced environmental impact and higher efficiency. In recent years, this has increased the interest in using autonomy for tasks where human operators have traditionally been in control. The most prominent example of this is the researcher effort into autonomy in the automotive industry, where the goal is to develop self driving technology for transportation of both passengers and goods [6, 7].

Similar to other industries, research into autonomy for surface vessels has also increased in recent years, with applications such as surveying and mapping, surveillance, and transportation, being of interest both for commercial and government use. In the defence sector a number of countries are looking at small autonomous surface vehicles (ASVs) and unmanned surface vehicles (USVs) for surveillance and reconnaissance, such as the Norwegian Defence Research Establishment's *Odin* platform [8]. Larger vessels have also been built, such as the Defense Advanced Research Projects Agency (DARPA) *Sea Hunter*, designed to detect and track submarines over long periods of time [9]. In the commercial sector, there has been increasing interest in autonomous passenger and cargo transport, with the world's first autonomous car ferry *Falco*, developed by Rolls-Royce Commercial Marine, entering into service in 2018

[10]. Since then, both the Finnish company Wärtsilä and Norwegian company Kongsberg Maritime have demonstrated similar autonomy solutions on car ferries in Norway [11, 12]. A number fully electric autonomous cargo vessels have also been proposed, including the *ReVolt* by DNV (then DNV GL) [13] (see Figure 1.1), Yara Birkeland [14], and ASKO zero-emission autonomous vessel [15]. Autonomous passenger vessels have also been proposed, such as the small autonomous urban passenger ferry *milliAmpere* (see Figure 1.2), designed as an alternative to bridges or manned ferries [16].

With autonomy offering a number of benefits, such as increased safety, better consistency and reliability, reduced environmental impact and higher efficiency, it is not surprising that there has been an increase in interest and research on the topic. Despite this, we are arguably still at a crossroads between autonomy and automation, with one of the major challenges being that of trusting the systems to safely perform the desired task in a wide range of operational conditions. This is further complicated when introducing learning based components into the autonomy systems, resulting in the need for trusting any future changes that the learning system may make. This means that most existing systems are still considered automatic, as they rely on having a human in the loop to ensure the system operates safely. In order to deliver on the promises of full autonomy for ASVs, one of the challenges that needs to be overcome, is designing robust, efficient and safe control system, enabling the vehicle to plan its mission, make decisions based on sensory feedback, and command the vehicle control surfaces. In this thesis we look at different approaches to the different layers of the control system, with a particular focus on optimization based methods.

1.2 Contributions

While working towards this thesis, several algorithms and methods for optimization based motion control, reactive planning, collision avoidance and trajectory planning, were developed. These contributions are discussed in detail in Chapter 3, with a summary given below.

- Two reinforcement learning (RL) based motion control methods for ASVs, based on approximate dynamic programming (ADP) and model predictive control (MPC) respectively. The methods rely on RL and system identification (SYSID) in order to optimize the closed loop performance of the control system, and were tested in simulations as

Introduction



Figure 1.1: The *ReVolt*, a 1 : 20 scale model autonomous cargo concept vessel developed by DNV (then DNV GL).



Figure 1.2: The *milliAmpere*, an experimental autonomous urban passenger ferry developed by NTNU.

well as full-scale experiments. Based on the results, we show how our proposed methods are able to perform online learning in order to optimize the closed loop performance, and outperform other traditional control approaches.

- A mid-level local trajectory planning method for docking and berthing of ASVs, which can use map data and ranging sensors in order to plan safe collision free docking maneuvers. In addition to testing the method in simulations, the method was also implemented on a small urban passenger ferry and tested with full scale experiments, with very good results. To the authors knowledge, this is the most comprehensive treatment of the docking problem in academia to date.
- A space-time obstacle representation for predicting the movement of dynamic obstacles under both measurement and behavioural uncertainty. Using the proposed representation, we show how it can be used in various optimization based mid-level local trajectory planners for collision avoidance (COLAV), in a way that is both robust and computationally efficient.
- Two different high-level optimization based trajectory planners, used to plan optimal model based trajectories for ASVs in static environments with polygonal spatial constraints. Compared to existing methods, the methods proposed in this thesis allow for planning model based optimal trajectories using an exact polygon representation of the geographical data.

1.3 Publications

Given below, is the list of original publications which were written as a result of the work on the thesis. There is a total of nine publications, consisting of three conference articles and six journal articles. The articles are ordered chronologically by date of publication, however the recommended (thematic) reading order, illustrated in Figure 1.3, is high-level planning (Paper G and Paper E), mid-level planning (Paper A, Paper D, Paper F and Paper H), and finally low-level motion control (Paper B, Paper C and Paper I).

Conference publications

- A Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Autonomous docking using direct optimal control”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 97–102. DOI: 10.1016/j.ifacol.2019.12.290
- C Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Combining system identification with reinforcement learning-based MPC”. in: *IFAC-PapersOnLine* 53.2 (2020), pp. 8130–8135. DOI: 10.1016/j.ifacol.2020.12.2294
- D Glenn Bitar, Andreas B Martinsen**, Anastasios M Lekkas, and Morten Breivik. “Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 14488–14494. DOI: 10.1016/j.ifacol.2020.12.1451

Journal publications

- B Andreas Bell Martinsen**, Anastasios Lekkas, Sébastien Gros, Jon Arne Glomsrud, and Tom Arne Pedersen. “Reinforcement learning-based tracking control of USVs in varying operational conditions”. In: *Frontiers in Robotics and AI* 7 (2020), p. 32. DOI: 10.3389/frobt.2020.00032
- E Glenn Bitar, Andreas B Martinsen**, Anastasios M Lekkas, and Morten 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
- F Andreas B Martinsen**, Glenn Bitar, Anastasios M Lekkas, and Sébastien 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
- G Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Optimal Model-Based Trajectory Planning With Static Polygonal Constraints”. In: *IEEE Transactions on Control Systems Technology* 29.5 (2021). DOI: 10.1109/TCST.2021.3094617

I Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments”. In: *Review* (2021)

H Andreas B Martinsen and Anastasios M Lekkas. “Two space-time obstacle representations based on ellipsoids and polytopes”. In: *IEEE Access* 9 (2021). DOI: 10.1109/ACCESS.2021.3103323

1.4 Outline

The rest of the thesis is structured as follows: Chapter 2 contains background on the topics covered in the publications. Chapter 3 gives an in-depth presentation of the contributions of the publications. Chapter 4 concludes, summarizes and reflects on the work, and discusses some directions for future work. Finally, Chapter 5 contains the publications that were written as a result of the work on this thesis.

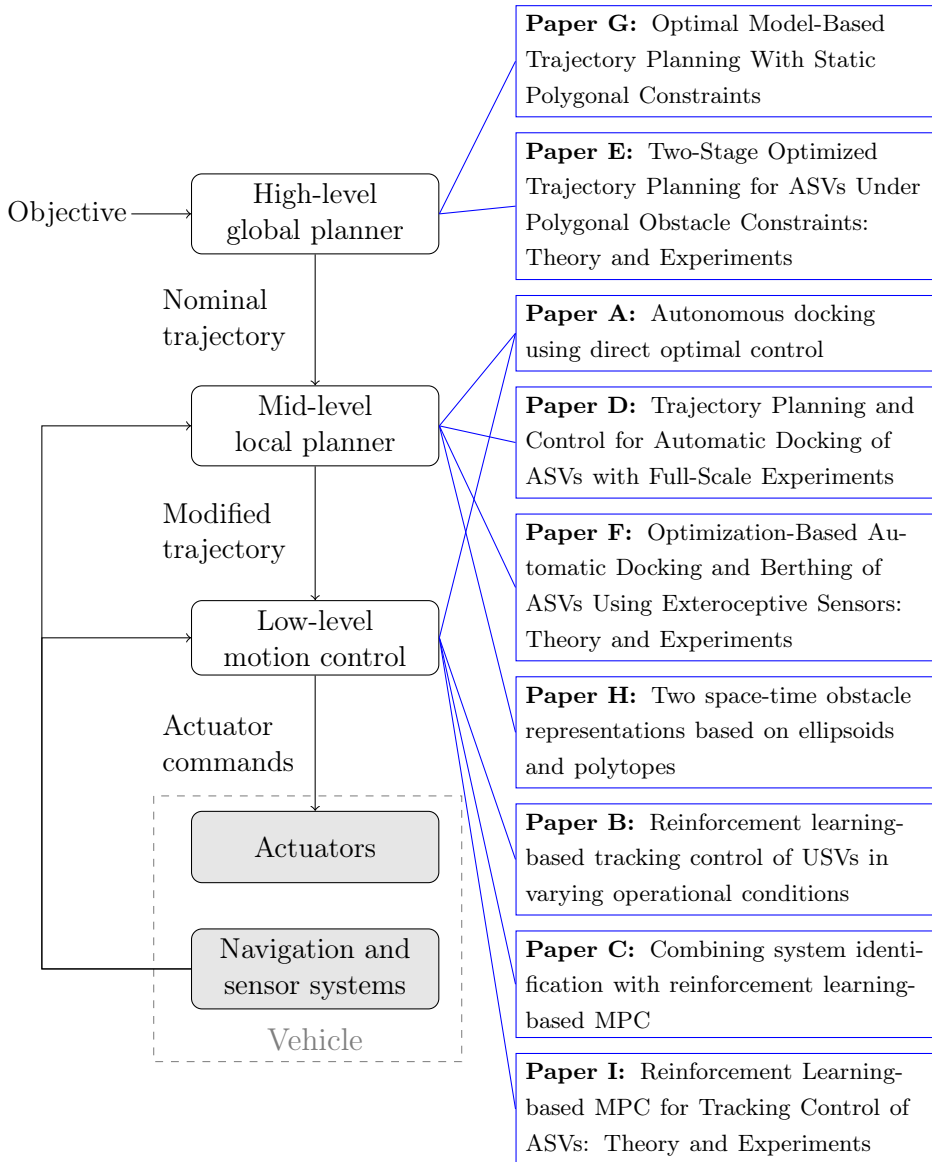


Figure 1.3: Hierarchical control system illustrating the relationship between the different abstraction layers and publications.

2 | Background

The control system for an ASV is typically designed hierarchically, where the different abstraction layers are responsible for specific tasks [26–28]. Depending on the task, the abstraction layers may differ significantly, with the two main layers traditionally being guidance and control. In this architecture the guidance layer is responsible for calculating the desired behaviour such as speed and heading, while the control layer is responsible for moving the actuators in order to follow the desired behaviour. With the growing complexity of ASVs, it is useful to split guidance in two, giving a three layered approach. These levels consist of high-level global planning, mid-level local trajectory planning, and low-level motion control, as illustrated in Figure 2.1.

The main task of the high-level global trajectory planner is to plan a feasible trajectory given a specific mission or objective. High level planning is typically only executed once, at the start of the mission, and considers only known static obstacles such as geographical map data.

The resulting nominal trajectory from the high-level global trajectory planner is then passed to the mid-level local trajectory planner, which is tasked with re-planning a modified trajectory when the need arises, in order to ensure collision avoidance. Mid-level local planning typically involves using exteroceptive sensors and situational awareness, in order to identify obstacles which were not known to the high-level planner. To account for the changing environment and the limited field of view of the exteroceptive sensors, the mid-level local planning is typically performed iteratively as new information is gathered from the exteroceptive sensor and situational awareness systems.

The modified trajectory from the mid-level local planner is then passed on to the low-level motion control, which is tasked with controlling the vessel actuators and control surfaces in order to accurately track the modified trajectory. The low-level motion control is performed using feedback control, allowing the

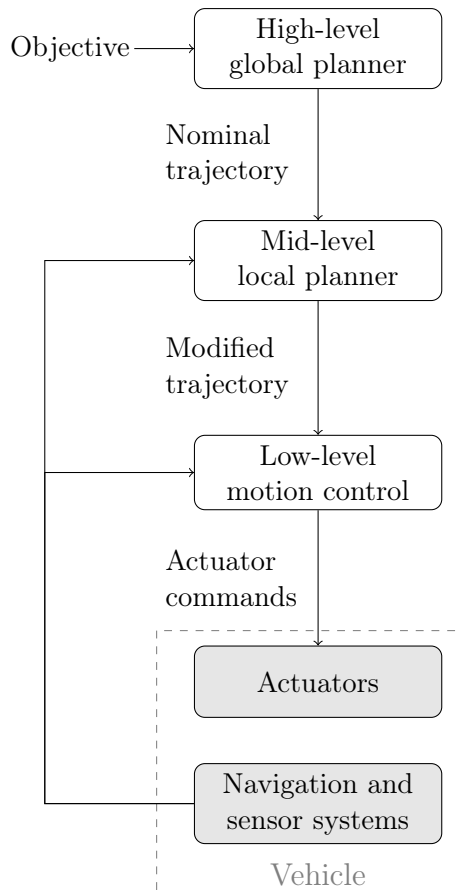


Figure 2.1: Hierarchical control system illustrating the three main abstraction layers of an ASV control system.

vessel to account for model uncertainty and environmental disturbances.

In the rest of this chapter, we will present some background on each of these three abstraction layers, with a particular focus on optimization-based approaches.

2.1 High-level global trajectory planning

In robotics, global trajectory planning, illustrated in Figure 2.2, is the process of finding a time-parametric continuous sequence of configurations, called a

2.1. High-level global trajectory planning

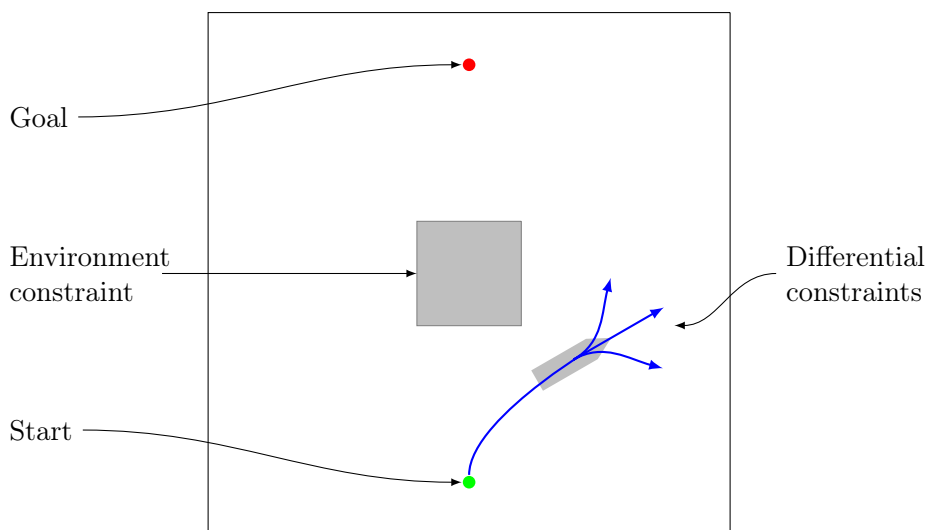
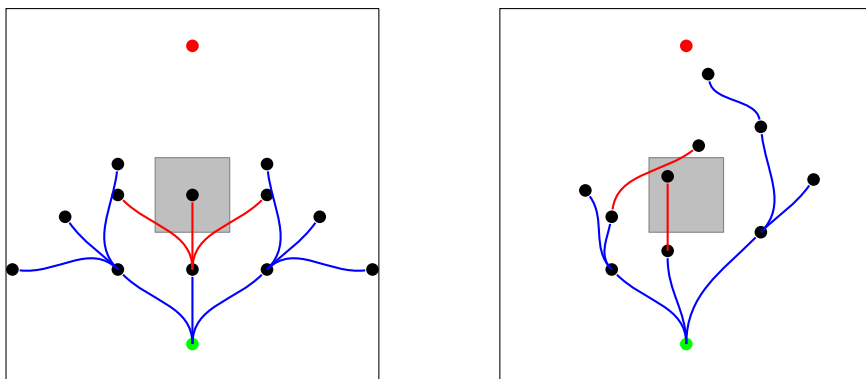


Figure 2.2: Trajectory planning problem

trajectory, which move the robot safely from some initial configuration to a goal configuration [29]. To be successful in the real world, the trajectory planner must be able to consider a variety of different constraints. This includes *environment constraints*, such as static and dynamic obstacles, and *differential constraints*, which arise from the kinematics and dynamics of the robot. A trajectory is called a feasible trajectory if it connects the initial configuration and the goal configuration in a way that satisfies both the environment constraints and the differential constraints. This means that the robot should be able to follow the feasible trajectory since it satisfies the differential constraints, and that the trajectory should not lead to any collisions since it satisfies the environment constraints. In general, trajectory planning problems can have more than one feasible trajectory. This allows for searching through the trajectories in order to find the "best" feasible trajectory with respect to some performance measure. This is called optimal trajectory planning, and the objective is often to find trajectories that minimize either energy, distance or time. Due to a potentially large number of obstacles, actuators, as well as complex kinematics and dynamics, trajectory planning is in general a difficult problem, and finding only a single feasible trajectory can be computationally expensive. This is further complicated with optimal trajectory planning, as it requires searching all feasible trajectories in order to find the trajectory that optimizes a given performance measure. In order to solve the global trajectory planning problem, a wide range of methods have been proposed, with most methods falling



(a) Combinatorial methods check all combinations of a given discretization to generate a search tree.

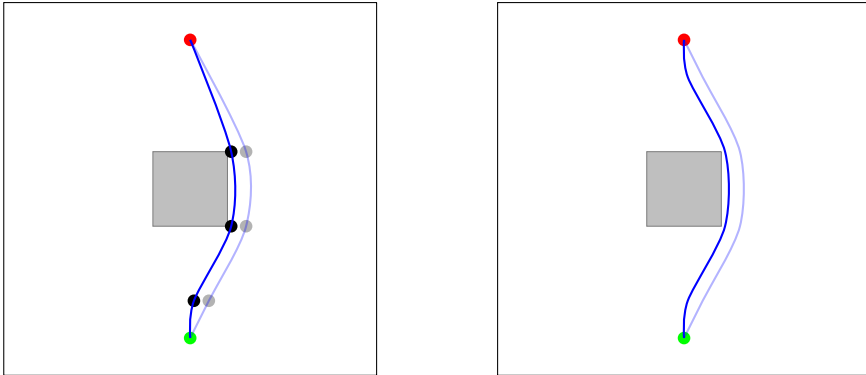
(b) Sampling-based methods randomly sample actions/states to generate a search tree.

Figure 2.3: Illustration of the different classes of roadmap methods, with black dots representing the waypoints, and the blue and red curves representing collision-free and colliding trajectory segments respectively.

into one of two main categories, namely *roadmap methods* and *complete path methods* [29–31].

The main goal of roadmap methods is to find a sequence of waypoints, which, when connected, result in an obstacle-free piecewise-linear path. The path can then be smoothed and turned into a feasible trajectory that complies with the vehicle dynamics. Roadmap methods can be further split into two distinct categories, namely, combinatorial methods and sampling-based methods, as illustrated in Figure 2.3. Combinatorial methods, divide the continuous space into structures that capture all spatial information needed to solve the motion planning using simple graph search algorithms such as Dijkstra [32] or A* [33]. For many complex problems however, combinatorial methods may lead to search spaces so large that the methods are not be computationally feasible. For these problems, sampling-based methods are often used instead. Sampling-based methods, rely on using randomly sampled subset of states or actions. This creates a randomly sampled discretization of the continuous search space, and hence limits the computational complexity at the cost of accuracy and completeness of the discretization. Some notable combinatorial methods include coarse planning with path smoothing, in where a mesh, grid or potential field is used to plan a course path [34–36], and then a method using curve segments, splines or motion primitives is used to refine the trajectory [37–43]. Notable sampling-based methods include probabilistic roadmap (PRM) [44],

2.1. High-level global trajectory planning



(a) Piecewise methods such as multiple shooting and collocation connect multiple trajectory segments to form a full trajectory.

(b) Continuous methods such as pseudospectral and single shooting, use high-degree polynomials and simulation to represent the entire trajectory.

Figure 2.4: Illustration of the different classes of complete path methods, with the dots representing the start and endpoints of the trajectory segments, and the blue curves showing how the trajectory can be optimized from one iteration to the next.

rapidly-exploring random tree (RRT) [45–47], and random-walk planners [48, 49].

Complete path methods on the other hand, produce a continuous parameterized trajectory by explicitly taking into account the motion equations of the robot and the full continuous search space. As a result, these methods generate a trajectory that is both obstacle-free and feasible, without further need of refinement or smoothing. Most complete path methods rely on some form of mathematical optimization. For some simple problems an analytical solution exists, as is the case for Dubins paths [50] and Reeds-Shepp [51]. In general, however, researchers must resort to numerical optimization, where handling complex constraints is challenging and getting stuck in local optima is not uncommon. Notable numerical methods, illustrated in Figure 2.4, include particle swarm optimization (PSO) [52, 53], single and multiple shooting methods [54] which are based on simulation, collocation methods [55], which are based on function approximation of low-degree polynomials, and pseudospectral methods [56], which are based on function approximation of high-degree polynomials.

In the context of high-level global trajectory planning for ASVs, the goal is to

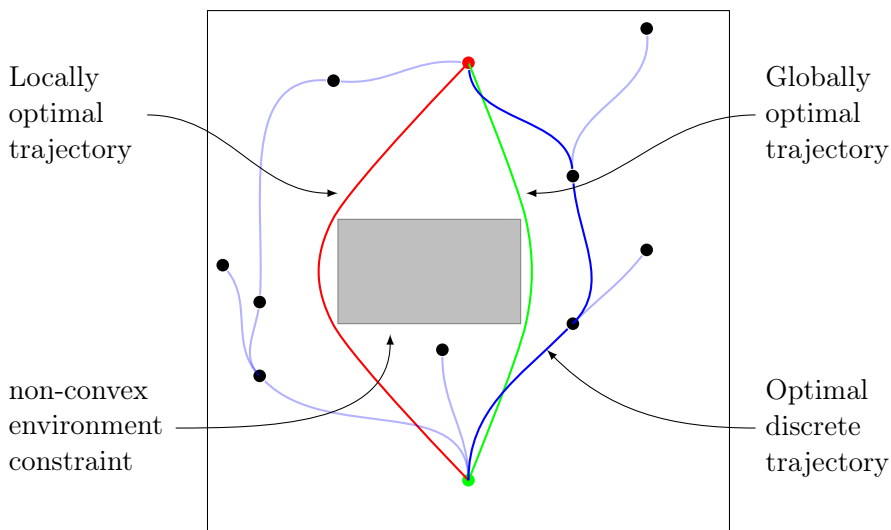


Figure 2.5: Complete path methods can only guarantee convergence to locally optimal trajectories, while roadmap methods are only able to find the optimal trajectory of the discretization.

use geographical data such as maps, in order to plan a feasible trajectory from an initial starting location to a goal destination. In coastal regions, trajectory planning is often complicated by the complex structure of the geographical data making up the environment constraints. These constraints will often make the trajectory planning problem non-convex, see Figure 2.5, meaning that most optimization-based complete path methods can not be guaranteed to find the globally optimal trajectory. For roadmap methods the non-convexity is typically not a problem, as they search the entire discretized search space. However, the optimality of roadmap methods is limited by the underlying discretization. This has led to hybrid trajectory planning methods, where both roadmap methods and complete path methods are combined in order to get improved performance and optimality. In recent years, these hybrid methods have shown a lot of promise when it comes to optimal trajectory planning for ASVs [21, 23, 57, 58], giving methods with optimality similar to complete path methods, and performance, in terms of computational efficiency, similar to roadmap methods.

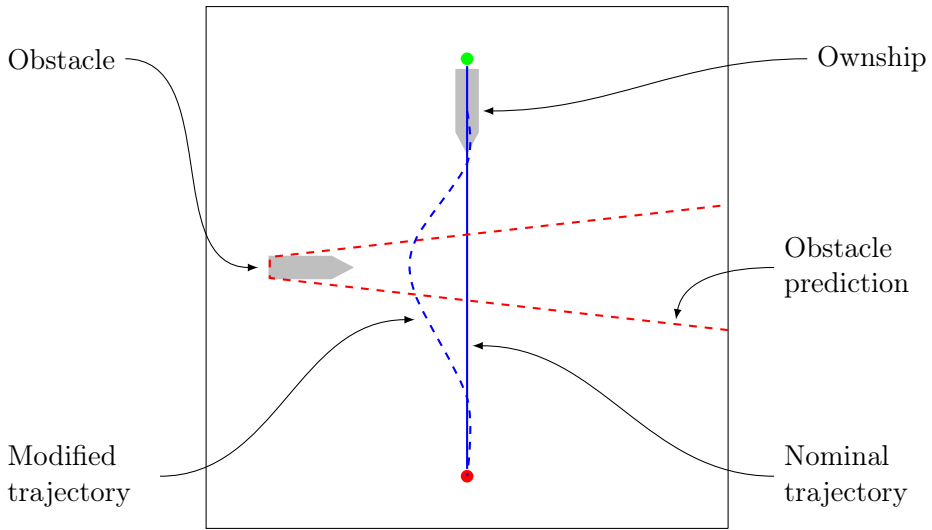


Figure 2.6: Local planning for collision avoidance.

2.2 Mid-level local trajectory planning

In conceptual autonomous marine operations, global trajectory planning can be implemented as a first step, in order to find an initial trajectory which will complete the desired objective, and is feasible with respect to known static obstacles and the ASV kinematics and dynamics. Once the ASV is following the initial global trajectory new obstacles such as other vessels and unmapped landmasses may be detected, leaving the initial global trajectory no longer feasible. To solve this problem, local trajectory planning is used to re-plan parts of the trajectory in order to account for the additional obstacles.

In the rest of this section we will discuss two different scenarios where local trajectory planning is of importance for ASVs. The first scenario pertains to maneuvering in open waters, where we need to perform COLAV with respect to dynamic obstacles, such as other vessels. The second scenario pertains to precision maneuvering in confined areas, such as docking and berthing, where both unmapped static obstacles and the ASV geometry must be considered for safely performing the desired objective.

Dynamic obstacles and COLAV

With increasing interest in autonomy solutions in the maritime industry, it becomes increasingly important to develop robust and efficient methods for risk assessment and collision avoidance. This is especially true for dynamic obstacles, for which accurate obstacle predictions are complicated by measurement and tracking uncertainties, as well as uncertainties in the future behaviour of the obstacle, as illustrated in Figure 2.6. A major component of developing robust and efficient methods for obstacle avoidance, is the underlying obstacle representation. In order for the obstacle representation to be practical, it needs to be able to capture the shape and movement of the obstacle in a way that is robust, allowing for both measurement uncertainties, as well as uncertainties in the obstacle behavior. Additionally, the obstacle representation must be suitable for planning, in order to allow for performing COLAV.

With the commercialisation of radar after World War II, there was a growing interest in studying how these systems could be used to aid mariners when navigating at sea. Into the 1960s and 1970s, technology had progressed to where vessel tracking and risk assessment systems could be integrated into the radar systems. These early systems, mostly relied on the method known as closest point of approach (CPA) for assessing collision risk, by computing the distance and point in time when two vessels are at their closest, given that the vessels have a known constant velocity [59]. While these early systems provided valuable feedback to the operator, they were not true COLAV system, as they still relied on the operator to take appropriate action in order to avoid collisions. Since these early days, research into COLAV has seen significant interest, and has resulted in a wide variety of COLAV algorithms, with most methods falling into one of two main categories, namely *reactive* and *deliberate* COLAV methods [27].

Reactive COLAV methods, often called sense-act methods, perform little to no planning, and are for the most part designed to only perform short term maneuvers in order to avoid collisions. These methods are typically computationally cheap, which makes them well suited for responding to sudden changes in the environment in a way that works well for avoiding immediate danger. However, since these methods only consider a short planning horizon, they are prone to making sub-optimal decisions in terms of the overall mission and objective. One of the most notable reactive COLAV methods, is velocity obstacle (VO), which has been rediscovered and published multiple times [60–62]. VO works by computing the set of all velocities that will result in a collision, and hence collision avoidance can be performed by choosing a velocity which

2.2. Mid-level local trajectory planning

does not fall within the VO. Additional extension to the VO representation allow for kinematic constraints and obstacle behaviour and uncertainty [63–65], with other similar reactive methods such as dynamic window (DW) methods, allowing for accounting for the vessel dynamics [66]. Other notable reactive planners include artificial potential fields [67], multi-objective optimization [68] and set-based methods [69] which can incorporate the international regulations for preventing collisions at sea (COLREGs), and control barrier functions [70], which can consider vessel dynamics and actuator constraints.

Deliberate methods, as opposed to reactive methods, consider larger amounts of information in order to plan multiple maneuvers, typically over a longer time horizon. This makes deliberate methods more computationally expensive, but allow for better long term planning with respect to the overall mission objective, as well as better behaviour in terms of following COLREGs. Deliberate COLAV methods have much in common with the high-level global trajectory planning methods, with the main difference being the inclusion of dynamic obstacles, and a typically shorter planning horizon. Similarly to the high-level global planning methods, deliberate COLAV methods can be split into roadmap methods and complete path methods. Early methods were mostly optimization-based complete path methods [71–74]. In recent years however, focus has shifted to roadmap methods, as they are more computationally efficient, and reliable for physical implementations. Notable roadmap methods include RRT [75], scenario-based MPC, and branching-course MPC [76].

When designing control architectures with COLAV systems today, both deliberate and reactive methods are often used in a two layer approach [27]. This allows for a COLAV system which builds on the complimentary strengths of the different algorithms. Typically this leads to a robust and fast reactive layer which is used mostly as a last resort, and a slower and more complex deliberate layer which performs trajectory planning in accordance with the COLREGs, i.e. the rules of the road at sea.

Static obstacles and Docking and Berthing

The problem of automatics docking and berthing is an important part of performing autonomous transportation. Planning a docking trajectory is in general a local trajectory planning problem, where the goal is to move from an initial pose to a target docking pose and eventually performing a controlled collision with the quay or berth, as seen in Figure 2.7. Since docking typically is performed in confined waters and close proximity to both mapped and

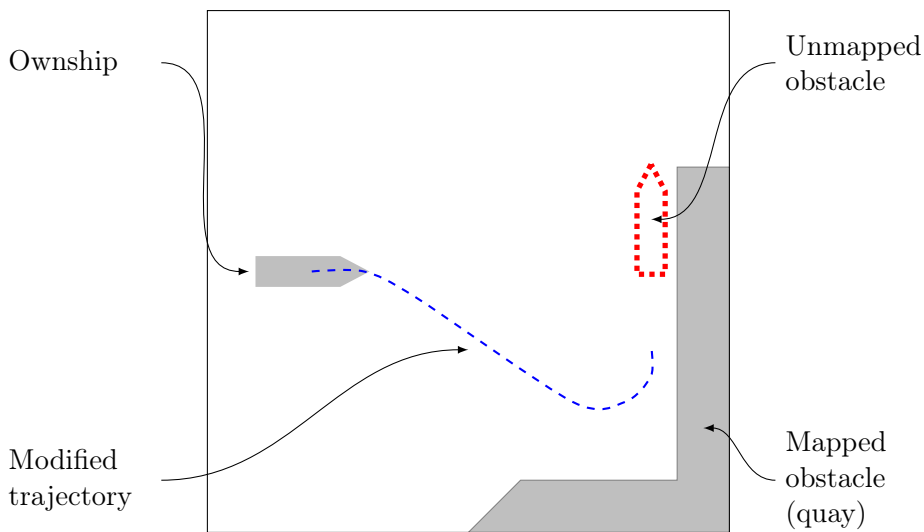


Figure 2.7: Local planning for docking and berthing.

unmapped obstacles, it is important to have accurate positioning data and situational awareness in order to plan and perform safe docking maneuvers.

Traditionally, docking large underactuated vessels have required the use of support vessels, such as tug boats, in order to push and pull the vessel to perform the docking maneuver. This has led to research into synchronizing the movement of multiple tugboats, in order to perform the desired maneuvers [77–80]. 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. This has led to a number of different approaches, including artificial potential field methods [81], fuzzy control systems that change behaviour based on predetermined rules [82–84], Learning-based methods using artificial neural networks (ANNs) [77, 80, 85–91] and deep learning (DL) [92–94], as well as rule-based expert systems [95]. However, the most promising methods rely on optimization-based planning [17, 19, 22, 96–103], 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 performing docking it is important to have accurate and reliable positioning systems in place, in order to determine the position of the vessel hull relative to the quay or berth. Unfortunately most methods proposed in academia lack experimental validation, with only a handful having performed

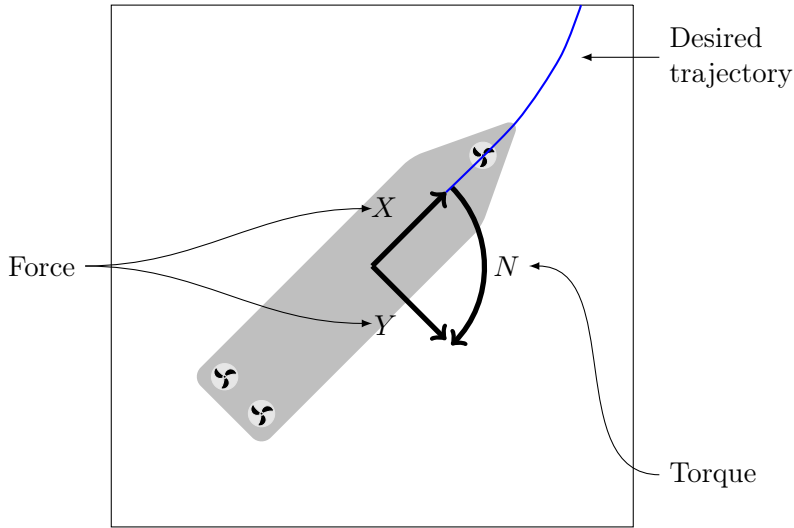


Figure 2.8: Low-level motion control of surface vessels, where the objective is to control the thrusters, and hence the forces X and Y , and torque N , in order to follow a desired course, path or trajectory.

experiments [19, 22, 93, 104]. Within industry, several companies have developed and demonstrated automatic docking systems [10–12], however details about the different approaches remain sparse. While high precision global navigation satellite system (GNSS) can be used to perform docking, it is important to note that this also requires the position of the berth to 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 [104–106] is often used in larger ports, in order to independently identify the position and velocity of the vessel relative to the quay. For full autonomy, relying on quay mounted positioning systems may not be sufficient, and additional vessel mounted ranging systems must be used to ensure that the docking operation can be performed safely and without relying on land-based infrastructure[22]. Relying on additional exteroceptive sensors with a limited field of view, means that only a local region around the vessel can be considered during the planning. This typically also necessitates the use of re-planning as new information is gathered, a trait that is common for most local planners.

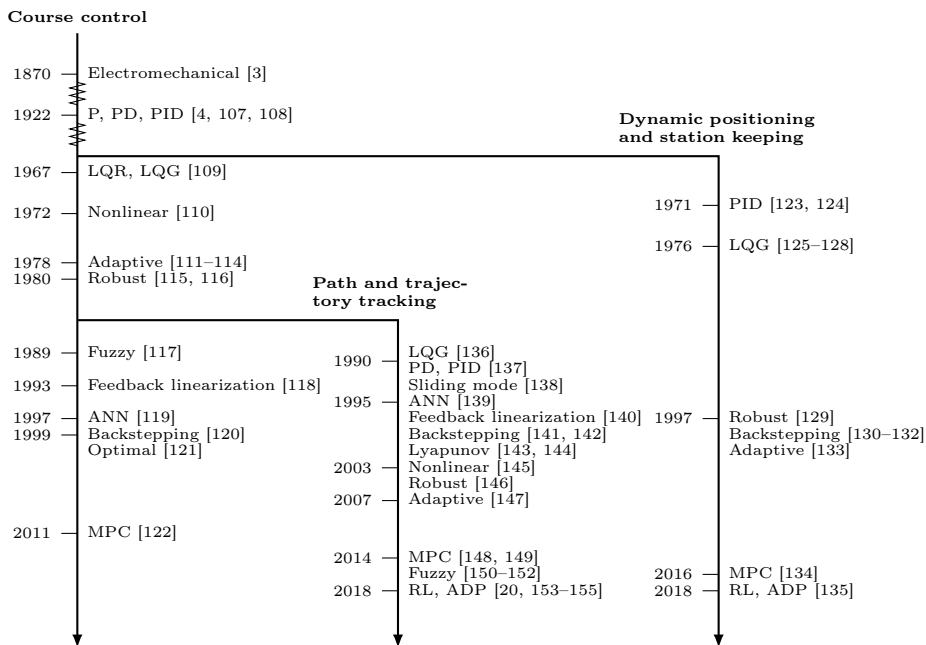


Figure 2.9: Timeline of vessel motion control, based on [156].

2.3 Low-level motion control

Once a trajectory has been planned, it is tracked by computing and executing appropriate maneuvers. This process, called motion control, involves controlling the actuators and control surfaces of the vessel in a way that follows a course, path or trajectory (see Figure 2.8). Designing efficient motion control system for ASVs typically requires the need for an accurate mathematical model describing the dynamics of the vessel. This is complicated by the unpredictable nature of the marine environment, and the complex hydrodynamic interactions, which can vary significantly during operations. This has led to extensive research on the topic of motion control for marine vessels, utilizing ideas from virtually every branch of control engineering (see Figure 2.9).

The first commercially successful motion control system was the *gyropilot*, developed by the Sperry Corporation [3]. This was enabled by the invention of the *gyrocompass*, which as opposed to traditional magnetic compasses, was not affected by magnetic disturbances generated by electrical equipment on steel vessels. As the gyrocompass enabled reliable compass measurement, early research mostly focused on the problem of course control using simple three

term feedback control laws, which today is known as proportional-integral-derivative (PID) control [4].

With the rise of offshore drilling in the early 1960s, DP was invented in order for the drilling vessels to be able maintain the position and heading by using its own thrusters. The first DP systems were manually controlled by human operators, but this was quickly changed in favour of using three decoupled PID controllers, to control the horizontal motion of the vessel (surge, sway and yaw). For these early systems, a challenging problem was wave induced disturbances entering the feedback loop. This however changed with the development of the Kalman filter and the linear–quadratic regulator (LQR), motivating the use of linear–quadratic–Gaussian (LQG) controllers for optimal filtering and control for both DP and course control.

The successful results with LQG controllers for both course control and DP systems, and the commercial availability of GNSS systems, such as GPS, resulted in a growing interest for path and trajectory tracking control for vessels in transit. In the mid 1990s, the problem of trajectory tracking for underactuated vessels gained significant attention. Since underactuated vessels have fewer independent controls than degrees of freedom, linearizing the vessel model about the desired constant position and orientation results in a linear model that is not controllable, and hence controlling underactuated vessels is an inherently nonlinear problem. This led to research into nonlinear control methods, including feedback linearization, backstepping and Lyapunov based methods, for path and trajectory tracking as well as dynamic positioning and course control.

With the increase in processing power the last decade, more computationally demanding control methods have been made possible. One of these methods is MPC, which is a popular approach for optimizing the closed loop performance of complex systems subject to constraints. MPC works by solving an optimal control problem (OCP) at each control interval in order to find an optimal policy. The optimal control problem seeks to minimize the sum of stage costs over a horizon, provided a model of the system and the current observed state. While MPC is a well-studied approach, and an extensive literature exists on analysing its properties [157, 158], the closed loop performance heavily relies on the accuracy of the underlying system model, which naturally presents challenges when significant unmodeled uncertainties are present.

As early as the 1970s, adaptive control methods were used to adapt vessel motion control systems in order to account for uncertain and time varying model parameters. In recent years, the availability of large amounts of data combined with processing power has allowed for new learning-based control

Background

methods. One of these methods is RL, which is a subfield of machine learning (ML), designed to tackle the problem of optimal sequential decision making under uncertainty. The roots of RL can be traced back to the artificial intelligence (AI) community in the 60's [159, 160]. Since then the field has come a long way, evolving in several directions to become one of the most active research areas at the intersection of ML, AI and control theory. Contrary to other machine learning methods, RL does not rely on a prerecorded dataset, but rather learns from evaluative feedback through a process of trial and error. Similarly to optimal control, this feedback comes in the form of a hand-engineered reward or cost function, which assigns a reward, or penalty, to the actions that result in desired, or undesired, outcomes, respectively. Given the reward or cost function, the job of the RL algorithm is to find a state-action mapping, known as the policy (the analog of a controller, in control engineering terminology), that maximizes the expected future reward given the problem constraints and uncertainties. In recent years, RL has proved to be useful as an adaptive control approach for motion control of marine vessels.

3 | Contributions

The main contributions of this thesis can be split into four separate categories, with each publication contributing to one of the categories. The four categories, and respective publications are given as follows:

- Optimization-based trajectory planning in static polygonal environments
 - **Paper E** "Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments"
 - **Paper G** "Optimal Model-Based Trajectory Planning With Static Polygonal Constraints"
- Docking and berthing of ASVs
 - **Paper A** "Autonomous docking using direct optimal control"
 - **Paper D** "Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments"
 - **Paper F** "Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments"
- Obstacle representation for collision avoidance and risk assessment
 - **Paper H** "Two space-time obstacle representations based on ellipsoids and polytopes"
- Reinforcement learning-based motion control
 - **Paper B** "Reinforcement learning-based tracking control of USVs in varying operational conditions"
 - **Paper C** "Combining system identification with reinforcement learning-based MPC"

- **Paper I** "Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments"

In the rest of this chapter, we will provide an in-depth description of the contributions to each of the four categories, as well as the individual contributions of each publication.

3.1 Optimization-based trajectory planning in static polygonal environments

When performing high-level global trajectory planning for ASVs, the goal is to find a feasible trajectory from an starting location to a goal destination. Due to the complexity of the geographical data making up the environment constraints, trajectory planners have traditionally relied on roadmap methods [35, 161]. In recent years, as the focus on lower emissions and more energy and cost efficient solutions has increased, complete path methods have gained popularity, as they allow for planning optimal trajectories. In order to use optimization based approaches, a common approach is to simplify the environmental constraints using for example constraint ellipses [31, 57, 162], as they offer a more computationally efficient representation when using numerical optimization methods. Unfortunately, using simplified environment constraints will in general lead to sub-optimal solutions, and does not solve the problem of getting stuck in local optimal solutions. As part of this thesis, we have developed two different hybrid planning approaches, which combine both roadmap methods and complete path methods, for planning optimal trajectories subject to polygonal environment constraints. Using hybrid approaches we are able to get around the problem of complete path methods getting stuck in local optimum, while also improving computation efficiency. Additionally, allowing for the use of polygonal constraints, eliminates the need for simplifying the environment constraints, resulting in globally optimal trajectories.

In Paper E "Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments", we develop a high-level global trajectory planner, for planning energy-optimal trajectories for ASVs under the influence of external disturbances, subject to polygonal environment constraints. The method is hybrid planning approach, which finds the optimal trajectory by solving an OCP, i.e. a complete path methods, which is warm-started by the solution of a hybrid A* search algorithm (See Figure 3.1). Similarly to the planning method in [58, 163], we use a set of motion primi-

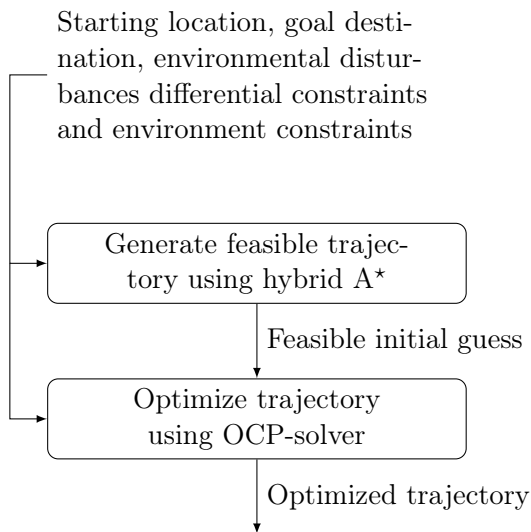


Figure 3.1: Block diagram of the high-level global trajectory planner functionality in Paper E "Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments".

tives in order to compute an initial feasible trajectory guess, before optimizing the final trajectory. Our method however, improves upon this by employing a search heuristic, which allows for the addition of external disturbances such as wind. Additionally, we use an alternative method to calculate the convex envelopes in preparation for the trajectory optimization stage, and we propose an alternative obstacle representation, which scales more efficiently with the number of polygons and polygon edges in the environment constraints, in terms of the number of optimization variables.

In Paper G "Optimal Model-Based Trajectory Planning With Static Polygonal Constraints" we consider the problem of optimal motion planning for a particle-like vehicle, moving on a 2D surface with polygonal obstacles. To this end, we introduce a hybrid method, which combines graph search on a pre-computed constrained Delaunay triangulation (CDT), with convex optimization for path refinement (see Figure 3.2). The proposed method allows for planning a globally optimal trajectory for a dynamical system subject to static polygonal constraints. The main contributions in this paper is how we combine hybrid planning with polygonal constraints and triangulation based spatial discretization. Contrary to other hybrid methods such as [57, 58, 164], where initial trajectories are planned using motion primitives and state space

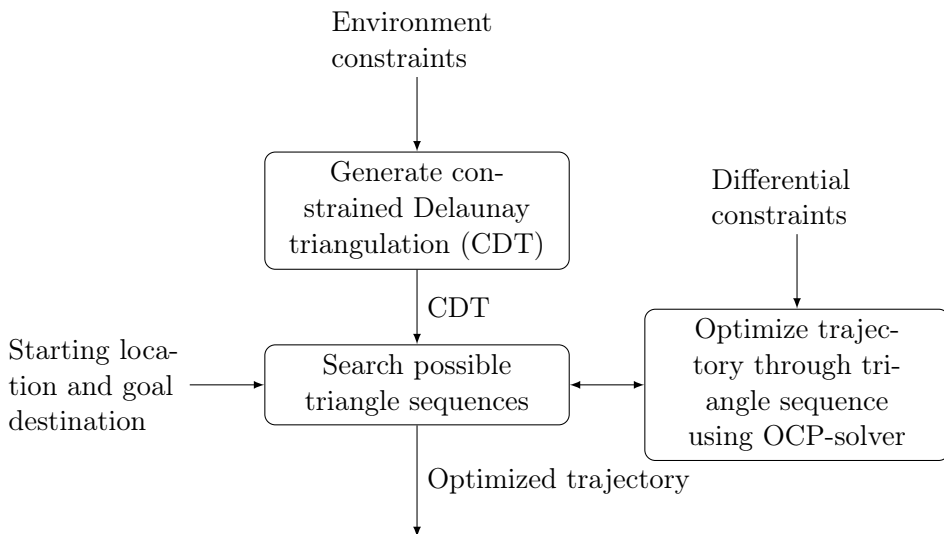


Figure 3.2: Block diagram of the high-level global trajectory planner functionality in Paper G "Optimal Model-Based Trajectory Planning With Static Polygonal Constraints".

discretizations, and refined using numerical optimization, our method employs an iterative approach of searching various triangle sequences, and refinement by optimizing the trajectory through the sequence of triangles. Polygonal constraints allow for complex constraints to be used in the planning algorithm. Very few optimization-based planning methods exist that are able to handle these types of constraints. Existing methods often lead to computationally expensive mixed integer optimization problems [165], rely on using inner approximations of the free space [17, 166], or non-convex elliptical approximations [31]. Our method relies on using a triangulation of the environment, similar to [34, 167] but instead of straight-line paths, it optimizes the path as a polynomial spline, similar to [168]. Combining these concepts, the proposed method is able to efficiently plan globally optimal trajectories for a dynamical system subject to static polygonal constraints.

3.2 Docking and berthing of ASVs

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. Our main contribution to this field, is a

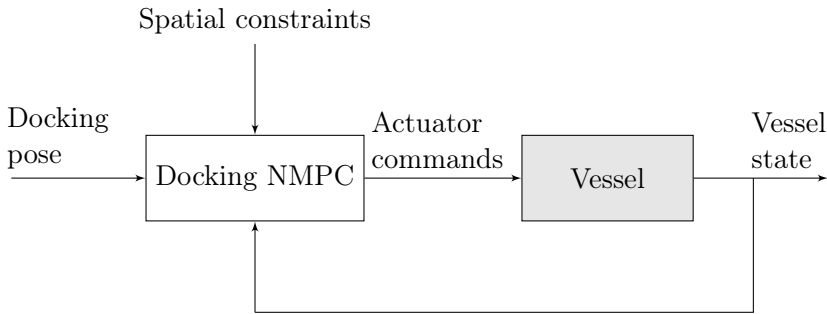


Figure 3.3: Block diagram of the mid-level docking planner in Paper A "Autonomous docking using direct optimal control".

novel approach for formulating the docking problem as an OCP, which can be used as a mid-level local trajectory planner in order to find an optimal collision free docking trajectory. Due to the complexity of performing docking, most of the existing methods rely on simplifying the docking problem, making the approaches unsuitable for real world use. We address this by proposing a state of the art method which considers both the differential constraints arising from the vessel dynamics, and the environment constraints given by a map of the harbor layout, range data from exteroceptive sensors, and the vessel geometry. Additionally, we provide full-scale experiments on the experimental autonomous urban passenger ferry *milliAmpere*, seen in Figure 1.2, showing that the proposed method is suitable for real world use.

In Paper A "Autonomous docking using direct optimal control", we present a method for framing the problem of autonomous docking, by formulating an OCP 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. In order to execute the trajectory, the problem is formulated as a nonlinear model predictive control (NMPC) problem, where the OCP is solved iteratively, with the first control action applied to a vessel simulation at each time step, illustrated in Figure 3.3. Inspired by the MPC-based DP approaches in [134, 169], the docking problem is formulated as a NMPC, with the addition of robust singularity avoiding control allocation [170], for an overactuated model supply vessel. In order to ensure that the planned docking maneuvers are safe and collision free, a novel approach for adding spatial constraints is proposed. This approach ensures the vessel stays within a safe convex region, in a way that also accounts for the shape and size of the vessel.

Contributions

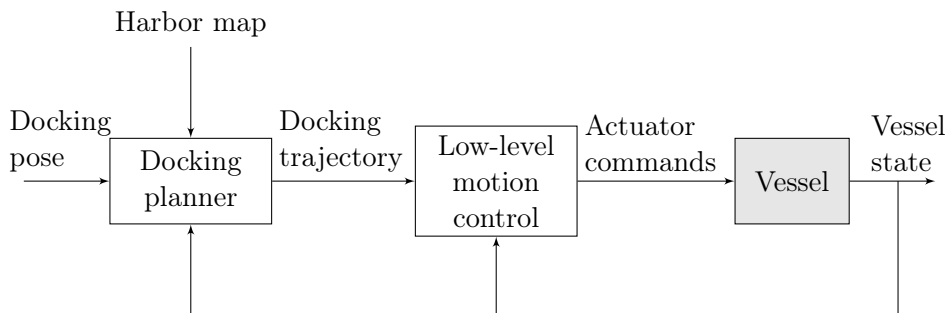


Figure 3.4: Block diagram of the mid-level docking planner in Paper D "Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments".

In Paper D "Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments", we build on the docking method proposed in Paper A in order to make the docking planner possible to run in real time on the experimental urban passenger ferry *milliAmpere*, seen in Figure 1.2. In order to make the control system more robust to external disturbances and computation delays caused by the solving the OCP, the docking planner and motion control system are decoupled, as illustrated in Figure 3.4. Additionally, slack variables are added to deal with feasibility issues that can arise when running real-world experiments, and the cost function is changed to give more desirable docking trajectories. The last addition is an algorithm for dynamically updating the convex spatial constraints, based on the position of the vessel, and a map of the harbor. By making these modifications, we show that the proposed docking planner is able to plan successful collision-free docking maneuvers in full-scale experiments on the experimental autonomous urban passenger ferry *milliAmpere*.

In Paper F "Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments", we further developed the docking planner from from Paper A and Paper D. The main contribution from the previous papers, is the addition of ranging data from on board exteroceptive sensors (see Figure 3.5), such as light detection and ranging (LIDAR) point clouds, and ultrasonic distance sensors. Including these additional exteroceptive sensors, we show how the vessel is able to plan and perform docking maneuvers in a harbor area, without the need for land-based sensor systems or manually updating the harbor map, even if the harbor layout changes. In this paper, we also provide additional improvements to the docking planner

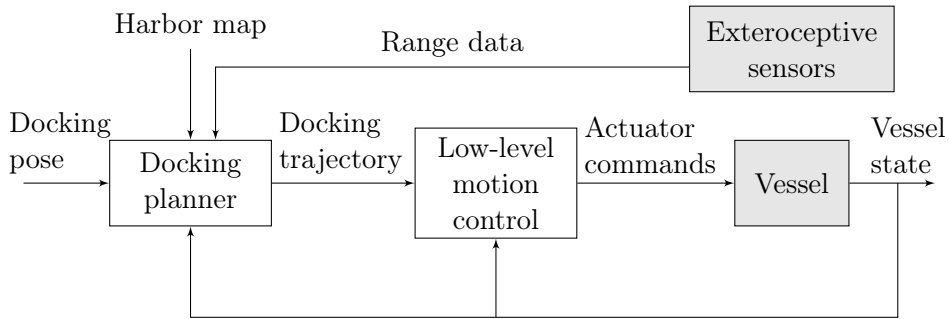


Figure 3.5: Block diagram of the mid-level docking planner in Paper F "Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments".

cost function in order to get improved docking trajectories. Additionally we provide details on the algorithm for dynamically creating the convex spatial constraints, in a way that combines known map data, together with the range data from the exteroceptive sensors. In order to validate the method, we provide additional full-scale experiments on the experimental autonomous urban passenger ferry *milliAmpere*, seen in Figure 1.2.

3.3 Obstacle representation for risk assessment and collision avoidance

With increasing interest in autonomy solutions in the maritime industry, it becomes increasingly important to develop robust and efficient methods for risk assessment and collision avoidance. This is especially true for dynamic obstacles, for which accurate trajectory predictions is complicated by both measurement and behavioral uncertainties.

In Paper H "Two space-time obstacle representations based on ellipsoids and polytopes", we develop a novel method for generating a space-time obstacle representation, which accounts for uncertainty in both measurements, as well as the future behaviour of the obstacle. The method relies on projecting the area occupied by the obstacle forward in time, using a set of velocities representing the possible maneuvers that the obstacle may take. Additionally, we show how the proposed space-time obstacle can be efficiently implemented both as convex polytopes and ellipsoids, which can be used for both risk assessment

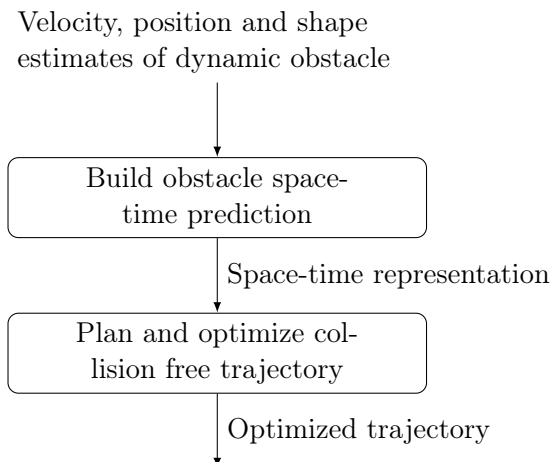


Figure 3.6: Block diagram of the space-time obstacle prediction and mid-level local trajectory planner in Paper H "Two space-time obstacle representations based on ellipsoids and polytopes".

and collision avoidance. In order to show the flexibility of the method, we provide several examples of how the proposed space-time representation can be used in mid-level local trajectory planners, in order to plan optimal collision free trajectories for a surface vessels.

3.4 Reinforcement learning-based motion control

Control of marine vehicles is a challenging problem, mostly due to the unpredictable nature of the sea and the difficulty in developing accurate mathematical models to represent the varying marine vehicle dynamics. As discussed in section 2.3, this has lead to a wide variety of methods, utilizing virtually every branch of control engineering.

One promising class of method for performing motion control for marine vessels, is RL, which has seen a resurgence in interest over the past few years, motivated by breakthroughs in deep reinforcement learning (DRL). Compared to conventional methods, RL has several advantages. Similarly to adaptive control, RL is a learning-based control method. This means that it can be used to learn how to control the vessel without the need for modeling the complex vessel dynamics. RL is also an optimization based method, this means the not only can RL be used to learn how to perform motion control, but it

3.4. Reinforcement learning-based motion control

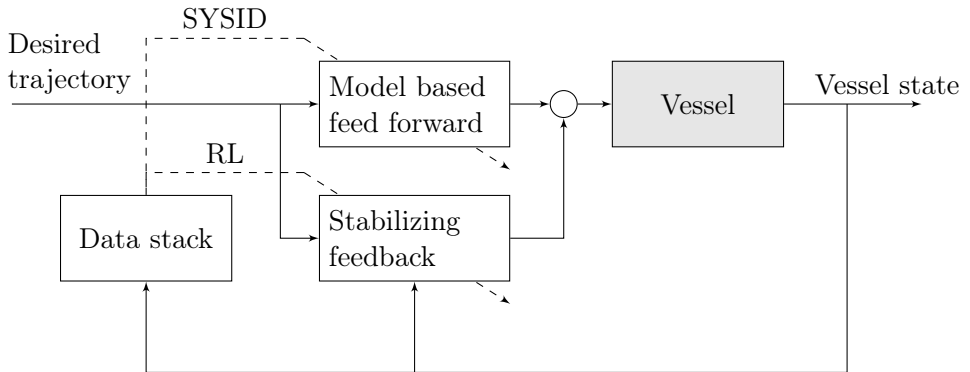


Figure 3.7: Block diagram of the motion control structure in Paper B "Reinforcement learning-based tracking control of USVs in varying operational conditions".

can learn to perform motion control in a way that optimizes the closed loop performance.

In Paper B "Reinforcement learning-based tracking control of USVs in varying operational conditions", we present a RL based motion control system for trajectory tracking control of fully-actuated surface vessels. The approach is based on approximate dynamic programming (ADP), which uses RL to optimize a stabilizing nonlinear control law. Our paper extend the work by Kamalapurkar et al. [135, 171] in order to build a trajectory tracking control system for a fully-actuated USV. Conceptually, the approach is quite similar to DP, but extends to higher velocity operational domains, while also trying to optimize tracking performance and compensate for environmental forces. The proposed method combines elements from RL, Lyapunov stability theory and SYSID, in order to learn a stabilizing feedback control law and a model based feedforward control law, as illustrated in Figure 3.7. In addition to validating the proposed control scheme in simulations, the method was also implemented on the *ReVolt* test platform (see Figure 1.1), allowing for experimental validation.

In Paper C "Combining system identification with reinforcement learning-based MPC", we propose a novel method for combining MPC, RL and SYSID, in order to optimize the closed loop performance of a MPC based control scheme. Inspired by the work of Gros and Zanon [172, 173], where RL and MPC are combined by allowing RL to use MPC as a function approximator. Our paper further extends this by using SYSID in order to aid the RL by

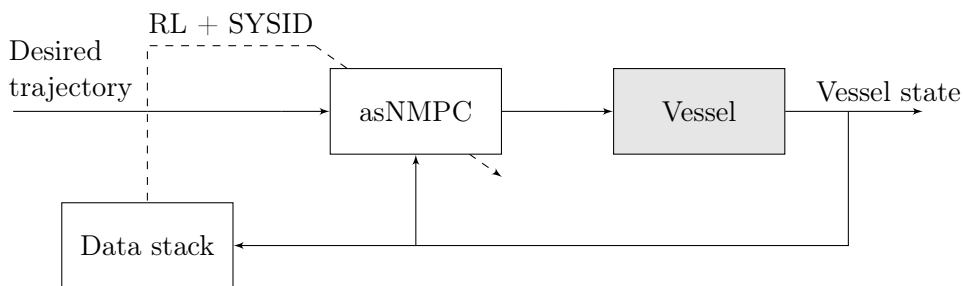


Figure 3.8: Block diagram of the motion control structure in Paper I "Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments".

improving the the accuracy of the MPC model. The paper explores a number of methods for combing the RL and SYSID objectives, in order to minimize the plant model mismatch while not affecting the closed loop performance of the MPC.

In Paper I "Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments", we propose a model based RL approach for trajectory tracking of surface vessels, illustrated in Figure 3.8. The approach builds on the work in Paper C, and extends it use a NMPC in order to perform the trajectory tracking in combination with control allocation. In order to optimize performance, the NMPC and model parameters are updated using RL and SYSID. This allows the proposed method to compensate for model mismatch and environmental forces, with a focus on optimizing the closed loop performance of the trajectory tracking controller, rather than simply fitting the MPC model to the real system dynamics. In order to run the proposed control scheme in real-time, we implemented it using advanced-step nonlinear model predictive control (asNMPC). Additionally, simulations were performed on the USV *ReVolt* (see Figure 1.1), and both simulations as well as sea trials were performed on the autonomous urban passengers ferry *milliAmpere* (see Figure 1.2).

4 | Discussion

In this chapter we will conclude the thesis by summing up some of its main contributions. Additionally, we will include some reflections around the work, and finally look at the current state of autonomous marine operations, and suggest some future research directions in terms of the topics discussed in this thesis.

4.1 Conclusion

This thesis contains contributions with novel solutions to the problems of high-level global trajectory planning, mid-level local trajectory planning, including docking and COLAV, and low-level motion control for ASVs. The contributions are centered around optimization-based methods, with the aim of developing safer, more efficient, and more robust algorithms for enabling autonomous marine operations.

The high-level global trajectory planning methods in this thesis were designed with the purpose of planning optimal trajectories for ASVs in static environments. This means that the methods needed to be model-based and consider constraints and dynamics imposed by the vessel, in order to allow the planner to find the optimal trajectory. In terms of static constraints, the proposed methods rely on a polygon representation of geographical data. This type of representation has several advantages, such as facilitating easy import of existing map data, typically represented as polygons, and allowing to use exact high resolution geographical data, which results in more optimal trajectories compared with other common map representations. The polygon representation does however come with some drawbacks, with the main issues being the non-convex nature of general polygons, and how to represent them as constraints in an optimization problem. In our case, we solve this by uti-

Discussion

lizing different hybrid planning approaches, that combine roadmap methods with optimization-based complete path methods. The two approaches that we present in this thesis build on similar principals, with the main difference being the underlying discretization. In Paper E discretization is performed using motion primitives. This approach is less computationally demanding than the spatial discretization in Paper G, however using the spatial discretization offers better guarantees in terms of global optimality.

A high-level global trajectory planner that only considers static obstacles is insufficient when unmapped obstacles and dynamic obstacles, such as other vessels, are present. This motivates the need for a mid-level local trajectory planner, tasked with re-planning the trajectory in order to account for the additional obstacles.

In terms of mid-level local trajectory planning, our main focus has been on the problem of docking and berthing. The progress on this problem has been iterative, where our first contribution to the problem was a novel approach to handling the vessel and harbor geometry that allowed us to develop a NMPC-based docking controller. This approach relies on using a convex outer approximation of the vessel geometry, and a convex inner approximation of the harbor, in order to ensure the maneuvers planned by the NMPC are collision free. With our second publication on the topic, we built on the approach by proposing a method for automatically generating the spatial constraints as convex inner approximations of the harbor layout, based on map data of the harbor. Additionally, we proposed separating the low-level control and the model-based docking planner. This has several advantages, including better steady-state disturbances rejection, faster feedback control, and improved robustness to failures in the NMPC planner, allowing for the method to be implemented on a physical platform. In the third and final paper we include exteroceptive sensor data, in order to account for unmapped obstacles. This resulted in a fully autonomous docking system, capable of planning and executing the necessary maneuvers for safely docking an ASV. To the author's best knowledge, this is the most comprehensive treatment of the docking and berthing problem for ASVs in the current literature.

In addition to the docking problem, we also look at the COLAV problem for dynamic obstacles, which also falls under the mid-level local trajectory planning. Our main contribution is a space-time obstacle representation, which accounts for uncertainty in both measurements, as well as the future behaviour of the obstacle. The method works by projecting the area occupied by the obstacle forward in time, in order to find the space-time volume which the obstacle can occupy. Additionally, we show how the proposed obstacle representation can

be used for both reactive as well as optimization-based deliberate COLAV.

In order to accurately follow the planned trajectories and compensate for disturbances, low-level motion control is needed. In this thesis, the main focus in terms of motion control has been on model-based RL approaches, allowing for optimizing the closed loop performance of the motion control system through evaluative feedback. The first publication on the topic is based on ADP, where the parameters of a model-based feedforward controller and a stabilizing feedback controller are learned from data collected online. The second approach uses a parametric NMPC, combined with SYSID and RL to learn model parameters online. Using both simulations and experiments, we show that adding a learning component can improve the control performance over conventional motion control methods.

While planning, COLAV and motion control are all topics that have seen significant interest over the years, it is only recently that the push for autonomy has seen a need for developing systems which incorporate all these elements in a robust and efficient way. This is especially true when developing methods which work not only in simulations, but also work experimentally. One of the main contributions of this thesis is the inclusion of experimental results for a number of the proposed methods. These results demonstrate the feasibility of the methods in practice, and highlight some important aspects to consider when designing and implementing the methods on physical platforms.

4.2 Reflections

While our work has focused on optimization-based trajectory planning and control for ASVs, the topics that have been discussed have given the work a broad scope which encompasses several fields, including optimization, control theory, ML, AI, planning, COLAV, situational awareness, and more. This has led to a number of insights and reflections which deserve their own discussion.

When starting the work on this thesis, I had recently finished my masters thesis on using DRL for end-to-end path following and control of marine vehicles [153–155]. While the end-to-end approach was interesting from a research perspective, the approach would typically involve hours of training on a simulator, equating to weeks and months of real world training, limiting the practicality of the method. An other major drawback of using DRL for an end-to-end approach, was certifying the stability and safety of the closed loop control, as the resulting control structure was typically too complex to analyze with classical

methods. These problems were my main inspiration to research model-based RL approaches, where we impose a model and control structure into the RL framework. The benefits of this is that training can typically be done much faster, as the search space is limited, and the closed loop stability can be investigated using known methods. On the other hand, this restricts the model and controller, however this is arguably a small price to pay, when considering that these methods can be used not only in simulations, but in real-world experiments, as we demonstrate in Paper B and Paper I.

One of the main benefits of using RL-based methods for control, is the ability to make self-optimizing closed-loop controllers, which can learn how to best control a plant without the need for comprehensive model identification and controller tuning. Unfortunately, it is not as simple as just replacing a classical control scheme with a RL-based method. Typically, it is still necessary to model the plant as it is used for training purposes in model-free RL approaches, and integrated into the control scheme for model-based RL approaches. Learning, which is one of the key ideas that make RL such a powerful tool, is also one of its biggest weaknesses. Since learning in general requires a trial and error procedure in order to explore and improve, additional safety measures need to be put into place when learning is performed on-line. The learning process can also be sensitive to measurement noise and external disturbances, further complicating the learning process. The reward or cost function, which is also one of the main strengths of RL, can also be considered one of its drawbacks. Depending on the task we want to perform, designing a reward or cost function to achieve the desired goal, can be quite difficult and require multiple iterations of careful hand engineering to get right. In many applications, applying RL can significantly improve the closed loop performance of a control problem. However, applying RL in a way that is robust and efficient, is typically time consuming and adds significant complexity to the controller design, when compared with classic control methods.

While the work on this thesis was initially going to be mostly theoretical and simulation based, the availability of different experimental platforms enabled us to also perform experimental validation. Before starting the work on this thesis I had no prior experience performing experimental work. Having experienced the limited practicality of the methods from the work on my masters thesis, one of my goals was to develop methods which were not limited to simulations, but could also be used in practice. While developing the RL-based approach in Paper B, we were given access to a high fidelity simulator of the *ReVolt* platform by DNV (then DNV GL). After being able to show that the method worked in simulations, I was eager to see if the method would be able

to work on the physical platform. Thinking the implementation would be fairly straightforward, I quickly realized that that was not the case, running into numerous errors in the existing codebase, hardware failures, a leaking hull and poor weather conditions. Later on, similar problems also arose when working with the *milliAmpere* platform, causing me to spend a lot of time on issues not necessarily directly related to the methods we were testing. While this work was quite frustrating at times, it was definitely worth it when we were able to run the planned experiments, with results similar to those of the simulations. This also gave me a huge appreciation for experimental work, seeing the additional work needed to get things running, as well as the additional robustness requirements needed for the control methods to work with the additional process and measurement noise.

For our work on both high-level and mid-level planning, our main focus was on optimization-based complete path methods. These methods have the benefit of being able to plan optimal trajectories, which for ASVs can be especially useful for energy optimization. The optimal trajectories are however only as accurate as the models they are based on, this means that for a practical application, with plant model mismatch and unmodeled environmental forces, the planned optimal trajectory will no longer be the true optimal trajectory. In the case of energy optimization for ASVs, trying to follow the planned optimal trajectory may in fact require significantly more energy to accurately track, due to the mismatch. This begs the question of how much benefit, if any, there is to planning an optimal trajectory as opposed to a suboptimal trajectory. In combination with the added complexity and computational demand of the optimization-based complete path methods, certain problems may be better solved using simpler combinatorial methods. This may be especially true for COLAV, where it is crucial to be able to quickly act on changes in the environment in order to avoid collisions, and the optimality of the maneuvers may not be as important. In the existing literature, this problem is often dealt with by having a two-layered approach to the mid-level planning, where reactive methods are used to react to immediate dangers, without any significant planning, while deliberate methods are used to plan over a longer horizon, when there is no immediate danger. In this approach, relatively slow optimization-based complete path methods may be better suited for deliberate planning than for reactive planning.

Our work contributes towards the vision of fully autonomous vessels. We never combined the contributions into a fully autonomous system, while our methods could theoretically be combined using the control hierarchy in Figure 2.1, as shown in [28], to form a complete automatic transit and docking system. We

are still far from a fully autonomous system able to handle complex human machine interactions, condition monitoring, situational awareness, and reacting to a plethora of unforeseen events. Luckily, these are all active areas of research, and we will likely see integration taking place in the near future.

4.3 Future work

In terms of high-level global planning for ASVs, there is still much work that can be done. Furthering our proposed approaches, there is still a question of computational demand. For optimization-based methods, the number of decision variables will tend to grow linearly with the length of the path, something that may make the methods computationally infeasible for planning long trajectories. An other area of of improvement is to incorporate better models of the vessel energy consumption, this may include hotel loads, as well as incorporating the true energy expenditure of the propulsion system, including various losses. Our proposed approaches are hybrid planning approaches, and combine the benefits of both roadmap methods and complete path methods, which is still a relatively new area of research and deserves more attention in future work.

For docking and berthing, our proposed approach is based on formulating the problem as an optimization problem. Formulating the optimization objective function and constraints in order to get "sensible" behaviour, is in general a difficult problem. To this end, setting up additional criteria in terms of the docking maneuvers, and formulating these as an objective function and constraints that can be used in the optimization problem, is an area of research where more work can be done. An other area to explore is the use of other methods then optimization for solving the planning and docking problem. While other approaches based on ANNs, fuzzy control, artificial potential fields and rule based expert systems exist, they do not solve the problem as comprehensively, and do not include experimental validation. Using some of the insights provided in our research, exploring the possibility of incorporating other planning approaches, including roadmap methods and hybrid methods, is an other avenue which can be explored. In real world docking and berthing, two of the main reasons for collisions, is the the close proximity to dynamic obstacles, such as other vessels, and correctly accounting for difficult environmental conditions. These are also two important areas which should be addressed in future research.

For COLAV, our proposed space-time obstacle representation is promising,

however it still remains to be seen how it performs in practice. It would also be interesting to look at the possibility of extending the proposed method to allow for time-varying velocity uncertainties, and thus make the method more general. Additionally, using the space-time obstacle representation together with other trajectory planning and collision avoidance methods would also be interesting, as well as further studying how to best represent the initial obstacle shapes in order to promote COLREGs compliance.

In the context of learning-based motion control, it would be interesting to look more into how to perform more robust and safe parameter updates. This is important as care must be taken in order to avoid problematic parameter updates caused by for example noisy and inaccurate measurements. This is mostly a problem when running on a physical platform, and in our case it was solved using batches of transitions, and sufficiently small learning rate. Additionally, considering different vessel models, including under-actuated vessels may be interesting. This may for the most part be applicable for MPC-based approaches, but is still an area which deserves more attention.

For optimization-based approaches in a hierarchical control structure such as the one suggested in this thesis (see Figure 2.1), the true performance in terms of optimality is highly dependant on how the different layers of the control hierarchy interact. In general this could mean that, while the planned trajectory is optimal with respect to an objective, the actual performance may be degraded by a motion control system struggling to accurately follow the desired trajectory. In this thesis we have for the most part looked at the different hierarchical layers separately, without considering the coupling between layers. Studying the control structure as a whole when designing and optimizing the performance is an area which has received little attention, but should be considered in future works.

While we are getting closer to fully autonomous marine operations, there are still a number of hurdles to overcome. Today, technology has progressed to the point where there are commercially available systems that have proven themselves capable of performing marine operations such as ferry crossings without the need for human intervention. However the systems are mostly considered automatic, as they are still being monitored and supervised by crew. In order to get to a point where vessels are fully autonomous, we need to continue improving everything from planning, COLAV, motion control, navigation, sensor systems, situational awareness, condition monitoring and human machine interactions. This is key in order to get to a point where the systems are robust and redundant enough that that we can overcome the final hurdle of building trust in them.

5 | Publications

This chapter contains the original publications which were written as a result of the work on this thesis. The articles are reprints of the original publications, formatted to fit the thesis. The work is ordered chronologically by date of publication, however the recommended (thematic) reading order, illustrated in Figure 1.3, is high-level planning (Paper G and Paper E), mid-level planning (Paper A, Paper D, Paper F and Paper H), and finally low-level motion control (Paper B, Paper C and Paper I).

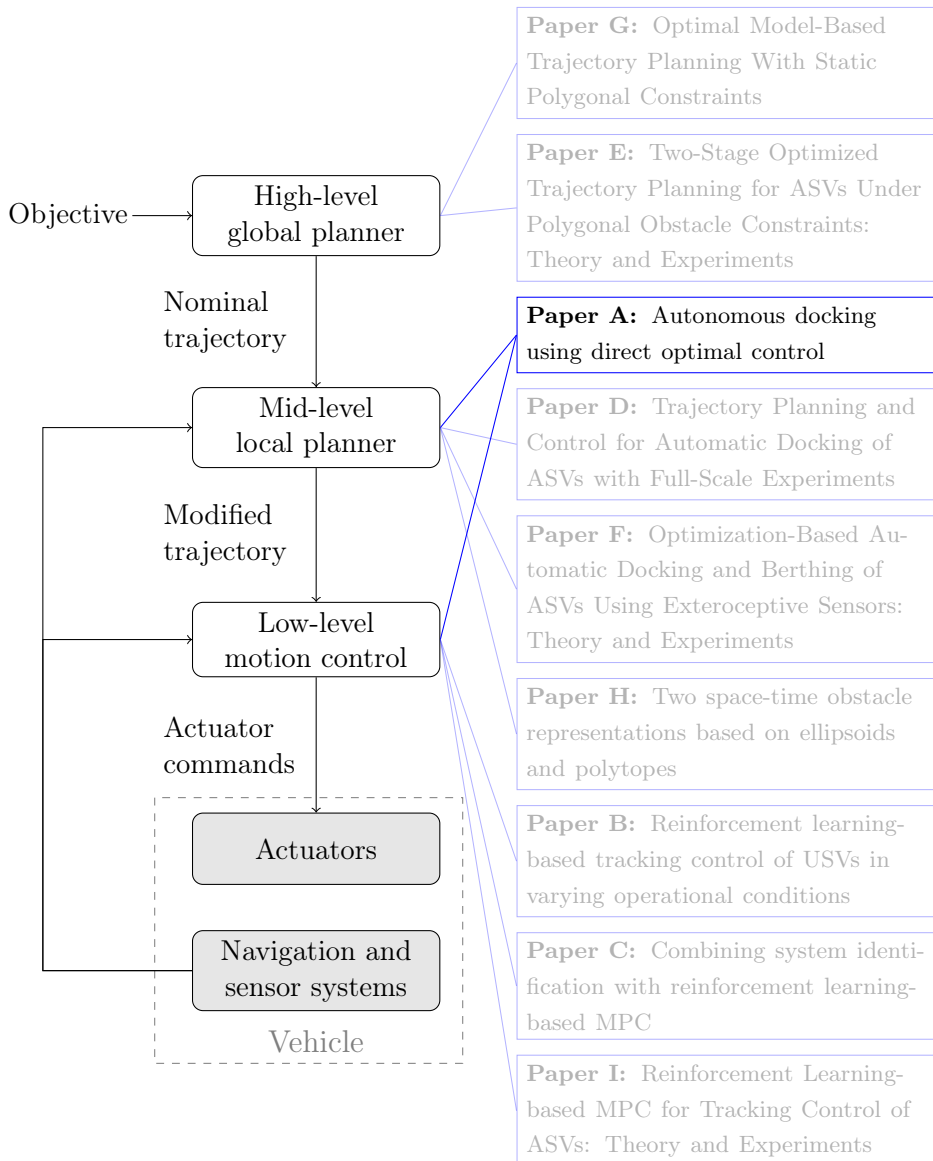
Publications

A	Autonomous docking using direct optimal control .	43
B	Reinforcement learning-based tracking control of USVs in varying operational conditions	61
C	Combining system identification with reinforcement learning-based MPC	95
D	Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments . . .	113
E	Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments	133
F	Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments	171
G	Optimal Model-Based Trajectory Planning With Static Polygonal Constraints	203
H	Two space-time obstacle representations based on ellipsoids and polytopes	235
I	Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments	259

A Autonomous docking using direct optimal control

Postprint of [17] **Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Autonomous docking using direct optimal control”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 97–102. DOI: 10.1016/j.ifacol.2019.12.290

©2019 IFAC-PapersOnLine. Reprinted and formatted to fit the thesis with permission from Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros.



Autonomous docking using direct optimal control

Andreas B. Martinsen¹, Anastasios M. Lekkas¹, and Sebastien Gros¹

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Abstract: We propose a method for performing autonomous docking of marine vessels using numerical optimal control. The task is framed as a dynamic positioning problem, with the addition of spatial constraints that ensure collision avoidance. The proposed method is an all-encompassing procedure for performing both docking, maneuvering, dynamic positioning and control allocation. In addition, we show that the method can be implemented as a real-time MPC-based algorithm on simulation results of a supply vessel.

Keywords: Docking, Optimal Control, Autonomous vehicles, Numerical Optimization, Path planning

1 Introduction

For most larger vessels, docking has historically been performed by utilizing external help from support vessels such as tug boats. The main reasons for this has been limits in terms of maneuverability as well as limits in the accuracy of the human operators when dealing with relatively slow dynamical systems. With the increasing usage of azimuth thrusters, marine vessels have become increasingly maneuverable. In addition to this, interest in autonomous ferries, and cargo vessels has increased in recent years. Despite this, and contrary to topics such as path following/tracking and control allocation, research on autonomous docking for surface vessels has seen little attention. While there are some methods such as [1-3] developed for Autonomous Underwater Vehicles (AUVs), which use fuzzy control schemes for different stages of the docking process. While [4] and [5] have developed methods for Unmanned Surface Vehicles (USVs) based on target tracking and artificial potential fields respectively. These existing approaches are usually quite limited, do not take into account the underlying vessel model, and make few guarantees in terms of safety.

In this paper, we present a method for framing the problem of autonomous docking as a optimal control problem. Our proposed method is similar to methods used for dynamic positioning [6, 7], with the addition of control allocation optimization [8], and spatial constraint, which ensure the vessel operates safely without colliding.

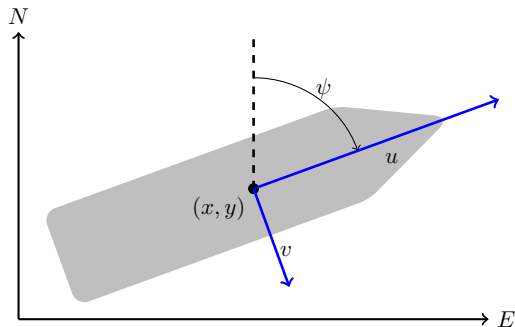


Figure 1: 3-DOF vessel centered at (x, y) , with surge velocity u , sway velocity v , heading ψ in a North-East-Down (NED) reference frame.

2 Vessel Model

2.1 Kinematics

When modeling vessels for the purpose of autonomous docking, we assume the vessel moves on the ocean surface at relatively low velocities. In addition to this we assume that effects of the roll and pitch motions of the vessel are negligible, and hence have little impact on the surge, sway and yaw of the vessel. The mathematical model used to describe the system can then be kept reasonably simple by limiting it to the planar position and orientation of the vessel. The motion of a surface vessel can be represented by the pose vector $\boldsymbol{\eta} = [x, y, \psi]^T \in \mathbb{R}^2 \times \mathbb{S}$, and velocity vector $\boldsymbol{\nu} = [u, v, r]^T \in \mathbb{R}^3$. Here, (x, y) 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 1. Using the notation in [9] we can describe a 3-DOF vessel model as follows

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\psi)\boldsymbol{\nu}, \quad (1)$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \quad (2)$$

where $\mathbf{M} \in \mathbb{R}^{3 \times 3}$, $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\boldsymbol{\tau}$ and $\mathbf{J}(\psi) \in SO(3)$ are the inertia matrix, dampening matrix, control input vector, and rotation matrix respectively. The rotational matrix $\mathbf{J}(\psi) \in SO(3)$ is given by

$$\mathbf{J}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

and is the rotation from the body frame to the earth-fixed reference frame.

2.2 Thrust configuration

The control surfaces of the vessel are specified by the thrust configuration matrix $\mathbf{T}(\boldsymbol{\alpha}) \in \mathbb{R}^{3, n_{thrusters}}$ which maps the thrust \mathbf{f} from each thruster into the surge, sway and yaw forces and moments in the body frame of the vessel given the thruster angles $\boldsymbol{\alpha}$.

$$\boldsymbol{\tau} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{f} \quad (4)$$

Each column $\mathbf{T}_i(\alpha_i)$ in $\mathbf{T}(\boldsymbol{\alpha})$ gives the configuration of the forces and moments of a thruster i as follows:

$$\mathbf{T}_i(\boldsymbol{\alpha})\mathbf{f}_i = \begin{bmatrix} F_x \\ F_y \\ F_y l_x - F_x l_y \end{bmatrix} = \begin{bmatrix} f_i \cos(\alpha_i) \\ f_i \sin(\alpha_i) \\ f_i(l_x \sin(\alpha_i) - l_y \cos(\alpha_i)) \end{bmatrix} \quad (5)$$

where α_i is the orientation of the thruster in the body frame, and f_i is the force it produces. Selecting the orientation $\boldsymbol{\alpha}$ and force \mathbf{f} of the thrusters in order to generate the desired force $\boldsymbol{\tau}$ is called the thrust allocation problem. While there are numerous ways of solving the thrust allocation problem [10], for our purpose we want to include the thrust allocation as part of the optimization for performing the docking operations. This allows us to take into account physical thruster constraints such as force saturation and feasible azimuth sectors.

$$\begin{aligned} \alpha_{i,min} &\leq \alpha_i \leq \alpha_{i,max} \\ f_{i,min} &\leq f_i \leq f_{i,max} \end{aligned}$$

In order to avoid singular thruster configurations, we add a penalty on the rank deficiency of the thrust configuration matrix, as proposed by [8]. The singular configuration cost is given as the following.

$$\frac{\rho}{\epsilon + \det(\mathbf{T}(\boldsymbol{\alpha})\mathbf{W}^{-1}\mathbf{T}^\top(\boldsymbol{\alpha}))} \quad (6)$$

Here $\epsilon > 0$ is a small constant in order to avoid division by 0, $\rho > 0$ is the weighting of the maneuverability, and \mathbf{W} is typically diagonal matrix, weighting each individual thruster. A constraint on the singular configuration may alternatively be added, however in our implementation this is added as a cost, which means that avoiding singular thrust configurations become more important when close to the desired docking position.

It should be noted that both the singular configuration cost in (6) and the thrust configuration matrix in (4) are both highly nonlinear due to the trigonometric functions, adding them as costs and constraints in an optimization problem will therefore in general cause the problem to become non-convex.



Figure 2: Thruster configuration for vessel, where 1 and 2 are azimuth thrusters, and 3 is a tunnel thruster.

2.3 Summary of model

The model used for the simulations is based on the SV Northern Clipper from [11]. The model is a 3 Degree of Freedom (3-DOF) linear model on the form:

$$\begin{aligned}\dot{\eta} &= \mathbf{J}(\psi)\boldsymbol{\nu}, \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} &= \mathbf{T}(\alpha)\mathbf{f}\end{aligned}$$

For thruster configuration, we used two azimuth thrusters in the stern and one tunnel thruster in the bow, giving configuration seen in Figure 2. Additionally saturations were added to the force generated by the thrusters, where the maximum thrust for the azimuth thrusters and tunnel thrusters respectively were $1/30$ and $\pm 1/60$ of the dry ships weight. For the azimuth thrusters additional constraints were added, this included a maximum turnaround time of 30s per revolution, and a maximum angle of $\pm 170^\circ$ giving a 20° forbidden sector illustrated in figure 2, which ensures the thrusters do not produce thrust that directly work against each other, which may cause damage, this additionally reflects the movement of real world azimuth thrusters which have a finite turning radius. Additional details on the vessel model, and specific parameters are given in Appendix A.

3 Autonomous Docking

3.1 Obstacle avoidance

Docking of autonomous vessels is a complex problem, which includes planning and performing maneuvers to control a vessel to a desired orientation and position, while adhering to spatial constraint in order to avoid collisions. 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 to the desired pose as possible, with out the vessel going aground, or running into obstacles, i.e. adhering to spatial constraints.

In order to ensure the vessel does not collide we define a safety margin around the vessel which obstacles should not enter. Given a set \mathbb{S}_v representing the vessel, the convex hull

$$\text{Conv}(\mathbb{S}_v)$$

A. Autonomous docking using direct optimal control

gives the boundary points of the vessel which form a polyhedron around the vessel. By dilating the set representing the vessel by a desired safety margin \mathbb{M} , we get the following polyhedron representing the safety boundary surrounding the vessel.

$$\mathbb{S}_b = \text{Conv}(\mathbb{S}_v \oplus \mathbb{M}) \quad (7)$$

For our simulations we used a safety margin of 10% giving the safety boundary \mathbb{S}_b seen in Figure 3, which is a polyhedron in the body frame of the vessel consisting of five vertices.

In order to ensure safe operating conditions, we define a operating region in terms of spatial constraints \mathbb{S}_s for the vessel. The operating region is chosen as the largest convex region that encompasses the desired docking position, while not intersecting with obstacles or land. Choosing the spatial constraints and vessel boundary in this way, safe operations are ensured when $\mathbb{S}_b \subseteq \mathbb{S}_s$, i.e. the vessel with the safety margin is contained within the spatial constraints, this is illustrated in Figure 3. Using the fact that the spatial constraints are a convex polyhedron:

$$\mathbb{S}_s = \{x | \mathbf{A}_s x \leq \mathbf{b}_s\}$$

we have that the vessel is within the spatial constraints so long as all the vertices of the vessel boundary follow the linear inequality representing the spatial constraints.

$$\mathbb{S}_b \subseteq \mathbb{S}_s \iff \mathbf{A}_s \mathbf{x}_i^{NED} \leq \mathbf{b}_s \quad \forall \mathbf{x}_i^{NED} \in \text{Vertex}(\mathbb{S}_b) \quad (8)$$

Since the Vertices of the 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.

$$\mathbf{A}_s \left(\mathbf{R}(\psi) \mathbf{x}_i^b + \begin{bmatrix} x \\ y \end{bmatrix} \right) \leq \mathbf{b}_s \quad \forall \mathbf{x}_i^b \in \text{Vertex}(\mathbb{S}_b) \quad (9)$$

Where \mathbf{R} is the rotation from the body frame to NED.

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \quad (10)$$

This can directly be implemented as inequality constraints in an optimization problem, and ensures the vessel is contained within a predefined safe region.

While this constraint is easily implemented in a nonlinear programming (NLP) problem, the constraint is not convex. This means the constraint will enforce safety requirements, however the NLP may not converge to a global optimum.

3.2 Optimal control problem (OCP)

Using the model, and constraints discussed in the previous sections, with the desired docking pose $\boldsymbol{\eta}_d = [x_d, y_d, \psi_d]^\top$, we can formulate the following nonlinear continuous

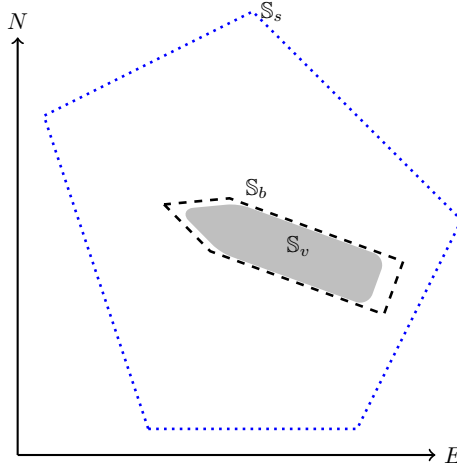


Figure 3: Vessel \mathbb{S}_v with safety boundary \mathbb{S}_b with black dashed line, and spatial constraints \mathbb{S}_s as blue dotted line, in the NED frame. The vessel will always lie within the spatial constraints \mathbb{S}_s as long as all the vertices of \mathbb{S}_b lie within the spatial constraints.

time optimal control problem.

$$J^* = \min_{\boldsymbol{\eta}, \boldsymbol{\nu}, \mathbf{f}, \boldsymbol{\alpha}} \int_0^T \left\{ \|\boldsymbol{\eta} - \boldsymbol{\eta}_d\|_{\mathbf{Q}_\eta}^2 + \|\boldsymbol{\nu}\|_{\mathbf{Q}_\nu}^2 + \|\mathbf{f}\|_{\mathbf{R}_f}^2 + \frac{\rho}{\epsilon + \det(\mathbf{T}(\boldsymbol{\alpha})\mathbf{W}^{-1}\mathbf{T}^\top(\boldsymbol{\alpha}))} \right\} dt \quad (11a)$$

subject to:

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\psi)\boldsymbol{\nu} \quad (11b)$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{f} \quad (11c)$$

$$\mathbf{A}_s \left(\mathbf{R}(\psi)\mathbf{x}_i^b + \begin{bmatrix} x \\ y \end{bmatrix} \right) \leq \mathbf{b}_s \quad \forall \mathbf{x}_i^b \in \text{Vertex}(\mathbb{S}_b) \quad (11d)$$

$$\mathbf{f}_{min} \leq \mathbf{f} \leq \mathbf{f}_{max} \quad (11e)$$

$$\boldsymbol{\alpha}_{min} \leq \boldsymbol{\alpha} \leq \boldsymbol{\alpha}_{max} \quad (11f)$$

$$|\dot{\boldsymbol{\alpha}}| \leq \dot{\boldsymbol{\alpha}}_{max} \quad (11g)$$

$$\text{Initial conditions on } \boldsymbol{\eta}, \boldsymbol{\nu}, \mathbf{f}, \boldsymbol{\alpha} \quad (11h)$$

Where we minimize cost (11a), subject to the dynamic model constraints (11b) and (11c), the spatial constraints (11d), the saturation constraint (11e), (11f) and (11g), and the initial conditions (11h) over the time horizon T . For this problem we have opted to use a simple quadratic penalty in order to ensure the vessel converges to

the desired pose, however Huber penalty functions as discussed in [12, 13] may give better performance for large pose deviations.

3.3 Implementation

In order to implement the proposed docking system we need to solve the OCP in the previous section. This can be done in multiple ways, however the two main classes of methods are sequential methods, such as direct single shooting [14], and simultaneous methods such as direct multiple shooting [15], and direct collocation [16]. For this approach we chose to use direct collocation, in where implicit numerical integration of the ODE constraints (11b) and (11c), as well as the objective function (11a), is performed as part of the nonlinear optimization. In the collocation method, the numerical integration is performed by fitting the derivatives to a degree d Legendre polynomial, with known integral, within N set time intervals called shooting intervals. The shooting intervals are then connected to create the full time horizon, by enforcing constraints on the shooting gaps between intervals.

For this problem we opted to use direct collocation for several reasons. Comparing direct collocation with multiple shooting, they both offer the same stability in terms of the optimization, however direct collocation offers a speedup, as the numerical integration is performed as part of the optimization, and not offloaded to a separate integration routine, giving the optimization problem a nice sparsity structure. While multiple shooting offers more flexibility in terms of the integrator used, the implicit integrator of the direct collocation is sufficient for our purpose. Comparing single shooting to direct collocation the single shooting problem has much fewer decision variables, however the problem often becomes very dense, and hence increases the computation time, single shooting is also more unstable, as propagating the gradients through a long time horizon often cause them to become very small (vanish) or very large (explode), and hence the optimization steps may be oscillatory and unstable.

For the implementation we used CasADi [17] a software framework for easy implementation of nonlinear optimization and optimal control problems, with IPOPT [18] an interior point optimizer, for solving the resulting NLP.

Solving the OCP once, gives a open loop trajectory over a time horizon T , which can be used to perform open loop control, or trajectory tracking. We however wish to use the OCP as the basis for a Nonlinear Model Predictive Control (NLMPC). Where at each time step the OCP is solved with the vessel state as initial conditions, and then only the first predicted control action is performed. This gives a closed loop control scheme, which makes the method more robust to modeling errors, and external disturbances due to the feedback.

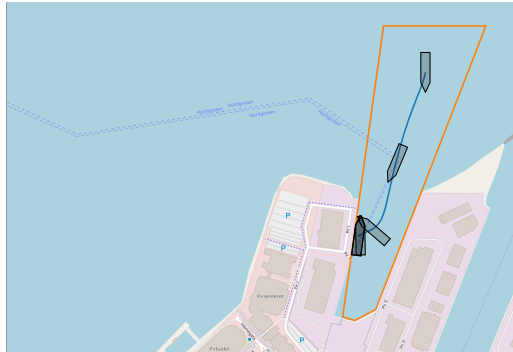


Figure 4: Vessel docking performed at Hurtigruten terminal in Trondheim Norway.

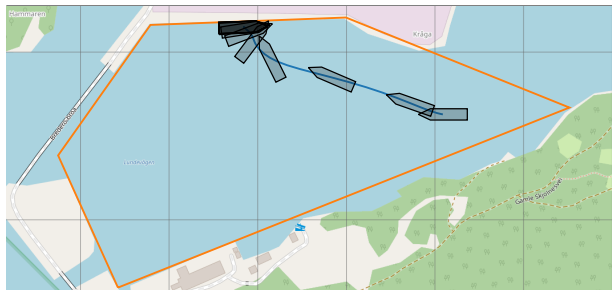


Figure 5: Vessel docking performed at Lundevågen harbour in Farsund Norway.

4 Simulation

As a proof of concept, simulations were performed, where the OCP was run as a closed loop Nonlinear Model Predictive Control (NLMPCC). For the OCP we used a time horizon of $T = 300$ seconds, with $N = 30$ time steps, making each time step $T/N = 10$ seconds. Using this we performed docking simulations at two different locations, namely Trondheim harbour and Lundevågen harbour, as seen in Figure 4 and 5 respectively. For the docking at Lundevågen harbour, the vessel state and control inputs are shown in Figure 6, 7 and 8, and for the docking at Trondheim harbour, the vessel state and control inputs are shown in Figure 9, 10 and 11. From the simulations we see an expected behaviour, where the vessel will turn and face the bow in the direction of travel, as this is the most efficient way of traveling. As the vessel closes in on the target position, it will start initiating the turn such that it faces in the desired heading, while simultaneously adhering to the defined spatial constraints in order to avoid colliding.

A. Autonomous docking using direct optimal control

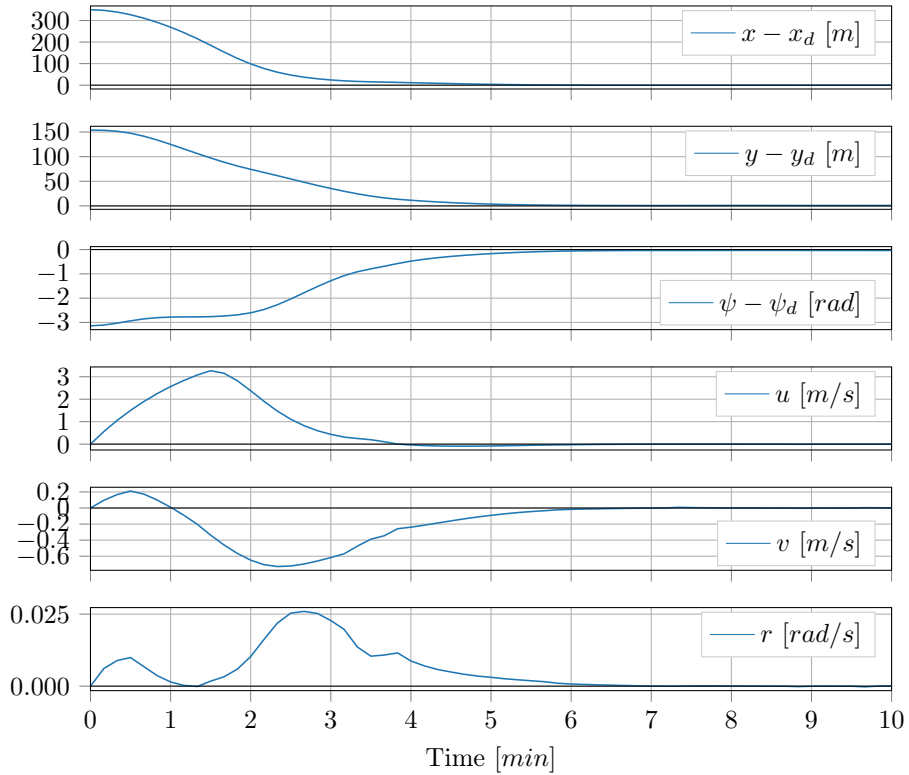


Figure 6: Vessel pose error $\boldsymbol{\eta} - \boldsymbol{\eta}_d$ and velocity $\boldsymbol{\nu}$ when docking at Lundeavågen harbour.

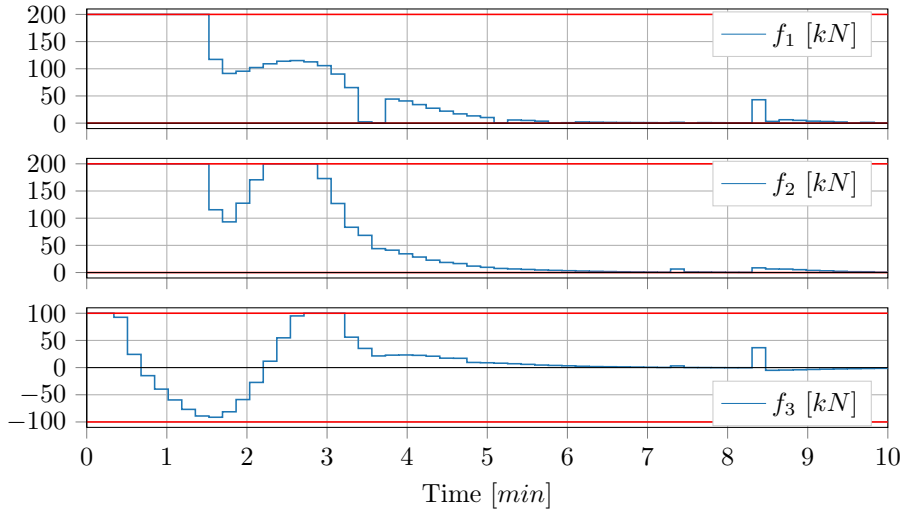


Figure 7: Thruster force for docking at Lundevågen harbour, with saturation constraints indicated in red.

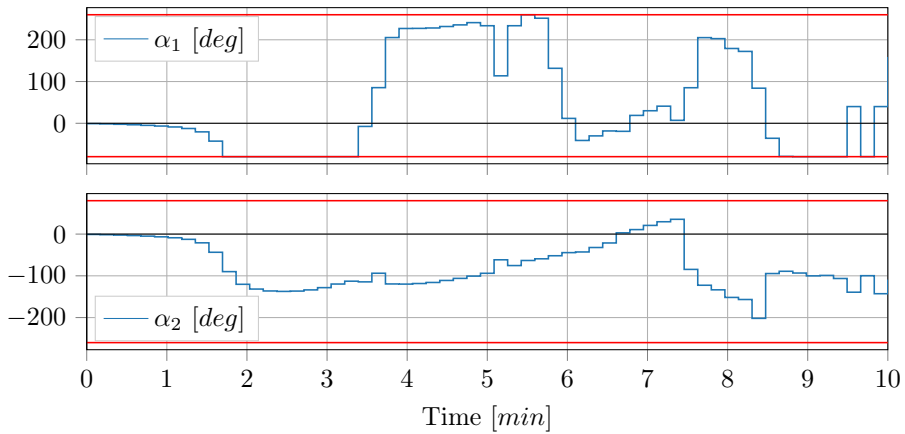


Figure 8: Azimuth angles when docking at Lundevågen harbour, with saturation constraints indicated in red.

A. Autonomous docking using direct optimal control

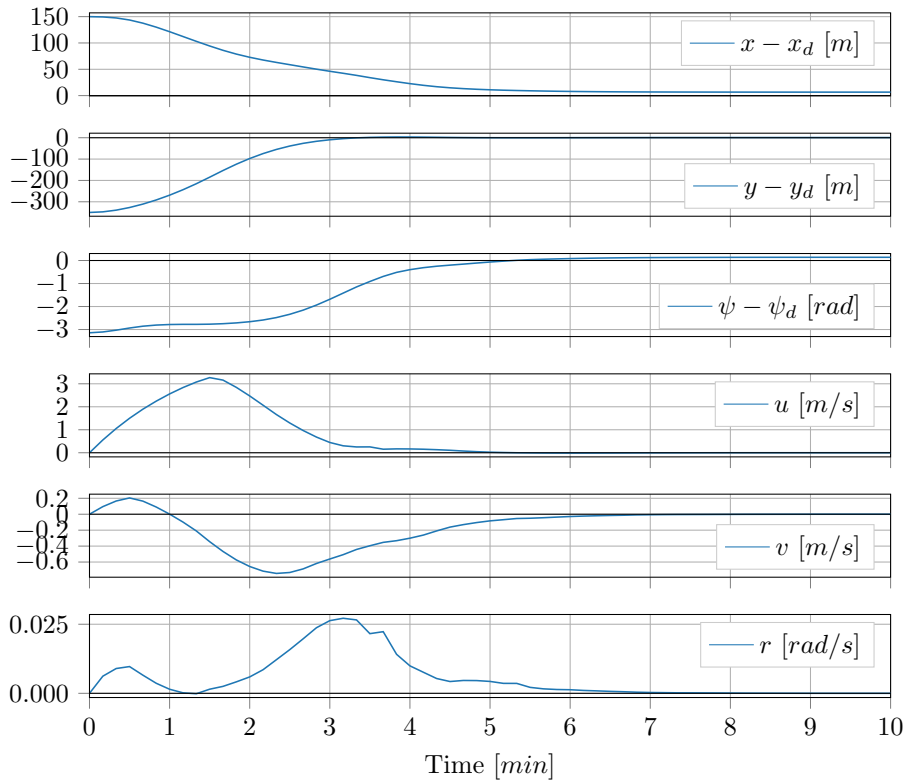


Figure 9: Vessel pose error $\eta - \eta_d$ and velocity ν when docking at Trondheim harbour.

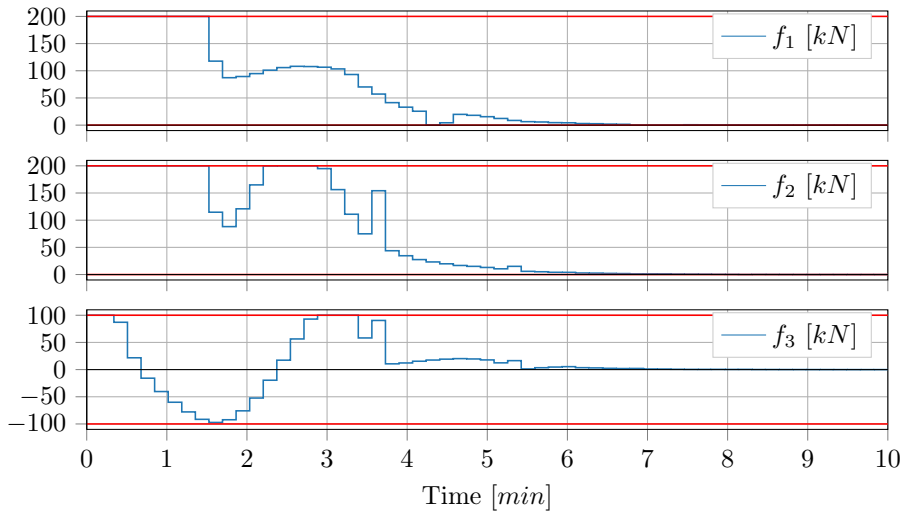


Figure 10: Thruster force for docking at Trondheim harbour, with saturation constraints indicated in red.

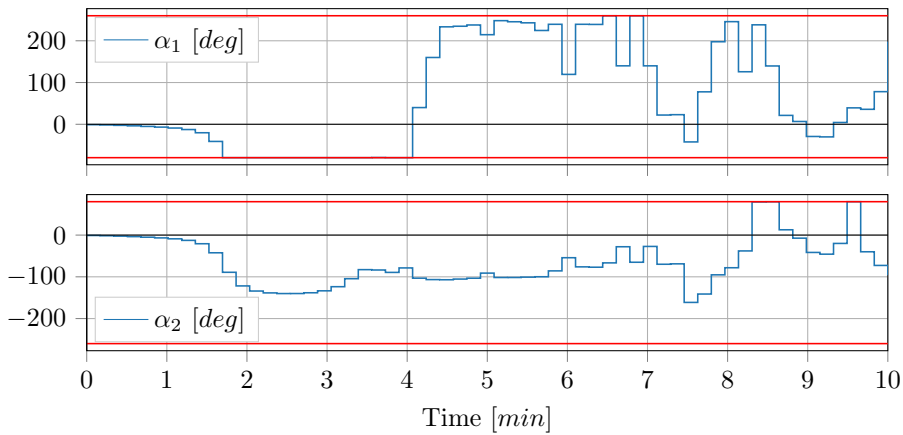


Figure 11: Azimuth angles when docking at Trondheim harbour, with saturation constraints indicated in red.

5 Conclusion

Based on the results of the simulation, the proposed method works very well, with the vessel approaching the target poses without violating the spatial constraints. Solving the open loop optimization problem with zeros as a trivial initial guess takes 2 – 4 seconds, while solving the problem using a warm start, a solution is found in about 0.5 seconds. With a purpose build solver this should take even less time, and ensures real time feasibility, as demonstrated by [19]. NLP solver for the problem should be chosen carefully. We found that IPOPT worked the best, as it was able to consistently solve the problem from a number of tested initial points, within a reasonable amount of time. With other solvers outright failing, or using excessive amounts of time.

The method does however have some drawbacks, since the proposed problem is non-convex due to the rotation of the azimuth thrusters and the vessel rotation, this means convergence to a global optimum can not be guaranteed. The method will however converge to a locally optimal solution, which in practise may be good enough, and will most importantly ensure safe operations. It is also worth noting that the problem is a finite horizon optimization problem, meaning we are only optimizing over a horizon T . This means that maneuvers that are optimal over a time horizon longer than T , may no longer be optimal over T , meaning the horizon must also be carefully chosen to get the desired behaviour.

The proposed method seems very promising, however many improvements can be made. Future research can be done on using more complex nonlinear vessel models, which may include thrust and azimuth dynamics. Different objective functions may be implemented, such as minimizing time, until the vessel reaches a terminal set, or energy expended. The method may also be further generalized by having dynamic spatial constraints, that use the largest convex set that does not intersect obstacles centered about the vessel as constraints. This may make the method not only suitable for docking, but also for general obstacle avoidance while in transit. While the proposed docking method has some measures ensuring robustness and safety while performing docking, future research can be done into making the method able to handle external environmental forces such as wind waves and currents using for example a scenario based MPC.

References

- [1] G. J. S. Rae and S. M. Smith. “A fuzzy rule based docking procedure for autonomous underwater vehicles”. In: *OCEANS 92 Proceedings@ m_Mastering the Oceans Through Technology*. Vol. 2. IEEE. 1992, pp. 539–546.
- [2] Ken Teo, Benjamin Goh, and Oh Kwee Chai. “Fuzzy docking guidance using augmented navigation system on an AUV”. In: *IEEE journal of oceanic engineering* 40.2 (2015), pp. 349–361.

Publications

- [3] Young-Hwa Hong et al. “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. 2003.
- [4] Morten Breivik and Jon-Erik Loberg. “A virtual target-based underway docking procedure for unmanned surface vehicles”. In: *IFAC Proceedings Volumes 44.1* (2011), pp. 13630–13635.
- [5] Joohyun Woo, Nakwan Kim, et al. “Vector Field based Guidance Method for Docking of an Unmanned Surface Vehicle”. In: *The Twelfth ISOPE Pacific/Asia Offshore Mechanics Symposium*. International Society of Offshore and Polar Engineers. 2016.
- [6] Aleksander Veksler et al. “Dynamic positioning with model predictive control”. In: *IEEE Transactions on Control Systems Technology* 24.4 (2016), pp. 1340–1353.
- [7] Margarita V Sotnikova and Evgeny I Veremey. “Dynamic positioning based on nonlinear MPC”. In: *IFAC Proceedings Volumes 46.33* (2013), pp. 37–42.
- [8] Tor Arne Johansen, Thor I Fossen, and Stig P Berge. “Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming”. In: *IEEE Transactions on Control Systems Technology* 12.1 (2004), pp. 211–216.
- [9] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.
- [10] Tor A Johansen and Thor I Fossen. “Control allocation—a survey”. In: *Automatica* 49.5 (2013), pp. 1087–1103.
- [11] Thor I Fossen, Svein I Sagatun, and Asgeir J Sørensen. “Identification of dynamically positioned ships”. In: (1996).
- [12] S. Gros and M. Diehl. “NMPC based on Huber penalty functions to handle large deviations of quadrature states”. In: *2013 American Control Conference*. June 2013, pp. 3159–3164.
- [13] S. Gros and M. Zanon. “Penalty Functions for Handling Large Deviation of Quadrature States in NMPC”. In: *IEEE Transactions on Automatic Control* 62.8 (Aug. 2017), pp. 3848–3860.
- [14] GA Hicks and WH Ray. “Approximation methods for optimal control synthesis”. In: *The Canadian Journal of Chemical Engineering* 49.4 (1971), pp. 522–528.
- [15] Peter Deuffhard. “A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting”. In: *Numerische Mathematik* 22.4 (1974), pp. 289–315.
- [16] TH Tsang, DM Himmelblau, and TF Edgar. “Optimal control via collocation and non-linear programming”. In: *International Journal of Control* 21.5 (1975), pp. 763–768.

- [17] Joel A E Andersson et al. “CasADi – A software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* (In Press, 2018).
- [18] Andreas Wächter and Lorenz T Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106.1 (2006), pp. 25–57.
- [19] M. Vukov et al. “Real-time nonlinear MPC and MHE for a large-scale mechatronic application”. In: *Control Engineering Practice* 45 (2015), pp. 64–78.
- [20] T I Fossen and T Perez. *Marine Systems Simulator (MSS)*. 2004.

A Vessel model

The vessel model used in the simulations was based on the SV Northern Clipper [11], where the model parameters were taken from the Marine System Simulator (MSS) Toolbox [20]. The model used has the following vessel dynamics

$$\begin{aligned}\dot{\boldsymbol{\eta}} &= \mathbf{J}(\boldsymbol{\psi})\boldsymbol{\nu}, \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}\boldsymbol{\nu} &= \mathbf{T}(\boldsymbol{\alpha})\mathbf{f}\end{aligned}$$

With the diagonal normalization matrix $\mathbf{N} = \text{diag}([1, 1, L])$, and the non-dimensional (bis-system) given by \mathbf{M}_{bis} and \mathbf{D}_{bis} , the mass and dampening matrix are given by the following.

$$\mathbf{M} = m\mathbf{N}\mathbf{M}_{bis}\mathbf{N}, \quad \mathbf{D} = m\sqrt{\frac{g}{L}}\mathbf{N}\mathbf{D}_{bis}\mathbf{N}$$

$$\mathbf{M}_{bis} = \begin{bmatrix} 1.1274 & 0 & 0 \\ 0 & 1.8902 & -0.0744 \\ 0 & -0.0744 & 0.1278 \end{bmatrix}, \quad \mathbf{D}_{bis} = \begin{bmatrix} 0.0358 & 0 & 0 \\ 0 & 0.1183 & -0.0124 \\ 0 & -0.0041 & 0.0308 \end{bmatrix}$$

Where the normalization parameters of length gravity and mass are given as $L = 76.2(m)$, $g = 9.8(m/s^2)$ and $m = 6000e3(kg)$ respectively.

For the vessel, we assume two azimuth thrusters in the aft, with one tunnel thruster in the front giving the thruster position and angle given in Table 1, and the thrust configuration matrix $\mathbf{T}(\boldsymbol{\alpha})$ is as follows.


$$\begin{bmatrix} \cos(\alpha_1) & \cos(\alpha_2) & 0 \\ \sin(\alpha_1) & \sin(\alpha_2) & 1 \\ l_{x_1} \sin(\alpha_1) - l_{y_1} \cos(\alpha_1) & l_{x_2} \sin(\alpha_2) - l_{y_2} \cos(\alpha_2) & l_{x_3} \end{bmatrix}$$

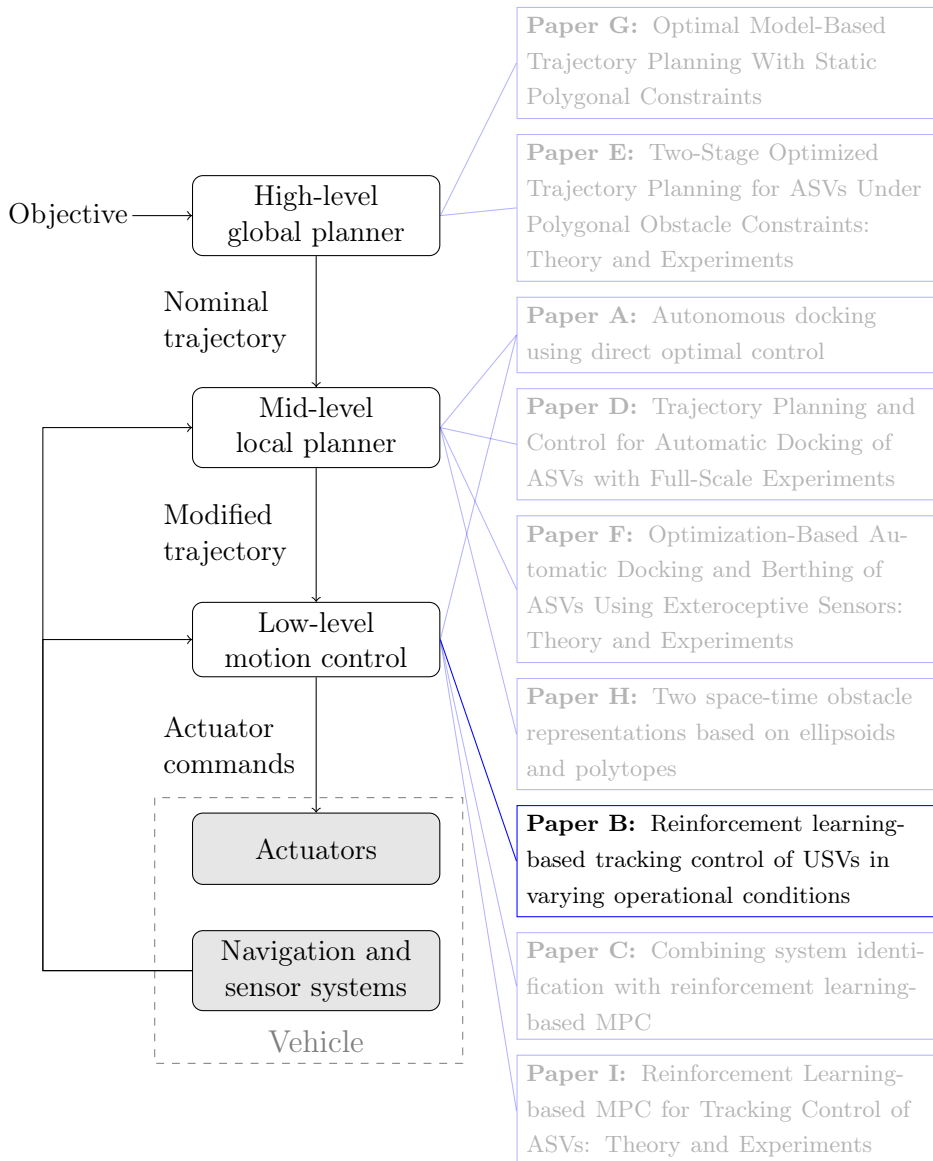
Table 1: Thruster position and angle

Truster	x -position	y -position	angle
Azimuth 1	$l_{x_1} = -35m$	$l_{y_1} = 7m$	α_1
Azimuth 2	$l_{x_2} = -35m$	$l_{y_2} = -7m$	α_2
Tunnel 3	$l_{x_3} = 35m$	$l_{y_3} = 0m$	$\alpha_3 = \frac{\pi}{2}$

B Reinforcement learning-based tracking control of USVs in varying operational conditions

Postprint of [20] **Andreas Bell Martinsen**, Anastasios Lekkas, Sébastien Gros, Jon Arne Glomsrud, and Tom Arne Pedersen. “Reinforcement learning-based tracking control of USVs in varying operational conditions”. In: *Frontiers in Robotics and AI* 7 (2020), p. 32. DOI: 10.3389/frobt.2020.00032

©2020 Andreas Bell Martinsen, Anastasios Lekkas, Sébastien Gros, Jon Arne Glomsrud, and Tom Arne Pedersen. Reprinted and formatted to fit the thesis under the terms of the Creative Commons Attribution License 



Reinforcement learning-based tracking control of USVs in varying operational conditions

Andreas B. Martinsen¹, Anastasios M. Lekkas¹, Sébastien Gros¹, Jon Arne Glomsrud², and Tom Arne Pedersen²

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

²Digital Assurance Program, Group Technology and Research, DNV GL, Trondheim, Norway

Abstract: We present a reinforcement learning-based (RL) control scheme for trajectory tracking of fully-actuated surface vessels. The proposed method learns online both a model-based feedforward controller, as well an optimizing feedback policy in order to follow a desired trajectory under the influence of environmental forces. The method's efficiency is evaluated via simulations and sea trials, with the unmanned surface vehicle (USV) *ReVolt* performing three different tracking tasks: The four corner DP test, straight-path tracking and curved-path tracking. The results demonstrate the method's ability to accomplish the control objectives and a good agreement between the performance achieved in the Revolt Digital Twin and the sea trials. Finally, we include an section with considerations about assurance for RL-based methods and where our approach stands in terms of the main challenges.

Keywords: reinforcement learning, trajectory tracking, optimal control, model-based adaptive control, approximate dynamic programming (ADP), dynamic positioning (DP), autonomous ships, system identification

1 Introduction

Control of marine vehicles is a challenging problem, mostly due to the unpredictable nature of the sea and the difficulty in developing accurate mathematical models to represent the varying marine vehicle dynamics. As a result, considerable research effort has been dedicated to the topic since the early 90's [1], resulting in a vast literature utilizing ideas from virtually every branch of control engineering: Linear, nonlinear, adaptive, intelligent, optimal, fuzzy and stochastic control approaches, to name a few, have been developed and tested over the years, and many of their properties are well understood [2–9]. Due to the fact that the hydrodynamic coefficients, and consequently the behavior, of a marine vehicle can vary significantly in different

Publications

speed regimes, a common approach has been to design controllers for specific motion control scenarios. This approach simplifies the vessel modeling process and has led to dynamic positioning (DP) and station keeping controllers for speeds close to zero, and trajectory tracking or path following (depending on whether temporal constraints are considered) controllers when a vessel is in transit mode. Naturally, the main drawback is that, when moving from one speed regime to another, controllers and/or models with different properties are needed. Two well-researched ways to achieve such performance diversity with conventional methods are to design numerous controllers and switch among them when needed, or to use adaptive approaches. To this end, research effort has been dedicated to developing flexible methods for updating the model parameters by, for instance, using system identification methods or parameter estimation via neural networks [10–16]. In the majority of the aforementioned works, model-based approaches exploiting human knowledge on hydrodynamics and the laws of motion were considered.

Reinforcement learning (RL), also known as neuro-dynamic programming or approximate dynamic programming, is a field of research developed by the Artificial Intelligence (AI) community for achieving optimal sequential decision making under system and environment uncertainty. The roots of RL can be traced back to the 60's and a thorough overview of its evolution can be found in [17, 18]. Contrary to optimal control theory, RL is based on *evaluative*, rather than *instructive*, feedback and comes in different forms, which may or may not include partial knowledge of the environment or the system. The process typically involves hand-engineering a reward function, which assigns a reward, or penalty, to the actions that induce desired, or undesired, outcomes, respectively. An RL algorithm is then assigned to find a policy (or controller, in control engineering terminology) that solves the control objective optimally, given the problem constraints and uncertainties. To sum up, RL algorithms use the reward function as a guide, and through trial and error, learn to model the system and its environment, which then leads to a policy that provides an optimal solution to the assigned problem.

Despite a number of successes for RL on simple problems, including algorithms such as *Q-learning* and *REINFORCE*, the field has seen limited interest. In recent years there has however been a resurgence of interest due to the development of Deep Reinforcement Learning (DRL), starting with Deep Mind developing the *Deep Q-Network* (DQN) algorithm that achieved superhuman performance in several Atari games [19], followed by Deep Mind's *AlphaGo* algorithm becoming the first computer program to beat a human champion in the game of *Go* [20]. Since then, DRL has been successful in surpassing all previous computer programs in chess and learning how to accomplish complex robotic tasks [21, 22]. Given DRL's ability to tackle problems with high uncertainty, implementations to motion control scenarios involving marine vessels have been presented recently [23–29]. In most of these works the authors implemented algorithms pertaining to the class of *actor-critic* RL methods, which involves two parts [30]: The *actor*, where the gradient of the performance is estimated and the policy parameters are directly updated in a direction of improvement. The main drawbacks of the actor are that it is prone to variance and the new gradient is

estimated independently of past estimates. The *critic*, learns an approximation of the value function, leading to an approximate solution to the Bellman or Hamilton-Jacobi-Bellman equation, which then is expected to prescribe a near-optimal policy. The critic's main drawback is that it lacks reliable guarantees in terms of near-optimality of the resulting policy. The actor-critic approach involves the actor improving the policy parameters' estimation based on the approximations learned by the critic. In the case of DRL, one main novelty was the use of two DNNs as function approximators of the policy and the value function, which resulted in considerably improved performance compared to previous approaches. However, DNNs have drawbacks, with some of the most important being lack of transparency and interpretability, lack of robustness, and inability to generalize to situations beyond their past experiences.

In this paper, we follow and extend the work by Kamalapurkar et al. [31, 32] in order to build a trajectory tracking control system for a fully-actuated unmanned surface vehicle (USV). Conceptually, the approach is quite similar to dynamic positioning (DP)[33], but extends to higher velocity operational domains, while also trying to optimize tracking performance and compensate for environmental forces[34]. The method combines elements from reinforcement learning, Lyapunov stability theory and system identification: We assume the structure of the vessel model is known but all of its parameters are unknown and have to be estimated online, as well as updated accordingly when the operational conditions change. Then we derive the tracking error dynamics for a generic reference trajectory and a stabilizing parametric control law (the *actor*), whose parameters are estimated during operation.

In order to validate the control scheme, the proposed method was tested in both in simulations, and on a physical model of DNV GL's *ReVolt* platform.

2 Reinforcement learning-based trajectory tracking

In this section we will derive a trajectory tracking control system for fully-actuated USVs. Since the approach is a model based reinforcement learning approach, we will start by looking at how ASVs can be modeled, and how the models can be approximated online using system identification. We will derive a feedforward control law for tracking the desired trajectory, and a feedback control law based on reinforcement learning, for controlling the drift of the vessel in a way that minimizes a given cost function.

2.1 Vessel Model

The mathematical model used to describe the system can then be kept reasonably simple by limiting it to the planar position and orientation of the vessel. 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, (x, y) describe the Cartesian position in the

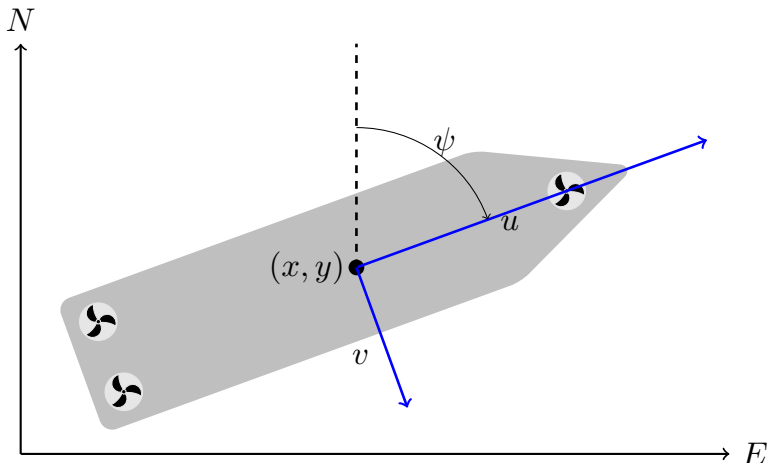


Figure 1: 3-DOF vessel centered at (x, y) , with surge velocity u , sway velocity v , heading ψ in a North-East-Down (NED) reference frame.

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 1. Using the notation in [35] we can describe a 3-DOF vessel model as follows

$$\begin{aligned} \dot{\eta} &= \mathbf{J}(\eta)\boldsymbol{\nu}, \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau}_{\text{Thrust}} + \boldsymbol{\tau}_{\text{Environment}} \end{aligned} \quad (1)$$

where $\mathbf{M} \in \mathbb{R}^{3 \times 3}$, $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\boldsymbol{\tau}_{\text{Thrust}}, \boldsymbol{\tau}_{\text{Environment}} \in \mathbb{R}^3$ and $\mathbf{J}(\eta) \in SO(3)$ are the inertia matrix, damping matrix, coriolis matrix, control input vector, environmental forces and rotation matrix respectively. The rotational matrix $\mathbf{J}(\eta) \in SO(3)$ is given by

$$\mathbf{J}(\eta) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

and is the rotation from the body frame to the earth-fixed North East Down (NED) reference frame.

2.2 Model approximation

While the structure of a vessel model, as given above, is well known, the model parameters are often difficult to find. For our approach we wish to make as few assumptions on the parameters of the vessel model as possible, and use online system identification in order to model the vessel based on gathered data. For this we assume

B. Reinforcement learning-based tracking control of USVs in ...

that we know the model structure as given in (1), but that the model parameters are unknown. Splitting the model into a known and unknown part, we get the following:

$$\dot{\mathbf{x}} = f_{\theta}(\mathbf{x}) + f_1(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \quad (3)$$

where $f_1(\mathbf{x})$ and $g(\mathbf{x})$ are known, and $f_{\theta}(\mathbf{x})$ is unknown. For the vessel model in (1), with the state vector $\mathbf{x} = [\boldsymbol{\eta}, \boldsymbol{\nu}]^{\top}$ and the control vector $\mathbf{u} = \boldsymbol{\tau}_{\text{Thrust}}$. We have the following:

$$\begin{aligned} f_{\theta}(\mathbf{x}) &= \begin{bmatrix} \mathbf{0}_{3 \times 1} \\ -\mathbf{M}^{-1} (\mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{\tau}_{\text{Environment}}) \end{bmatrix} \\ f_1(\mathbf{x}) &= \begin{bmatrix} \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathbf{0}_{3 \times 1} \end{bmatrix} \\ g(\mathbf{x}) &= \begin{bmatrix} \mathbf{0}_{3 \times 3} \\ \mathbf{M}^{-1} \end{bmatrix} \end{aligned}$$

hence we assume the mass matrix is known, but the damping and coriolis matrix are unknown. For the damping and coriolis matrices we assume the vessel has port starboard symmetry, from [35] this gives the following structure.

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} \cdot v + Y_{\dot{r}} \cdot r \\ 0 & 0 & -X_{\dot{u}} \cdot u \\ -Y_{\dot{v}} \cdot v + Y_{\dot{r}} \cdot r & X_{\dot{u}} \cdot u & 0 \end{bmatrix} \quad (4)$$

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} -X_u - X_{|u|u} \cdot |u| & 0 & 0 \\ 0 & -Y_v - Y_{|v|v} \cdot |v| - Y_{|r|v} \cdot |r| & -Y_r - Y_{|v|r} \cdot |v| - Y_{|r|r} \cdot |r| \\ 0 & -N_v - N_{|v|v} \cdot |v| - N_{|r|v} \cdot |r| & -N_r - N_{|v|r} \cdot |v| - N_{|r|r} \cdot |r| \end{bmatrix} \quad (5)$$

For the damping matrix $\mathbf{D}(\boldsymbol{\nu})$, both linear and nonlinear terms are included. The linear terms are important for low speed maneuvering and station keeping, while ensuring the velocity converges exponentially to zero. The nonlinear terms are required as they dominate at higher velocities. This ensures that the model is able to handle a large range of velocities, i.e. it can be used for both high speed trajectory tracking and low speed station keeping and dynamic positioning. For the coriolis matrix, we use only the added mass terms. Since the structure of the rigid body, and added mass is the same for the coriolis matrix, the coriolis matrix given above will be able to capture both the added mass and rigid body dynamics.

In addition to learning the vessel dynamics, we also wanted to be able to compensate for environmental forces. In order to allow for the environmental forces to be learned, they are modeled as an additional unknown pressure vector $\mathbf{p}_{\text{env}}^{\text{NED}} = [p_{\text{North}}, p_{\text{East}}, 0]^{\top}$ assumed constant in the NED frame. The resulting force in the body frame is then assumed to be proportional to the cross sectional area of the vessel times the pressure in the body frame, giving the following relationship.

$$\boldsymbol{\tau}_{\text{Environment}}^{\text{body}} = \text{diag}([w, l, 0])\mathbf{J}^{\top}(\boldsymbol{\nu})\mathbf{p}_{\text{Environment}}^{\text{NED}} \quad (6)$$

Publications

where w and l are the width and length of the vessel respectively, note that for better accuracy calculated pressure coefficients based on the design of the hull may be used instead of the width and length. The unknown parameters are

$$\boldsymbol{\theta} = [X_{\dot{u}}, Y_{\dot{v}}, Y_{\dot{r}}, X_u, Y_v, Y_r, N_v, N_r, X_{|u|u}, Y_{|v|v}, Y_{|v|r}, Y_{|r|v}, Y_{|r|r}, N_{|v|v}, N_{|v|r}, N_{|r|v}, N_{|r|r}, p_{\text{North}}, p_{\text{East}}]^\top \quad (7)$$

and the function $f_{\boldsymbol{\theta}}(\mathbf{x})$ can be written as a linear function in $\boldsymbol{\theta}$:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = Y(\mathbf{x})\boldsymbol{\theta} \quad (8)$$

where $Y(\mathbf{x})$ is:

$$Y(\mathbf{x}) = \begin{bmatrix} \mathbf{0}_{3 \times 3} \\ -\mathbf{M}^{-1} \end{bmatrix} \begin{bmatrix} 0 & v \cdot r & r^2 & -u & 0 & 0 & 0 & 0 & -|u|u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w \cos \psi & w \sin \psi \\ -u \cdot r & 0 & 0 & 0 & -v & -r & 0 & 0 & 0 & -|v|v & -|r|r & -|r|v & -|r|r & 0 & 0 & 0 & 0 & 0 & -l \sin \psi & l \cos \psi \\ u \cdot v & -v \cdot u & -r \cdot u & 0 & 0 & 0 & -v & -r & 0 & 0 & 0 & 0 & 0 & -|v|v & -|v|r & -|r|v & -|r|r & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

We therefore obtain the following parametric model:

$$\dot{\mathbf{x}} = Y(\mathbf{x})\boldsymbol{\theta} + f_1(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \quad (10)$$

which is linear in the parameters $\boldsymbol{\theta}$.

Model assumptions

- The vessel is port starboard symmetric, with a structure as given as in (1).
- The vessel dampening is linear and quadratic with respect to the linear and angular velocity.
- Environmental forces are constant in the NED frame, and proportional to vessel cross section.
- The vessel is fully actuated.

2.3 Trajectory tracking

In this section we will develop an adaptive feedforward control law which given a time-varying trajectory, finds the control inputs required to follow the trajectory, given the model approximation found in the previous section.

When the control objective is to track a bounded continuously differentiable signal \mathbf{x}_d , the dynamics of the tracking error $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$ can be written as

$$\dot{\mathbf{e}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} - \dot{\mathbf{x}}_d \quad (11)$$

Assuming $g(\mathbf{x})$ is bounded and has full column rank for all \mathbf{x} [31], then the system is controllable, which in this case holds as the vessel is fully actuated. This gives the feedforward control for the reference trajectory as :

$$\mathbf{u}_d(\mathbf{x}_d, \dot{\mathbf{x}}_d) = g^+(\mathbf{x}_d)(\dot{\mathbf{x}}_d - f(\mathbf{x}_d)) \quad (12)$$

B. Reinforcement learning-based tracking control of USVs in ...

where g^+ is the left Moore–Penrose pseudo-inverse, given as $g^+ = (g^\top g)^{-1} g^\top$. Using a reference model $\dot{\mathbf{x}}_d = h_d(\mathbf{x}_d)$, the feedforward control for the reference trajectory can be written as:

$$\mathbf{u}_d(\mathbf{x}_d) = g^+(\mathbf{x}_d)(h_d(\mathbf{x}_d) - f(\mathbf{x}_d)) \quad (13)$$

We can then formulate the tracking problem as the following time-invariant optimal control problem.

$$\underbrace{\begin{bmatrix} \dot{e} \\ \dot{\mathbf{x}}_d \end{bmatrix}}_{\zeta} = \underbrace{\begin{bmatrix} f(e + \mathbf{x}_d) + g(e + \mathbf{x}_d)\mathbf{u}_d(\mathbf{x}_d) \\ h_d(\mathbf{x}_d) \end{bmatrix}}_{F(\zeta)} + \underbrace{\begin{bmatrix} g(e + \mathbf{x}_d) \\ 0 \end{bmatrix}}_{G(\zeta)} \boldsymbol{\pi} \quad (14)$$

Where $\boldsymbol{\pi}$ is an input correction for the drift dynamics, which we will define in the next section. Given the parametric model in (10), the parametric version of the tracking problem is given as:

$$\underbrace{\begin{bmatrix} \dot{e} \\ \dot{\mathbf{x}}_d \end{bmatrix}}_{\zeta} = \underbrace{\begin{bmatrix} Y(e + \mathbf{x}_d)\boldsymbol{\theta} + f_1(e + \mathbf{x}_d) + g(e + \mathbf{x}_d)\mathbf{u}_d(\mathbf{x}_d; \boldsymbol{\theta}) \\ h_d(\mathbf{x}_d) \end{bmatrix}}_{F(\zeta; \boldsymbol{\theta})} + \underbrace{\begin{bmatrix} g(e + \mathbf{x}_d) \\ 0 \end{bmatrix}}_{G(\zeta)} \boldsymbol{\pi} \quad (15)$$

where the parametric feedforward control for the reference trajectory $\mathbf{u}_d(\mathbf{x}_d; \boldsymbol{\theta})$ is given as:

$$\mathbf{u}_d(\mathbf{x}_d; \boldsymbol{\theta}) = g^+(\mathbf{x}_d)(h_d(\mathbf{x}_d) - Y(\mathbf{x}_d)\boldsymbol{\theta} - f_1(\mathbf{x}_d)) \quad (16)$$

Given the formulation above, with the feedforward control for the reference trajectory $\mathbf{u}_d(\mathbf{x}_d)$, and the optimal model parameters $\boldsymbol{\theta}^*$, the exact feedforward control for the reference trajectory is possible to compute. The dynamics above guarantee trajectory tracking when $\dot{e} = 0$, i.e. when the tracking error is zero. When the tracking error is not zero however, we need to control the drift dynamics in order to ensure convergence to the desired trajectory by designing the feedback control $\boldsymbol{\pi}(t)$ such that $\lim_{t \rightarrow \infty} e(t) = 0$. The objective of the optimal control problem is to design the feedback control law $\boldsymbol{\pi}(t)$ such that it minimizes a given cost function.

2.4 Approximate optimal control of drift dynamics

In the previous section we developed a feedforward control law $\mathbf{u}_d(\mathbf{x}_d; \boldsymbol{\theta})$ for tracking a desired trajectory. Due to inaccuracies in model approximation and disturbances, using only the feedforward control law, the vessel will experience drift. In order to compensate for the inevitable drift, we will in this section develop a feedback control law $\boldsymbol{\pi}(\cdot)$, which controls the drift dynamics in a way that optimizes a given cost function. We will additionally show how the parameters of the feedback control law can be learned by using reinforcement learning.

The optimal control problem we wish to solve is that of minimizing the cost function:

$$J(\zeta, \boldsymbol{\pi}) = \int_{t_0}^{\infty} r(\zeta(\tau), \boldsymbol{\pi}(\tau)) d\tau \quad (17)$$

Publications

Where $r(\cdot)$ is scalar function defining the local cost, and should not be confused with the yaw rate. The cost function is defined as:

$$r(\zeta, \boldsymbol{\pi}) = Q(\zeta) + \boldsymbol{\pi}^\top \mathbf{R}\boldsymbol{\pi} \quad (18)$$

where $\mathbf{R} \succ 0$ is a positive definite symmetric matrix. And $Q(\zeta)$ is a positive definite function. Assuming that a minimizing control policy $\boldsymbol{\pi}(\cdot)$ exists, the optimal value function is given as:

$$V^*(\zeta) = \min_{\boldsymbol{\pi}(\tau), \tau \in [t_0, \infty)} \int_{t_0}^{\infty} r(\zeta(\tau), \boldsymbol{\pi}(\tau)) d\tau \quad (19)$$

We can now note that for a small time step Δt , the above expression can be formulated as:

$$V^*(\zeta(t)) = \min_{\boldsymbol{\pi}(\tau), \tau \in [t, t+\Delta t)} \int_t^{t+\Delta t} r(\zeta(\tau), \boldsymbol{\pi}(\tau)) d\tau + V^*(\zeta(t + \Delta t))$$

Taking the limit of this as $\Delta t \rightarrow 0$, for the optimal value function under the optimal policy, we get[36]:

$$V^*(\zeta(t)) = \min_{\boldsymbol{\pi}(t)} r(\zeta(t), \boldsymbol{\pi}(t)) + V^*(\zeta(t)) + \dot{V}^*(\zeta(t))$$

Simplifying this we get the Hamilton-Jacobi-Bellman (HJB) equation for the optimal control problem as follows:

$$\begin{aligned} H^* &= \dot{V}^*(\zeta) + r(\zeta, \boldsymbol{\pi}^*(\zeta)) \\ &= \nabla_{\zeta} V^*(\zeta)^\top \dot{\zeta} + r(\zeta, \boldsymbol{\pi}^*(\zeta)) \\ &= \nabla_{\zeta} V^*(\zeta)^\top (F(\zeta) + G(\zeta)\boldsymbol{\pi}^*(\zeta)) + r(\zeta, \boldsymbol{\pi}^*(\zeta)) = 0 \end{aligned} \quad (20)$$

Where H^* , $\boldsymbol{\pi}^*$ and V^* is the optimal hamiltonian, policy and value function respectively. From calculus of variation [37] we have the Hamiltonian minimization condition, which states that a value function V is the optimal Value function if and only if there exists a controller $\boldsymbol{\pi}(\cdot)$ and trajectory $\zeta(\cdot)$ under $\boldsymbol{\pi}(\cdot)$ satisfy the equation:

$$\begin{aligned} &\nabla_{\zeta} V(\zeta)^\top (F(\zeta) + G(\zeta)\boldsymbol{\pi}(\zeta)) + r(\zeta, \boldsymbol{\pi}(\zeta)) \\ &= \min_{\hat{\boldsymbol{\pi}} \in U} \{ \nabla_{\zeta} V(\zeta)^\top (F(\zeta) + G(\zeta)\hat{\boldsymbol{\pi}}(\zeta)) + r(\zeta, \hat{\boldsymbol{\pi}}(\zeta)) \} \end{aligned} \quad (21)$$

The necessary conditions for this to hold are:

$$\nabla_{\boldsymbol{\pi}} (\nabla_{\zeta} V(\zeta)^\top (F(\zeta) + G(\zeta)\boldsymbol{\pi}(\zeta)) + r(\zeta, \boldsymbol{\pi}(\zeta))) = 0 \quad (22)$$

which gives the closed form solution of the optimal controller as:

$$\begin{aligned} G^\top(\zeta) (\nabla_{\zeta} V(\zeta)) + \nabla_{\boldsymbol{\pi}} r(\zeta, \boldsymbol{\pi}) &= 0 \Leftrightarrow \\ 2\mathbf{R}\boldsymbol{\pi} &= -G^\top(\zeta) (\nabla_{\zeta} V(\zeta)) \Leftrightarrow \end{aligned}$$

$$\boldsymbol{\pi}^*(\boldsymbol{\zeta}) = -\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta})(\nabla_{\boldsymbol{\zeta}}V(\boldsymbol{\zeta})) \quad (23)$$

Hence assuming that an optimal controller exists, the closed form solution given by the HJB equation is given by (23). Note that the value function is assumed time independent, and hence we are looking for a stationary solution of the HJB equation. This holds true, as the reformulation into a trajectory tracking problem (15) gives a time independent system.

The Universal Approximation theorem [31, Property 2.3] states that a single layer neural network can simultaneously approximate a function and its derivative given a sufficiently large number of basis functions. Using this, we can approximate any continuous function as:

$$V(\mathbf{x}) = \mathbf{W}^\top \boldsymbol{\sigma}(\mathbf{x}) + \epsilon(\mathbf{x}) \quad (24)$$

where \mathbf{W} is the weighting matrix, $\boldsymbol{\sigma}(\mathbf{x})$ is the vector of basis functions, and $\epsilon(\mathbf{x})$ is the approximation error, which can be made arbitrarily small by increasing the number of basis functions. Note that the basis functions can here be chosen to be any parameterization, such as Radial-Basis functions, polynomials or even a Fourier series. Using this we can represent the value function as a neural network which is linear in the parameters, giving the optimal value function:

$$V^*(\boldsymbol{\zeta}) = \mathbf{W}^\top \boldsymbol{\sigma}(\boldsymbol{\zeta}) + \epsilon(\boldsymbol{\zeta}) \quad (25)$$

and the optimal policy as a feedback control law on the form:

$$\boldsymbol{\pi}^*(\boldsymbol{\zeta}) = -\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta})(\nabla_{\boldsymbol{\zeta}}\boldsymbol{\sigma}(\boldsymbol{\zeta})^\top \mathbf{W} + \nabla_{\boldsymbol{\zeta}}\epsilon^\top(\boldsymbol{\zeta})) \quad (26)$$

By making the parameterizations sufficiently rich, we make the approximation error small. We can then use the approximations given below, for the value function and control policy respectively.

$$\hat{V}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_c) = \hat{\mathbf{W}}_c^\top \boldsymbol{\sigma}(\boldsymbol{\zeta}) \quad (27)$$

$$\hat{\boldsymbol{\pi}}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_a) = -\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta})\nabla_{\boldsymbol{\zeta}}\boldsymbol{\sigma}(\boldsymbol{\zeta})^\top \hat{\mathbf{W}}_a \quad (28)$$

In order to find the parameters $\hat{\mathbf{W}}_c$ and $\hat{\mathbf{W}}_a$, we will in the next section find update laws, based on reinforcement learning, to be able to optimize performance online.

Unfortunately, policy (28) does not account for the saturating constraints, such as the maximum force the actuators of the physical vessel can produce. In order to account for the actuator limitations, we propose a different control policy which uses a saturating function[36] in order to avoid this problem. Using the following cost function:

$$r(\boldsymbol{\zeta}, \boldsymbol{\pi}) = Q(\boldsymbol{\zeta}) + 2 \sum_{i=1}^m r_i \int_0^{\pi_i} \tanh^{-1}(\xi) d\xi \quad (29)$$

where r_i is the i^{th} entry of the diagonal of \mathbf{R} , i.e. $\mathbf{R} = \text{diag}([r_1, r_2 \dots r_m])$. Figure 2 shows a comparison of the saturating input cost, and a pure quadratic cost. Performing the same analysis as for the quadratic penalty, we can get the following saturating control law:

$$\boldsymbol{\pi}^*(\boldsymbol{\zeta}) = -\tanh\left(\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta})(\nabla_{\boldsymbol{\zeta}}V(\boldsymbol{\zeta}))\right) \quad (30)$$

Since $\tanh(\cdot)$ saturates at ± 1 , this means that the feedback control law $\boldsymbol{\pi}$ will saturate at ± 1 , the outputs can then be easily scaled to fit other bounds. It can be shown that since $\tanh(\cdot)$ is a monotonically increasing continuously differentiable function, the control law satisfies the first order necessary conditions, and the second order sufficient conditions of the Hamiltonian minimization condition. This means that if an optimal controller exists the closed form solution is given by (30). Using an approximation we get the following approximate optimal policy

$$\hat{\boldsymbol{\pi}}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_a) = -\tanh\left(\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta})\nabla_{\boldsymbol{\zeta}}\sigma(\boldsymbol{\zeta})^\top \hat{\mathbf{W}}_a\right) \quad (31)$$

It should be noted, that while the policy in (31) uses a value function approximation in order to approximate the optimal policy, the parameters $\hat{\mathbf{W}}_a$ are not the same as the parameters $\hat{\mathbf{W}}_c$ in the value function approximation in (27). In this way we can separate the learning of the policy and value function, this is known as an actor critic method, where the value function is known as the critic, and the policy is known as an actor. The intuitive reason for doing this, is that it allows the critic to learn the value function resulting from the behaviour of the policy, and in this way it can critique the policy. Similarly, the policy or actor, can learn to improve its performance based on the criticism of the critic. How the learning is performed is further discussed in the next section.

2.5 Update laws

Now that we have expressed the control laws $\mathbf{u}_d(\mathbf{x}_d; \hat{\boldsymbol{\theta}})$, $\hat{\boldsymbol{\pi}}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_a)$ and value function $\hat{V}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_c)$, the challenge becomes finding update laws for the parameters of the system identification $\hat{\boldsymbol{\theta}}$, the critic $\hat{\mathbf{W}}_c$ and the actor $\hat{\mathbf{W}}_a$. For the model parameters $\hat{\boldsymbol{\theta}}$, we will use methods from system identification and adaptive control, to try to optimize the fit between the parameterized model, and the observed vessel states. For the actor and critic parameters $\hat{\mathbf{W}}_a$ and $\hat{\mathbf{W}}_c$, we will use model based reinforcement learning to find the parameters that gives the optimal value function, and consequently the optimal feedback control policy.

For the system identification parameters $\hat{\boldsymbol{\theta}}$, the goal is to find the parameters for which the model behaves as similarly as possible to the observed behaviour. Running our physical system, and collecting observations $(\dot{\mathbf{x}}_i, \mathbf{x}_i, \mathbf{u}_i) \quad i \in 1, 2, \dots, N$, we can formulate a least squares optimization problem for finding the parameters that minimize the difference between the observed state derivative $\dot{\mathbf{x}}_i$ and the parametric model (3)

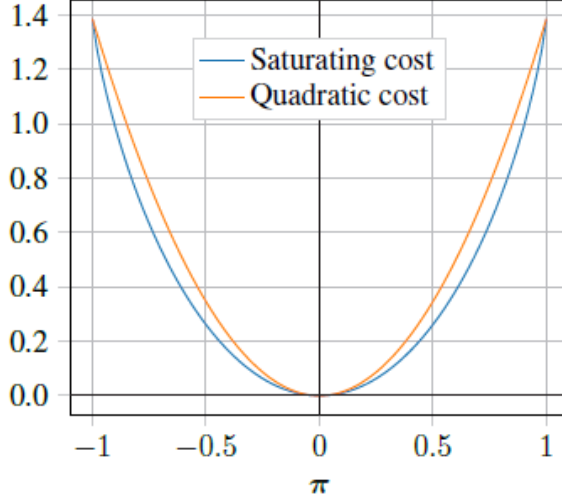


Figure 2: Comparison between saturating cost (29) and quadratic cost (18). Here the quadratic cost is scaled by $\ln(4)$.

as follows.

$$\boldsymbol{\theta}^* = \arg \min_{\hat{\boldsymbol{\theta}}} \underbrace{\sum_{i=1}^N \frac{1}{2} \|\dot{\mathbf{x}}_i - Y(\mathbf{x}_i)\hat{\boldsymbol{\theta}} - f_1(\mathbf{x}_i) - g(\mathbf{x}_i)\mathbf{u}_i\|_2^2}_{L(\hat{\boldsymbol{\theta}})}$$

This is a linear least squares optimization problem for which there exists a closed form solution, however we can also solve the problem by performing stochastic gradient decent on the parameters $\hat{\boldsymbol{\theta}}$, as follows:

$$\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \nabla_{\hat{\boldsymbol{\theta}}} L(\hat{\boldsymbol{\theta}})$$

The gradient decent law above, works in discrete iteration, however we can reformulate it as a an ordinary differential equation (ODE). Doing some further changes motivated by the stability analysis of the convergence of the parameter estimates, we get the concurrent learning based approach proposed in [38] as:

$$\dot{\hat{\boldsymbol{\theta}}}(t) = \boldsymbol{\Gamma}_\theta Y^\top(\mathbf{x}(t))\tilde{\mathbf{x}}(t) + \frac{k_\theta}{N} \boldsymbol{\Gamma}_\theta \sum_{i=1}^N Y^\top(\mathbf{x}_i) \left(\dot{\mathbf{x}}_i - f_1(\mathbf{x}_i) - g(\mathbf{x}_i)\mathbf{u}_i - Y(\mathbf{x}_i)\hat{\boldsymbol{\theta}} \right) \quad (32)$$

where $\boldsymbol{\Gamma}_\theta$ is a parameter weight matrix, and k_θ is a scalar weight factor. Assuming that the prerecorded data is sufficiently rich such that the matrix $\sum_{i=1}^N Y^\top(\mathbf{x}_i)Y(\mathbf{x}_i)$ is full rank, the parameter error can be shown to converge. As the convergence rate of the system identifier is proportional to the minimum singular value of $\sum_{i=1}^N Y^\top(\mathbf{x}_i)Y(\mathbf{x}_i)$, replacing data in the data stack can be done by using a singular value maximizing

algorithm [39] in order to get faster convergence. Note, that since we are assuming a sufficiently rich prerecorded data set, we no longer need persistence of excitation (PE), in order to guarantee parameter convergence.

In order to find the update laws for the critic or value function parameters $\hat{\mathbf{W}}_c$, we need a way of evaluating the optimality of the value function given the the current parameters. For this we look back at the HJB equation (20) given as:

$$0 = r(\zeta, \pi^*(\zeta)) + \nabla_{\zeta} V^*(\zeta)^{\top} (F(\zeta) + G(\zeta)\pi^*(\zeta))$$

Substituting the estimates \hat{V} and $\hat{\pi}$ for the optimal value function V^* and optimal policy π , we can formulate the Bellman error as the error in the HJB equation as follows:

$$\begin{aligned} \delta(\zeta; \hat{\theta}, \hat{\mathbf{W}}_c, \hat{\mathbf{W}}_a) &= \underbrace{Q(\zeta) + \hat{\pi}^{\top}(\zeta; \hat{\mathbf{W}}_a) \mathbf{R} \hat{\pi}(\zeta; \hat{\mathbf{W}}_a)}_{r(\zeta, \hat{\pi}(\zeta; \hat{\mathbf{W}}_a))} \\ &+ \nabla_{\zeta} \hat{V}(\zeta; \hat{\mathbf{W}}_c)^{\top} (F(\zeta; \hat{\theta}) + G(\zeta) \hat{\pi}(\zeta; \hat{\mathbf{W}}_a)) \end{aligned} \quad (33)$$

The Bellman error can intuitively be thought of as the error between the optimal value function under the policy, and the estimates. Since the goal for the value function or critic is to find the parameters \mathbf{W}_c that best approximates the value function, a natural choice becomes to find the parameters that minimize the bellman error. With reinforcement learning we can use a data stack of prerecorded state transitions $\zeta_i(t) = [\mathbf{x}_i - \mathbf{x}_{d,i}, \mathbf{x}_{d,i}]^{\top}$ $i \in 1, 2, \dots, N$, to formulate the following optimization problem:

$$\min_{\hat{\mathbf{W}}_c} \sum_{i=1}^N \frac{1}{2} \delta(\zeta_i; \hat{\theta}, \hat{\mathbf{W}}_c, \hat{\mathbf{W}}_a)^2$$

This is a nonlinear optimization problem, but we may again use a methods like gradient decent in order to iteratively learn parameters that improve the optimization problem given above. Writing the gradient decent in terms of an ODE, and making some changes motivated by a stability analysis[31]. A least-squares update law with forgetting factor[40] can be formulated for the critic as follows:

$$\dot{\hat{\mathbf{W}}}_c(t) = -k_{c,1} \mathbf{\Gamma}(t) \frac{\omega(\zeta(t), t)}{\rho(\zeta(t), t)} \hat{\delta}(\zeta(t), t) - \frac{k_{c,2}}{N} \mathbf{\Gamma}(t) \sum_{i=1}^N \frac{\omega(\zeta_i(t), t)}{\rho_i(\zeta_i(t), t)} \hat{\delta}(\zeta_i(t), t) \quad (34)$$

$$\dot{\mathbf{\Gamma}}(t) = \begin{cases} \beta \mathbf{\Gamma}(t) - k_{c,1} \mathbf{\Gamma}(t) \frac{\omega(\zeta(t), t) \omega^{\top}(\zeta(t), t)}{\rho^2(\zeta(t), t)} \mathbf{\Gamma}(t) & \text{If } \|\mathbf{\Gamma}\| \leq \bar{\Gamma} \\ 0 & \text{Otherwise} \end{cases} \quad (35)$$

In critic update law above $k_{c,1}$ and $k_{c,2}$ are scalar learning rates, while $\mathbf{\Gamma}$ is an adaptive weight matrix, and β is a scalar forgetting factor, which controls how previous data samples are discounted. For brevity of notation we used the functions $\omega(\cdot)$, $\rho(\cdot)$ and

$\hat{\delta}(\cdot)$ defined as:

$$\begin{aligned}\omega(\zeta, t) &= \nabla_{\zeta} \sigma(\zeta) \left(F(\zeta; \hat{\theta}(t)) + G(\zeta) \hat{\pi}(\zeta; \hat{\mathbf{W}}_a(t)) \right) \\ \rho(\zeta, t) &= 1 + \omega^{\top}(\zeta, t) \mathbf{\Gamma}(t) \omega(\zeta, t) \\ \hat{\delta}(\zeta, t) &= \delta(\zeta; \hat{\theta}(t), \hat{\mathbf{W}}_c(t), \hat{\mathbf{W}}_a(t))\end{aligned}$$

Here, ω can be considered a regressor vector, while ρ is a normalization factor, and $\hat{\delta}$ the Bellman error.

The actor update law (36) is chosen such that it learns from the critic, while at the same time trying to stay close to the initial control law.

$$\dot{\hat{\mathbf{W}}}_a(t) = \text{proj} \left(-k_{a,1} \left(\hat{\mathbf{W}}_a(t) - \hat{\mathbf{W}}_c(t) \right) - k_{a,2} \left(\hat{\mathbf{W}}_a(t) - \mathbf{W}_0 \right) \right) \quad (36)$$

In the actor update law above, the first term will make the actor parameters follow the critic parameters, while the second term will try to keep the actor parameters close to the initial parameters \mathbf{W}_0 . $k_{a,1}$ and $k_{a,2}$ are scalar learning rates for the two terms. A smooth projection[40] is added such that the actor weights are within a predefined region, for which the control law is stable. Any smooth projection can be chosen, however we chose a projection ensuring the actor weights were bounded within a region of the initial weights \mathbf{W}_0 .

2.6 Stability analysis

For the system identification parameters θ , we consider the candidate Lyapunov function:

$$V_p(\mathbf{x}) = \tilde{\theta}^{\top} \mathbf{\Gamma}_{\theta}^{-1} \tilde{\theta}, \quad (37)$$

where $\tilde{\theta} = \hat{\theta} - \theta^*$ is the difference between the predicted and optimal model parameters. Assuming the system is time invariant, (including time invariant environmental forces in the NED frame), and given a positive definite weighting matrix $\mathbf{\Gamma}_{\theta}$. The time derivative of the candidate lyapunov function is:

$$\begin{aligned}\dot{V}_P(\mathbf{x}) &= 2\tilde{\theta}^{\top} \mathbf{\Gamma}_{\theta}^{-1} \dot{\tilde{\theta}} \\ &= 2\tilde{\theta}^{\top} \mathbf{\Gamma}_{\theta}^{-1} \mathbf{\Gamma}_{\theta} Y(\mathbf{x})^{\top} \tilde{\mathbf{x}} + \frac{2k_{\theta}}{N} \tilde{\theta}^{\top} \mathbf{\Gamma}_{\theta}^{-1} \mathbf{\Gamma}_{\theta} \sum_{i=1}^N Y^{\top}(\mathbf{x}_i) \tilde{\mathbf{x}}_i.\end{aligned} \quad (38)$$

Using the fact that: $\tilde{\mathbf{x}} = \dot{\mathbf{x}} - f_1(\mathbf{x}) - Y(\mathbf{x})\hat{\theta} - g(\mathbf{x})\tau = -Y(\mathbf{x})\tilde{\theta}$ we get:

$$\dot{V}_P(\mathbf{x}) = -2\tilde{\mathbf{x}}^{\top} \tilde{\mathbf{x}} - \frac{2k_{\theta}}{N} \tilde{\theta}^{\top} \sum_{i=1}^N (Y^{\top}(\mathbf{x}_i) Y(\mathbf{x}_i)) \tilde{\theta} \leq 0, \quad (39)$$

hence the model error $\tilde{\mathbf{x}}$ and parameter error $\tilde{\theta}$ converge exponentially to zero as $t \rightarrow \infty$. We can also note that the rate of the parameter convergence is given by the singular values of $\sum_{i=1}^N (Y^{\top}(\mathbf{x}_i) Y(\mathbf{x}_i))$.

For the RL update laws in (34), (35) and (36), it can be shown that under a number of strict assumptions, a system on the form given in (15), with an unconstrained policy, is uniformly ultimately bounded in terms of the error dynamics \mathbf{e} , as well as the weights and parameters \mathbf{W}_a , \mathbf{W}_c and $\boldsymbol{\theta}$. The stability analysis can be found in [31]. For our purposes, we further constrain the parameters \mathbf{W}_a of the feedback control law by projecting them into a region close to a known stable initial parameterization. Closed loop stability is important for assurance of the control system, this is further discussed in section 4.

2.7 Reference model

When generating a reference path, we must ensure that it is sufficiently smooth in order to be able to say something about the convergence to the path. In practice however, we may have a signal which is discrete, defining the desired pose only at certain times. In order to smooth the trajectory we therefore use a reference model, which tracks the discrete reference pose, and generates a continuous reference trajectory pose $\boldsymbol{\eta}_d = [x_d, y_d, \psi_d]^\top$ and velocity vector $\boldsymbol{\nu}_d = [u_d, v_d, r_d]^\top$. For the pose we can make a reference model on the following form:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_d \\ \ddot{\boldsymbol{\eta}}_d \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \\ -\boldsymbol{\Omega}^3 & -(2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}^2 & -(2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_d \\ \dot{\boldsymbol{\eta}}_d \\ \ddot{\boldsymbol{\eta}}_d \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{\Omega}^3 \end{bmatrix} \boldsymbol{\eta}_{ref} \quad (40)$$

Where $\boldsymbol{\Omega} = \text{diag}([\omega_1, \dots, \omega_n])$ and $\boldsymbol{\Delta} = \text{diag}([\delta_1, \dots, \delta_n])$. Choosing $\boldsymbol{\Delta} = \mathbf{I}$ ensures the reference model is critically damped, while $\boldsymbol{\Omega}$ controls the rate of convergence of the states. We must also generate the velocity vector, however based on the pose, the velocity can be calculated as:

$$\begin{aligned} \boldsymbol{\nu}_d &= \mathbf{J}^\top(\boldsymbol{\eta}_d)\dot{\boldsymbol{\eta}}_d \\ \dot{\boldsymbol{\nu}}_d &= -\mathbf{S}([0, 0, r_d]^\top)\mathbf{J}^\top(\boldsymbol{\eta}_d)\dot{\boldsymbol{\eta}}_d + \mathbf{J}^\top(\boldsymbol{\eta}_d)\ddot{\boldsymbol{\eta}}_d \end{aligned} \quad (41)$$

where $-\mathbf{S}([0, 0, r_d]^\top)\mathbf{J}^\top(\boldsymbol{\eta}_d) = \dot{\mathbf{J}}^\top(\boldsymbol{\eta}_d)$, and $\mathbf{S}(\boldsymbol{\omega})$ is the skew symmetric matrix:

$$\mathbf{S}([\omega_1, \omega_2, \omega_3]^\top) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

The reason we here use a third order filter for the reference model, is to ensure a smooth pose, velocity, and acceleration, even when a step in the reference is observed. This ensures that the feedforward control for the reference trajectory (16) can track the reference.

A block diagram of the final control structure is given in Figure 3. The diagram shows how the controller is split into a feedback control law $\boldsymbol{\pi}$, and a feedforward control law \mathbf{u}_d . Where the Reference filter is used to generate the pose and velocity reference \mathbf{x}_d , and the the data stack collected from observing vessel transitions, is used to update the parameters of the control laws.

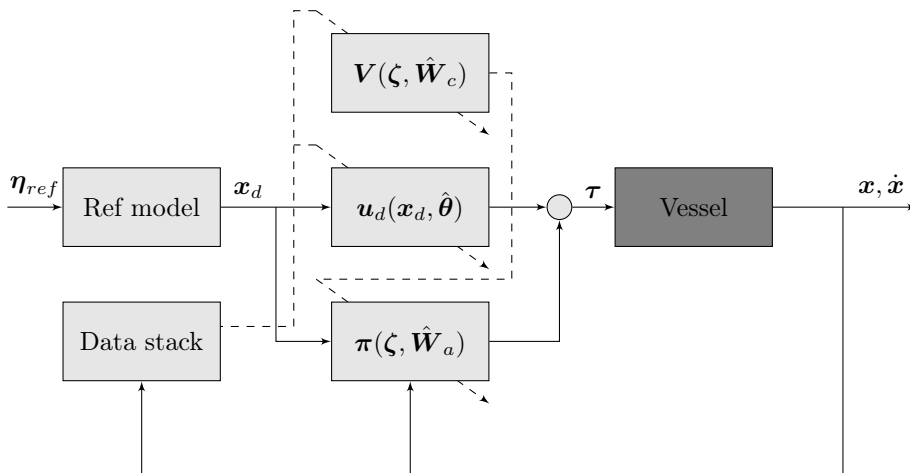


Figure 3: Block diagram of the proposed control scheme, solid lines represent state and control signals, while dashed lines represent adaption signals. Note that the underlying vessel dynamics considered are unknown, and includes thrust allocation and state estimation.

3 Experiments

In this section we present the results from simulations, and sea trials on the *ReVolt* test platform (see Figure 4), when using the control scheme proposed in the previous section. We will first present the implementation details for the control algorithm. After that we will briefly present the experimental platform, before finally presenting the simulation, and sea trial results for varying operational conditions. The experiments include both low speed dynamic positioning, and high speed trajectory tracking.

3.1 Implementation details

For the implementation the parameter update laws (32), (36), (35) and (34) were implemented with a 4th order Runge-Kutta integration scheme, with a timestep of 0.1 seconds. Additionally the reference model in (40) and (41) were implemented, also using a 4th order Runge-Kutta scheme, in order to generate the reference trajectory $\mathbf{x}_d = [\boldsymbol{\eta}_d, \boldsymbol{\nu}_d]^\top$ and its derivative $\dot{\mathbf{x}}_d = h_d = [\dot{\boldsymbol{\eta}}_d, \dot{\boldsymbol{\nu}}_d]^\top$.

For the parameterization of the system identifier, the $\boldsymbol{\theta}$ and $Y(\mathbf{x})$ were chosen as in (7) and (9), while for the actor and critic, the parameterization $\sigma(\boldsymbol{\zeta})$ was chosen as the vector of all the second order cross terms of the position and velocity error in the body

frame $\mathbf{e}^{\text{body}} = [\tilde{\boldsymbol{\eta}}^{\text{body}}, \tilde{\boldsymbol{\nu}}]$ where $\tilde{\boldsymbol{\eta}}^{\text{body}} = \mathbf{J}^\top(\boldsymbol{\eta})\tilde{\boldsymbol{\eta}}$, giving the following expression:

$$\mathbf{W}\sigma(\boldsymbol{\zeta}) = \sum_{x_i \in \mathbf{e}^{\text{body}}} \sum_{x_j \in \mathbf{e}^{\text{body}}} w_{i,j} x_i x_j \quad (42)$$

The reason that we use the error in the body frame, is the assumption that the cost is invariant to rotations when in the body frame, as this is the same frame the dynamics of the system are given in. The initial conditions for the actor and critic weights were chosen such that they matched the continuous time algebraic Riccati equation for a simplified linear model of the vessel.

For the control law, the constrained closed form controller (30) was used. And the output was scaled to fit the max thrust and torque $\bar{\boldsymbol{\tau}} = \frac{1}{\sqrt{3}}[50.0, 20.0, 32.0]^\top$ the vessel is able to produce. While $[50.0, 20.0, 32.0]^\top$ is the max force the vessel is able to produce in each direction individually, due to the coupling between thrusters, we assume the maximum thrust in any given coupled direction can be approximated by the an ellipse with axis lengths 50.0, 20.0 and 32.0. Since the proposed method only allows us to constrain thrust in each individual direction, we use the largest inner approximation of the ellipse as our thrust bound, giving the max thrust and torque as $\bar{\boldsymbol{\tau}}$ given above. It should be noted, that while this constrains the thrust, we can still not guarantee that the vessel is able to produce the desired amount of thrust as $\bar{\boldsymbol{\tau}}$ is only an inner approximation of the elliptic approximation, whereas the true thrust bound may be much more complex. It should also be noted that using the inner approximation $\bar{\boldsymbol{\tau}}$ as a bound, means we are not able to fully utilize the full thrust that the vessel has to offer. One way of solving these issues would be to include the thrust allocation as part of the problem formulation, however this is beyond the scope of this paper. It should also be noted that the desired thrust vector includes both the path tracking control law and drift correction $\boldsymbol{\tau}_{\text{Thrust}} = \mathbf{u}_d(\mathbf{x}_d; \hat{\boldsymbol{\theta}}) + \hat{\boldsymbol{\pi}}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_a)$ where the saturation is only considered in the drift controller and not the path tracking control law. This means the desired path should be generated in a way that satisfies the thrust constraints.

For the state cost function $Q(\boldsymbol{\zeta})$ a quadratic cost on the form $Q(\boldsymbol{\zeta}) = [\tilde{\boldsymbol{\eta}}^{\text{body}}, \tilde{\boldsymbol{\nu}}]^\top \mathbf{Q} [\tilde{\boldsymbol{\eta}}^{\text{body}}, \tilde{\boldsymbol{\nu}}]$ was chosen, where \mathbf{Q} is a positive definite weight matrix. Given as:

$$\mathbf{Q} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 10.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 10.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 10.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 10.0 \end{bmatrix}$$

The weight matrix is given as a mostly diagonal matrix, with a small cross term between the position error in sway direction, and heading error. The cross term is added in order to encourage the vessel to travel in the surge direction when there is a large position error, as this is the most efficient direction of travel.



Figure 4: *ReVolt* test platform courtesy of DNV GL

The data stack that was used consisted of 100 samples, and a singular value maximization scheme was implemented in order to increase the convergence rate. Using a purely singular value maximization based data selection scheme, while giving good performance on a stationary system, does not work for time varying system, and hence does not allow for estimating the slowly varying environmental forces. In order to account for this, weighting of the singular value maximization, and data sample age was used in order to save recent samples with high singular values.

3.2 Experimental platform

The *ReVolt*, shown in Figure 4, is a 1 : 20 scale model of a autonomous concept vessel developed by DNV GL in collaboration with NTNU. The model is 3 meters long, 0.72 meters wide, and weighs 257 kg. *ReVolt* has a top speed of 2 knots (approximately 1 m/s) with a total combined engine power of 360 W. The thrust configuration is given as in Figure 1, with two identical stern thrusters, and one slightly less powerful bow thruster, all of which are fully rotatable azimuth thrusters, and are controlled by an optimization based thrust allocation (TA) algorithm. The vessel state is estimated using a nonlinear observer consisting of an Extended Kalman Filter (EKF), and combines measurements from a Global Navigation Satellite System (GNSS) with Real-Time Kinematic (RTK) correction data, on board accelerometer, gyroscope and compass. This provides accurate heading and position down to $\pm 0.2^\circ$ and ± 1 cm. A description of the *ReVolt* hardware and software is given in table 1.

While the physical vessel was used for the sea trials, a high fidelity Digital Twin of *ReVolt*, developed by DNV GL, was used for simulation. The Digital Twin is based on a full 6DOF model, with parameters identified through tow-tank experiments, as well as frequency domain analysis of a 3D model of the vessel hull. The Digital Twin allowed for rapidly testing how the proposed control scheme performed under ideal conditions, as well as under different sea states, ocean currents and wind conditions.

Publications

Onboard computer:	Tank-720
Sensors:	Xsens MTI-G-710 IMU Vector VS330 GNSS Receiver
Software:	Linux Ubuntu LTS 16.04 ROS Kinetic Kame

Table 1: *ReVolt* hardware and software specifications

3.3 Simulations and Sea trials

In order to test the proposed control scheme, a number of experiments were devised. As the control scheme was built to be able to handle both high speed and low speed maneuvering, we wanted to test both, by doing low speed Dynamic Positioning (DP), as well as higher speed path tracking.

3.3.1 Dynamic Positioning (DP)

In order to test the Dynamic positioning capabilities of the control method, the four corner test seen in Figure 5 is used. The four corner DP test is used, as it shows the tracking capabilities of the vessel for individual degrees of freedom, as well as the coupled motion of all degrees of freedom, it is also worth noting that the vessel returns to the initial pose, meaning the test can easily be repeated. The four corner test starts with the vessel pointing north 0° , then performs the following commands:

1. Change position l meters due north, and come to a complete stop. This tests the surge motion of the vessel.
2. Change position l meters due east, and come to a complete stop. This tests the sway motion of the vessel.
3. Change heading 45° , and come to a complete stop. This tests the yaw motion of the vessel.
4. Change position l meters due south, and come to a complete stop. This tests the coupled surge and sway motion of the vessel.
5. Change position l meters due west, and heading to 0° and come to a complete stop. This tests coupled motion of all degrees of freedom.

For the box test we chose the box side length l to be 5 meters, and the reference path was generated by linearly interpolating the the pose between the the commands, with 55 seconds to execute each command and a 5 second pause between commands in order for the reference filter to catch up to the reference, and ensure that the vessel comes to a stop. The reference poses used for the experiments are given in Table 2.

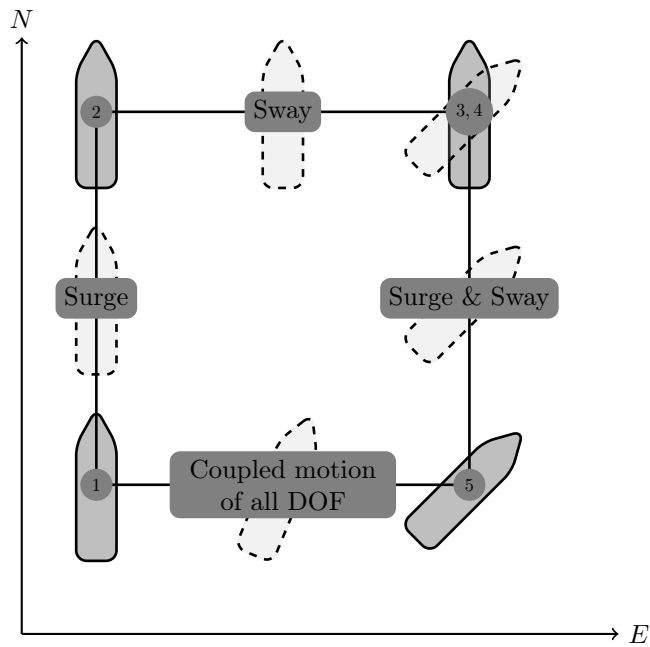


Figure 5: Four corner DP test, for testing trajectory tracking in individual, and coupled degrees of freedom

Publications

Time [s]	0	55	60	115	120	175	180	235	240	295	300
x_r [m]	0	5	5	5	5	5	5	0	0	0	0
y_r [m]	0	0	0	5	5	5	5	5	5	0	0
ψ_r [deg]	0	0	0	0	0	45	45	45	45	0	0

Table 2: Reference pose for four corner DP test, note that the reference that was used was a linear interpolation of the poses in the table

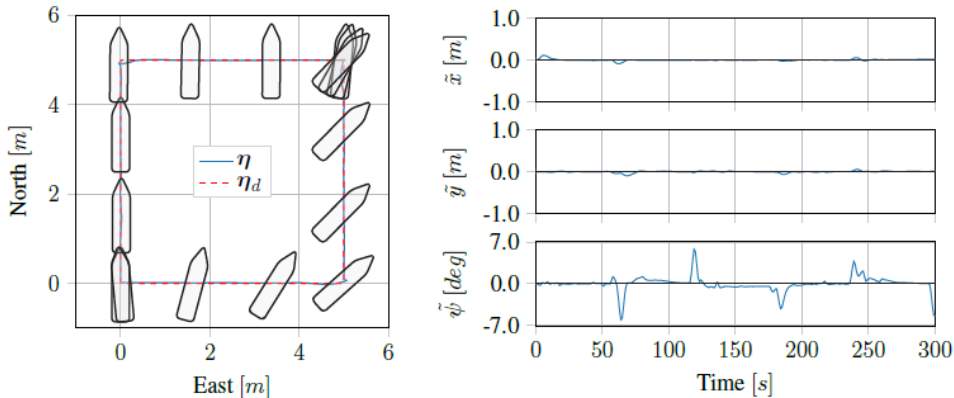


Figure 6: Simulation results for four corner DP tests

In order to evaluate the performance of the dynamic positioning, The Integral Absolute Error (IAE) given in (43) was used.

$$IAE(t) = \int_0^t \sqrt{(\bar{\eta} - \bar{\eta}_d)^\top (\bar{\eta} - \bar{\eta}_d)} dt \quad (43)$$

Where $\bar{\eta}$ and $\bar{\eta}_d$ are the normalized pose vectors, normalized between ± 5 meters in north and east direction, and $\pm 50^\circ$ in heading, giving the following.

$$\bar{\eta} = \left[\frac{x}{5}, \frac{y}{5}, \frac{\psi}{50} \right]^\top, \quad \bar{\eta}_d = \left[\frac{x_d}{5}, \frac{y_d}{5}, \frac{\psi_d}{50} \right]^\top$$

Running the proposed control scheme in simulations on the Digital-Twin of the *ReVolt* vessel, we got the trajectory and errors seen in Figure 6. For the same test performed on the physical vessel during the sea trials, we got the trajectory and errors seen in Figure 7.

3.3.2 Path tracking

For both straight line path tracking and curved path tracking, the way-points in Table 3 were used to generate a linearly, and quadratically interpolated path respectively.

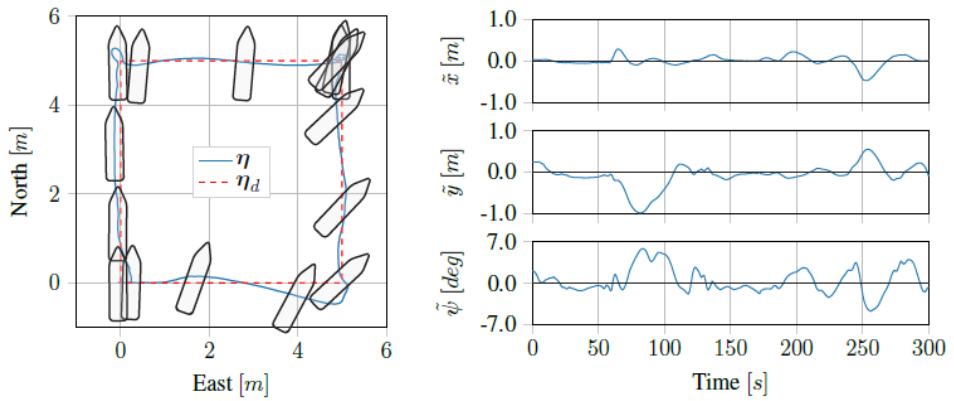


Figure 7: Sea trial results for four corner DP tests

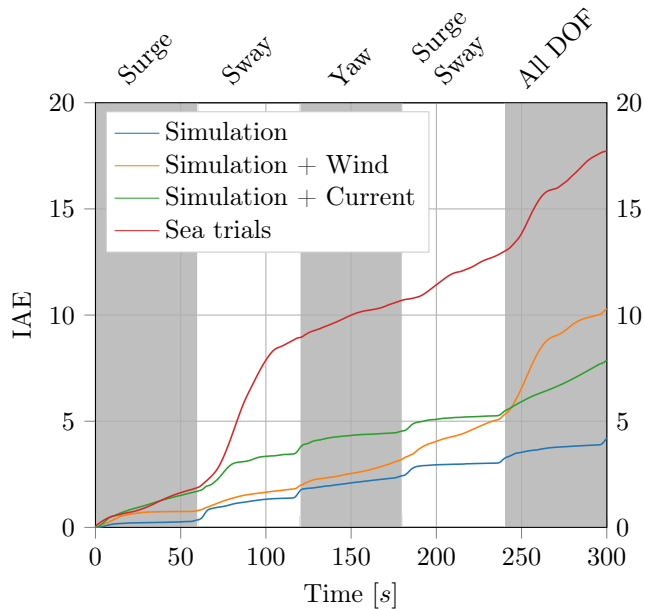


Figure 8: Integral Absolute Error (IAE) for the Dynamic Positioning task, the gray and white bands mark the different commands/phases of the four corner test.

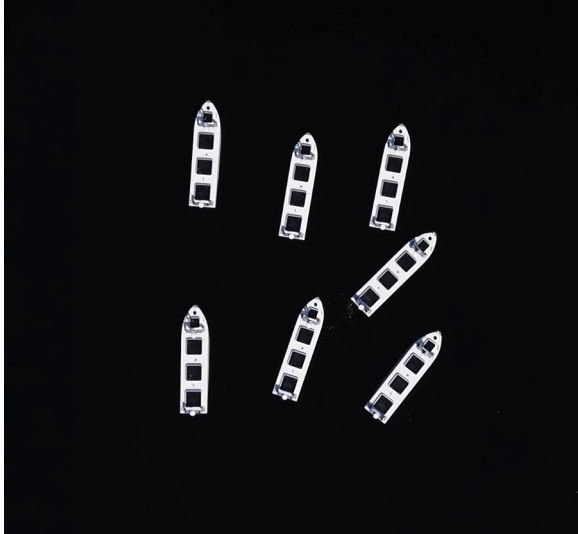


Figure 9: Time-lapse drone photo of four corner DP test. It should be noted that the time-lapse above is of an early test, where errors in the navigation system resulted in poor performance.

For the heading, the path direction was used to generate the desired heading, giving the following reference heading.

$$\psi_r = \text{atan2}(\dot{y}_r, \dot{x}_r) \quad (44)$$

In order to encourage the vessel to converge to path in the surge direction, a small cross term was added in the state cost function $Q(\zeta)$ between the heading error, and the position error in the surge direction of the body frame. The key insight here, is that the for large errors in surge, this term will encourage the vessel to turn the bow towards the desired position, meaning the vessel is encouraged to travel in the surge direction, which is the most efficient direction of travel, due to the design of the hull. For our implementation, where pose error is given in the body frame of the vessel, and the state penalty is given as a quadratic function $Q(\zeta) = \zeta^T Q \zeta$, this penalty is added by simply adding a term to the off-diagonals of Q corresponding to the cross terms between position error in the y direction, and the heading error.

Running the straight line path tracking on the Digital Twin of the *ReVolt* vessel we got the results seen in Figure 10. Running the same tests on the physical vessel, we got the results seen in Figure 11. As we can see, the proposed control scheme is able to follow the path quite well.

B. Reinforcement learning-based tracking control of USVs in ...

Time [s]	0	100	200	300
x_r [m]	0	50	100	150
y_r [m]	0	0	50	50

Table 3: Reference pose for the straight line path and curved path, note that the reference that was used for straight line path tracking was a linear interpolation of the poses, and the reference pose for the curved path, was a quadratic interpolation of the poses.

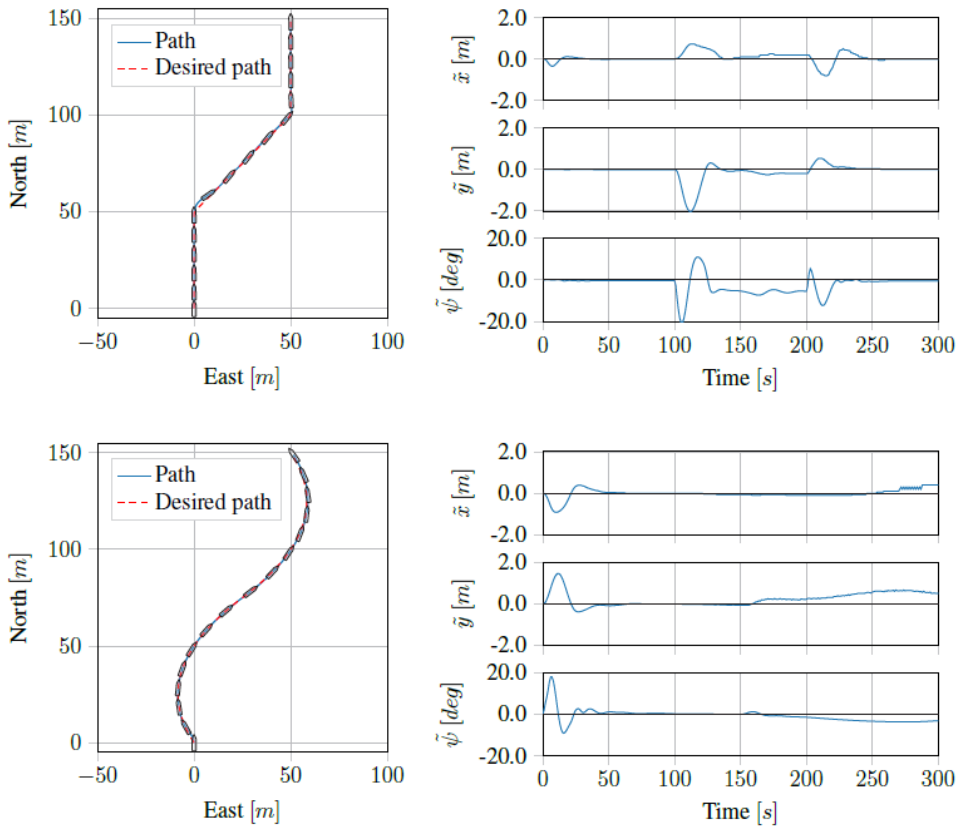


Figure 10: Simulation results for straight line and curved path tracking

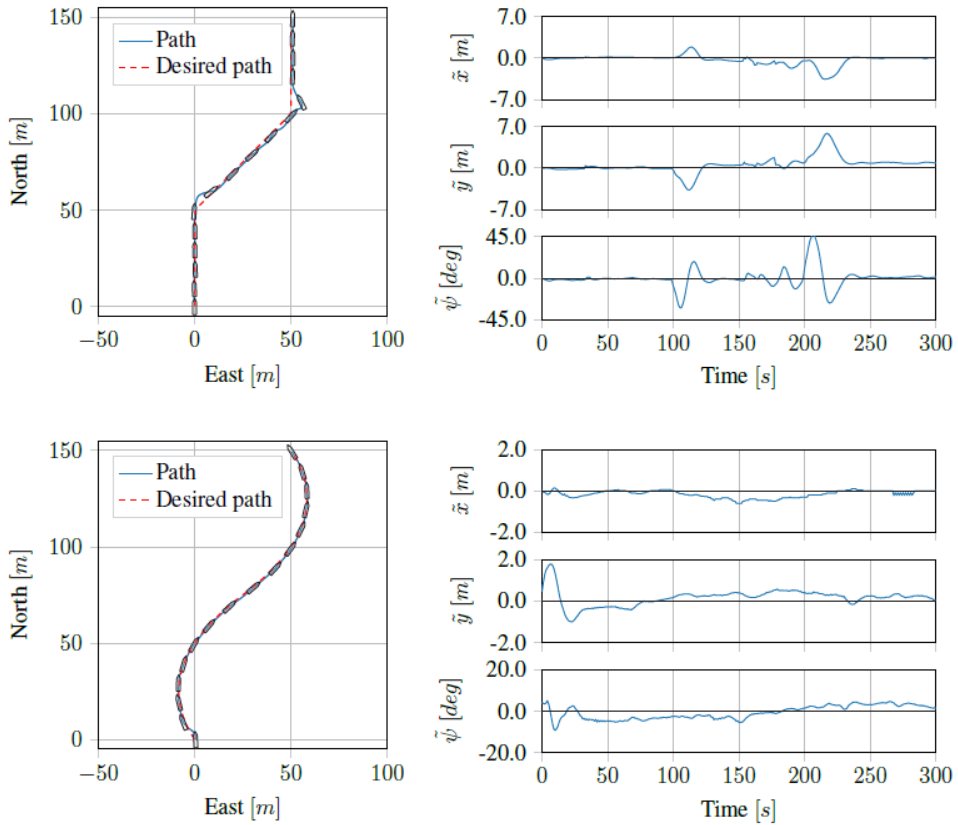


Figure 11: Sea trial results for straight line and curved path tracking

3.4 Results

Based on the results, the proposed method seems to work very well, in both simulations and the physical platform. While the simulator has been designed to perform as closely as possible to the physical platform, there are slight discrepancies that may explain the performance drop. The main factors of the performance drop is however most likely due to the measurement and observation noise that is present on the physical vessel. While the RTK GNSS is able to give a good measurement for the pose of the vessel, the estimated vessel velocities that the algorithm is dependent on become very inaccurate, especially at low speeds when the signal to noise ration becomes small. Another error source is likely the thruster dynamics. While the algorithm above assumes the desired thrust is produced immediately, in reality producing the desired thrust vector takes time, as the thrust allocation involves rotating the thrusters to a given angle, as well as spinning up to a desired motor RPM. An additional source of error may also have been a vertical stabilizer, which had recently been added to the vessel between the two rear thrusters, but had not been taken into account in the thrust allocation algorithm. Overall, the results are quite good, especially considering the size of the vessel, the relatively low thrust capability, and the precision to which the maneuvers are performed, even under the uncertainty created by the sensor noise, and environmental forces.

4 Assurance of RL-based controllers

Assurance is the structured collection of evidence supporting claims and arguments that a system is safe or fit for its intended purpose. Assurance is required to develop trustworthy systems and solutions for use in real-world applications. Principles of assurance can be found in any certification or verification framework, where claims and arguments most often can be considered requirements of verification, while evidence is the result from this verification. Two types of verification are used: 1) *Product verification*, which performs direct verification of the developed product or system and produces *primary evidence*; 2) *process verification*, which performs verification of some part of the development process and produces *circumstantial evidence*. Using established verification frameworks applied to conventional marine control systems, experience has shown what requirements and evidence are most important when verifying these conventional control systems. With novel technology, such as data-driven methods, the verification requirements and evidence that is needed for assurance are still unknown, as they pose a new set of challenges when assuring the system.

Data-driven approaches are not new, but with increasing computational power and abundance of data there has been an increasing interest in these methods. Within control theory, the field of system identification has been a key part of control engineering for many years [41, 42], and data-driven modelling for control purposes has been practiced since. Such models are typically based on the physical properties

that govern the system, and hence the parameters estimated by such methods may reflect measurable properties of the system, thus providing benchmarks for verification. However, most models represent the physical system only within certain operational limits, e.g. weather or sea states, or for certain vessel speed ranges, which restricts the validity of the models accordingly.

Contrary to the more static nature of classical data-driven approaches, where tuning the control parameters relies on offline estimation of the model parameters, in this paper the key difference is that model based-RL is used for online tuning of both the vessel model (including an estimation of unknown disturbances) and the control policy parameters. The vessel model and the control policy are based on proven methods used in the maritime industry for vessel station keeping and guidance, but there are still some key issues that must be considered. For instance, the control policy is highly dependent on an instantaneously valid vessel model, which in turn means the behavior of the vessel is highly dependant on the validity of the learned model. Both the vessel model and the control policy parameters are all continuously learned, but it is critical that all allowed parameter combinations give a sufficiently safe behavior. The proposed control scheme in this paper continuously learns and updates the parameters in order to optimize the tracking behaviour. In terms of safety, the main concern is whether the learned model and policy parameters lead to a safe and acceptable behaviour. Verifying this in a setting where the parameters are learned online is still an open problem.

Amodei, et. al.[43] discusses five basic concrete problem areas related to RL and safety, which must be taken into account for any application of RL.

1. **Avoiding Reward Hacking:** The first problem, is that of hacking or gaming the cost function. For the tracking problem in this paper, a positive definite quadratic penalty on the error dynamics is used. From control theory these methods are known to converge to the origin, i.e. where the error is zero. This means the intended behaviour is guaranteed when the policy converges to the optimal policy.
2. **Avoiding Negative Side Effects:** The second problem of avoiding negative side effects, is similar to the first, but addresses the issue of choosing the cost function such that the optimal policy does not give bad or unintended behaviour. For the method proposed in this paper, making such guarantees is quite difficult, as tuning the parameters of the quadratic cost function will still have an effect on the vessel behaviour when converging to the origin. One example of this is that we typically want the vessel to approach the path head on if we have a large deviation between the position we are at, and the desired position. Tuning the parameters of cost function in order to get this behaviour is not trivial.
3. **Scalable Oversight:** This pertains to how we can ensure that the RL agent respects aspects of the objective that are encountered infrequently. In terms of the trajectory tracking problem, the environment is quite limited, and the

objective is clearly defined, hence the problem of scalable oversight is of limited relevance to the work presented in this paper.

4. **Safe Exploration:** Exploration is necessary in order to improve performance, but bears risk, and thus performing exploration in a safe manner is not trivial. Safe exploration also encompasses the evaluation of the quality of the training data that is gathered. For a real world application, this means accounting for faulty hardware, and noisy measurements, which may lead to problematic training data. For the method proposed in this paper, where the system is learning continuously online, the problem of safe exploration and learning is highly relevant. Some measures are taken, such as using batches of training data and restricting the values that the policy parameterization may take. However, these measures only serve to mitigate potential problems, hence safe exploration and learning is still an open problem.
5. **Robustness to Distributional Shift:** This refers to how we can ensure the agent is robust to changes in the operating environment. For the proposed method, this is mostly solved by continuously learning online, which ensures that the agent learns the distributional shift when the environment changes. However as discussed in the fourth problem of safe exploration, learning online complicates the matter of ensuring data quality.

In order to produce the evidence needed for verification of data driven methods, there are two main approaches, namely *scenario based verification*, and *theoretical verification*. *Scenario based verification* would be to conduct extensive testing in representative scenarios, which in practice would mean simulation-based testing as this would be the only feasible online solution. More limited real testing should also be used, but targeted towards validating the simulation accuracy. Many RL solutions are in practice not viable without simulation-based training or development and this would mean the same tool can be utilized both for testing, and offline training. The challenge in this case would be to induce a representative set of scenarios to prove the safety or validity of the solution, and such scenario selection is an open question for testing AI or systems operating in complex environments in general. The second approach *theoretical verification*, would be to impose constraints on the RL algorithm in order to avoid unwanted behaviour. This would entail combining methods from control theory, a physical or mathematical understanding of the system, and experience or insights of the control scheme, in order to express and implement various constraints on the learnable model and policy parameters. This may not conceptually be a new approach since similar methods already exist, but this approach is difficult to use in practise, as finding parameter constraints that ensure safe operations is nontrivial.

In conclusion, an assurance framework for technologies such as the one presented in this paper is an open research question. However, one can with confidence state that it should include both process and product verification, i.e. considering not only what is developed, but also how it is developed. This would mean that adequate development and assurance processes should be developed, including verification methods that can

produce the required evidence both for efficient development, as well as assurance. In addition, novel data-driven methods should be combined with prior knowledge, verified solutions and proven physical or mathematical relations [44]. This, in order to be able to explain the behaviour, and in turn guarantee that the methods are safe and fit for the intended purpose.

5 Conclusion

The proposed method performed very well in all three tested tracking scenarios both in simulations and in sea trials. The method is also versatile, as using it on different vessels only requires knowledge of the inertia matrix, with the update laws providing a tool for learning the other model parameters, and a control policy. For future work, it may be interesting to improve and update the thrust allocation algorithm to get a smaller error between produced and desired thrust, and investigate whether this results in better accuracy. Alternatively, feedback from the thrusters can be used to get better estimates of the thrust vector for use in the data stack, and model estimation. For the value function, polynomial basis functions were used, and the estimator was linear in the parameters, which leads to a limited estimation capability. Deep learning methods could lead to more accurate value function approximation, albeit at the expense of transparency and interpretability. Procedures for implementing online assurance would add great value to current research practices. One possible way to do this is by using a new parameterization of the control law once it has been verified, either by simulation, or by constraining the parameterization to a set of parameters that is known to be safe.

Conflict of interest statement

Author Jon Arne Glomsrud and Tom Arne Pedersen were employed by the company DNV GL. The remaining 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.

Author Contributions

ABM and AML conceived the presented idea. ABM developed the theory and performed the simulations, while ABM TAP and AML carried out the experiments. JAG and TAM contributed with the assurance section from a classification society perspective. ABM AML and SG discussed the results and methods. All authors contributed to the final manuscript.

References

- [1] Thor I Fossen. *Guidance and control of ocean vehicles*. John Wiley & Sons Inc, 1994.
- [2] Kristin Y Pettersen and Olav Egeland. “Exponential stabilization of an under-actuated surface vessel”. In: *Proceedings of 35th IEEE Conference on Decision and Control*. Vol. 1. IEEE. 1996, pp. 967–972.
- [3] Thor I Fossen. “A survey on nonlinear ship control: from theory to practice”. In: *IFAC Proceedings Volumes 33.21* (2000), pp. 1–16.
- [4] Kazuhiko Hasegawa et al. “Ship auto-navigation fuzzy expert system (SAFES)”. In: *Journal of the Society of Naval Architects of Japan* 1989.166 (1989), pp. 445–452.
- [5] MR Katebi, MJ Grimble, and Y Zhang. “H- ∞ robust control design for dynamic ship positioning”. In: *IEE Proceedings-Control Theory and Applications* 144.2 (1997), pp. 110–120.
- [6] Danny Soetanto, Lionel Lapierre, and Antonio Pascoal. “Adaptive, non-singular path-following control of dynamic wheeled robots”. In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*. Vol. 2. IEEE. 2003, pp. 1765–1770.
- [7] Khac Duc Do. “Global path-following control of underactuated ships under deterministic and stochastic sea loads”. In: *Robotica* 34.11 (2016), pp. 2566–2591.
- [8] Euan W McGookin et al. “Ship steering control system optimisation using genetic algorithms”. In: *Control Engineering Practice* 8.4 (2000), pp. 429–443.
- [9] Ning Wang et al. “Adaptive robust finite-time trajectory tracking control of fully actuated marine surface vehicles”. In: *IEEE Transactions on Control Systems Technology* 24.4 (2015), pp. 1454–1462.
- [10] Thor I Fossen, Svein I Sagatun, and Asgeir J Sørensen. “Identification of dynamically positioned ships”. In: *Control Engineering Practice* 4.3 (1996), pp. 369–376.
- [11] Claes G Kallstrom. *Identification and adaptive control applied to ship steering*. Tech. rep. 1982.
- [12] Claes G Källström and Karl Johan Åström. “Experiences of system identification applied to ship steering”. In: *Automatica* 17.1 (1981), pp. 187–198.
- [13] Nikola Mišković et al. “Fast in-field identification of unmanned marine vehicles”. In: *Journal of Field Robotics* 28.1 (2011), pp. 101–120.
- [14] Shi-Lu Dai, Cong Wang, and Fei Luo. “Identification and learning control of ocean surface ship using neural networks”. In: *IEEE Transactions on Industrial Informatics* 8.4 (2012), pp. 801–810.
- [15] R Sutton, GN Roberts, and SDH Taylor. “Tuning fuzzy ship autopilots using artificial neural networks”. In: *Transactions of the Institute of Measurement and Control* 19.2 (1997), pp. 94–106.


Publications

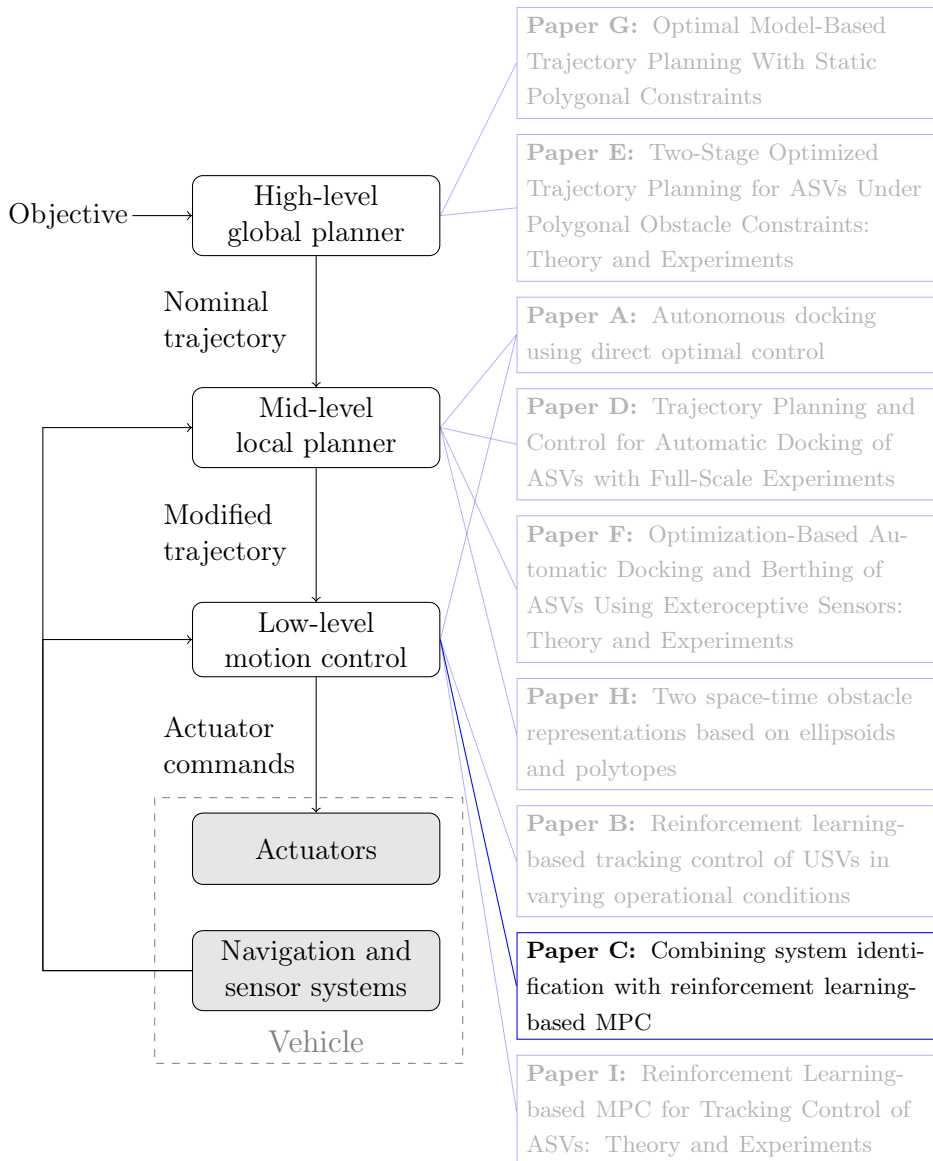
- [16] Ning Wang et al. “Global asymptotic model-free trajectory-independent tracking control of an uncertain marine vehicle: An adaptive universe-based fuzzy control approach”. In: *IEEE Transactions on Fuzzy Systems* 26.3 (2017), pp. 1613–1625.
- [17] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [18] Dimitri P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.
- [19] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).
- [20] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), p. 484.
- [21] David Silver et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. In: *arXiv preprint arXiv:1712.01815* (2017).
- [22] Marcin Andrychowicz et al. “Learning dexterous in-hand manipulation”. In: *arXiv preprint arXiv:1808.00177* (2018).
- [23] Haiqing Shen and Chen Guo. “Path-following control of underactuated ships using actor-critic reinforcement learning with MLP neural networks”. In: *Sixth International Conference on Information Science and Technology (ICIST)*. IEEE. 2016, pp. 317–321.
- [24] Lixing Zhang et al. “Neural-network-based reinforcement learning control for path following of underactuated ships”. In: *35th Chinese Control Conference (CCC)*. IEEE. 2016, pp. 5786–5791.
- [25] Yin Cheng and Weidong Zhang. “Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels”. In: *Neurocomputing* 272 (2018), pp. 63–73.
- [26] L Pham Tuyen et al. “Deep reinforcement learning algorithms for steering an underactuated ship”. In: *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE. 2017, pp. 602–607.
- [27] Runsheng Yu et al. “Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle”. In: *36th Chinese Control Conference (CCC)*. IEEE. 2017, pp. 4958–4965.
- [28] Andreas B. Martinsen and Anastasios M. Lekkas. “Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning”. In: *11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)*. 2018.
- [29] Andreas B. Martinsen and Anastasios M. Lekkas. “Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models”. In: *OCEANS 2018 MTS/IEEE Charleston*. IEEE. 2018, pp. 1–8.
- [30] Vijay R Konda and John N Tsitsiklis. “Actor-critic algorithms”. In: *Advances in neural information processing systems*. 2000, pp. 1008–1014.

- [31] Rushikesh Kamalapurkar et al. *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Springer, 2018.
- [32] Patrick Walters et al. “Online approximate optimal station keeping of a marine craft in the presence of an irrotational current”. In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 486–496.
- [33] Asgeir J Sørensen. “A survey of dynamic positioning control systems”. In: *Annual reviews in control* 35.1 (2011), pp. 123–136.
- [34] Anastasios M Lekkas and Thor I Fossen. “Trajectory tracking and ocean current estimation for marine underactuated vehicles”. In: *2014 IEEE Conference on Control Applications (CCA)*. IEEE. 2014, pp. 905–910.
- [35] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.
- [36] Kenji Doya. “Reinforcement learning in continuous time and space”. In: *Neural computation* 12.1 (2000), pp. 219–245.
- [37] Daniel Liberzon. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2011.
- [38] Girish V Chowdhary and Eric N Johnson. “Theory and flight-test validation of a concurrent-learning adaptive controller”. In: *Journal of Guidance, Control, and Dynamics* 34.2 (2011), pp. 592–607.
- [39] Girish Chowdhary and Eric Johnson. “A singular value maximizing data recording algorithm for concurrent learning”. In: *Proceedings of the 2011 American Control Conference*. IEEE. 2011, pp. 3547–3552.
- [40] Petros A Ioannou and Jing Sun. *Robust adaptive control*. Courier Corporation, 2012.
- [41] BL Ho and Rudolf E Kálmán. “Effective construction of linear state-variable models from input/output functions”. In: *at-Automatisierungstechnik* 14.1-12 (1966), pp. 545–548.
- [42] Karl Johan Åström, Torsten Bohlin, and Sture Wensmark. “Automatic construction of linear stochastic dynamic models for stationary industrial processes with random disturbances using operating records”. In: *Technical Paper TP 18: 150* (1965).
- [43] Dario Amodei et al. “Concrete problems in AI safety”. In: *arXiv preprint arXiv:1606.06565* (2016).
- [44] Simen Eldevik. *AI + SAFETY (Available at: <https://ai-and-safety.dnvgl.com/>)*. 2018. URL: <https://ai-and-safety.dnvgl.com/> (visited on 08/30/2019).

C Combining system identification with reinforcement learning-based MPC

Postprint of [18] **Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Combining system identification with reinforcement learning-based MPC”. in: *IFAC-PapersOnLine* 53.2 (2020), pp. 8130–8135. DOI: 10.1016/j.ifacol.2020.12.2294

©2020 Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. Reprinted and formatted to fit the thesis under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License 



Combining system identification with reinforcement learning-based MPC

Andreas B. Martinsen¹, Anastasios M. Lekkas¹, and Sébastien Gros¹

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Abstract: In this paper we propose and compare methods for combining system identification (SYSID) and reinforcement learning (RL) in the context of data-driven model predictive control (MPC). Assuming a known model structure of the controlled system, and considering a parametric MPC, the proposed approach simultaneously: a) Learns the parameters of the MPC using RL in order to optimize performance, and b) fits the observed model behaviour using SYSID. Six methods that avoid conflicts between the two optimization objectives are proposed and evaluated using a simple linear system. Based on the simulation results, hierarchical, parallel projection, nullspace projection, and singular value projection achieved the best performance.

Keywords: Reinforcement Learning, Model predictive control, System identification

1 Introduction

Reinforcement Learning (RL) is a powerful tool for tackling Markov Decision Processes (MDP) without depending on a model of the probability distributions underlying the state transitions. Most RL methods rely purely on observed state transitions, and realizations of the stage cost in order to increase the performance of the control policy. RL has drawn increasing attention due to recent high profile accomplishments made possible using function approximators [1]. Notable examples include performing at super human levels in games such as Go, chess and Atari [2–4], and robots learning to walk, fly without supervision, and perform complex manipulation [5–7]. Most of these recent advances have been the result of RL with Deep Learning (DL) by using Deep Neural Networks (DNNs) as function approximators. While systems controlled by DNNs show a lot of promise, they are difficult to analyze, and in turn their behaviour is difficult to certify and trust.

Model Predictive Control (MPC) is a popular approach for optimizing the closed loop performance of complex systems subject to constraints. MPC works by solving an optimal control problem at each control interval in order to find an optimal policy. The optimal control problem seeks to minimize the sum of stage costs over a horizon,

provided a model of the system and the current observed state. While MPC is a well-studied approach, and an extensive literature exists on analysing its properties [8, 9], the closed loop performance heavily relies on the accuracy of the underlying system model, which naturally presents challenges when significant unmodeled uncertainties are present.

In recent works, such as [10, 11], RL and MPC have been combined, by allowing RL to use a MPC as a function approximator. This approach allows to combine the benefits of data-driven optimization from RL with the tools available for analysing and certifying the closed loop performance of MPC. In this paper we extend the work by [10], by using a parametric MPC as a function approximator for performing RL, and combining it with on-line system identification (SYSID). The SYSID component is added with the purpose of aiding RL when there is a large model mismatch, as well as helping to improve the accuracy from the resulting MPC trajectory prediction. The main contribution of the paper are the methods for combining the competing optimization objectives of the RL and the SYSID in a way that minimizes plant model mismatch while not affecting the closed loop performance of the MPC. This paper focuses on the Q-learning approach to RL.

The paper is organized into five sections. Section 2 gives a brief overview of data-driven MPC, reinforcement learning and system identification. Section 3 describes several approaches for combining RL and SYSID in order to avoid loss in performance due to conflicting objectives. Section 4 shows simulation results for the different proposed methods, and finally, Section 5 concludes the paper.

2 Background

2.1 MPC as function approximator

As in [10], we will use a parametric optimization problem as a function approximator for reinforcement learning. Given a stage cost $L(\mathbf{x}, \mathbf{u})$ we can express the following MPC problem

$$\min_{\mathbf{x}, \mathbf{u}, \boldsymbol{\sigma}} \quad \lambda_{\boldsymbol{\theta}}(\mathbf{x}_0) + \sum_{i=0}^{N-1} \gamma^i (L(\mathbf{x}_i, \mathbf{u}_i) + L_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{u}_i) + \boldsymbol{\omega}^{\top} \boldsymbol{\sigma}_i) + \gamma^N V_{\boldsymbol{\theta}}^f(\mathbf{x}_N) \quad (1a)$$

$$\text{s.t.} \quad \mathbf{x}_{i+1} = f_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{u}_i), \quad (1b)$$

$$h(\mathbf{x}_i, \mathbf{u}_i) + h_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{u}_i) \leq \boldsymbol{\sigma}_i, \quad (1c)$$

$$\mathbf{x}_0 = \mathbf{s}, \quad (1d)$$

where we optimize the state, \mathbf{x} , action \mathbf{u} and slack variables $\boldsymbol{\sigma}$ over the time horizon N . In the optimization problem, $\lambda_{\boldsymbol{\theta}}(\mathbf{x})$ is an initial cost modifier, $L(\mathbf{x}, \mathbf{u})$ is the stage

cost, $L_{\theta}(\mathbf{x}, \mathbf{u})$ is a parametric stage cost modifier, $V_{\theta}^f(\mathbf{x})$ is a parametric terminal cost approximation, $f_{\theta}(\mathbf{x}, \mathbf{u})$ is a parametric model approximation, $h(\mathbf{x}, \mathbf{u})$ and $h_{\theta}(\mathbf{x}, \mathbf{u})$ are inequality constraints and inequality constraint modifiers, and $\gamma \in (0, 1]$ is the discount factor. The goal of the RL component is to modify the parameters θ of the parametric optimization problem in order to find a policy $\pi_{\theta}(\mathbf{x})$ that minimizes the expected cumulative discounted baseline stage cost:

$$\min_{\theta} \mathbb{E} \left[\sum_{i=0}^{\infty} \gamma^i \bar{L}(\mathbf{x}_i, \pi_{\theta}(\mathbf{x}_i)) \right],$$

where the baseline stage cost \bar{L} is defined as:

$$\bar{L}(\mathbf{x}_i, \mathbf{u}_i) = L(\mathbf{x}_i, \mathbf{u}_i) + \boldsymbol{\omega}^{\top} \max(0, h(\mathbf{x}_i, \mathbf{u}_i)).$$

Here the second term penalizes the constraint violations.

Ideally we would like strict constraints, however this would mean the MPC problem can become infeasible when model mismatch or disturbances cause constraint violations. In order to mitigate this problem, a slack penalty $\boldsymbol{\omega}$ is used, which is chosen large enough such that the constraints are only violated when the MPC becomes infeasible. For the RL, adding slack constraints is also important, as strict constraints means a penalty of ∞ for constraint violations, which most RL algorithms are not able to deal with.

2.2 Value functions and policy

Given the parametric optimization problem (1), we define the parametric action-value function as:

$$Q_{\theta}(\mathbf{s}, \mathbf{a}) = \min_{\mathbf{x}, \mathbf{u}, \boldsymbol{\sigma}} \quad (1a) \tag{2a}$$

$$\text{s.t.} \quad (1b) - (1d), \tag{2b}$$

$$\mathbf{u}_0 = \mathbf{a}, \tag{2c}$$

which trivially satisfies the fundamental equalities underlying the Bellman equation:

$$V_{\theta}(\mathbf{s}) = \min_{\mathbf{a}} Q_{\theta}(\mathbf{s}, \mathbf{a}), \tag{3}$$

$$\pi_{\theta}(\mathbf{s}) = \arg \min_{\mathbf{a}} Q_{\theta}(\mathbf{s}, \mathbf{a}). \tag{4}$$

2.3 Q-Learning

A classical RL approach is Q-Learning [12]. To perform Q-Learning for MPC we can use semi-gradient methods [13], which are based on parameter updates driven by

minimizing the temporal-difference error δ :

$$\delta_t = y_t - Q_{\boldsymbol{\theta}}(\mathbf{s}_t, \mathbf{a}_t),$$

where $y_t = \bar{L}(\mathbf{x}_t, \mathbf{u}_t) + \gamma V_{\boldsymbol{\theta}}(\mathbf{x}_{t+1})$ is the fixed target value. Defining the squared temporal-difference error as the minimization objective, and assuming that the target value is independent of the parameterization $\boldsymbol{\theta}$, we get the semi-gradient update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{u}_t), \quad (5)$$

where $\alpha > 0$ is the step-size or learning rate. For the classical semi-gradient Q-learning scheme given in (5), a second order method can be implemented by using quasi-Newton steps instead of gradient steps. This results in the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \delta \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{u}_t), \quad (6)$$

where $\mathbf{H} = \nabla_{\boldsymbol{\theta}}^2 (y_t - Q_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{u}_t))^2$ is the Hessian of the error between the targets and the action-value function. For a batch of transitions, the problem becomes a nonlinear least squares problem:

$$\min_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}), \quad \text{where} \quad \psi(\boldsymbol{\theta}) = \sum_t \delta_t^2$$

which may be solved using a Gauss-Newton method, as proposed in [14]. The modified Gauss-Newton method gives the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \underbrace{(\mathbf{J}_Q^{\top} \mathbf{J}_Q + \lambda_Q \mathbf{I})^{-1} \mathbf{J}_Q^{\top} \boldsymbol{\delta}}_{:= \Delta \boldsymbol{\theta}_Q}, \quad (7)$$

where \mathbf{J}_Q is the Jacobian of the action-value function over the batch in use, and $\boldsymbol{\delta}$ is the vector of temporal difference errors:

$$\mathbf{J}_Q = \begin{bmatrix} \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\mathbf{x}_{t,1}, \mathbf{u}_{t,1}) \\ \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\mathbf{x}_{t,2}, \mathbf{u}_{t,2}) \\ \vdots \\ \nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(\mathbf{x}_{t,B}, \mathbf{u}_{t,B}) \end{bmatrix}, \quad \boldsymbol{\delta} = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_B \end{bmatrix}$$

over the batch $\mathcal{B} = \{(\mathbf{x}_{t,i}, \mathbf{u}_{t,i}, \mathbf{x}_{t+1,i}) | i \in 1 \dots B\}$. The diagonal matrix $\lambda_Q \mathbf{I}$ is added such that $\mathbf{J}_Q^{\top} \mathbf{J}_Q + \lambda_Q \mathbf{I}$ is positive definite, and acts as a regularization of the Gauss-Newton method.

It is worth noting that the semi-gradient Q-Learning method given above yields no guarantee to find the global optimum of the parameter for nonlinear function approximators $Q_{\boldsymbol{\theta}}$. This limitation pertains to most applications of RL relying on nonlinear function approximators such as the commonly used DNN. It is also worth noting that in practice the parameterization $\boldsymbol{\theta}$ is limited. This means we in general are not able to fit the Q function globally, but rather that the formulation above fits the Q function to the distribution from which the samples are drawn.

2.4 System Identification

System identification offers a large set of tools for building mathematical models of dynamic systems, using measurements of the systems input and output signals. Based on the data-driven MPC scheme outlined in the previous section, we want an on-line parameter estimation method compatible with the parametric model. A classical SYSID approach is the Prediction Error Method (PEM) where the objective is to minimize the difference between the observed state and the predicted state given the observed transition $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$. For a parametric model approximation of the form:

$$\hat{\mathbf{x}}_{t+1} = f_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{u}_t),$$

the state error \mathbf{e} between the parametric model and the observed state can then be expressed as follows:

$$\mathbf{e}_t = \mathbf{x}_{t+1} - \hat{\mathbf{x}}_{t+1} = \mathbf{x}_{t+1} - f_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{u}_t).$$

In the simplest case, where the state vector \mathbf{x} is fully observable, PEM can be performed by minimizing the squared error between the observed state, and the predicted state:

$$\min_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}), \quad \text{where} \quad \phi(\boldsymbol{\theta}) = \mathbf{e}^\top \mathbf{e}$$

where \mathbf{e} collects a batch of measurements \mathbf{e}_i . This optimization problem can be tackled via gradient descent, giving the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \mathbf{e}^\top \mathbf{e},$$

where β is the learning rate. Since $\boldsymbol{\theta}$ are all the parameters appearing in the MPC (1), PEM is in practise only modifying the subset of the parameters $\boldsymbol{\theta}$ that appear in the parametric model. For faster learning, we propose using a second order approach, and perform quasi-Newton steps on the parameters. One such method is the modified Gauss-Newton method, which for a batch of transitions reads as follows:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \underbrace{(\mathbf{J}_f^\top \mathbf{J}_f + \lambda_f \mathbf{I})^{-1} \mathbf{J}_f^\top \mathbf{e}}_{:= \Delta \boldsymbol{\theta}_f}, \quad (8)$$

where \mathbf{J}_f is the Jacobian of parametric system model, and \mathbf{e} is the vector of model errors over a batch $\mathcal{B} = \{(\mathbf{x}_{t,i}, \mathbf{u}_{t,i}, \mathbf{x}_{t+1,i}) | i \in 1 \dots B\}$.

$$\mathbf{J}_f = \begin{bmatrix} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}_{t,1}, \mathbf{u}_{t,1}) \\ \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}_{t,2}, \mathbf{u}_{t,2}) \\ \vdots \\ \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}_{t,B}, \mathbf{u}_{t,B}) \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} \mathbf{x}_{t+1,1} - f_{\boldsymbol{\theta}}(\mathbf{x}_{t,1}, \mathbf{u}_{t,1}) \\ \mathbf{x}_{t+1,2} - f_{\boldsymbol{\theta}}(\mathbf{x}_{t,2}, \mathbf{u}_{t,2}) \\ \vdots \\ \mathbf{x}_{t+1,B} - f_{\boldsymbol{\theta}}(\mathbf{x}_{t,B}, \mathbf{u}_{t,B}) \end{bmatrix}$$

Similarly to RL, for a batch of transitions, the problem becomes a least-squares problem, fitting the observed transitions to the parametric model.

It is worth noting that for a linear parameterization, the Gauss-Newton method gives convergence to the least squares solution over the batch in one step, when $\beta = 1$ and $\lambda_f = 0$. It is also worth noting that since PEM only works on a subset of the parameters $\boldsymbol{\theta}$, \mathbf{J}_f is rank deficient and hence $\mathbf{J}_f^\top \mathbf{J}_f$ is singular by construction. Choosing the regularization term $\lambda_f > 0$, will ensure $\mathbf{J}_f^\top \mathbf{J}_f + \lambda_f \mathbf{I}$ is nonsingular and hence invertible. For the regularization parameter λ_f and λ_Q we typically want to choose a small value, to get performance close to the pure Gauss-Newton method, while only slightly regularizing in order to avoid issues arising from a singular Hessian approximation.

3 System identification for data-driven MPC

The Prediction Error Method and Reinforcement Learning are modifying the same parameter vector $\boldsymbol{\theta}$, but operate using two different objectives. RL is targeting policy optimization by minimizing the temporal difference error against a fixed target, while PEM is fitting the parametric model to the observed state transitions.

A combination of the two methods then becomes a multi-objective optimization problem. The simplest approach is to directly combine the steps from both the Q-Learning and PEM. Using the second order update laws in (7) and (8), with the parameter updates $\Delta\boldsymbol{\theta}_Q$ and $\Delta\boldsymbol{\theta}_f$ respectively, we get the following:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \Delta\boldsymbol{\theta}_Q + \beta \Delta\boldsymbol{\theta}_f \quad (9)$$

Here the step-lengths α and β can be thought of as the weighting between the Q-Learning and SYSID respectively. However, the end goal is arguably to maximize the closed-loop performance of the MPC scheme rather than minimizing the prediction error of the model, hence if the two objectives are competing, the RL objective should be prioritized. This suggests that an alternative to the naive sum of update laws approach must be considered.

3.1 Hierarchical multi-objective approach

In order to introduce a hierarchy between minimizing the PEM and RL objectives, we can consider the optimization problem:

$$\min_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}), \quad (10a)$$

$$\text{s.t. } \nabla_{\boldsymbol{\theta}} \psi(\boldsymbol{\theta}) = 0, \quad (10b)$$

which requires $\boldsymbol{\theta}$ to minimize the PEM while being a stationary point of the RL objective. If $\nabla_{\boldsymbol{\theta}}^2 \psi(\boldsymbol{\theta}) \geq 0$ is satisfied at the solution of (10), then $\boldsymbol{\theta}$ is a (local)

C. Combining system identification with reinforcement ...

minimizer of the RL objective. The KKT conditions associated to (10) read as:

$$\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}}^2\psi(\boldsymbol{\theta})\boldsymbol{\lambda} = 0, \quad (11a)$$

$$\nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}) = 0. \quad (11b)$$

A quasi-Newton step on (11) reads as:

$$\nabla_{\boldsymbol{\theta}}^2\phi(\boldsymbol{\theta})\Delta\boldsymbol{\theta} + \nabla_{\boldsymbol{\theta}}^2\psi(\boldsymbol{\theta})\boldsymbol{\lambda} = -\nabla_{\boldsymbol{\theta}}\phi(\boldsymbol{\theta}), \quad (12a)$$

$$\nabla_{\boldsymbol{\theta}}^2\psi(\boldsymbol{\theta})\Delta\boldsymbol{\theta} = -\nabla_{\boldsymbol{\theta}}\psi(\boldsymbol{\theta}). \quad (12b)$$

Let us consider a (possibly $\boldsymbol{\theta}$ -dependent) nullspace / full-space decomposition of the RL Hessian $\nabla_{\boldsymbol{\theta}}^2\psi$, i.e. \mathcal{N}, \mathcal{F} such that:

$$\nabla_{\boldsymbol{\theta}}^2\psi(\boldsymbol{\theta})\mathcal{N} = 0, \quad [\mathcal{N} \ \mathcal{F}] \text{ full rank}, \quad \mathcal{N}^\top \mathcal{F} = 0, \quad (13)$$

and the associated decomposition of the primal step $\Delta\boldsymbol{\theta}$:

$$\Delta\boldsymbol{\theta} = \mathcal{N}\boldsymbol{n} + \mathcal{F}\boldsymbol{f}. \quad (14)$$

We then observe that the primal quasi-Newton step given by (12) can be decomposed into:

$$\mathcal{N}^\top \nabla_{\boldsymbol{\theta}}^2\phi \mathcal{N} \boldsymbol{n} + \mathcal{N}^\top \nabla_{\boldsymbol{\theta}}^2\phi \mathcal{F} \boldsymbol{f} = -\mathcal{N}^\top \nabla_{\boldsymbol{\theta}}\phi, \quad (15a)$$

$$\mathcal{F}^\top \nabla_{\boldsymbol{\theta}}^2\psi \mathcal{F} \boldsymbol{f} = -\mathcal{F}^\top \nabla_{\boldsymbol{\theta}}\psi. \quad (15b)$$

One can then verify that:

$$\boldsymbol{n} = -\left(\mathcal{N}^\top \nabla_{\boldsymbol{\theta}}^2\phi \mathcal{N}\right)^\dagger \left(\mathcal{N}^\top \nabla_{\boldsymbol{\theta}}\phi - \mathcal{N}^\top \nabla_{\boldsymbol{\theta}}^2\phi \mathcal{F} \boldsymbol{f}\right), \quad (16a)$$

$$\boldsymbol{f} = -\left(\mathcal{F}^\top \nabla_{\boldsymbol{\theta}}^2\psi \mathcal{F}\right)^{-1} \mathcal{F}^\top \nabla_{\boldsymbol{\theta}}\psi, \quad (16b)$$

where \cdot^\dagger stands for the Moore-Penrose pseudo-inverse. Let us label:

$$\nabla_{\boldsymbol{\theta}}^2\phi_\perp^\dagger = \mathcal{N} \left(\mathcal{N}^\top \nabla_{\boldsymbol{\theta}}^2\phi \mathcal{N}\right)^\dagger \mathcal{N}^\top, \quad (17)$$

the pseudo-inverse of the SYSID Hessian $\nabla_{\boldsymbol{\theta}}^2\phi$ projected in the nullspace of the RL Hessian. Let us additionally label

$$\Delta\boldsymbol{\theta}_Q^H = \mathcal{F}\boldsymbol{f}, \quad \Delta\boldsymbol{\theta}_f^H = \mathcal{N}\boldsymbol{n}. \quad (18)$$

The primal step $\Delta\boldsymbol{\theta}$ then reads as:

$$\Delta\boldsymbol{\theta} = \Delta\boldsymbol{\theta}_Q^H + \Delta\boldsymbol{\theta}_f^H, \quad (19a)$$

$$\Delta\boldsymbol{\theta}_Q^H = -\mathcal{F}^\top \left(\mathcal{F}^\top \nabla_{\boldsymbol{\theta}}^2\psi \mathcal{F}\right)^{-1} \mathcal{F}^\top \nabla_{\boldsymbol{\theta}}\psi = -\nabla_{\boldsymbol{\theta}}^2\psi^\dagger \nabla_{\boldsymbol{\theta}}\psi, \quad (19b)$$

$$\Delta\boldsymbol{\theta}_f^H = -\nabla_{\boldsymbol{\theta}}^2\phi_\perp^\dagger \nabla_{\boldsymbol{\theta}}\phi + \nabla_{\boldsymbol{\theta}}^2\phi_\perp^\dagger \nabla_{\boldsymbol{\theta}}^2\phi \Delta\boldsymbol{\theta}_Q^H. \quad (19c)$$

In practice, pseudo-inverses are not always numerically stable. In order to alleviate this potential issue, we can use regularizations of the PEM and RL Hessians instead, i.e. we can select $\lambda_Q, \lambda_f > 0$ and use:

$$\Delta\theta_Q^H = -(\nabla_{\theta}^2\psi + \lambda_f\mathbf{I})^{-1}\nabla_{\theta}\psi, \quad (20a)$$

$$\nabla_{\theta}^2\phi_{\perp}^{\dagger} = \mathcal{N}\left(\mathcal{N}^{\top}(\nabla_{\theta}^2\phi + \lambda_Q\mathbf{I})\mathcal{N}\right)^{-1}\mathcal{N}^{\top}, \quad (20b)$$

together with (19c) instead of (19b) and (17). For $\lambda_{Q,f} \rightarrow 0$, (20)-(19c) asymptotically deliver the same steps as (19).

3.2 Projected steps

In this section we will discuss several projections we can perform in order to mitigate conflicts between the two optimization objectives. As discussed earlier, we typically want the RL updates to dominate, as these are directly related to the MPC closed-loop performance.

3.2.1 Parallel projection

We first consider a parallel projection, where the PEM step $\Delta\theta_f$ is projected along the RL step $\Delta\theta_Q$, giving the following projected PEM step

$$\Delta\theta_f^{\parallel} = \frac{\Delta\theta_Q\Delta\theta_Q^{\top}}{\Delta\theta_Q^{\top}\Delta\theta_Q}\Delta\theta_f$$

Intuitively, this projection can be thought of as an adaptive step-length for the RL step, i.e. the SYSID modifies the RL step-length in the direction that improves the SYSID loss.

3.2.2 Orthogonal projection

Similar to the parallel projection, we may use the orthogonal projection:

$$\Delta\theta_f^{\perp} = \left(\mathbf{I} - \frac{\Delta\theta_Q\Delta\theta_Q^{\top}}{\Delta\theta_Q^{\top}\Delta\theta_Q}\right)\Delta\theta_f$$

The orthogonal projection is dual to the parallel projection, in that it does not effect the length of the of the RL step. It may however have the drawback of working against the RL step since we do not account for the sensitivity of the RL objective in the orthogonal direction. This can be easily be seen in the case where a optimum of the RL objective is achieved, i.e. $\delta\nabla_{\theta}Q_{\theta}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$, where any PEM step will in general push the parameters away form the RL optimum.

3.2.3 Nullspace projection

Based on the heriarcical optimization problem in (19c), we see that in the particular case that $\nabla_{\theta}^2 \phi = c^{-1} \cdot \mathbf{I}$ holds, the hierarchical optimization problem reduces to simply projecting the PEM step into the nullspace of the RL step:

$$\nabla_{\theta}^2 \phi_{\perp}^{\dagger} = c \mathcal{N} \mathcal{N}^{\top}, \tag{21a}$$

$$\Delta \theta_f^H = -c \mathcal{N} \mathcal{N}^{\top} \nabla_{\theta} \phi = -\mathcal{N} \mathcal{N}^{\top} \nabla_{\theta}^2 \phi^{-1} \nabla_{\theta} \phi. \tag{21b}$$

Using this nullspace projection, with the gauss newton approach in (8) we get the following update law:

$$\Delta \theta_f^N = \mathcal{N} \mathcal{N}^{\top} \Delta \theta_f.$$

Choosing this simplified nullspace projection, the PEM step is projected into a direction for which the value function is not sensitive, hence the gradient step for the SYSID will not effect the primary goal of optimizing the RL objective. The nullspace projection may also be thought of as a regularization of the RL objective.

3.2.4 Smallest singular value projection

The nullspace projects the PEM steps into the nullspace of \mathbf{H} , i.e. the space where the singular values of \mathbf{H} are zero. As a generalization of the nullspace projection, we can project the PEM steps into the space where the Hessian is the least sensitive. Using the singular value decomposition of the Hessian of the temporal difference loss.

$$\mathbf{U} \Sigma \mathbf{V} = \mathbf{H}$$

We can extract an orthonomal basis of the p smallest singular values $\underline{\mathbf{V}}$ as the last p rows of \mathbf{V} . The projection into the p smallest singular values is then given by the following.

$$\Delta \theta_f^S = \underline{\mathbf{V}}^{\top} \underline{\mathbf{V}} \Delta \theta_f$$

We can alternatively choose p to be the number of singular values under a certain threshold. Note that if we choose p to be the number of singular values equal to zero, the projection becomes equivalent to the nullspace projection. While the nullspace projection will give no progress if \mathbf{H} is full rank, the singular value projection ensures some progress on the PEM objective, at a small cost to to the RL objective.

4 Simulations

In this section we will compare the performance of the different RL MPC modifications proposed above. In order to gauge the results we consider the following simple linear

MPC problem:

$$\min_{\mathbf{x}, \mathbf{u}, \boldsymbol{\sigma}} \sum_{i=0}^{N-1} \gamma^i \left(\|\mathbf{x}_i\|^2 + \frac{1}{2} \|\mathbf{u}_i\|^2 + \mathbf{f}^\top \begin{bmatrix} \mathbf{x}_i \\ \mathbf{u}_i \end{bmatrix} + \boldsymbol{\omega}^\top \boldsymbol{\sigma}_i \right) + V_0 + \gamma^N \mathbf{x}_N^\top \mathbf{S} \mathbf{x}_N \quad (22a)$$

$$\text{s.t.} \quad \mathbf{x}_{i+1} = \mathbf{A} \mathbf{x}_i + \mathbf{B} \mathbf{u}_i + \mathbf{b}, \quad (22b)$$

$$\begin{bmatrix} 0 \\ -1 \end{bmatrix} + \underline{\mathbf{x}} - \boldsymbol{\sigma}_i \leq \mathbf{x}_i \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \bar{\mathbf{x}} + \boldsymbol{\sigma}_i, \quad (22c)$$

$$-1 \leq \mathbf{u}_i \leq 1 \quad (22d)$$

where the parameters $\boldsymbol{\theta}$ of the optimization problem are given as:

$$\boldsymbol{\theta} = (V_0, \mathbf{f}, \mathbf{S}, \mathbf{A}, \mathbf{B}, \mathbf{b}, \underline{\mathbf{x}}, \bar{\mathbf{x}})$$

For the initial model parameter guess used in the MPC we have the following:

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0.25 \\ 0.0 & 1.0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.0312 \\ 0.25 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Additionally the terminal cost matrix \mathbf{S} was chosen as the solution to the discrete-time algebraic Riccati equation, while the rest of the parameters were initialized to zero. For the real process we used the following dynamics:

$$\mathbf{x}_{i+1} = \begin{bmatrix} 0.9 & 0.35 \\ 0.0 & 1.1 \end{bmatrix} \mathbf{x}_i + \begin{bmatrix} 0.0813 \\ 0.2 \end{bmatrix} \mathbf{u}_i + \begin{bmatrix} e_k \\ 0 \end{bmatrix}$$

where e_k is uniformly distributed on the interval $[-0.1, 0]$. The disturbance will have the effect of pushing the first state towards the lower bound such that the constraint is violated, and in turn incurring a large cost. To prevent this from happening and perform optimally, the RL algorithm must modify the parameters $\boldsymbol{\theta}$. In Figure 1 the states \mathbf{x} and action \mathbf{u} are shown for the baseline method, which only uses pure RL steps. As seen in the figure, the constraints on the first state x_1 are violated in the beginning, but by updating the parameters using RL, the system quickly learns to avoid the constraints, while at the same time staying as close to them as possible in order to minimize the discounted stage cost.

Running the on-line RL together with the proposed PEM methods, we get the results seen in Figures 2, 3 and 4. Figure 2 shows the moving average stage cost, which is a good performance measure of the closed loop performance of the MPC. From the results we see the the hierarchical, parallel, singular value and nullspace projections all converge to a slightly better performance than the baseline, while the orthogonal projection, and weighted sum of steps perform worse then the baseline. The drop in performance of the orthogonal projection, weighted sum of steps, and to a certain degree the parallel projection, is the result of competing objectives. This is also reflected in the parameter error as seen in Figure 3, where the model fit comes at

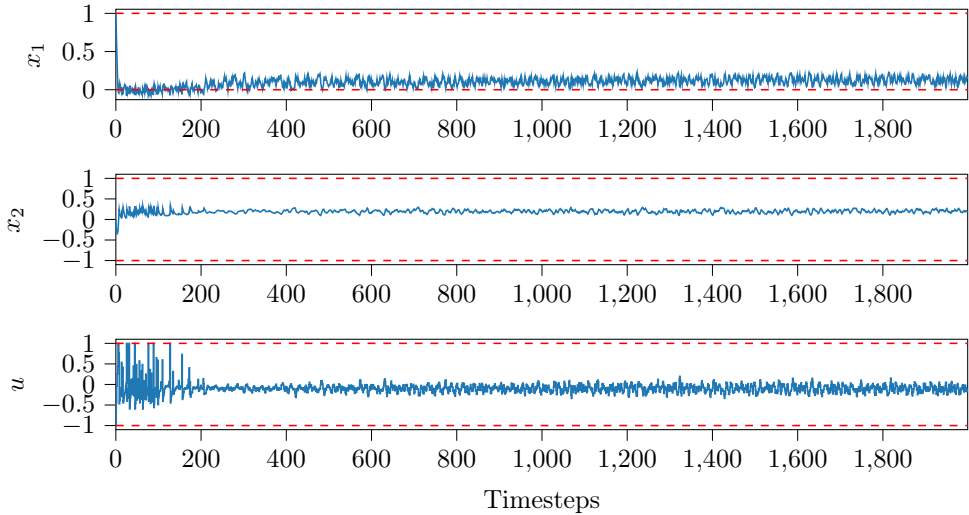


Figure 1: Baseline when using only reinforcement learning (7). The optimal unconstrained solution would be to regulate the system to $x_1 = 0$, however due to the constraint, and disturbances this is no longer the case.

the expense of closed loop performance. Looking at the temporal difference error in Figure 4, we see that most of the proposed methods give faster initial convergence. This is a result of the improved plant model mismatch which in turn gives better value function estimates from the MPC. A similar observation can be made in Figure 5 and 6, where the initial model parameters were chosen as a double integrator:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

giving a larger plant model mismatch. From the results we see that all the proposed methods have a better initial convergence of the closed loop performance, with the parallel, singular value and nullspace projections, also giving better final closed loop performance. For the parameter error, we see a significant improvement of all methods, except for the hierarchical and nullspace projection, in comparison with the baseline. The results are in line with the constraints imposed by the different projections, where the hierarchical and nullspace projection being the most conservative, and the summation of gradients being the least conservative.

5 Conclusion

In this paper we proposed and tested a number of strategies for combining RL, PEM and data-driven MPC in order to perform on-line learning and control. The main contribution is the addition of PEM as an on-line system identification method, which

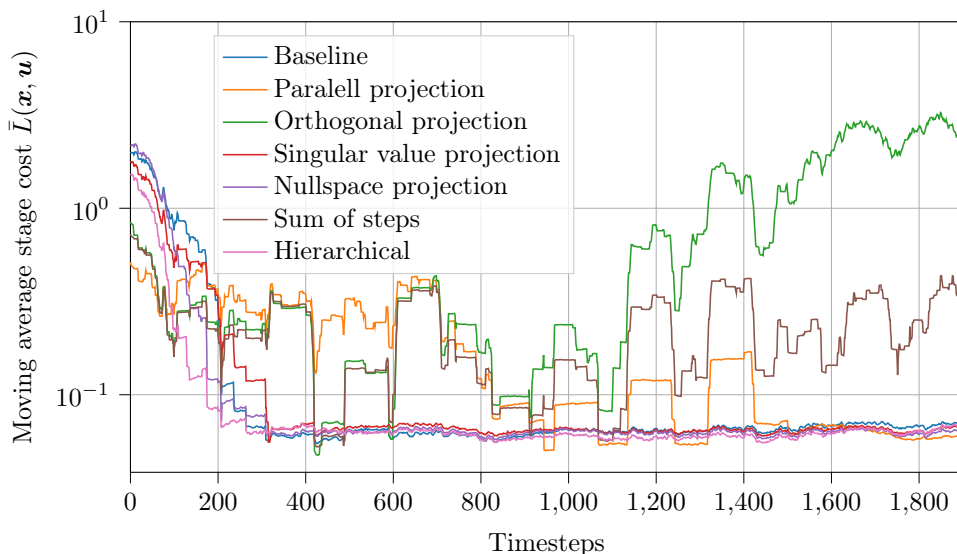


Figure 2: Moving average stage cost over 100 steps. Jumps/steps in performance indicates constraint violations, which results in a large cost.

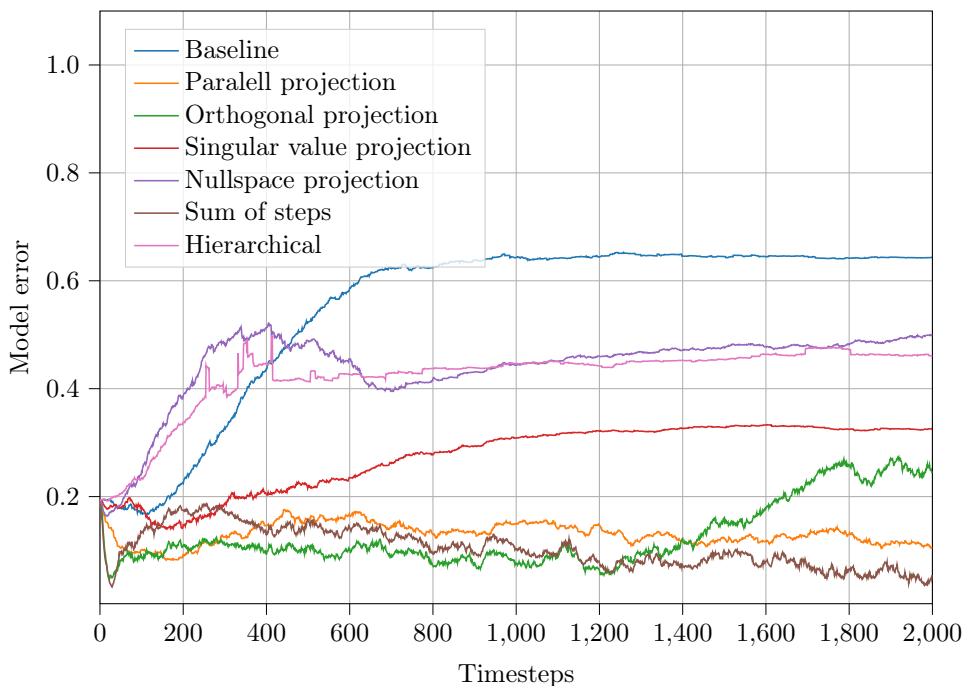


Figure 3: Norm of the parameter error for the model parameters \mathbf{A} , \mathbf{B} and \mathbf{b} . Lower error means the parametric model in the MPC is closer to the simulated model.

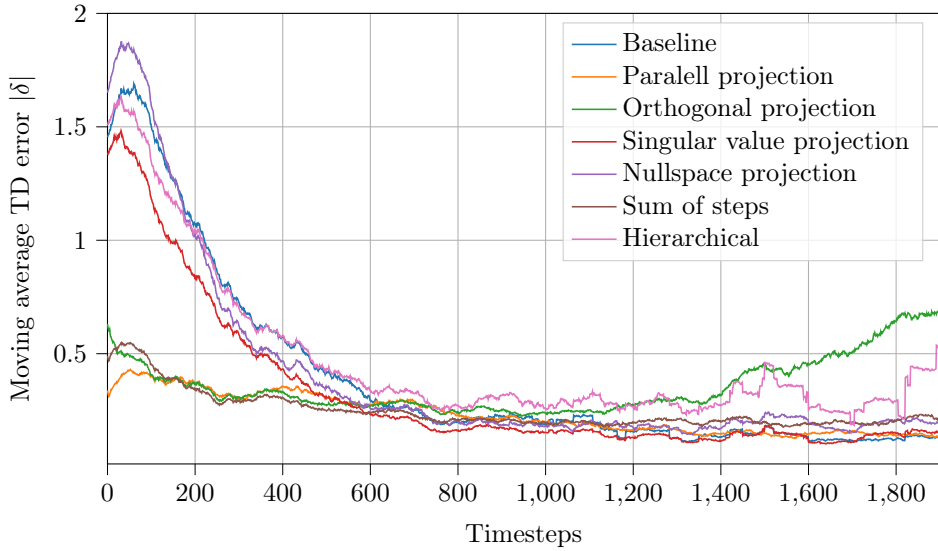


Figure 4: Moving average absolute temporal difference error $|\delta|$ over 100 steps.

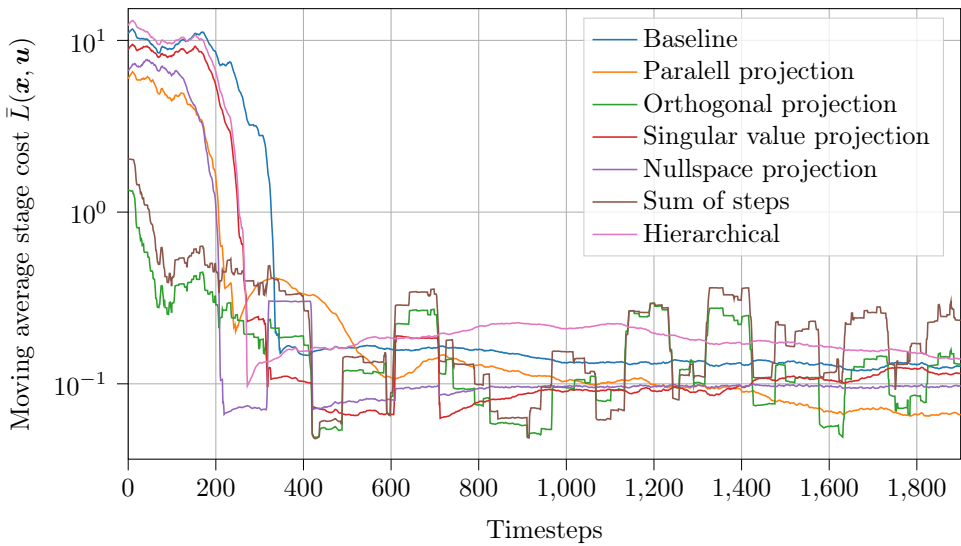


Figure 5: Moving average stage cost over 100 steps using poor initial model parameters, we see a clear improvement in performance in the early learning stage.

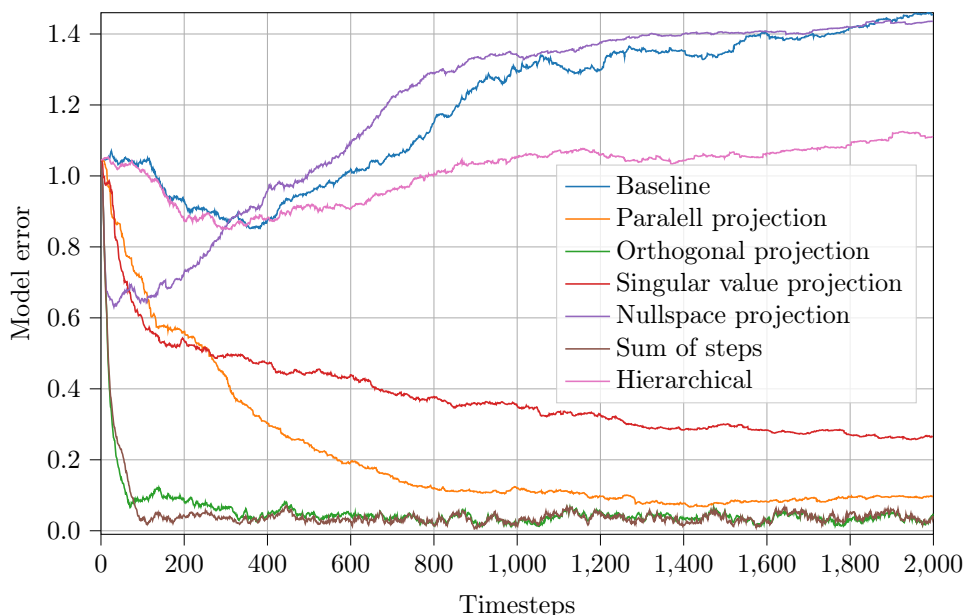


Figure 6: Norm of the parameter error for the parametric model with poor initial model parameters.

is added in order to aid the RL when there is a large plant model mismatch, as well as help to get better accuracy from the resulting MPC trajectory prediction. The proposed parallel, singular value and nullspace projection methods show promising results in terms of decreasing plant model miss-match, and giving slightly better closed loop MPC performance than using pure RL, while the orthogonal projection, and sum of steps resulted in improved model fit, however at the cost of closed loop performance of the proposed MPC scheme. In conclusion, combing PEM with RL, can give better initial learning when we do not have a good initial guess for the parameters, as well as lead to better overall performance of the closed loop MPC, without significant additional computational overhead.


For future work, it is of interest to look at methods for adaptively changing the step-length of the two objectives. For example choosing a step-length β dependant on the RL step, may help mitigate the problem of competing objectives, and in turn improve the performance of the proposed methods. Combining the proposed method with policy gradient, is also an area of interest, as policy gradient methods offer a way of directly optimizing the policy.

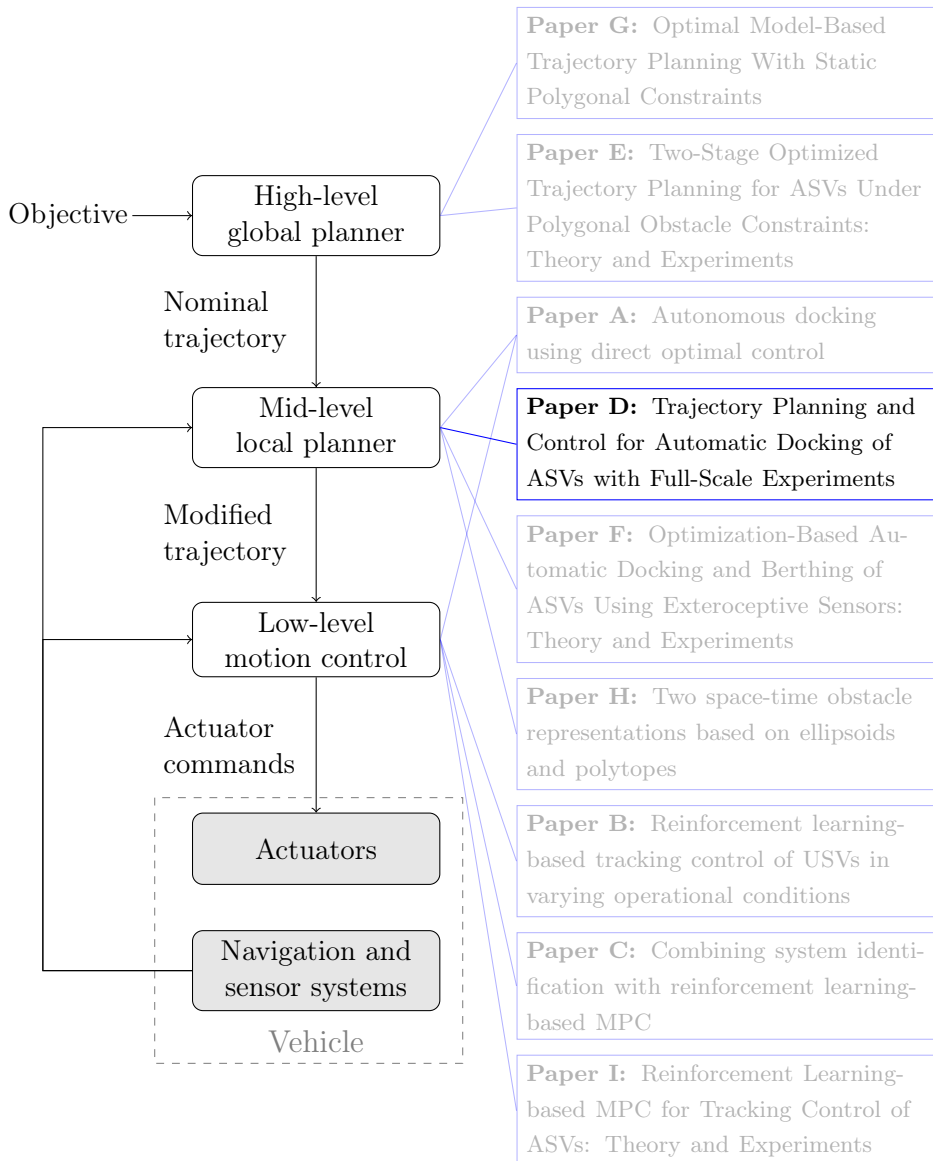
References

- [1] Lucian Busoniu et al. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.
- [2] David Silver et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. In: *arXiv preprint arXiv:1712.01815* (2017).
- [3] David Silver et al. “Mastering the game of go without human knowledge”. In: *Nature* 550.7676 (2017), p. 354.
- [4] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).
- [5] Shouyi Wang, Wanpracha Chaovaitwongse, and Robert Babuska. “Machine learning algorithms in bipedal robot control”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.5 (2012), pp. 728–743.
- [6] Pieter Abbeel et al. “An application of reinforcement learning to aerobatic helicopter flight”. In: *Advances in neural information processing systems*. 2007, pp. 1–8.
- [7] Marcin Andrychowicz et al. “Learning dexterous in-hand manipulation”. In: *arXiv preprint arXiv:1808.00177* (2018).
- [8] David Q Mayne et al. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814.
- [9] James B Rawlings and Rishi Amrit. “Optimizing process economic performance using model predictive control”. In: *Nonlinear model predictive control*. Springer, 2009, pp. 119–138.
- [10] Sébastien Gros and Mario Zanon. “Data-driven Economic NMPC using Reinforcement Learning”. In: *IEEE Transactions on Automatic Control* (2019).
- [11] Mario Zanon and Sébastien Gros. “Safe Reinforcement Learning Using Robust MPC”. In: *arXiv preprint arXiv:1906.04005* (2019).
- [12] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. In: *PhD thesis, Cambridge University* (1989).
- [13] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [14] Mario Zanon, Sébastien Gros, and Alberto Bemporad. “Practical reinforcement learning of stabilizing economic MPC”. In: *arXiv preprint arXiv:1904.04614* (2019).

D Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments

Postprint of [19] Glenn Bitar, **Andreas B Martinsen**, Anastasios M Lekkas, and Morten Breivik. “Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 14488–14494. DOI: 10.1016/j.ifacol.2020.12.1451

©2020 Glenn Bitar, Andreas B Martinsen, Anastasios M Lekkas, and Morten Breivik. Reprinted and formatted to fit the thesis under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License 



Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments*

Glenn Bitar¹, Andreas B. Martinsen¹, Anastasios M. Lekkas¹, and Morten Breivik¹

¹Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

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:

*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.

Publications

- 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 [1], Teo, Goh, and Chai [2] and Hong et al. [3], 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 [4] 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, Uchida, and Okazaki [5] 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 Wärtsilä¹ and Rolls-Royce² (now Kongsberg Maritime). Details about the methods used in these approaches are unavailable to the public.

The docking method from [6] 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 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 [6] 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.

¹Wärtsilä press release: <https://www.wartsila.com/twentyfour7/innovation/look-ma-no-hands-auto-docking-ferry-successfully-tested-in-norway>.

²Rolls-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>.



Figure 1: The experimental autonomous urban passenger ferry *milliAmpere*, developed at NTNU, moored near Brattøra in Trondheim, Norway.

- 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 [6], 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.

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

Publications

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

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 [7].

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 [6]. 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:

$$\min_{\mathbf{x}_p(\cdot), \mathbf{u}_p(\cdot), \mathbf{s}(\cdot)} \int_{t_0}^{t_0+T} \left(F(\mathbf{x}_p(t), \mathbf{u}_p(t)) + \mathbf{k}_s^\top \mathbf{s}(t) \right) dt \quad (1a)$$

³Autoferry website: <https://www.ntnu.edu/autoferry>.

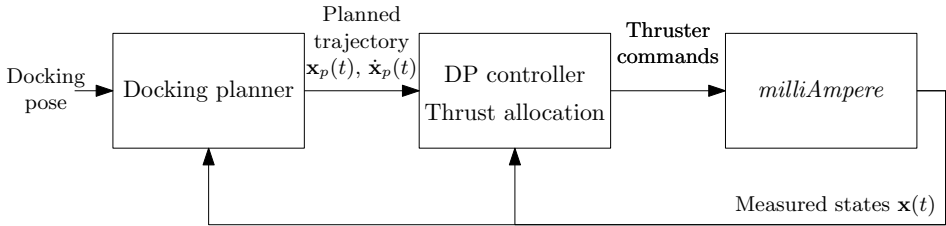


Figure 2: Block diagram of the docking system setup. The DP controller and thrust allocation are existing functions on *milliAmpere*.

subject to

$$\dot{\mathbf{x}}_p(t) = \mathbf{f}(\mathbf{x}_p(t), \mathbf{u}_p(t)) \quad \forall t \in [t_0, t_0 + T] \quad (1b)$$

$$\mathbf{h}(\mathbf{x}_p(t), \mathbf{u}_p(t)) - \mathbf{s}(t) \leq \mathbf{0} \quad \forall t \in [t_0, t_0 + T] \quad (1c)$$

$$\mathbf{s}(t) \geq \mathbf{0} \quad \forall t \in [t_0, t_0 + T] \quad (1d)$$

$$\mathbf{x}_p(t_0) = \mathbf{x}(t_0). \quad (1e)$$

The planned states are denoted $\mathbf{x}_p = [\boldsymbol{\eta}_p^\top, \boldsymbol{\nu}_p^\top]^\top$, 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 $\mathbf{x}_d = [\boldsymbol{\eta}_d^\top, \mathbf{0}_3^\top]^\top$ while avoiding collisions, where $\boldsymbol{\eta}_d = [x_d, y_d, \psi_d]^\top$ is referred to as the *docking pose*. The vector $\mathbf{x}(t_0)$ is the vessel's measured state at time t_0 . The planning horizon is $T = 120$ s.

The input vector $\mathbf{u}_p = [f_{x1}, f_{y1}, f_{x2}, f_{y2}]^\top$ 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 $\mathbf{x}_p(\cdot)$, inputs $\mathbf{u}_p(\cdot)$ and slack variables $\mathbf{s}(\cdot)$. It consists of a cost-to-go function $F(\mathbf{x}_p(t), \mathbf{u}_p(t))$, as well as a cost-to-go on the slack variables $\mathbf{k}_s^\top \mathbf{s}(t)$ with the elements of \mathbf{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

$$\begin{aligned}
 F(\mathbf{x}_p(t), \mathbf{u}_p(t)) = & \\
 & H\left(\begin{bmatrix} x_p(t) - x_d \\ y_p(t) - y_d \end{bmatrix}\right) + \\
 & 20(1 - \cos(\psi_p(t) - \psi_d)) + \\
 & 10v_p(t)^2 + 10r_p(t)^2 + \\
 & \mathbf{u}_p(t)^\top \mathbf{u}_p(t) / m_{11}^2,
 \end{aligned} \tag{2}$$

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

$$H(\mathbf{a}) = \delta^2 \left(\sqrt{1 + \frac{\mathbf{a}^\top \mathbf{a}}{\delta^2}} - 1 \right) \tag{3}$$

with $\delta = 10$ 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.

3.2 Vessel model

Equation (1b) is a simplified model of the vessel dynamics. A diagonalized version of the surge-decoupled model in [7] is used, with details found in Appendix A. The kinematic and kinetic models are concatenated to

$$\dot{\mathbf{x}}_p = \mathbf{f}(\mathbf{x}_p, \mathbf{u}_p) = \begin{bmatrix} \mathbf{R}(\psi_p) \boldsymbol{\nu}_p \\ (\mathbf{S} \mathbf{M}_p)^{-1} (-\mathbf{C}_p(\boldsymbol{\nu}_p) \boldsymbol{\nu}_p - \mathbf{D}_p(\boldsymbol{\nu}_p) \boldsymbol{\nu}_p + \boldsymbol{\tau}_p(\mathbf{u}_p)) \end{bmatrix}, \tag{4}$$

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

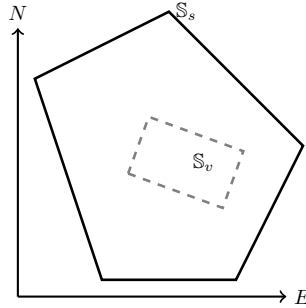


Figure 3: Spatial constraints illustration.

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

$$\mathbb{S}_s = \left\{ \mathbf{p} \in \mathbb{R}^2 \mid \mathbf{A}_s \mathbf{p} \leq \mathbf{b}_s \right\}, \quad (5)$$

where $\mathbf{A}_s \in \mathbb{R}^{k \times 2}$ and $\mathbf{b}_s \in \mathbb{R}^k$ and k is the number of vertices in the convex set. This results in the collision avoidance constraint being equivalent to

$$\mathbf{A}_s \left(\mathbf{R}_2(\psi_p(t)) \mathbf{v} + \begin{bmatrix} x_p(t) \\ y_p(t) \end{bmatrix} \right) \leq \mathbf{b}_s \forall \mathbf{v} \in \text{Vertex}(\mathbb{S}_v). \quad (6)$$

The rotation matrix $\mathbf{R}_2(\psi_p(t))$ is equal to the upper-left $\mathbb{R}^{2 \times 2}$ of (14) 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:

$$f_{xi}(t)^2 + f_{yi}(t)^2 \leq f_{\max}^2, \quad i \in \{1, 2\}. \quad (7)$$

There are also limits on the states \mathbf{x}_p , i.e.

$$\mathbf{x}_{lb} \leq \mathbf{x}_p(t) \leq \mathbf{x}_{ub}, \quad (8)$$

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 ($\pm 1.0 \text{ m s}^{-1}$) and the yaw rate ($\pm 5^\circ \text{ s}^{-1}$).

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:

$$\boldsymbol{\tau}_c(t) = \boldsymbol{\tau}_{\text{ff}}(t) + \boldsymbol{\tau}_{\text{fb}}(t). \quad (9)$$

The feed-forward term is

$$\boldsymbol{\tau}_{\text{ff}}(t) = \mathbf{M}_p \dot{\boldsymbol{\nu}}_p(t) + \mathbf{D}_p(\boldsymbol{\nu}_p(t)) \boldsymbol{\nu}_p(t), \quad (10)$$

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

$$\boldsymbol{\tau}_{\text{fb}}(t) = -\mathbf{R}(\psi(t))^\top \cdot \left(\mathbf{K}_p \tilde{\boldsymbol{\eta}}(t) + \int_0^t \mathbf{K}_i \tilde{\boldsymbol{\eta}}(\tau) d\tau + \mathbf{K}_d \dot{\tilde{\boldsymbol{\eta}}}(t) \right), \quad (11)$$

with $\tilde{\boldsymbol{\eta}}(t) = \boldsymbol{\eta}(t) - \boldsymbol{\eta}_p(t)$. The controller gains are $\mathbf{K}_p = \text{diag}\{100, 100, 200\}$, $\mathbf{K}_i = \text{diag}\{10, 10, 20\}$ and $\mathbf{K}_d = \text{diag}\{1000, 1000, 1500\}$ 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 $\pm[150 \text{ N}, 150 \text{ N}, 200 \text{ N m}]^\top$.

The control command $\boldsymbol{\tau}_c(t)$ is sent to *milliAmpere*'s thrust allocation system, detailed in [8], 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 m s^{-1} to 3 m s^{-1} and rare gusts of 5 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

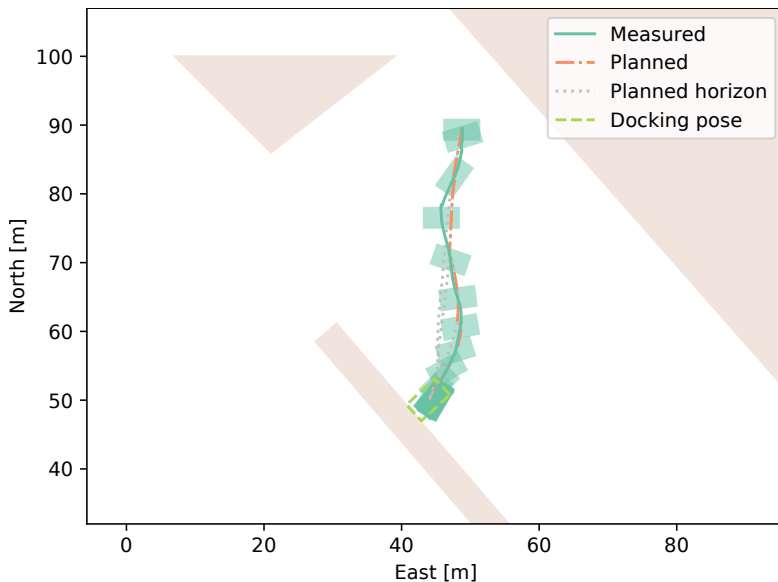


Figure 4: Overview of *milliAmpere*'s trajectory during a docking experiment. The vessel's pose is depicted at 5 s intervals with green rectangles. The measured position is drawn in solid green, while the active planned reference is in dash-dotted orange. The dotted gray lines show the trajectory planner's 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.

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. 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 the vessel is able to track the commanded trajectory well, since the speed

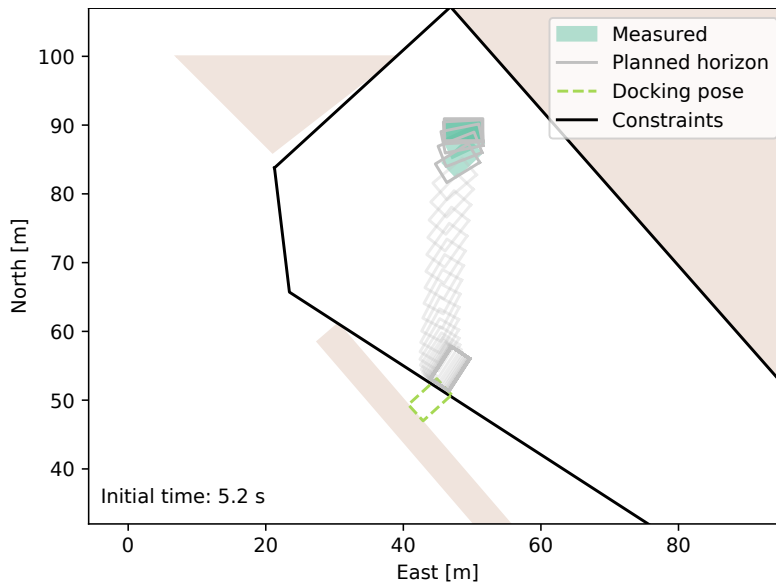


Figure 5: Planned and measured positions from the first step of the planner. In 2 s intervals, the plot shows the entire planned pose trajectory as gray outlines, and the first 10 s of the measured pose as green rectangles. The black solid polyhedron shows the current convex area that represents the spatial constraints from (6).

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.

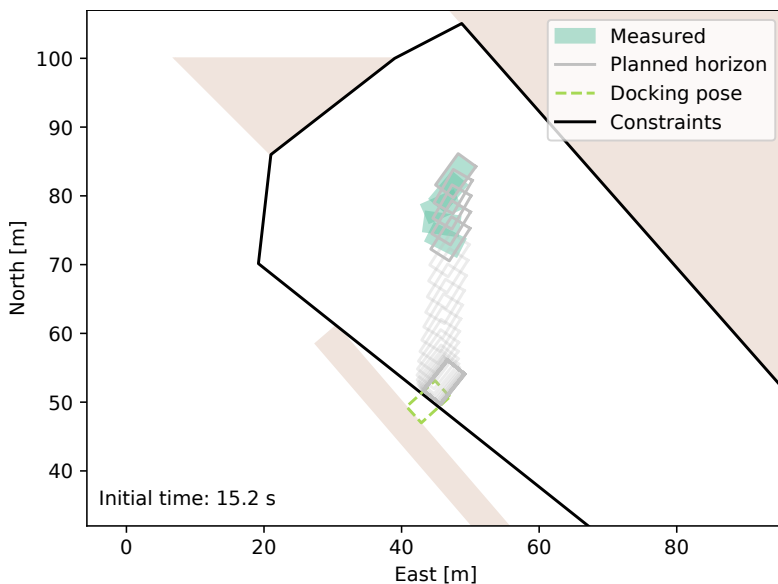


Figure 6: As Figure 5, but at the second planning step. In this step we see that the tracking controller struggles to follow the heading commands. We believe this is due to *milliAmpere*'s lack of natural stability in heading, as well as due to poor performance of the DP controller at high velocities.

5 Conclusions and future work

We have demonstrated the capabilities for docking an ASV using an OCP-based trajectory planner in combination 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:

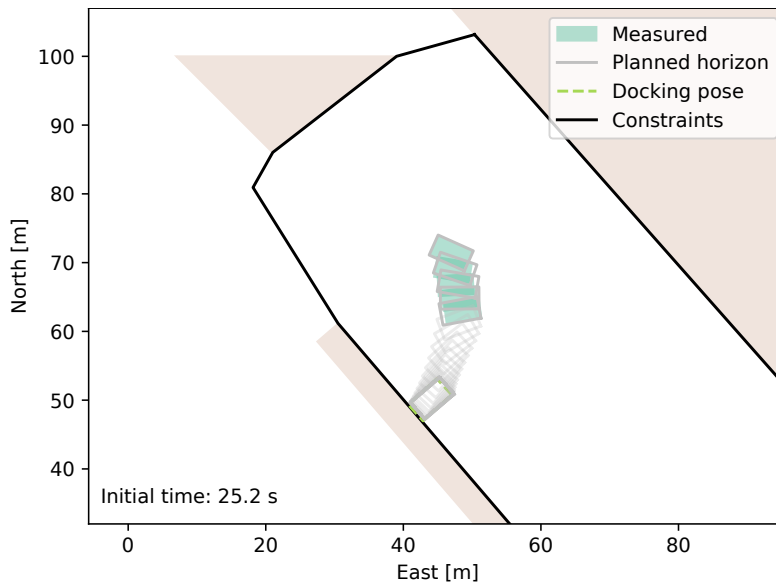


Figure 7: As Figure 5, but at the third planning step. At this slow speed, the tracking controller is able to follow the planned trajectory well.

- 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 [9], which can bring the vessel to a location suitable for docking.

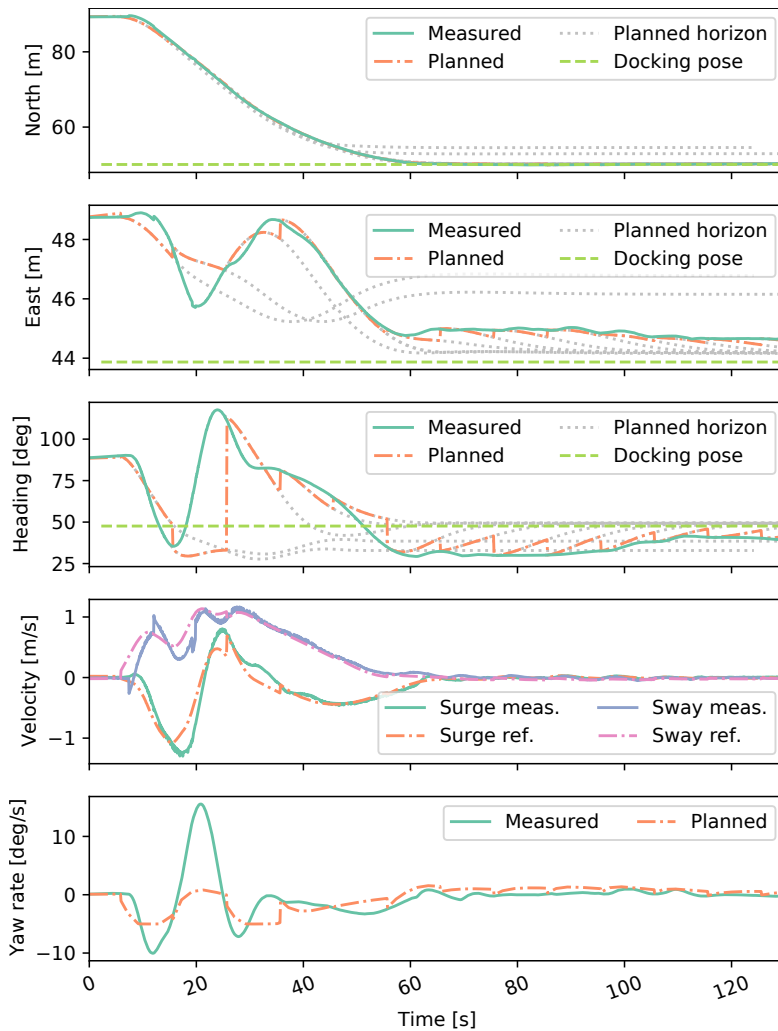


Figure 8: Measured pose and velocity states during the docking experiments, along with reference trajectories and docking pose. As in Figure 4, we include the full horizon of the planned trajectories in dotted gray lines.

References

- [1] G. J. S. Rae and S. M. Smith. “A Fuzzy Rule Based Docking Procedure For Autonomous Underwater Vehicles”. In: *Proceedings of OCEANS, Newport, RI, USA* (Oct. 26, 1992). Newport, RI, USA, 1992.

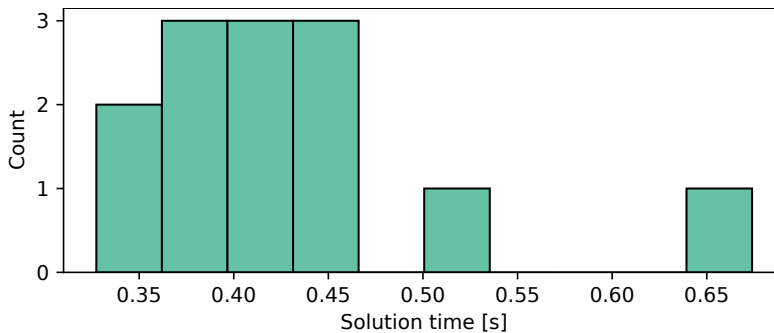


Figure 9: Histogram of solution times of the trajectory planner.

- [2] Ken Teo, Benjamin Goh, and Oh Kwee Chai. “Fuzzy Docking Guidance Using Augmented Navigation System on an AUV”. In: *IEEE Journal of Oceanic Engineering* 40.2 (2015), pp. 349–361.
- [3] Young-Hwa Hong et al. “Development of the homing and docking algorithm for AUV”. In: *Proceedings of the 13th International Offshore and Polar Engineering Conference, Honolulu, Hawaii, USA*. 2003.
- [4] Van Luong Tran and Namkyun Im. “A study on ship automatic berthing with assistance of auxiliary devices”. In: *International Journal of Naval Architecture and Ocean Engineering* 4.3 (Sept. 2012), pp. 199–210.
- [5] Naoki Mizuno, Yosuke Uchida, and Tadatsugi Okazaki. “Quasi Real-Time Optimal Control Scheme for Automatic Berthing”. In: *Proceedings of the 10th IFAC MCMC, Copenhagen, Denmark*. 2015, pp. 305–312.
- [6] Andreas B. Martinsen, Anastasios M. Lekkas, and Sebastien Gros. “Autonomous docking using direct optimal control”. In: *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*. arXiv:1910.11625 [eess.SY]. 2019, pp. 97–102. arXiv: 1910.11625 [eess.SY].
- [7] Anders Aglen Pedersen. “Optimization Based System Identification for the milliAmpere Ferry”. MA thesis. Trondheim, Norway: Norwegian University of Science and Technology (NTNU), 2019.
- [8] Tobias R. Torben, Astrid H. Brodtkorb, and Asgeir J. Sørensen. “Control allocation for double-ended ferries with full-scale experimental results”. In: *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*. 2019, pp. 45–50.
- [9] Glenn Bitar et al. “Warm-Started Optimized Trajectory Planning for ASVs”. In: *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*. arXiv:1907.02696 [eess.SY]. 2019, pp. 308–314. arXiv: 1907.02696 [eess.SY].

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 [7]. The models use the state vector

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\eta}^\top & \boldsymbol{\nu}^\top \end{bmatrix}^\top, \quad (12)$$

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

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu}, \quad (13)$$

where

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (14)$$

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 [7]. That model has the form

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}(\boldsymbol{\alpha}, \mathbf{n}), \quad (15)$$

where $\mathbf{M} \in \mathbb{R}^{3 \times 3}$ is the positive definite system inertia matrix, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$ is the skew symmetric Coriolis and centripetal matrix, and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$ 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 $\mathbf{n} = [n_1, n_2]^\top$. These values are modeled dynamically based on commanded values, with details in [7].

A.2 Planning model

For the dynamic constraints in the OCP, we use a simplified version of (15):

$$\mathbf{S}\mathbf{M}_p\dot{\boldsymbol{\nu}}_p + \mathbf{C}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p + \mathbf{D}_p(\boldsymbol{\nu}_p)\boldsymbol{\nu}_p = \boldsymbol{\tau}_p(\mathbf{u}_p), \quad (16)$$

where \mathbf{M}_p , \mathbf{C}_p and \mathbf{D}_p are diagonalized versions of \mathbf{M} , \mathbf{C} and \mathbf{D} from (15), respectively. The matrices are

$$\mathbf{M}_p = \text{diag}\{m_{11}, m_{22}, m_{33}\} > 0, \quad (17)$$

$$\mathbf{C}_p(\boldsymbol{\nu}_p) = \begin{bmatrix} 0 & 0 & -m_{22}v_p \\ 0 & 0 & m_{11}u_p \\ m_{22}v_p & -m_{11}u_p & 0 \end{bmatrix} \quad (18)$$

and

$$\mathbf{D}_p(\boldsymbol{\nu}_p) = \text{diag}\{d_{11}(u_p), d_{22}(v_p), d_{33}(r_p)\} > 0, \quad (19)$$

where

$$d_{11}(u_p) = -X_u - X_{|u|u}|u_p| - X_{u^3}u_p^2 \quad (20a)$$

$$d_{22}(v_p) = -Y_v - Y_{|v|v}|v_p| - Y_{v^3}v_p^2 \quad (20b)$$

$$d_{33}(r_p) = -N_r - N_{|r|r}|r_p|. \quad (20c)$$

The coefficient matrix

$$\mathbf{S} = \text{diag}\{2.5, 2.5, 5.0\} \quad (21)$$

is factored into (16) to amplify the inertia, making the planned trajectories more sluggish.

The dynamic thruster model from [7] 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:

$$\mathbf{u}_p = [f_{x1} \quad f_{y1} \quad f_{x2} \quad f_{y2}]^\top, \quad (22)$$

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

$$\boldsymbol{\tau}_p(\mathbf{u}_p) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & l_1 & 0 & l_2 \end{bmatrix} \mathbf{u}_p. \quad (23)$$

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 (15):

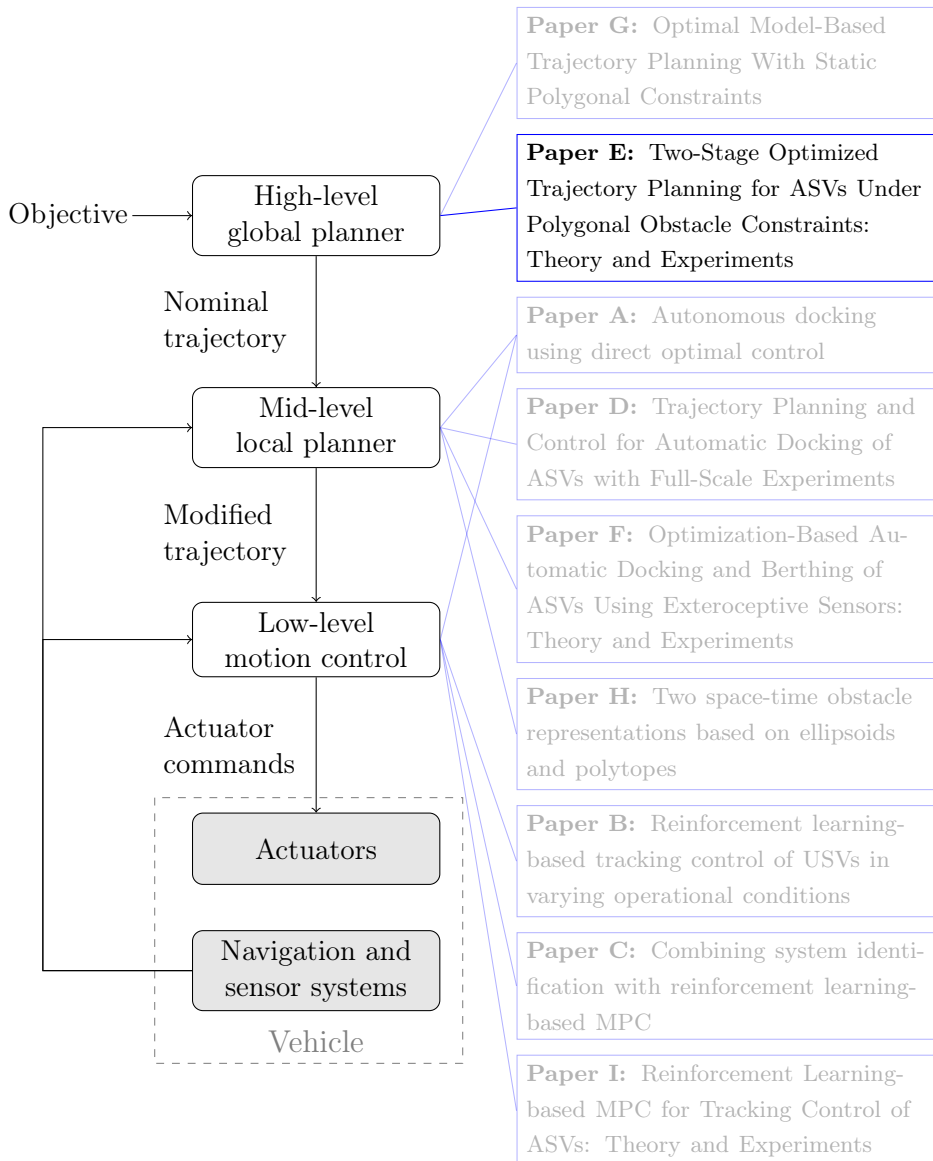
$$\mathbf{M}_p \boldsymbol{\nu}_p + \mathbf{D}_p(\boldsymbol{\nu}_p) \boldsymbol{\nu}_p = \boldsymbol{\tau}_{ff}. \quad (24)$$

The DP controller was originally developed for station keeping, and does not contain the \mathbf{C} matrix. Otherwise, the matrix values in (24) are equal to those in the planning model (16).

E Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments

Postprint of [21] Glenn Bitar, **Andreas B Martinsen**, Anastasios M Lekkas, and Morten 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

©2020 Glenn Bitar, Andreas B Martinsen, Anastasios M Lekkas, and Morten Breivik. Reprinted and formatted to fit the thesis under the terms of the Creative Commons Attribution License 



Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory & Experiments*

Glenn Bitar¹, Andreas B. Martinsen¹, Anastasios M. Lekkas¹, and Morten Breivik¹

¹Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway

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.

Keywords: Autonomous vehicles, collision avoidance, marine vehicles, motion planning, polygonal collision-avoidance constraints, trajectory optimization, trajectory planning.

*This work was supported in part 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 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.

1.1 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. 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.

¹<https://www.maritime-executive.com/article/rolls-royce-and-wartsila-in-close-race-with-autonomous-ferries> (accessed September 14, 2020).

²https://www.nyk.com/english/news/2019/20190930_01.html (accessed August 31, 2020).

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, Lekkas, and Sørensen [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 feasibility³, 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, Zhan, and Tomizuka [17] can represent

³We 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.

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. 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 subsequent from the geometrical path planning.

1.2 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

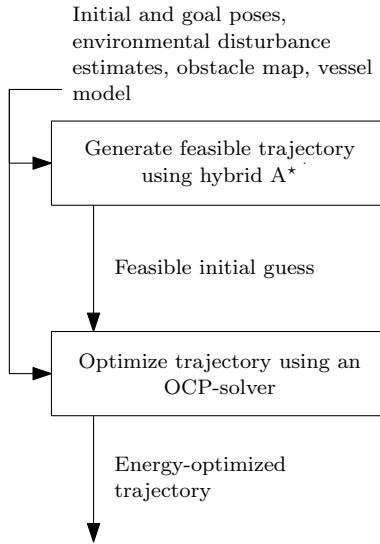


Figure 1: A block diagram of the high-level functionality of our proposed trajectory planning method.

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.

1.3 Outline

In Section 2 we cover preliminary information about notation and vessel modeling. Sections 3 and 4 present the development of our trajectory planning method. In Section 3, 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 4, 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 using an interior point method. Section 5 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 6, 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 7 gives concluding remarks.

2 Preliminaries

2.1 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

$$\boldsymbol{\eta} = [x \quad y \quad \psi]^\top \in \mathcal{C} := \mathbb{R}^2 \times S. \quad (1)$$

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

E. Two-Stage Optimized Trajectory Planning for ASVs Under ...

$\mathcal{A}(\boldsymbol{\eta}) \subset \mathcal{W}$, which defines the vessel's shape. The set of noncolliding configurations is thus

$$\mathcal{C}_{\text{free}} := \{ \boldsymbol{\eta} \in \mathcal{C} \mid \mathcal{A}(\boldsymbol{\eta}) \cap \mathcal{O} = \emptyset \}. \quad (2)$$

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

$$\mathbf{r}_{\mathcal{C}} := [r_p \quad r_p \quad r_h]^\top, \quad (3)$$

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 $\mathbf{r}_{\mathcal{C}}$.

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 2.2 introduces such constraints in the form of a mathematical vessel model.

2.2 ASV modeling

Our ASV is modeled as a surge-decoupled three-degree-of-freedom displacement vessel, with the state vector

$$\mathbf{x} := \begin{bmatrix} \boldsymbol{\eta}^\top & \boldsymbol{\nu}^\top \end{bmatrix}^\top \in \mathcal{X} := \mathcal{C} \times \mathbb{R}^3 \quad (4)$$

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

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu}, \quad (5)$$

where

$$\mathbf{R}(\psi) := \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

The kinetics of the ASV is described by

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}. \quad (7)$$

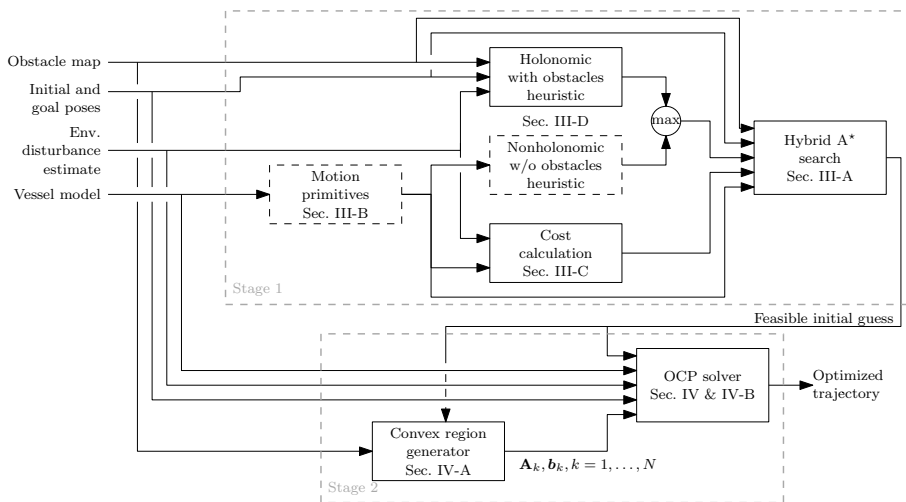


Figure 2: Block diagram of the trajectory planning method. Stage 1 refers to the generation of the initial guess, described in Section 3, and Stage 2 refers to the trajectory optimization from Section 4.

This notation is widely used for vessel models in the maritime control literature. Here, $\mathbf{M} \in \mathbb{R}^{3 \times 3}$ is the positive definite system inertia matrix, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$ is the skew-symmetric Coriolis and centripetal matrix, and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$ is the positive definite damping matrix. The force vector $\boldsymbol{\tau} = [X, Y, N]^T \in \mathcal{T} \subset \mathbb{R}^3$ 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:

$$\boldsymbol{\tau}_{\text{env}} = \boldsymbol{\tau}_{\text{env}}(\psi, \boldsymbol{\nu}, \mathbf{V}_w), \quad (8)$$

where $\mathbf{V}_w \in \mathbb{R}^2$ 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 [24].

The model is concatenated to

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\tau}, \mathbf{V}_w) := \begin{bmatrix} \mathbf{R}(\psi)\boldsymbol{\nu} \\ \mathbf{M}^{-1}[-(\mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu}))\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}(\psi, \boldsymbol{\nu}, \mathbf{V}_w)] \end{bmatrix} \quad (9)$$

for ease of reference when discussing OCPs later in the paper.

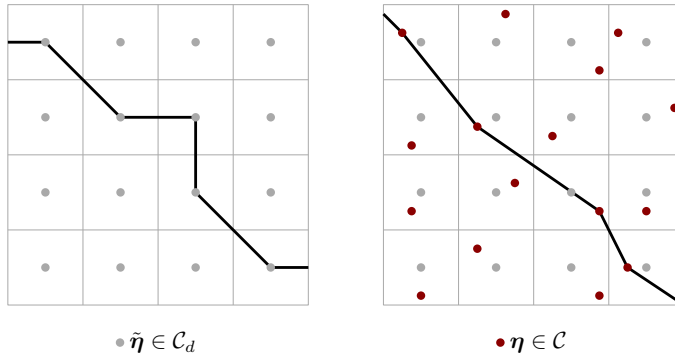


Figure 3: Comparison of traditional A* search space to the hybrid A* search space in a two-dimensional grid. To the left is the commonly found eight-connected uniform grid associated with A*, where states are associated with grid cell centers. To the right is the search space of hybrid A*, where states can lie anywhere in the cells.

3 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.

3.1 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 $\tilde{\eta} \in \mathcal{C}_d$. The mappings are

Algorithm 1 Hybrid A* search pseudocode.

```

1: function HYBRID A*( $\eta_0, \eta_f, \mathbf{V}_w, \mathcal{O}$ )
2:    $\tilde{\eta}_0 \leftarrow \text{KEY}(\eta_0), \tilde{\eta}_f \leftarrow \text{KEY}(\eta_f)$ 
3:    $O \leftarrow \emptyset, C \leftarrow \emptyset$ 
4:   PUSH( $O, \tilde{\eta}_0, 0$ )
5:   STATE( $\tilde{\eta}_0$ )  $\leftarrow \eta_0, \text{COST}(\tilde{\eta}_0) \leftarrow 0$ 
6:   while  $O \neq \emptyset$  do
7:      $\tilde{\eta} \leftarrow \text{POP}(O)$ 
8:      $C \leftarrow C \cup \{\tilde{\eta}\}$ 
9:     if  $\tilde{\eta} = \tilde{\eta}_f$  then
10:      return sequence from  $\tilde{\eta}_0$  to  $\tilde{\eta}_f$ 
11:      $\eta \leftarrow \text{STATE}(\tilde{\eta})$ 
12:     for all  $\mathcal{P}, c, \eta_n \in \text{PRIMITIVES}(\eta, \mathbf{V}_w)$  do
13:        $\tilde{\eta}_n \leftarrow \text{KEY}(\eta_n)$ 
14:       if COLLISION( $\mathcal{P}, \mathcal{O}$ ) or  $\tilde{\eta}_n \in C$  then
15:         continue
16:        $f \leftarrow \text{COST}(\tilde{\eta}) + c$ 
17:       if  $\tilde{\eta}_n \notin C \cup O$  then
18:          $\text{COST}(\tilde{\eta}_n) \leftarrow \infty$ 
19:       if  $f < \text{COST}(\tilde{\eta}_n)$  then
20:          $\text{COST}(\tilde{\eta}_n) \leftarrow f$ 
21:         PARENT( $\tilde{\eta}_n$ )  $\leftarrow \tilde{\eta}$ 
22:         STATE( $\tilde{\eta}_n$ )  $\leftarrow \eta_n$ 
23:          $O \leftarrow O \setminus \{\tilde{\eta}_n\}$ 
24:          $h \leftarrow f + \text{HEURISTICS}(\eta_n, \eta_f, \mathbf{V}_w)$ 
25:         PUSH( $O, \tilde{\eta}_n, h$ )
26:   return error, no path found

```

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 3.2, 3.4, and 3.5, respectively.

3.2 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 2.1, 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.

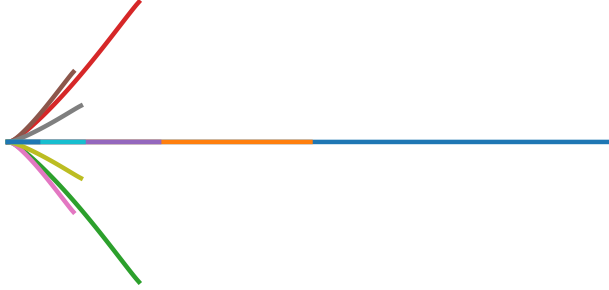
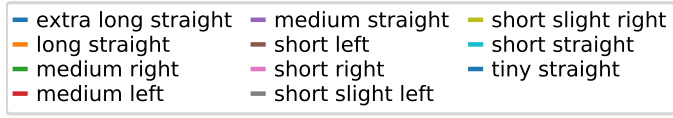


Figure 4: Motion primitives used in our results.

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.

The OCP used to generate motion primitives is

$$\min_{\mathbf{x}(\cdot), \boldsymbol{\tau}(\cdot)} \int_0^{t_f} F(\mathbf{x}(\tau), \boldsymbol{\tau}(\tau)) d\tau \quad (10a)$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\tau}(t), \mathbf{0}_2) \quad t \in [0, t_f] \quad (10b)$$

$$\mathbf{x}_{lb} \leq \mathbf{x}(t) \leq \mathbf{x}_{ub} \quad t \in [0, t_f] \quad (10c)$$

$$\boldsymbol{\tau}_{lb} \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_{ub} \quad t \in [0, t_f] \quad (10d)$$

$$\mathbf{x}(0) = \mathbf{x}_0 \quad (10e)$$

$$\mathbf{x}(t_f) = \mathbf{x}_f. \quad (10f)$$

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 m s^{-1} . For a specific primitive defined by (L, χ) , the

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

parameters of (10) are

$$t_f = L/U_{\text{nom}} \quad (11a)$$

$$\mathbf{x}_{lb} = \begin{bmatrix} \min(0, L \cos \chi) \\ \min(0, L \sin \chi) \\ \min(0, \chi) \\ u_{lb} \\ -v_{ub} \\ -r_{ub} \end{bmatrix} \quad (11b)$$

$$\mathbf{x}_{ub} = \begin{bmatrix} \max(0, L \cos \chi) \\ \max(0, L \sin \chi) \\ \max(0, \chi) \\ u_{ub} \\ v_{ub} \\ r_{ub} \end{bmatrix} \quad (11c)$$

$$\boldsymbol{\tau}_{lb} = [X_{lb} \quad -Y_{ub} \quad -N_{ub}]^\top \quad (11d)$$

$$\boldsymbol{\tau}_{ub} = [X_{ub} \quad Y_{ub} \quad N_{ub}]^\top \quad (11e)$$

$$\mathbf{x}_0 = [0 \quad 0 \quad 0 \quad U_{\text{nom}} \quad 0 \quad 0]^\top \quad (11f)$$

$$\mathbf{x}_f = [L \cos \chi \quad L \sin \chi \quad \chi \quad U_{\text{nom}} \quad 0 \quad 0]^\top. \quad (11g)$$

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.

E. Two-Stage Optimized Trajectory Planning for ASVs Under ...

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

The OCP (10) contains a cost-to-go function:

$$\begin{aligned}
 F(\mathbf{x}, \boldsymbol{\tau}) = & \overbrace{|\boldsymbol{\nu}|^{\top} \cdot |\boldsymbol{\tau}|}^{\text{energy}} \\
 & + 1000 \left((v/v_{ub})^2 + (r/r_{ub})^2 \right) \\
 & + 100 \left((X/X_{ub})^2 + (Y/Y_{ub})^2 + (N/N_{ub})^2 \right). \quad (12)
 \end{aligned}$$

The cost’s main contributor is energy spent but includes quadratic costs on velocity states and input forces. Without these quadratic costs, the OCP becomes significantly harder to solve. The pure energy part of the cost function makes up $\sim 95\%$ of the straight motion primitives’ costs and $\sim 80\%$ 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 $r_p = 10$ m, so we need at least one primitive longer than $\sqrt{2} \cdot 10 \text{ m} \approx 14.14$ 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 [25] 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 4.2.

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 3.3, and the new neighboring state $\eta_n \in \mathcal{C}$. The cost is dependent on the wind velocity \mathbf{V}_w . A mathematical description of the function is

$$\text{PRIMITIVES} : \mathcal{C} \times \mathbb{R}^2 \mapsto [\mathcal{W} \times \mathbb{R}^+ \times \mathcal{C}]_{1,\dots,M}, \quad (13)$$

where M is the number of motion primitives.

3.3 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:

$$W_r = \int_0^{t_f} |\boldsymbol{\tau}_r|^\top \cdot |\boldsymbol{\nu}_r| dt, \quad (14)$$

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

$$W_{\text{wind}} = \int_0^{t_f} |\boldsymbol{\tau}_w|^\top \cdot \left| \boldsymbol{\nu} - \mathbf{R}(\psi)^\top \begin{bmatrix} \mathbf{V}_w \\ 0 \end{bmatrix} \right| dt, \quad (15)$$

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

$$W_{\text{water}} = \int_0^{t_f} |\mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu}|^\top \cdot |\boldsymbol{\nu}| dt. \quad (16)$$

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 [26].

3.4 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

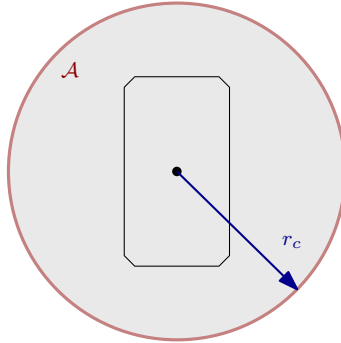


Figure 5: Vessel shape along with the clearance radius r_c which defines the footprint \mathcal{A} used for collision checking.

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 radius of $r_c = 10$ m. The COLLISION function in Algorithm 1 performs the collision checking:

$$\text{COLLISION} : \mathcal{W} \times \mathcal{W} \mapsto \{\text{true}, \text{false}\}. \quad (17)$$

3.5 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

$$\text{HEURISTICS} : \mathcal{C} \times \mathcal{C} \times \mathbb{R}^2 \mapsto \mathbb{R}^+, \quad (18)$$

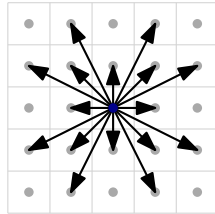


Figure 6: 16-connected graph. In this connectivity scheme, edges are added to all nodes two layers from the center node, unless the travel direction already exists in an inner layer.

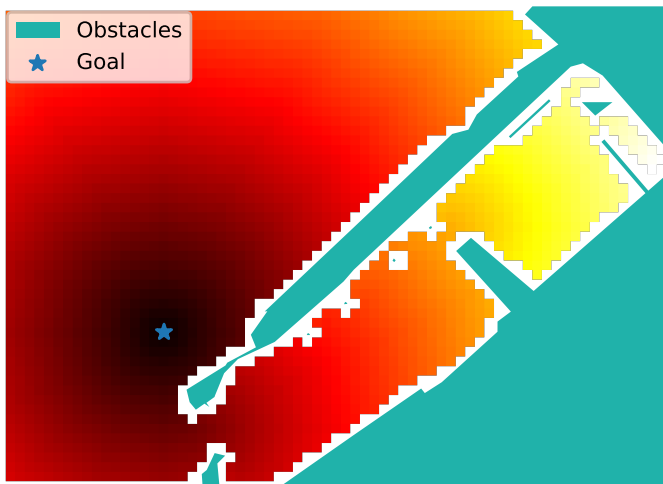


Figure 7: Example of the holonomic with obstacles heuristic function on a map. Brighter squares are more costly.

where the function maps the current state η , the goal state η_f , and the wind velocity \mathbf{V}_w to a positive scalar.

3.5.1 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 3.3, 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 connectivity in Figure 6, i.e., on $\sim 22^\circ$ 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 [27] 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 [28, 29]. 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.

3.5.2 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 3.2 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 in Section 4 locally optimizes the trajectory accounting for known or estimated disturbances.

3.6 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

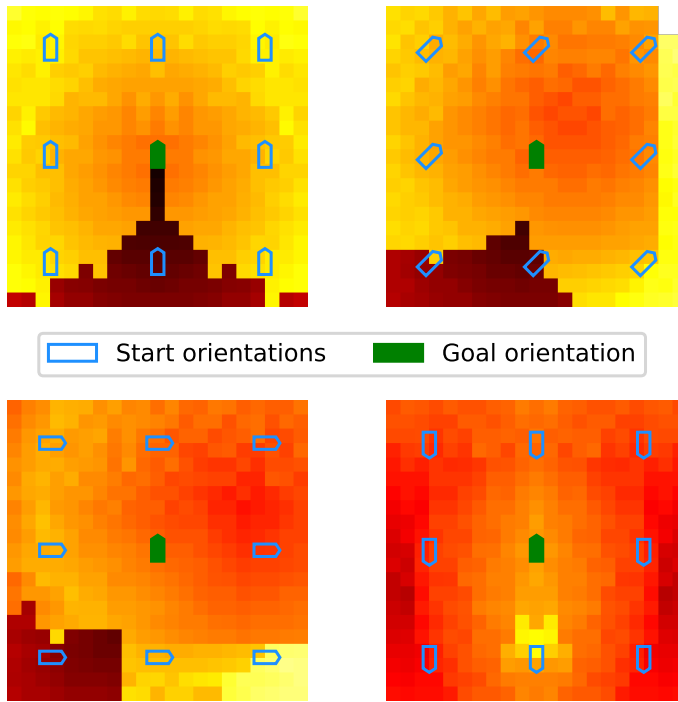


Figure 8: Plot of the nonholonomic without obstacles heuristic function for initial heading angles 0° , 45° , 90° and 180° . Brighter squares are more costly.

concatenated into forming the solution trajectories

$$\mathbf{x}^* : [0, t_f^*] \mapsto \mathcal{X} \quad (19a)$$

$$\boldsymbol{\tau}^* : [0, t_f^*] \mapsto \mathcal{T}, \quad (19b)$$

which are valid on the time interval $[0, t_f^*]$, where t_f^* 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.

4 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 3.2. 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:

$$\min_{\mathbf{x}(\cdot), \boldsymbol{\tau}(\cdot), t_f} \int_0^{t_f} F(\mathbf{x}(\tau), \boldsymbol{\tau}(\tau)) d\tau \quad (20a)$$

subject to

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \boldsymbol{\tau}(t), \mathbf{V}_w) \quad t \in [0, t_f] \quad (20b)$$

$$\mathbf{x}_{lb} \leq \mathbf{x}(t) \leq \mathbf{x}_{ub} \quad t \in [0, t_f] \quad (20c)$$

$$\boldsymbol{\tau}_{lb} \leq \boldsymbol{\tau}(t) \leq \boldsymbol{\tau}_{ub} \quad t \in [0, t_f] \quad (20d)$$

$$\mathbf{x}(0) = \mathbf{x}_0 \quad (20e)$$

$$\mathbf{x}(t_f) = \mathbf{x}_f \quad (20f)$$

$$\mathbf{A}_k \cdot [x(t_k) \quad y(t_k)]^\top \leq \mathbf{b}_k - r_c \quad k = 1, \dots, N \quad (20g)$$

$$0 \leq t_f \leq t_f^*. \quad (20h)$$

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 4.1. Since the final time is a free variable, we place bounds on it in (20h). The transcription and solution process is described in Section 4.2.

4.1 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 $t_k, k = 1, \dots, N$, 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 $\mathbf{x}^*(\cdot)$ from (19a) is denoted $\mathbf{p}^*(\cdot) = [x^*(\cdot), y^*(\cdot)]^\top$. For the points in time $t_k, k = 1, \dots, N$, the parameters $\mathbf{A}_k \in \mathbb{R}^{m_k \times 2}$ and $\mathbf{b}_k \in \mathbb{R}^{m_k}$ are generated based on the obstacle map \mathcal{O} with $\mathbf{p}^*(t_k)$ 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

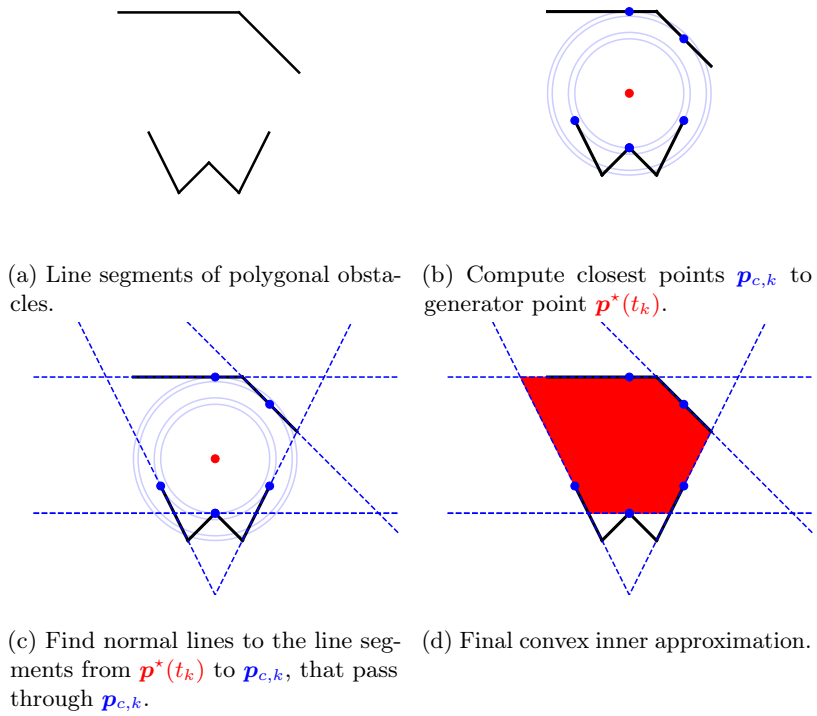


Figure 9: Illustration of how to compute the convex spatial constraints.

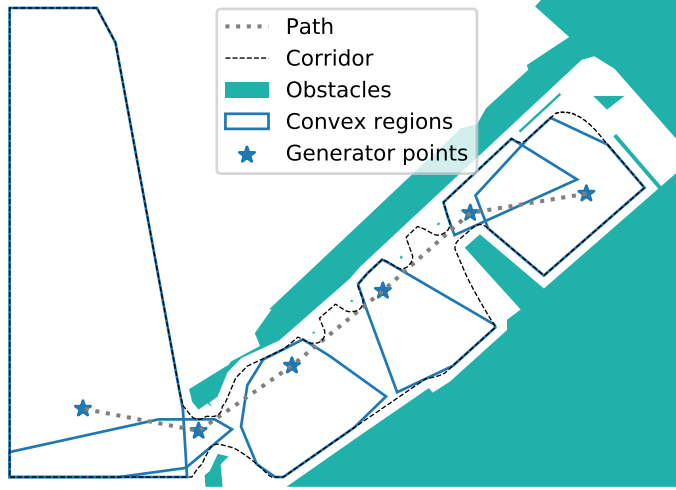


Figure 10: Example of convex regions along an arbitrary path with generator points spaced by 100 m. In the OCP the spacing would be ~ 1.5 m, causing dense overlapping, resulting in a corridor as depicted in the figure.

process is summarized as follows: Given a generator point $\mathbf{p}^*(t_k)$, grow a circle centered at $\mathbf{p}^*(t_k)$ until it reaches a point $\mathbf{p}_{c,k}$ where it touches an obstacle, and then create a constraint tangent to the expansion circle at the point at $\mathbf{p}_{c,k}$. Continue growing and create constraints until no further growth is possible. The process is illustrated in Figure 9. The parameters \mathbf{A}_k and \mathbf{b}_k are defined by

$$\underbrace{\begin{bmatrix} \frac{(\mathbf{p}_{c,k,1} - \mathbf{p}^*(t_k))^\top}{\|(\mathbf{p}_{c,k,1} - \mathbf{p}^*(t_k))\|_2} \\ \frac{(\mathbf{p}_{c,k,2} - \mathbf{p}^*(t_k))^\top}{\|(\mathbf{p}_{c,k,2} - \mathbf{p}^*(t_k))\|_2} \\ \vdots \\ \frac{(\mathbf{p}_{c,k,m_k} - \mathbf{p}^*(t_k))^\top}{\|(\mathbf{p}_{c,k,m_k} - \mathbf{p}^*(t_k))\|_2} \end{bmatrix}}_{\mathbf{A}_k} \mathbf{p} \leq \underbrace{\begin{bmatrix} \frac{(\mathbf{p}_{c,k,1} - \mathbf{p}^*(t_k))^\top \mathbf{p}_{c,k,1}}{\|(\mathbf{p}_{c,k,1} - \mathbf{p}^*(t_k))\|_2} \\ \frac{(\mathbf{p}_{c,k,2} - \mathbf{p}^*(t_k))^\top \mathbf{p}_{c,k,2}}{\|(\mathbf{p}_{c,k,2} - \mathbf{p}^*(t_k))\|_2} \\ \vdots \\ \frac{(\mathbf{p}_{c,k,m_k} - \mathbf{p}^*(t_k))^\top \mathbf{p}_{c,k,m_k}}{\|(\mathbf{p}_{c,k,m_k} - \mathbf{p}^*(t_k))\|_2} \end{bmatrix}}_{\mathbf{b}_k}. \quad (21)$$

A point $\mathbf{p} \in \mathbb{R}^2$ 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 $m_k, k = 1, \dots, N$, 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 ($\mathbf{x}(t_k), k = 1, \dots, N$). 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 $\mathbf{x}(t_k)$. Having a short shooting interval duration h gives satisfactory collision avoidance behavior. In our results, we use a density of $h \approx 1$ s.

4.2 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^* from the hybrid A* results in Section 3.6. An initial shooting interval duration of $h^* = 1$ s determines $N = \lfloor t_f^*/h^* \rfloor + 1$, while since the final time t_f is a free variable with upper bound t_f^* , 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

$$\min_{\mathbf{w}} \phi(\mathbf{w}) \tag{22a}$$

subject to

$$\mathbf{w}_{lb} \leq \mathbf{w} \leq \mathbf{w}_{ub} \tag{22b}$$

$$\mathbf{g}_{lb} \leq \mathbf{g}(\mathbf{w}) \leq \mathbf{g}_{ub}. \tag{22c}$$

The decision variables \mathbf{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 \mathbf{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 [25]. 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 \mathbf{w}^\diamond . These are converted to optimal trajectories

$$\mathbf{x}^\diamond : [0, t_f^\diamond] \mapsto \mathcal{X} \tag{23a}$$

$$\boldsymbol{\tau}^\diamond : [0, t_f^\diamond] \mapsto \mathcal{T}, \tag{23b}$$

where t_f^* 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.

4.3 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 3.2, 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.

5 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.

5.1 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 2.2, 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 [24].

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 [30,

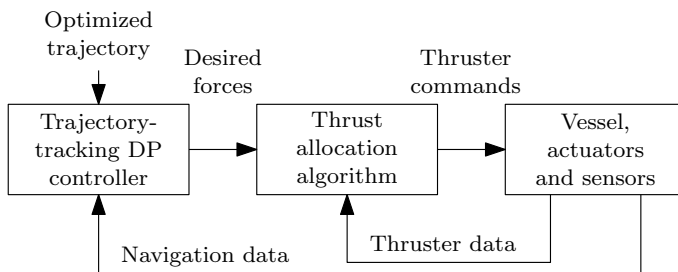


Figure 11: Vessel control system architecture.

Section 3.4]. The controller sends the desired force output to the thrust allocation algorithm, which in turn sends thruster commands to the vessel’s actuators. This thrust allocation algorithm is described in [31].

For evaluation, energy use is measured by integrating the simulated power output, similar to the energy-part of (12):

$$E = \int_{t_0}^{t_f} |\boldsymbol{\nu}(t)|^\top \cdot |\boldsymbol{\tau}(t)| dt. \quad (24)$$

5.2 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 $\mathbf{V}_w = [0, 0]$ when it is, in fact, $\mathbf{V}_w = [0, 3] \text{ m s}^{-1}$, 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.

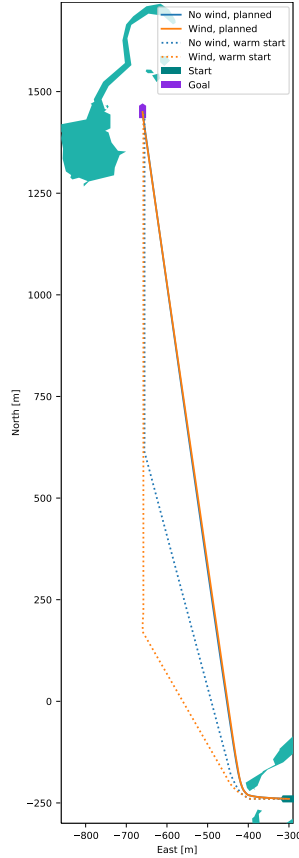


Figure 12: Comparison of trajectories planned without and with knowledge of simulated wind conditions. Described in Section 5.2.

5.3 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 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.

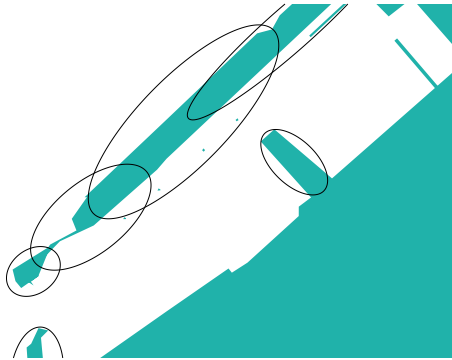


Figure 13: Elliptic obstacles that approximately match our map. Used when planning with the C1 method in Section 5.3.1.

5.3.1 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 Wh. 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

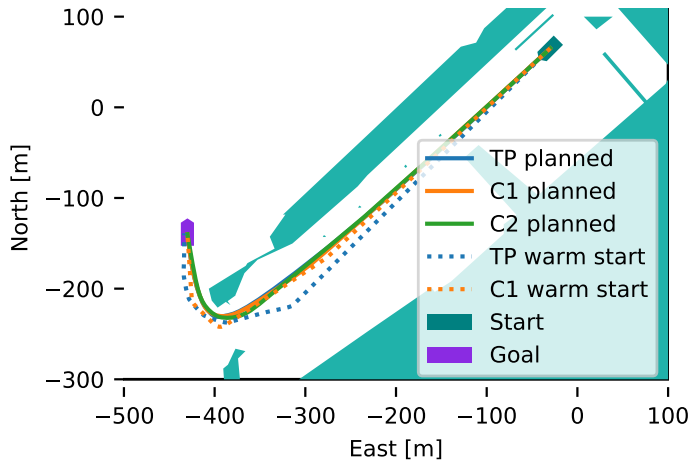


Figure 14: Comparison of trajectories planned with different methods from Section 5.3. Results from our method are labeled TP, while C1 and C2 denote the other planning methods.

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 $\| [x(t), y(t)] - [x^\circ(t), y^\circ(t)] \|_2$, while the heading error is $\psi(t) - \psi^\circ(t)$.

5.3.2 C2: Complete optimization-based cell decomposition

Martinsen, Lekkas, and Gros [23] have developed an optimization-based trajectory planner that searches for a trajectory by 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 5.3.1, 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.

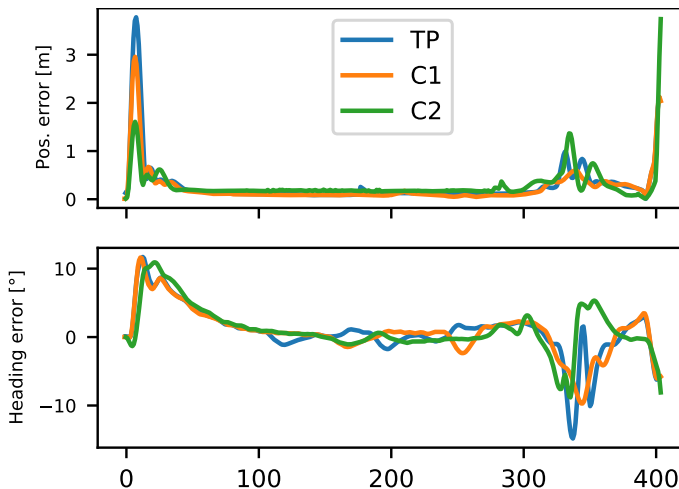


Figure 15: Tracking errors from the simulation comparisons. Results from our method are labeled TP, while C1 and C2 denote the other planning methods.

Table 3: Performance comparisons for simulated planning scenarios in Section 5.3.

Method	Energy usage [W h]	Planning time [s]
TP	52	31
C1	52	57
C2	52	785

5.4 Complex scenario

The previous planning scenarios have been simple, with obvious routing choices. Figure 16 shows a more complex 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.

5.5 Conclusions

Including available wind information when planning a scenario yielded negligible improvements, shown in Section 5.2. 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

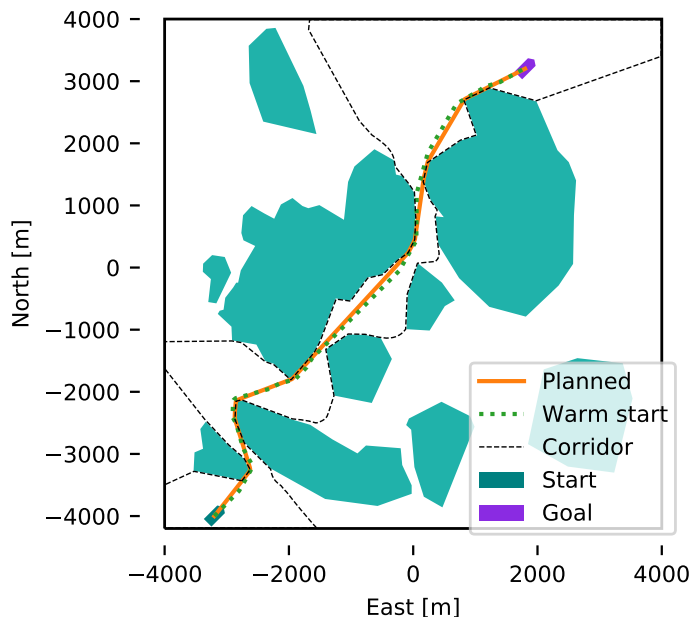


Figure 16: Planning in a map of Sjernerøyane, Norway – a more complex scenario with multiple routing options. Described in Section 5.4.

both wind estimates and wind force models.

Compared to two other optimization-based trajectory planning methods in Section 5.3, 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.

As we show in Section 5.4, 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.

6 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 5. We tested the planning method and tracking capabilities in the Trondheim harbor area, using the same scenario as in



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

Section 5.3. 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:

$$E = \int_{t_0}^{t_f} \left(|I_1(t) \cdot U_1(t)| + |I_2(t) \cdot U_2(t)| \right) dt, \quad (25)$$

where I_i and U_i are motor current and voltage, respectively, for $i \in \{1, 2\}$.

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 5.3, 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

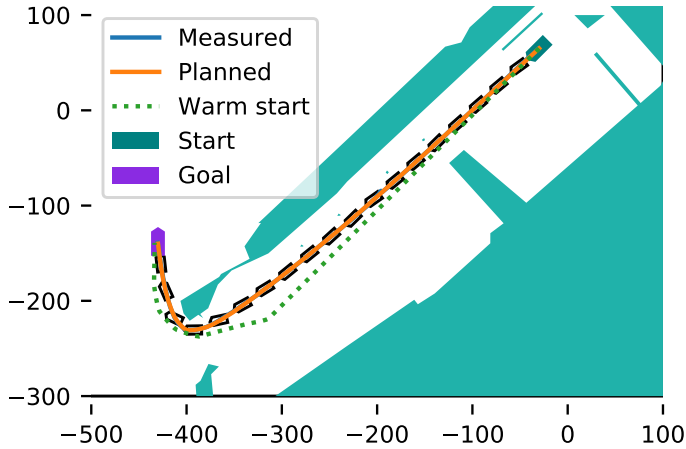


Figure 18: Measured and planned trajectories from validation experiment in Section 6.

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, the planned distance is more than the clearance of $r_c = 10$ 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 Wh, 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.

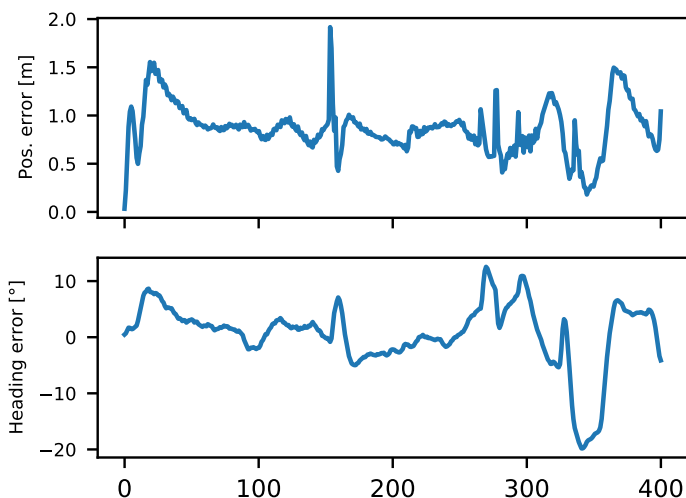


Figure 19: Tracking errors from the validation experiment in Section 6.

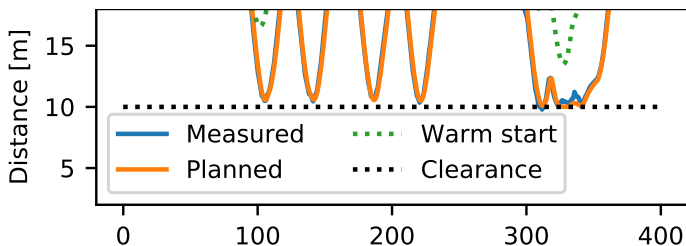


Figure 20: Measured and planned obstacle distances from the validation experiments in Section 6.

7 Conclusions

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-

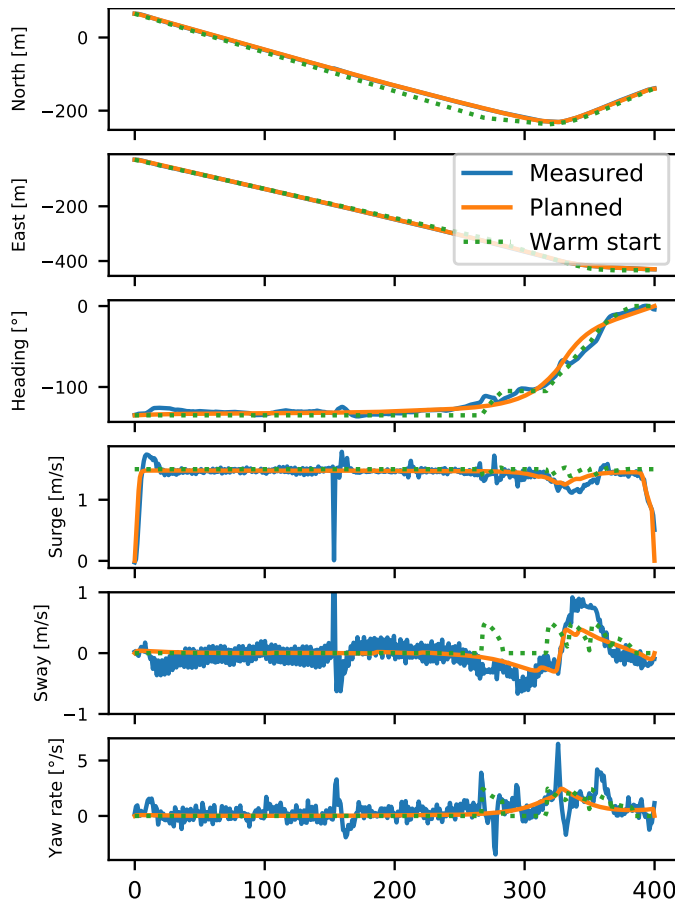


Figure 21: Measured and planned state trajectories from the validation experiment in Section 6.

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

Publications

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*. G. Bitar also thanks Marius Thoresen at the Norwegian Defence Research Establishment for helpful discussions during algorithm implementation.

References

- [1] Martin Ludvigsen and Asgeir J. Sørensen. “Towards integrated autonomous underwater operations for ocean mapping and monitoring”. In: *Annual Reviews in Control* 42 (2016), pp. 145–157.
- [2] Bjørn-Olav H. Eriksen et al. “Hybrid Collision Avoidance for ASVs Compliant with COLREGs Rules 8 and 13–17”. In: *Frontiers in Robotics and AI* 7 (2020), pp. 1–11. arXiv: 1907.00198 [eess.SY].
- [3] Mae L. Seto, ed. *Marine Robot Autonomy*. 2013.
- [4] Scott Pendleton et al. “Perception, Planning, Control, and Coordination for Autonomous Vehicles”. In: *Machines* 5.1 (2017), p. 6.
- [5] Steven M. LaValle. “Motion Planning, Part I: The Essentials”. In: *IEEE Robotics & Automation Magazine* 18 (1 2011), pp. 79–89.
- [6] Artur Wolek and Craig A. Woolsey. “Model-Based Path Planning”. In: *Sensing and Control for Autonomous Vehicles*. Ed. by Thor I. Fossen, Kristin Y. Pettersen, and Henk Nijmeijer. Springer International Publishing, 2017, pp. 183–206.


- [7] Peter Hart, Nils Nilsson, and Bertram 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.
- [8] 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.
- [9] L. E. Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.
- [10] Steven M. LaValle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. 1998.
- [11] Richard Bellman. *Dynamic Programming*. Courier Dover Publications, 2003.
- [12] Qi Gong, Ryan Lewis, and I. Michael Ross. “Pseudospectral Motion Planning for Autonomous Vehicles”. In: *Journal of Guidance, Control, and Dynamics* 32.3 (May 2009), pp. 1039–1045.
- [13] Glenn Bitar, Morten Breivik, and Anastasios M. Lekkas. “Energy-Optimized Path Planning for Autonomous Ferries”. In: *Proc. of the 11th IFAC CAMS*. Opatija, Croatia, 2018, pp. 389–394.
- [14] Glenn Bitar et al. “Warm-Started Optimized Trajectory Planning for ASVs”. In: *Proceedings of the 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles*. Daejeon, South Korea, 2019. arXiv: 1907.02696 [eess.SY].
- [15] Xiaojing Zhang et al. “Autonomous Parking Using Optimization-Based Collision Avoidance”. In: *IEEE Conference on Decision and Control (CDC)*. Miami Beach, FL, USA, 2018, pp. 4327–4332.
- [16] Kristoffer Bergman et al. *An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments*. 2020. arXiv: 2005.02674 [math.OC].
- [17] Jianyu Chen, Wei Zhan, and Masayoshi Tomizuka. “Autonomous Driving Motion Planning With Constrained Iterative LQR”. In: *IEEE Transactions on Intelligent Vehicles* 4.2 (2019), pp. 244–254.
- [18] Dmitri Dolgov et al. *Practical search techniques in path planning for autonomous driving*. Tech. rep. American Association for Artificial Intelligence, 2008.
- [19] Yu Zhang et al. “Hybrid Trajectory Planning for Autonomous Driving in Highly Constrained Environments”. In: *IEEE Access* 6 (2018), pp. 32800–32819.
- [20] Yu Meng et al. “A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization”. In: *IEEE Access* 7 (2019), pp. 130530–130551.

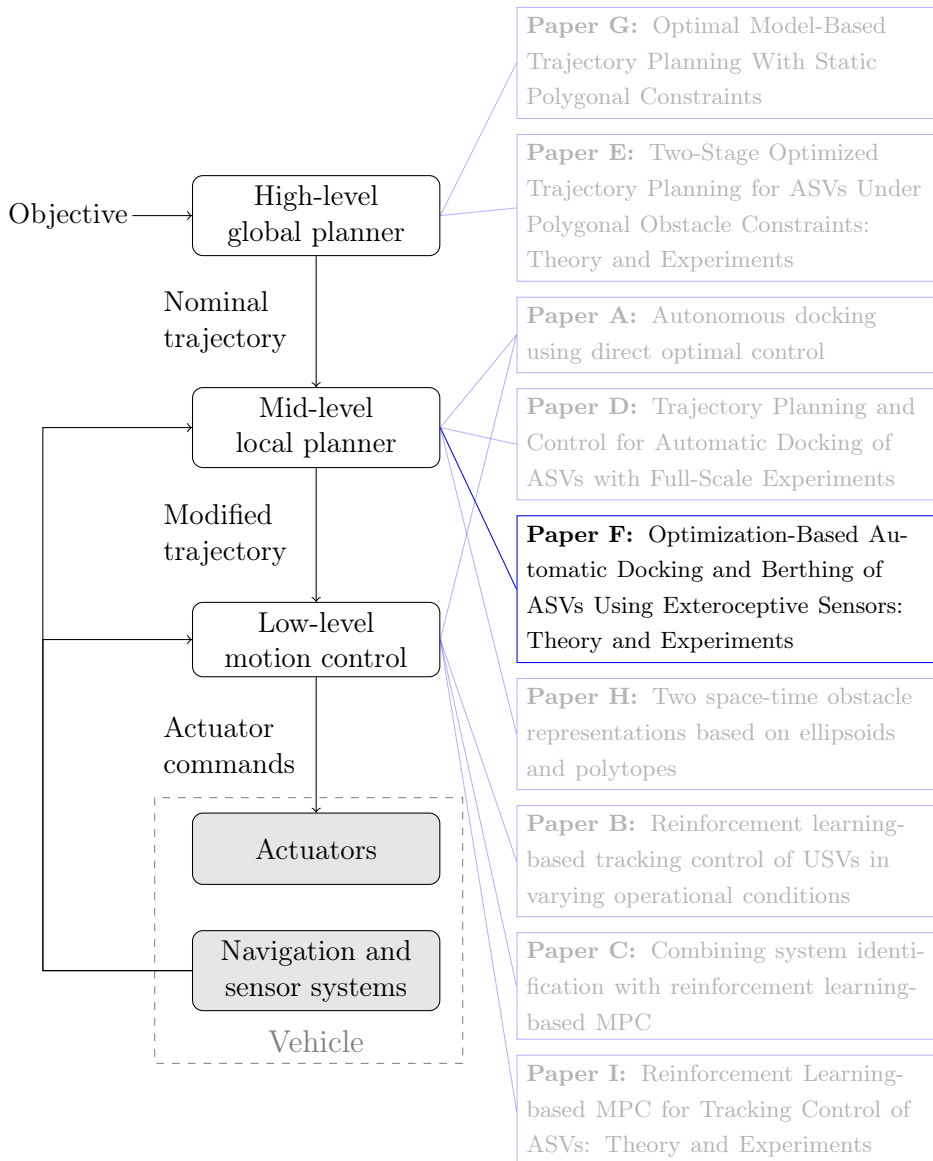
Publications

- [21] Giuseppe Casalino, Alessio Turetta, and Enrico 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 '09 Conference* (May 11, 2009). IEEE. Bremen, Germany, 2009, pp. 1–8.
- [22] Petr Švec et al. “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.
- [23] Andreas B. Martinsen, Anastasios M. Lekkas, and Sebastien Gros. *Optimal model-based trajectory planning with static polygonal constraints*. submitted for publication. arXiv: 2010.14428 [eess.SY].
- [24] Anders Aglen Pedersen. “Optimization based system identification for the milliAmpere ferry”. MA thesis. Norwegian University of Science and Technology, 2019.
- [25] 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.
- [26] T. Lee, H. Chung, and H. Myung. “Multi-resolution path planning for marine surface vehicle considering environmental effects”. In: *Proceedings of OCEANS*. Santander, Spain, 2011, pp. 1–9.
- [27] 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.
- [28] Clément Pêtrès et al. “Path Planning for Autonomous Underwater Vehicles”. In: *IEEE Transactions on Robotics* 23.2 (2007), pp. 331–341.
- [29] Santiago Garrido et al. “Fast marching subjected to a vector field – path planning method for Mars rovers”. In: *Expert Systems with Applications* 78 (2017), pp. 334–346.
- [30] Glenn Bitar et al. “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].
- [31] Tobias R. Torben, Astrid H. Brodtkorb, and Asgeir 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.

F Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments

Postprint of [22] **Andreas B Martinsen**, Glenn Bitar, Anastasios M Lekkas, and Sébastien 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

©2020 Andreas B Martinsen, Glenn Bitar, Anastasios M Lekkas, and Sébastien Gros. Reprinted and formatted to fit the thesis under the terms of the Creative Commons Attribution License 



Optimization-based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments*

Andreas B. Martinsen¹, Glenn Bitar^{1,2}, Anastasios M. Lekkas^{1,2}, and Sébastien Gros¹

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

²Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway

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.

Keywords: Autonomous surface vehicles, Berthing, Collision avoidance, Docking, Marine vehicles, Motion planning, Optimal control, Trajectory optimization

1 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

*This work is supported by the Research Council of Norway through the project number 269116 and through the Centres of Excellence funding scheme with project number 223254.

Publications

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 led 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 perform the docking maneuver. This has led 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.

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 4.1).
 - Improvements to the cost function, which give more refined docking maneuvers (Section 3.2).
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 4.2).
 - Improved interplay between the tracking controller and planner for better tracking performance (Section 5.1).
 - Improvements to the cost function, which gives better tracking performance (Section 3.2).



Figure 1: Experimental platform milliAmpere.

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.

2 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.

2.1 Kinematics and Dynamics

When modeling vessels for the purpose of autonomous docking, we consider only the vessel movement on the ocean 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 $SO(n) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{n \times n}, \mathbf{R}^\top \mathbf{R} = \mathbf{R} \mathbf{R}^\top = \mathbf{I}, \det(\mathbf{R}) = 1\}$ 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, $\mathbf{p}_c = [x, y]^\top$ 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 illustra-

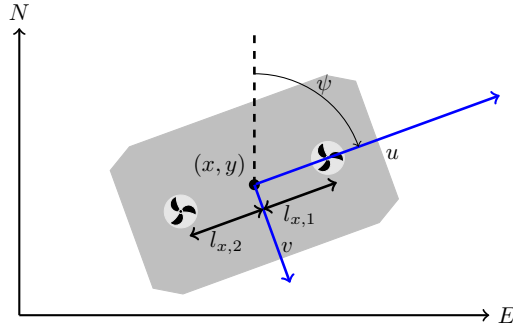


Figure 2: 3-DOF vessel centered at $\mathbf{p}_c = [x, y]^\top$, with surge velocity u , sway velocity v , heading ψ in a North-East-Down (NED) reference frame.

tion is given in Figure 2. Using the notation in [38] we can describe a 3-DOF vessel model as follows

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\psi)\boldsymbol{\nu}, \quad (1)$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \quad (2)$$

where $\mathbf{M} \in \mathbb{R}^{3 \times 3}$, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\boldsymbol{\tau} \in \mathbb{R}^3$ and $\mathbf{J}(\psi) \in SO(3)$ are the inertia matrix, Coriolis matrix, dampening matrix, control forces and moments, and transformation matrix, respectively. The transformation matrix $\mathbf{J}(\psi)$ is given by

$$\mathbf{J}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

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].

2.2 Thrust configuration

The control surfaces of the vessel are specified by the thrust configuration matrix $\mathbf{T} \in \mathbb{R}^{3, n_{\text{thrusters}}}$, which maps the thrust $\mathbf{f} \in \mathbb{R}^{n_{\text{thrusters}}}$ from each thruster into the surge, sway and yaw forces and moments in the body frame of the vessel

$$\boldsymbol{\tau} = \mathbf{T}\mathbf{f}. \quad (4)$$

Each column \mathbf{T}_i in \mathbf{T} gives the configuration of the forces and moments of a thruster i as follows:

$$\mathbf{T}_i \mathbf{f}_i = \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{y,i} \cdot l_{x,i} - F_{x,i} \cdot l_{y,i} \end{bmatrix}, \quad (5)$$

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:

$$\boldsymbol{\tau} = \begin{bmatrix} F_{x,1} \\ F_{y,1} \\ F_{y,1} \cdot l_{x,1} \end{bmatrix} + \begin{bmatrix} F_{x,2} \\ F_{y,2} \\ F_{y,2} \cdot l_{x,2} \end{bmatrix}. \quad (6)$$

3 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.

3.1 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

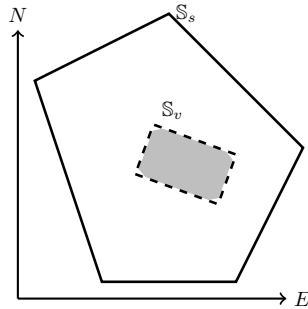


Figure 3: The gray convex polytope illustrates the vessel, whereas \mathbb{S}_v is the vessel boundary, and \mathbb{S}_s are the spatial constraints. The vessel will always lie within the spatial constraints \mathbb{S}_s as long as all the vertices of \mathbb{S}_v lie within the spatial constraints.

the following:

$$\mathbb{S}_v \subseteq \mathbb{S}_s \iff \mathbf{A}_s \mathbf{p}_i^n \leq \mathbf{b}_s \quad \forall \mathbf{p}_i^n \in \text{Vertex}(\mathbb{S}_v), \quad (7)$$

where \mathbf{p}_i^n denotes the position of vertex i in the North-East-Down (NED) reference frame. Since the vertices of the 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:

$$\mathbf{A}_s (\mathbf{R}(\psi) \mathbf{p}_i^b + \mathbf{p}_c) \leq \mathbf{b}_s \quad \forall \mathbf{p}_i^b \in \text{Vertex}(\mathbb{S}_v), \quad (8)$$

where \mathbf{p}_i^b denotes the position of vertex i in the body frame, $\mathbf{p}_c = [x, y]^\top$ is the vessel position and \mathbf{R} is the rotation from the body frame to NED.

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \quad (9)$$

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.

3.2 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:

$$\min_{\mathbf{x}_p(\cdot), \mathbf{u}_p(\cdot), \mathbf{s}(\cdot)} \int_{t_0}^{t_0+T} \left(l(\mathbf{x}_p(t), \mathbf{u}_p(t)) + \mathbf{k}_s^\top \mathbf{s}(t) \right) dt \quad (10a)$$

Publications

s.t.

$$\dot{\mathbf{x}}_p(t) = \mathbf{f}(\mathbf{x}_p(t), \mathbf{u}_p(t)) \quad \forall t \in [t_0, t_0 + T] \quad (10b)$$

$$\mathbf{h}(\mathbf{x}_p(t), \mathbf{u}_p(t)) - \mathbf{s}(t) \leq \mathbf{0} \quad \forall t \in [t_0, t_0 + T] \quad (10c)$$

$$\mathbf{s}(t) \geq \mathbf{0} \quad \forall t \in [t_0, t_0 + T] \quad (10d)$$

$$\mathbf{x}_p(t_0) = \mathbf{x}(t_0), \quad (10e)$$

where $\mathbf{x}_p(t) = [\boldsymbol{\eta}_p, \boldsymbol{\nu}_p]^\top$ is the planned state trajectory of the vessel, $\mathbf{u}(t) = [F_{x,1}, F_{y,1}, F_{x,2}, F_{y,1}]^\top$, are the planned thruster forces, and $\mathbf{s}(t)$ 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 $\mathbf{k}_s^\top \mathbf{s}(t)$, the gain \mathbf{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:

$$l(\mathbf{x}_p(t), \mathbf{u}_p(t)) = q_{x,y} \cdot c_{x,y}(\boldsymbol{\eta}_p(t)) + q_\psi \cdot c_\psi(\boldsymbol{\eta}_p(t)) + \boldsymbol{\nu}_p(t)^\top \mathbf{Q} \boldsymbol{\nu}_p(t) + \mathbf{u}_p(t)^\top \mathbf{R} \mathbf{u}_p(t). \quad (11)$$

The velocity and control actions are penalized with a quadratic penalty, with weight matrices \mathbf{Q} and \mathbf{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:

$$c_{x,y}(\boldsymbol{\eta}_p) = \delta^2 \left(\sqrt{1 + \frac{(x_p(t) - x_d)^2 + (y_p(t) - y_d)^2}{\delta^2}} - 1 \right). \quad (12)$$

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:

$$c_\psi(\boldsymbol{\eta}_p(t)) = \frac{1 - \cos(\psi_p(t) - \psi_d)}{2} e^{-\frac{(x_p(t) - x_d)^2 + (y_p(t) - y_d)^2}{2\delta^2}}. \quad (13)$$

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.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \phi(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{w}) = 0 \\ & \mathbf{h}(\mathbf{w}) \leq 0 \end{aligned}$$

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.

4 Automatic Constraint Generation

Given the OCP formulation in the previous section, we are faced with the problem of finding a convex set:

$$\mathbb{S}_s = \{\mathbf{p} \in \mathbb{R}^2 \mid \mathbf{A}_s \mathbf{p} \leq \mathbf{b}_s\},$$

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.

4.1 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 \mathbf{p}_c , grow an ellipse centered at \mathbf{p}_c , until it touches an environmental constraint (Section 4.1.1).
2. Create a linear constraint tangent to the expansion ellipse at the contact point between the ellipse and the environmental constraint (Section 4.1.2).
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 4.1.3)

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.

4.1.1 Computing the contact point between ellipse and environmental constraint

Given a line segment $(\mathbf{p}_a, \mathbf{p}_b)$ making up the environmental constraints, we want to find the contact point \mathbf{p}_{ab} between the line segment and the expansion ellipse centered at \mathbf{p}_c , this is illustrated in Figure 4. We can formulate this as an optimization problem which minimizes the distance between \mathbf{p}_c and the parametric line segment:

$$\mathbf{p}_{ab}(\omega) = \mathbf{p}_a(1 - \omega) + \mathbf{p}_b\omega, \quad (14)$$

where $\omega \in [0, 1]$. For the optimization problem we wish to find the parameter ω that minimizes a non-Euclidean distance from $\mathbf{p}_{ab}(\omega)$ to \mathbf{p}_c , giving the following:

$$\min_{\omega} f(\omega) = (\mathbf{p}_{ab}(\omega) - \mathbf{p}_c)^\top \Sigma (\mathbf{p}_{ab}(\omega) - \mathbf{p}_c) \quad (15)$$

$$s.t. \quad 0 \leq \omega \leq 1, \quad (16)$$

where Σ is a positive definite symmetric projection matrix, defining the expansion ellipse. Choosing $\Sigma = \mathbf{I}$ gives the Euclidean distance, while choosing Σ as a different positive definite symmetric matrix, allows prioritizing a direction, when minimizing the distance. For the unconstrained optimization problem, the necessary conditions give:

$$\begin{aligned} \frac{df(\omega)}{d\omega} &= (\mathbf{p}_a(1 - \omega) + \mathbf{p}_b\omega - \mathbf{p}_c)^\top \Sigma (\mathbf{p}_b - \mathbf{p}_a) = 0 \\ \Rightarrow \omega &= -\frac{(\mathbf{p}_a - \mathbf{p}_c)^\top \Sigma (\mathbf{p}_b - \mathbf{p}_a)}{(\mathbf{p}_b - \mathbf{p}_a)^\top \Sigma (\mathbf{p}_b - \mathbf{p}_a)}. \end{aligned} \quad (17)$$

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:

$$\omega = \text{clip} \left(-\frac{(\mathbf{p}_a - \mathbf{p}_c)^\top \Sigma (\mathbf{p}_b - \mathbf{p}_a)}{(\mathbf{p}_b - \mathbf{p}_a)^\top \Sigma (\mathbf{p}_b - \mathbf{p}_a)}, 0, 1 \right) \quad (18)$$

The closest point \mathbf{p}_{ab} , constrained to being on the line segment $(\mathbf{p}_a, \mathbf{p}_b)$ 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 \mathbf{p}_{ab} , which means it can be efficiently computed.

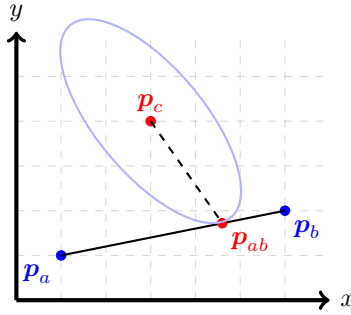


Figure 4: The contact point \mathbf{p}_{ab} is given by the shortest non-Euclidean distance from point \mathbf{p}_c to line segment $(\mathbf{p}_a, \mathbf{p}_b)$

4.1.2 Finding the tangent line to the expansion ellipse

Given the closest point \mathbf{p}_{ab} on a line segment, the next step is to find the line $\mathbf{A}\mathbf{x} = \mathbf{b}$ which is tangent to the expansion ellipse, and hence normal to the ellipse gradient $\Sigma(\mathbf{p}_{ab} - \mathbf{p}_c)$, as illustrated in Figure 5. Given the expansion ellipse gradient for a point \mathbf{p}_{ab} as:

$$\Sigma(\mathbf{p}_{ab} - \mathbf{p}_c) \quad (19)$$

we can note that any nonzero vector $[x, y]^\top$ is orthogonal to the expansion ellipse, if the inner product is zero.

$$(\mathbf{p}_{ab} - \mathbf{p}_c)^\top \Sigma \begin{bmatrix} x \\ y \end{bmatrix} = 0 \Rightarrow \text{Orthogonal}. \quad (20)$$

Using (20), the tangent line of the expansion ellipse, passing through the point $[x_0, y_0]^\top$ is given as the following.

$$(\mathbf{p}_{ab} - \mathbf{p}_c)^\top \Sigma \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \right) = 0 \quad (21)$$

Since we want the tangent line of the expansion ellipse to pass through the point \mathbf{p}_{ab} , the tangent line is given by all points $[x, y]^\top$ which fulfill the following equality.

$$\underbrace{(\mathbf{p}_{ab} - \mathbf{p}_c)^\top \Sigma}_A \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_x = \underbrace{(\mathbf{p}_{ab} - \mathbf{p}_c)^\top \Sigma \mathbf{p}_{ab}}_b \quad (22)$$

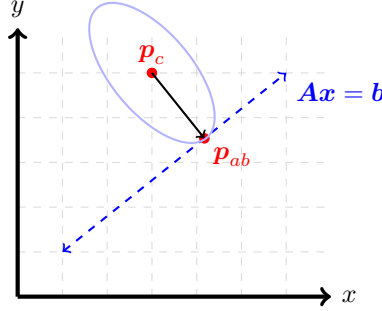


Figure 5: Finding the line $\mathbf{Ax} = \mathbf{b}$ tangent to the expansion ellipse at \mathbf{p}_{ab} , i.e. normal line to the vector $\Sigma(\mathbf{p}_{ab} - \mathbf{p}_c)$ passing through \mathbf{p}_{ab}

4.1.3 Linear inequality constraint generation

In order to generate the convex constraints around a point \mathbf{p}_c , our proposed method is based on finding the tangent line to the expansion ellipse (see Section 4.1.2) from the point \mathbf{p}_c to the closest point $\mathbf{p}_{ab,i}$ (see Section 4.1.1) on each line segment $i \in 1, 2, \dots, N$, 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.

$$\underbrace{\begin{bmatrix} (\mathbf{p}_{ab,1} - \mathbf{p}_c)^\top \Sigma \\ (\mathbf{p}_{ab,2} - \mathbf{p}_c)^\top \Sigma \\ \vdots \\ (\mathbf{p}_{ab,N} - \mathbf{p}_c)^\top \Sigma \end{bmatrix}}_A \mathbf{p} \leq \underbrace{\begin{bmatrix} (\mathbf{p}_{ab,1} - \mathbf{p}_c)^\top \Sigma \mathbf{p}_{ab,1} \\ (\mathbf{p}_{ab,2} - \mathbf{p}_c)^\top \Sigma \mathbf{p}_{ab,2} \\ \vdots \\ (\mathbf{p}_{ab,N} - \mathbf{p}_c)^\top \Sigma \mathbf{p}_{ab,N} \end{bmatrix}}_b \quad (23)$$

We can note that since (18) is piecewise linear and smooth, the constraints given above are continuous with respect to the center point \mathbf{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 \mathbf{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:

$$\frac{1}{\|(\mathbf{p}_{ab,i} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \quad (24)$$

The reason this normalizing factor is convenient, is the fact that the resulting matrix \mathbf{A} becomes dimensionless with every row having unit length, and hence the the constraint will have the same units as \mathbf{p} . This has several benefits, including that $\mathbf{A}\mathbf{p} - \mathbf{b}$ 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.

$$\underbrace{\begin{bmatrix} \frac{(\mathbf{p}_{ab,1} - \mathbf{p}_c)^\top \boldsymbol{\Sigma}}{\|(\mathbf{p}_{ab,1} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \\ \frac{(\mathbf{p}_{ab,2} - \mathbf{p}_c)^\top \boldsymbol{\Sigma}}{\|(\mathbf{p}_{ab,2} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \\ \vdots \\ \frac{(\mathbf{p}_{ab,N} - \mathbf{p}_c)^\top \boldsymbol{\Sigma}}{\|(\mathbf{p}_{ab,N} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \end{bmatrix}}_{\mathbf{A}} \mathbf{p} \leq \underbrace{\begin{bmatrix} \frac{(\mathbf{p}_{ab,1} - \mathbf{p}_c)^\top \boldsymbol{\Sigma} \mathbf{p}_{ab,1}}{\|(\mathbf{p}_{ab,1} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \\ \frac{(\mathbf{p}_{ab,2} - \mathbf{p}_c)^\top \boldsymbol{\Sigma} \mathbf{p}_{ab,2}}{\|(\mathbf{p}_{ab,2} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \\ \vdots \\ \frac{(\mathbf{p}_{ab,N} - \mathbf{p}_c)^\top \boldsymbol{\Sigma} \mathbf{p}_{ab,N}}{\|(\mathbf{p}_{ab,N} - \mathbf{p}_c)\boldsymbol{\Sigma}\|_2} \end{bmatrix}}_{\mathbf{b}} \quad (25)$$

4.1.4 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,

$$\mathbf{A}\mathbf{p} \leq \mathbf{b} \quad (26)$$

a given row $\mathbf{A}_k, \mathbf{b}_k$ can be identified as being a redundant constraint by solving the following Linear Programming (LP) problem:

$$\begin{aligned} \min_{\mathbf{p}} \quad & y_k = \mathbf{b}_k - \mathbf{A}_k \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}_i \mathbf{p} \leq \mathbf{b}_i \quad \forall i \neq k, \end{aligned} \quad (27)$$

and checking if the above problem has a solution \mathbf{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 constraints. It does this by first computing the dual points of the constraints as:

$$\mathbf{d}_i = \frac{\mathbf{A}_i}{\mathbf{b}_i - \mathbf{A}_i \mathbf{p}}, \quad (28)$$

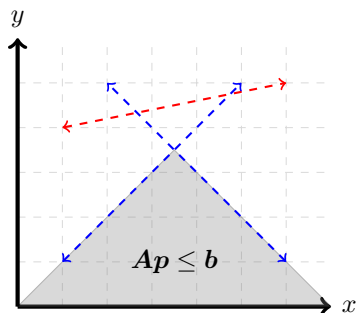


Figure 6: **Redundant constraints** are constraints that don't make up the **support** of the intersecting half-plane $\{\mathbf{p} \in \mathbb{R}^2 \mid \mathbf{A}\mathbf{p} \leq \mathbf{b}\}$.

where \mathbf{p} is an interior point of the constraints. Then computes the convex hull of the dual points as:

$$\mathcal{D} = \text{Conv}(\{\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_N\}). \quad (29)$$

The support of the constraints are then given by the rows i for which the dual points \mathbf{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.

4.2 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.

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 \mathbf{p}_{ab} . An example of a map augmented with LIDAR data can be seen in Figure 8b. For sensor data such as short range

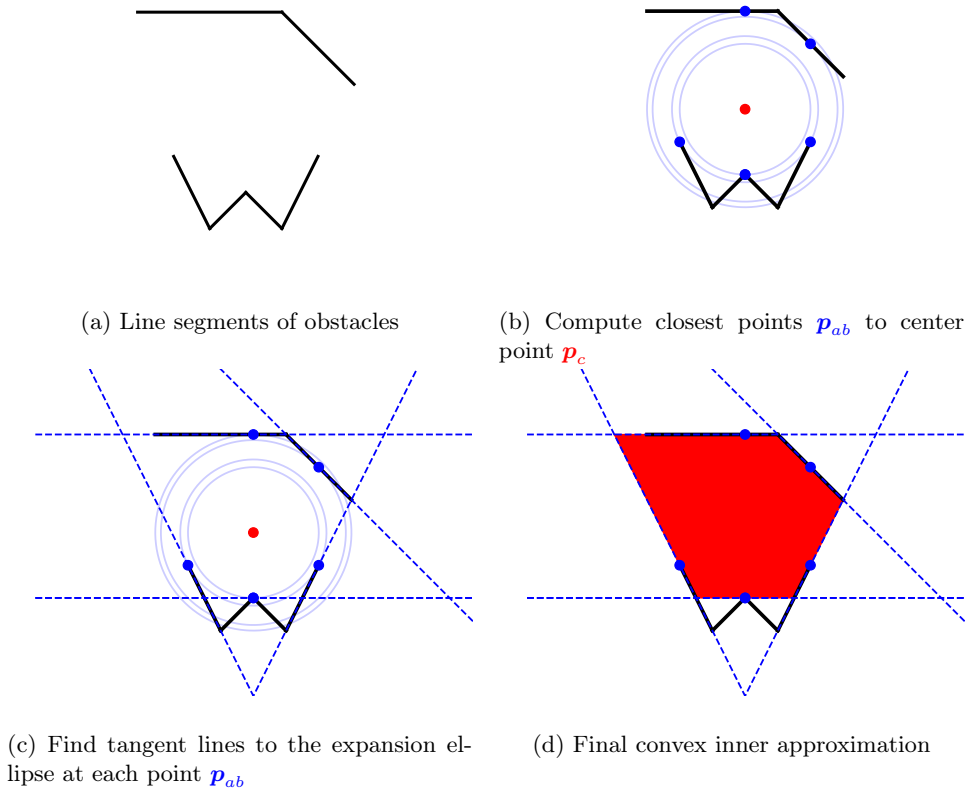


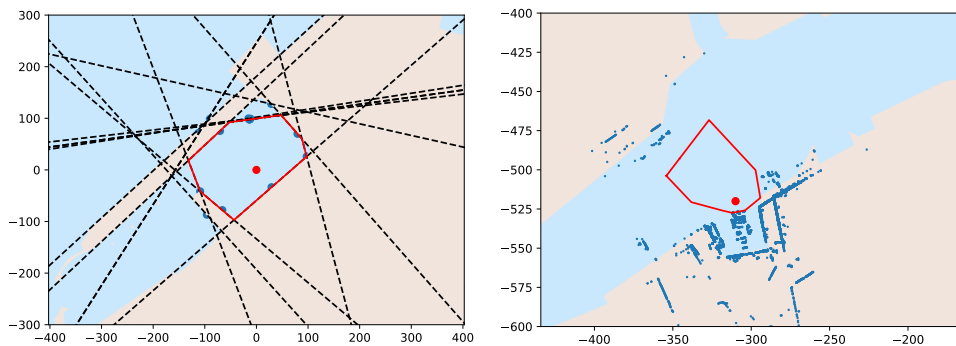
Figure 7: Illustration of how to compute the convex spatial constraints

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 Σ in order to create an axis for which we prioritize the set generation. In the case of a vessel, 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:

$$\Sigma = \mathbf{R}(\psi) \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix} \mathbf{R}(\psi)^\top, \quad (30)$$

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.



(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.

4.3 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

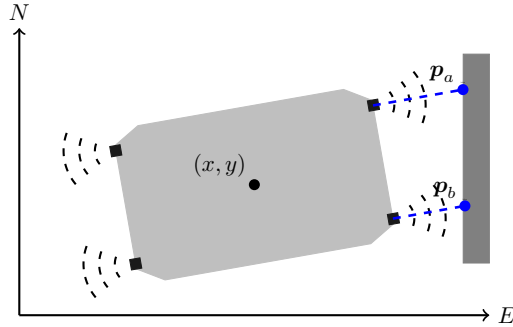


Figure 9: Ultrasonic distance sensors attached to the front and rear of the vessel, where the line segments $(\mathbf{p}_a, \mathbf{p}_b)$ can be added to the spatial constraints.

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 \mathbf{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 result 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 $\mathbf{p}_{ab} - \mathbf{p}_c$. This reduced constraint, will then be an outer approximation of the full constraints.

5 Experiments

5.1 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

Publications

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).

planning parameters given in Appendix A. 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 Autoferry¹, and is planned to operate as an on-demand ferry in the Trondheim harbor.

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.
- 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.

¹Autoferry website: <https://www.ntnu.edu/autoferry>.

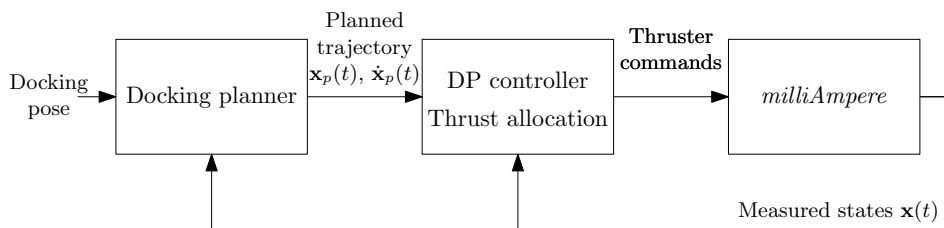


Figure 10: Block diagram of the docking system setup. The DP controller and thrust allocation are existing functions on *milliAmpere*.

- 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.

5.2 Results

Experiments² 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 $1m/s$ and $3m/s$. 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 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.

The experiments E1 were performed at the end of a floating dock, while E2 at the center of the floating dock, due 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

²Video of experiments is available at: <https://youtu.be/AyaW1JvI6K8>

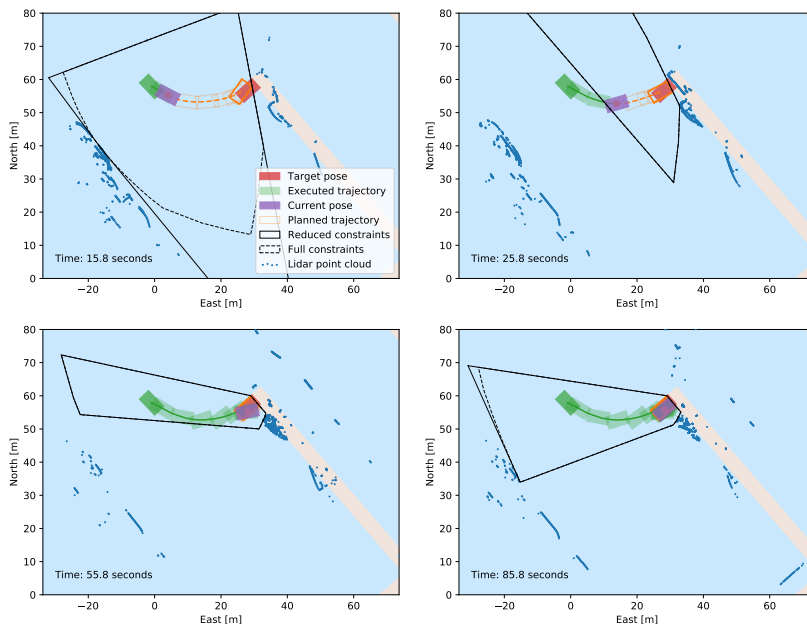


Figure 11: Visualisation of the docking motion during the experiments on September 7th 2020 (E1).

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 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

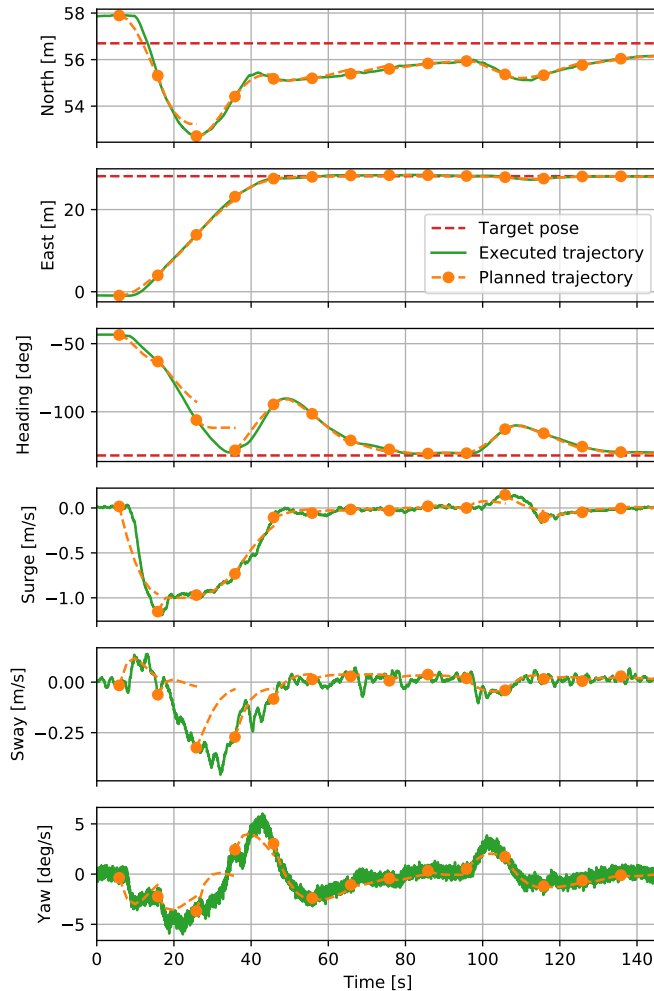


Figure 12: Planned and executed trajectory during the experiments on September 7th 2020 (E1).

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

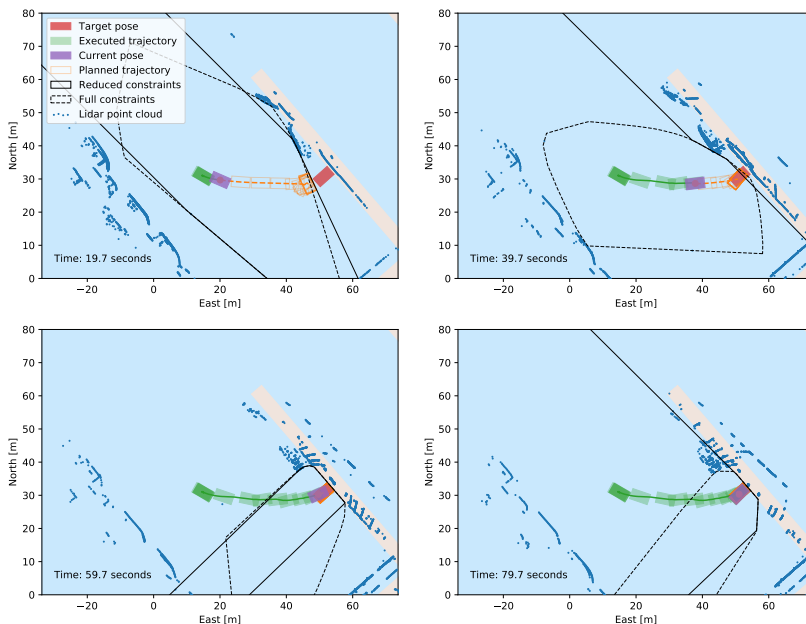


Figure 13: Visualisation of the docking motion during the experiments on September 11th 2020 (E2).

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.

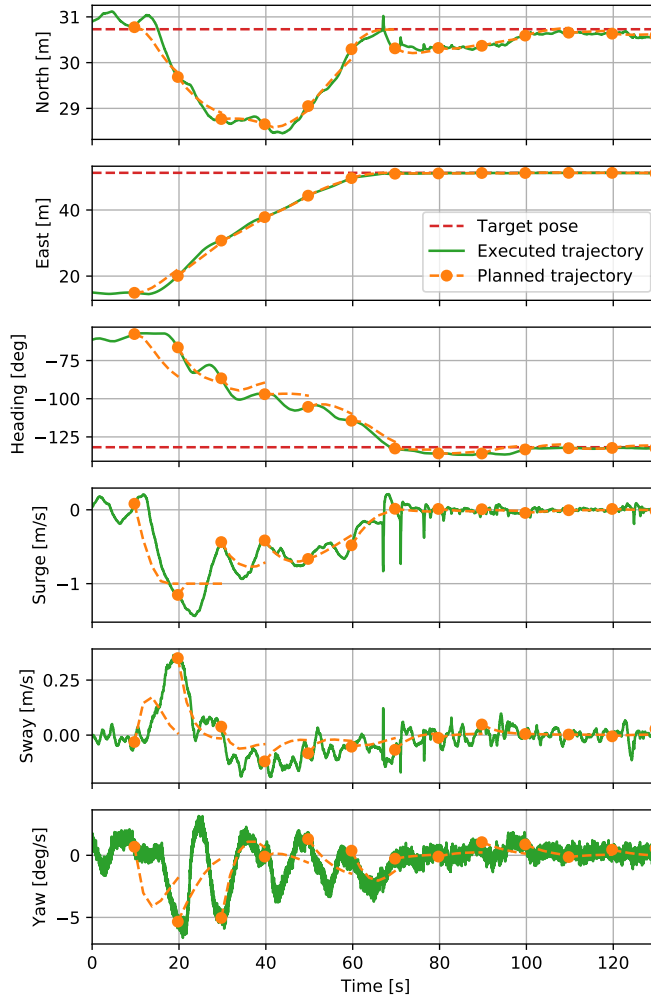


Figure 14: Planned and executed trajectory during the experiments on September 11th 2020 (E2).

6 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

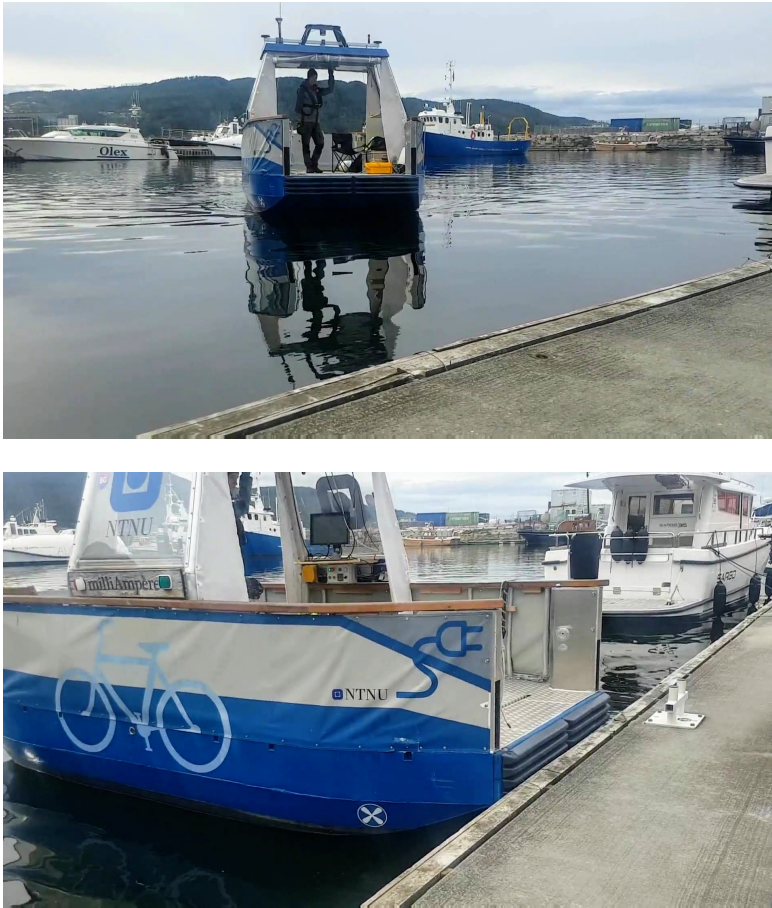


Figure 15: The *milliAmpere* while performing the docking maneuver, including closing in on the dock, and the final docking pose.

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.

A 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
\mathbf{Q}	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 100 \end{bmatrix}$	Velocity weight matrix
\mathbf{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
Σ	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	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] Eric Murdoch et al. *A Master's Guide to Berthing*. Witherby & Company, 2004.
- [2] Kazuhiko Hasegawa and T Fukutomi. "On Harbour Manoeuvring and Neural Control System for Berthing with Tug Operation". In: *Proc. 3rd International Conference on Manoeuvring and Control of Marine Craft*. 1994, pp. 197–210.
- [3] Hideki Kawai Van Phuoc Bui, Young Bok Kim, and Kwon Soon Lee. "A ship berthing system design with four tug boats". In: *Journal of Mechanical Science and Technology* 25.5 (2011), pp. 1257–1264.
- [4] Phuoc Van Bui and Young Bok Kim. "Development of Constrained Control Allocation for Ship Berthing by Using Autonomous Tugboats". In: *International Journal of Control, Automation and Systems* 9.6 (2011), pp. 1203–1208.
- [5] Van Luong Tran and Namkyun 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.
- [6] G. J. S. Rae et al. "A Fuzzy Rule Based Docking Procedure for Two Moving Autonomous Underwater Vehicles". In: *1993 American Control Conference*. IEEE, 1993, pp. 580–584.
- [7] Ken Teo, Benjamin Goh, and Oh Kwee Chai. "Fuzzy Docking Guidance Using Augmented Navigation System on an AUV". In: *IEEE Journal of Oceanic Engineering* 40.2 (2015), pp. 349–361.
- [8] Van-Suong Nguyen and Nam-Kyun Im. "Automatic Ship Berthing Based on Fuzzy Logic". In: *International Journal of Fuzzy Logic and Intelligent Systems* 19.3 (2019), pp. 163–171.
- [9] Hiroyuki Yamato. "Automatic Berthing by the Neural Controller". In: *Proc. Of Ninth Ship Control Systems Symposium*. Vol. 3. 1990, pp. 3183–3201.
- [10] K Hasegawa. "Automatic Berthing Control System Using Network and Knowledgebase". In: *Journal of the Society of Naval Architects of Japan* 220 (1993), pp. 135–143.
- [11] Yao Zhang, Grant E Hearn, and Pratyush Sen. "A Multivariable Neural Controller for Automatic Ship Berthing". In: *IEEE Control Systems Magazine* 17.4 (1997), pp. 31–45.
- [12] Namkyun Im and Kazuhiko Hasegawa. "A Study on Automatic Ship Berthing Using Parallel Neural Controller". In: *Journal of the Kansai Society of Naval Architects, Japan* 2001.236 (2001), pp. 65–70.

- [13] Yaseen Adnan Ahmed and Kazuhiko 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.
- [14] Nam-Kyun Im and Van-Suong Nguyen. “Artificial neural network controller for automatic ship berthing using head-up coordinate system”. In: *International Journal of Naval Architecture and Ocean Engineering* 10.3 (2018), pp. 235–249.
- [15] Van-Cuong Do Van-Suong Nguyen and Nam-Kyun Im. “Development of Automatic Ship Berthing System Using Artificial Neural Network and Distance Measurement System”. In: *International journal of Fuzzy logic and Intelligent systems* 18.1 (2018), pp. 41–49.
- [16] Daesoo Lee, Seung-Jae Lee, and Seo Yu-Jeong. “Application of Recent Developments in Deep Learning to ANN-based Automatic Berthing Systems”. In: *International Journal of Engineering and Technology Innovation* 10.1 (2019), p. 75.
- [17] Naoki Mizuno and Ryo Kuboshima. “Implementation and Evaluation of Non-linear Optimal Feedback Control for Ship’s Automatic Berthing by Recurrent Neural Network”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 91–96.
- [18] Ella-Lovise Hammervold Rørvik. “Automatic Docking of an Autonomous Surface Vessel”. MA thesis. Trondheim, Norway: Norwegian University of Science and Technology (NTNU), 2020.
- [19] Hiroyuki Yamato, Takeo Koyama, and Takehiko Nakagawa. “Automatic Berthing Using the Expert System”. In: *IFAC Proceedings Volumes* 25.3 (1992), pp. 173–184.
- [20] Morten Breivik and Jon-Erik Loberg. “A Virtual Target-Based Underway Docking Procedure for Unmanned Surface Vehicles”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 13630–13635.
- [21] Joohyun Woo and Nakwan Kim. “Vector Field based Guidance Method for Docking of an Unmanned Surface Vehicle”. In: *The Twelfth ISOPE Pacific/Asia Offshore Mechanics Symposium*. International Society of Offshore and Polar Engineers. 2016.
- [22] Alexander Farnsworth. “Auto-docking ferry successfully tested in Norway”. In: *Wärtsilä Press Release* (2018).
- [23] Rolls-Royce. “Rolls-Royce and Finferries demonstrate world’s first Fully Autonomous Ferry”. In: *Rolls-Royce Press Release* (2018).
- [24] Kongsberg. “AUTOMATIC FERRY ENTERS REGULAR SERVICE FOLLOWING WORLD-FIRST CROSSING WITH PASSENGERS ONBOARD”. In: *Kongsberg Press Release* (2020).
- [25] Kouichi Shouji, Kohei Ohtsu, and Sumitoshi Mizoguchi. “An Automatic Berthing Study by Optimal Control Techniques”. In: *IFAC Proceedings Volumes* 25.3 (1992), pp. 185–194.

Publications

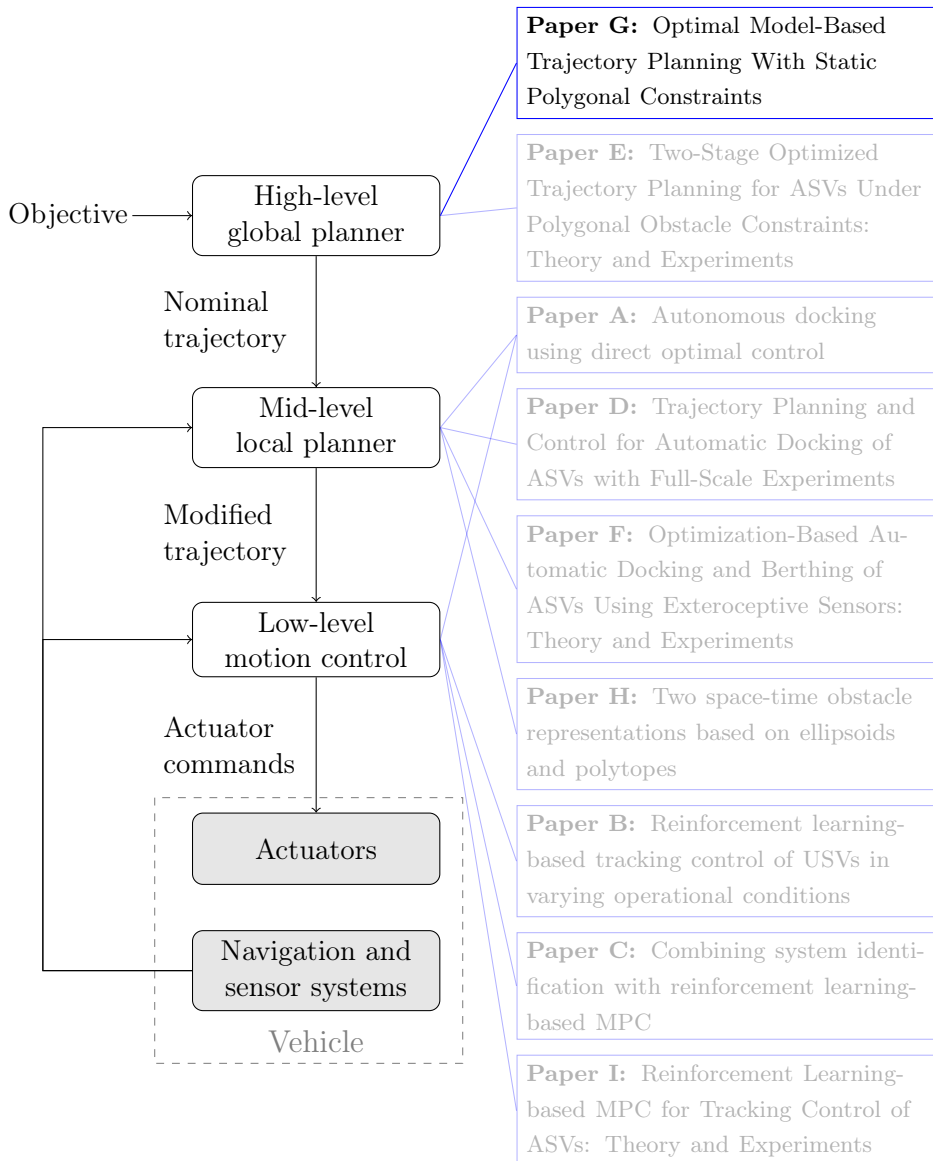
- [26] Karim Djouani and Yskandar Hamam. “Minimum time-energy trajectory planning for automatic ship berthing”. In: *IEEE Journal of Oceanic Engineering* 20.1 (1995), pp. 4–12.
- [27] K Ohtsu, K Shoji, and T Okazaki. “Minimum-time maneuvering of a ship, with wind disturbances”. In: *Control Engineering Practice* 4.3 (1996), pp. 385–392.
- [28] Tadatsugi Okazaki, Kohei Ohtsu, and Naoki Mizuno. “A Study of Minimum Time Berthing Solutions”. In: *IFAC Proceedings Volumes* 33.21 (2000), pp. 135–139.
- [29] Naoki Mizuno, Yosuke Uchida, and Tadatsugi Okazaki. “Quasi Real-Time Optimal Control Scheme for Automatic Berthing”. In: *IFAC-PapersOnLine* 48.16 (2015), pp. 305–312.
- [30] Mikkel Cornelius Nielsen, Tor Arne Johansen, and Mogens Blanke. “Cooperative Rendezvous and Docking for Underwater Robots using Model Predictive Control and Dual Decomposition”. In: *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 14–19.
- [31] N Mizuno, T Kita, and T Ishikawa. “A New Solving Method for Non-Linear Optimal Control Problem and Its Application to Automatic Berthing Problem”. In: *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 2183–2188.
- [32] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Autonomous docking using direct optimal control”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 97–102.
- [33] Glenn Bitar et al. “Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments”. In: *arXiv preprint arXiv:2004.07793* (2020).
- [34] Shijie Li et al. “Automatic Docking for Underactuated Ships Based on Multi-Objective Nonlinear Model Predictive Control”. In: *IEEE Access* 8 (2020), pp. 70044–70057.
- [35] Tadao Takai and Kohei Ohtsu. “Automatic Berthing Experiments Using "Shioji-Maru"”. In: *The Journal of Japan Institute of Navigation* 83 (1990), pp. 267–276.
- [36] L. Gucma et al. “Laser docking system integrated with pilot navigation support system. Background to high precision, fast and reliable vessel docking”. In: *17th Saint Petersburg International Conference on Integrated Navigation Systems, ICINS 2010 - Proceedings* (2010), pp. 268–275.
- [37] Marko Perkovic et al. “Accommodating larger container vessels using an integrated laser system for approach and berthing”. In: *Microprocessors and Microsystems* 52 (2017), pp. 106–116.
- [38] Thor I Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, 2011.

- [39] Tor A Johansen and Thor I Fossen. “Control allocation—A survey”. In: *Automatica* 49.5 (2013), pp. 1087–1103.
- [40] Sébastien Gros and Mario Zanon. “Penalty Functions for Handling Large Deviation of Quadrature States in NMPC”. In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3848–3860.
- [41] Sébastien Gros and Moritz Diehl. “NMPC based on Huber Penalty Functions to Handle Large Deviations of Quadrature States”. In: *2013 American Control Conference*. IEEE, 2013, pp. 3159–3164.
- [42] G. A. Hicks and W. H. Ray. “Approximation methods for optimal control synthesis”. In: *The Canadian Journal of Chemical Engineering* 49.4 (1971), pp. 522–528.
- [43] Peter Deuffhard. “A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting”. In: *Numerische Mathematik* 22.4 (1974), pp. 289–315.
- [44] T. H. Tsang, D. M. Himmelblau, and T. F. Edgar. “Optimal control via collocation and non-linear programming”. In: *International Journal of Control* 21.5 (1975), pp. 763–768.
- [45] Robin Deits and Russ 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] Sikang Liu et al. “Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments”. In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695.
- [47] Tobias Schoels et al. “CIAO*: MPC-based Safe Motion Planning in Predictable Dynamic Environments”. In: *arXiv preprint arXiv:2001.05449* (2020).
- [48] Kristoffer Bergman et al. “An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments”. In: *arXiv preprint arXiv:2005.02674* (2020).
- [49] Jan Telgen. “Identifying Redundant Constraints and Implicit Equalities in Systems of Linear Constraints”. In: *Management Science* 29.10 (1983), pp. 1209–1222.
- [50] S Paulraj and P Sumathi. “A Comparative Study of Redundant Constraints Identification Methods in Linear Programming Problems”. In: *Mathematical Problems in Engineering* 2010 (2010).
- [51] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. “The Quickhull algorithm for convex hulls”. In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483.
- [52] Tobias R. Torben, Astrid H. Brodtkorb, and Asgeir J. Sørensen. “Control allocation for double-ended ferries with full-scale experimental results”. In: *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*. 2019, pp. 45–50.

G Optimal Model-Based Trajectory Planning With Static Polygonal Constraints

Postprint of [23] **Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Optimal Model-Based Trajectory Planning With Static Polygonal Constraints”. In: *IEEE Transactions on Control Systems Technology* 29.5 (2021). DOI: 10.1109/TCST.2021.3094617

©2021 IEEE Transactions on Control Systems Technology. Reprinted and formatted to fit the thesis with permission from Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros.



Optimal Model-Based Trajectory Planning With Static Polygonal Constraints

Andreas B. Martinsen¹, Anastasios M. Lekkas^{1,2}, and Sébastien Gros¹

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

²Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway

Abstract: The main contribution of this paper is a novel method for planning globally optimal trajectories for dynamical systems subject to polygonal constraints. The proposed method is a hybrid trajectory planning approach, which combines graph search, i.e. a discrete roadmap method, with convex optimization, i.e. a complete path method. Contrary to past approaches, which have focused on using simple obstacle approximations, or sub-optimal spatial discretizations, our approach is able to use the exact geometry of polygonal constraints in order to plan optimal trajectories. The performance and flexibility of the proposed method is evaluated via simulations by planning distance-optimal trajectories for a Dubins car model, as well as time-, distance- and energy-optimal trajectories for a marine vehicle.

Keywords: Autonomous vehicles, Marine vehicles, Mobile robots, Motion planning, Optimal control, Path planning, Trajectory optimization

1 Introduction

In robotics, motion planning is the process of finding a sequence of valid configurations, which move the robot safely from some initial configuration to a goal configuration. To be successful in the real world, the motion planner must be able to consider a variety of constraints such as *environment constraints*, including static and dynamic obstacles, and *differential constraints*, which arise from the system kinematics and dynamics and are modeled with differential equations. Due to a potentially large number of obstacles, actuators, as well as complex kinematics and dynamics, motion planning is in general a difficult problem that has led to a wide range of methods and a vast literature.

Trajectory planning pertains to finding a time-parametric continuous sequence of configurations, called a trajectory, which is obstacle-free and satisfies the differential constraints (i.e. a feasible trajectory). Optimal trajectory planning has the addi-

tional task of finding the "best" feasible trajectory with respect to some performance measure, such as minimum energy, distance or time. The requirement of optimality is in general very demanding computationally since it requires an exhaustive search over the state space. One of many ways to categorize motion planning methods is to distinguish between *roadmap methods* and *complete path methods* [1–3].

The main goal of roadmap methods is to find a sequence of waypoints, which, when connected, result in an obstacle-free piecewise-linear path. The path can then be smoothed and turned into a feasible trajectory that complies with the vehicle dynamics. Roadmap methods can be further split into two distinct categories, namely, combinatorial methods and sampling-based methods. Combinatorial methods, divide the continuous space into structures that capture all spatial information needed to solve the motion planning using simple graph search algorithms. For many complex problems however, combinatorial methods may not be computationally feasible. For these problems, sampling based methods are often used instead. Sampling based methods, rely on using randomly sampled subset of states or actions. This creates a randomly sampled discretization of the continuous search space, and hence limits the computational complexity at the cost of accuracy and completeness of the discretization. Some notable combinatorial methods include coarse planning with path smoothing, in where a mesh, grid or potential field is used to plan a course path [4–7], and then a method using curve segments, splines or motion primitives is used to refine the trajectory [8–14]. Notable sampling based methods include probabilistic roadmap (PRM) [15], rapidly-exploring random tree (RRT) [16–18], and Random-walk planners [19, 20]

Complete path methods on the other hand, produce a continuous parameterized trajectory by explicitly taking into account the motion equations of the robot and the full continuous search space. As a result, these methods generate a trajectory that is both obstacle-free and feasible, without further need of refinement/smoothing. Most complete path methods rely on some form of mathematical optimization. For some simple problems an analytical solution exists, as is the case for Dubins paths [21] and Reeds-Shepp [22]. In general, however, researchers must resort to numerical optimization, where handling complex constraints is challenging and getting stuck in local optima is not uncommon. Notable numerical methods include dynamic programming [23], particle swarm optimization (PSO) [24, 25], shooting methods [26], which are based on simulation, collocation methods [27], which are based on function approximation of low-level polynomials, and pseudospectral methods [28], which are based on function approximation of high-level polynomials.

In this paper we consider the problem of optimal motion planning for a particle-like vehicle, moving on a 2D surface with polygonal obstacles. To this end, we introduce a hybrid method, which combines graph search on a pre-computed mesh, with convex optimization for path refinement. The proposed method allows for planning a globally optimal trajectory for a dynamical system subject to static polygonal constraints. The main contributions is this paper is how we combine hybrid planning with polygonal constraints and triangulation based spatial discretization. With hybrid planning,

we combine both roadmap and complete path methods. Contrary to other hybrid methods such as [29–31], where initial trajectories are planned using motion primitives and state space discretizations, and refined using numerical optimization, our method employs an iterative approach of planning and refinement. Polygonal constraints allow for complex constraints to be used in the planning algorithm. Very few optimization-based planning methods exist that are able to handle these types of constraints. Existing methods often lead to computationally expensive mixed integer optimization problems [32], rely on using inner approximations of the free space [33, 34], or non-convex elliptical approximations [2]. Our method relies on using a triangulation of the environment, similar to [4, 35] but instead of straight-line paths, it plans the path as a polynomial spline, similar to [36]. Combining the above concepts, our proposed method is able to efficiently plan globally optimal trajectories for a dynamical system subject to static polygonal constraints.

The rest of the paper is organized as follows: Section 2 outlines the method. Section 3 shows examples of distance-optimal paths for a simple kinematic car, as well as time-, distance- and energy-optimal paths for an unmanned surface vehicle. Finally Section 4 concludes the paper.

2 Method

The problem that we aim to solve in this paper, is that of planning optimal trajectories for dynamical systems in environments with static polygonal constraints. The proposed method is able to compute optimal time parameterized state trajectories:

$$\mathbf{x}(t), \quad t \in [t_0, t_f],$$

which connect some initial state \mathbf{x}_0 and final goal state \mathbf{x}_f such that:

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) = \mathbf{x}_f.$$

The trajectory is generated such that it satisfies the continuous time dynamics and kinematics of a given dynamical system on the form:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),$$

which in general may be nonlinear and have additional constraints on the states and actions. The optimized trajectory, is found such that it avoids polygonal spatial constraints that are present in the environment. This is ensured by having the path travel through a sequence of neighbouring triangles \mathcal{T}_i , with the sequence denoted $[\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N]$, where the interior of each triangle is collision free. The proposed method for solving this problem can be divided into three distinct stages.

1. **Triangulation and adjacency graph** is the first stage, where a triangulation of the environment is generated based on the polygonal constraints (Figure 1b), and an adjacency graph is calculated based on neighbouring triangles (Figure 1c).

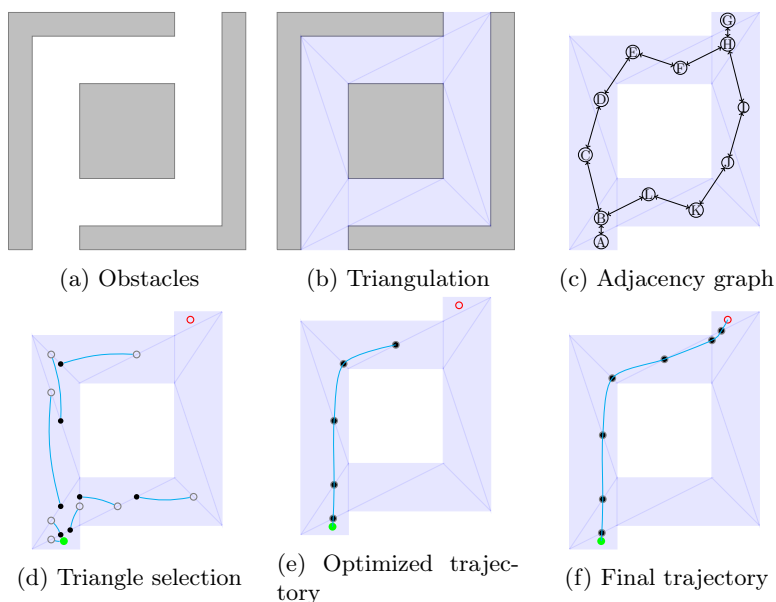


Figure 1: Given polygonal obstacles (a), the proposed algorithm finds the trajectory by creating a triangulation (b) and adjacency graph (c). Iteratively exploring different triangle sequences (d) where the refined trajectory is optimized as a spline (e). The exploration is performed until the goal is reached (f).

2. **Graph search** is the second phase, where a graph search algorithm is used to explore possible sequences of triangles in the triangulation (Figure 1c).
3. **Trajectory refinement** is the third phase, where a continuous trajectory is generated and optimized within the confinement of a sequence of triangles (Figure 1d and 1e).

2.1 Triangulation and adjacency graph

In this step, the objective is to generate a triangulation of the environment, given polygonal spatial constraints. The resulting triangulation must include the edges of the polygons, which is referred to as *constrained triangulation*. The reason for segmenting the environment into triangles in this way, is that any triangle in this type of triangulation, is either fully inside of the polygonal constraint, or fully outside of the polygonal constraint. This results in an exact, and efficient decomposition of the environment. We can then use the triangles that are fully outside of the polygonal constraints in order to plan a sequence of triangles for the trajectory to pass through, which is guaranteed to be collision free.

In this work, the triangulation that we use, is a Constrained Delaunay Triangulation (CDT) [37]. A regular Delaunay triangulation (DT) [38] will maximize the minimum angle of all the angles of the triangles in the triangulation, and hence tend to avoid sliver triangles. With CDT, certain segments are forced into the triangulation. This is necessary in order to ensure that the triangles of the triangulation are either fully inside the polygonal spatial constraints, or fully outside the spatial constraints. For the spatial constraints in Figure 1a, a constraint triangulation is given in Figure 1b.

After the triangulation is created, an adjacency graph is computed by connecting neighbouring triangles of the triangulation, where two triangles are considered neighbours if they share an edge. An illustration is shown in Figure 1c. The triangulation and adjacency graph are then used in the next phase for exploring and planning sequences of neighbouring triangles.

2.2 Graph search

Graph search can in general only be used for planning in discrete environments. In order to extend it to the continuous domain, we propose using a trajectory refinement strategy, where the graph search is performed by planning a sequence of neighbouring triangles $[\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N]$, and a continuous time parameterized trajectory $\mathbf{x}(t)$, is planned within the constraints of the sequence of triangles.

Given a CTD, we can construct a graph, where each node represents a triangle, and edges are given by neighbouring triangles, this is illustrated in Figure 1c. The goal of the graph search is to plan a sequence of triangles $[\mathcal{T}_0, \dots, \mathcal{T}_N]$, which optimizes a desired performance measure. In our case the goal is to optimize a time parameterized path integral on the form:

$$\int_{t_0}^{t_f} J(\cdot) d\tau, \quad (1)$$

where $J(\cdot)$ is a non-negative instantaneous cost. Given an initial starting point \mathbf{x}_0 , the proposed graph search method, works by starting with the initial triangle sequence $[\mathcal{T}_0]$, such that $\mathbf{x}_0 \in \mathcal{T}_0$. It then iteratively extends the sequence of triangles $[\mathcal{T}_0, \dots, \mathcal{T}_{N-1}]$, by adding new neighbouring triangles \mathcal{T}_N . This is performed until a feasible sequence of triangles $[\mathcal{T}_0, \dots, \mathcal{T}_N]$, connecting the initial state \mathbf{x}_0 and final goal state \mathbf{x}_f , is found, and a termination condition is met. The order in which potential sequences are extended, is determined by a heuristics based lower bound on the path integral. This ensures that the potentially best paths are explored first, and hence reducing the number of triangle sequences that need to be explored.

2.3 Trajectory refinement

In order to plan a continuous trajectory in an area divided into triangles, we can observe that the trajectory is constrained by the edge through which it enters, and the

edge through which it leaves any given triangle. The point at which it leaves and enters a triangle is also the point at which the trajectory enters and leaves its neighbours respectively. It is therefore possible to plan a refined trajectory through each triangle, with a given entrance and exit point along the triangle boundary (see Figure 2). This means that the final optimal trajectory, which may travel through a non-convex polygon, consists of trajectory segments constrained to lie within individual convex triangles.

Given a dynamical system on the form:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \quad (2)$$

where \mathbf{x} is the state vector, and \mathbf{u} is the control vector. The optimal trajectory through a sequence of neighbouring triangles, denoted $[\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N]$, can be written as the following optimization problem.

$$V(\mathbf{x}_0, [\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N]) = \min_{\mathbf{x}, \mathbf{u}, t} \sum_{i=0}^N \int_{t_i}^{t_{i+1}} J(\mathbf{x}, \mathbf{u}, \tau) d\tau \quad (3a)$$

$$\text{s.t. } \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \quad (3b)$$

$$\mathbf{x}(t) \in \mathcal{T}_i \quad \forall t \in [t_i, t_{i+1}] \quad (3c)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0. \quad (3d)$$

In the above optimization problem, (3b) ensures the trajectory is feasible with respect to the model, (3c) ensures each trajectory segment lies within its respective triangle, and (3d) gives the initial conditions for the optimization problem. Using the above formulation, we note that in the graph-search phase, the optimization problem is built by iteratively adding triangles to the triangle sequence $[\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N]$, and hence extending the horizon N . It should be noted that (3) can be extended to include additional state and input constraints. However, adding additional constraints may make the problem more difficult and time consuming to solve, and may lead to feasibility issues if the constraints are not chosen with care.

2.4 Complete method

Given a trajectory $\mathbf{x}(t)$, starting at \mathbf{x}_0 , and ending at \mathbf{x}_f , and going through a sequence of triangles $[\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N]$, we can define the value function of the sequence as the value that minimizes the cost along the optimal trajectory through the sequence

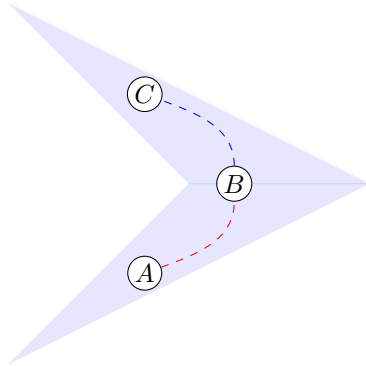


Figure 2: The trajectory ($A \rightarrow C$) through two triangles can be planned as the trajectory through each individual triangle ($A \rightarrow B$ and $B \rightarrow C$), constrained to meeting somewhere along the neighbouring edge.

of triangles, with fixed start and endpoint:

$$Q(\mathbf{x}_0, [\mathcal{T}_0, \mathcal{T}_1, \dots, \mathcal{T}_N], \mathbf{x}_f) = \min_{\mathbf{x}, \mathbf{u}, t} \sum_{i=0}^N \int_{t_i}^{t_{i+1}} J(\mathbf{x}, \mathbf{u}, \tau) d\tau \quad (4a)$$

$$\text{s.t. } \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \quad (4b)$$

$$\mathbf{x}(t) \in \mathcal{T}_i \quad \forall t \in [t_i, t_{i+1}] \quad (4c)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad (4d)$$

$$\mathbf{x}(t_{N+1}) = \mathbf{x}_f. \quad (4e)$$

Note, that this is the same optimization problem as in (3), but with the addition of the terminal constraint in (4e). Using this, we can get the result in Lemma 1.

Lemma 1. *The fixed endpoint value function $Q(\cdot)$ will always be lower bounded by the free endpoint value function $V(\cdot)$:*

$$Q(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N], \mathbf{x}_f) \geq V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N]) \quad (5)$$

Proof. The free endpoint value function $V(\cdot)$ where \mathbf{x}_f is free can be expressed in terms of minimizing the fixed endpoint value function $Q(\cdot)$ as follows:

$$\begin{aligned} V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N]) &= \min_{\mathbf{x}_f \in \mathcal{T}_N} Q(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N], \mathbf{x}_f) \\ &\leq Q(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N], \mathbf{x}_f) \quad \forall \mathbf{x}_f \in \mathcal{T}_N \end{aligned} \quad (6)$$

□

In order to determine the optimality of a sequence of triangles, we need to show that extending the sequence will not lower the cost of the trajectory. Using the value

Publications

function definitions in (3) and (4), and the following assumption, we get the result in Lemma 2.

Assumption 1. *The cost function $J(\cdot) \geq 0$ is a non-negative function. Meaning the integral of the cost can not decrease along the path.*

Lemma 2. *Given Assumption 1, the value function $V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N])$ is monotonically increasing with respect to the length of the sequence of triangles.*

Proof.

$$\begin{aligned} V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N]) &= Q(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{N-1}], \mathbf{x}_N) + V(\mathbf{x}_N, [\mathcal{T}_N]) \\ &\geq V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{N-1}]) + V(\mathbf{x}_N, [\mathcal{T}_N]) \\ &\geq V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{N-1}]) \end{aligned}$$

□

Definition 1 (Triangle sequence completeness). *We say that a sequence of triangles $[\mathcal{T}_0, \dots, \mathcal{T}_N]$ is complete if the initial state is within the initial triangle $\mathbf{x}_0 \in \mathcal{T}_0$, and the final goal state is within the final triangle $\mathbf{x}_f \in \mathcal{T}_N$. Similarly, a sequence is incomplete if the initial state is within initial triangle $\mathbf{x}_0 \in \mathcal{T}_0$, and the final goal state is not within the last triangle $\mathbf{x}_f \notin \mathcal{T}_N$.*

When searching sequences of triangles, it is useful to be able to approximate bounds on the cost to go, if the sequence is incomplete. In order to do this, we are using an admissible heuristic function $h(\mathbf{x}, \mathbf{x}_f)$ together with the optimization problem in (3) to estimate the cost to go from some state \mathbf{x} to the terminal goal state \mathbf{x}_f . Using the heuristic, we can define the following function:

$$\begin{aligned} \underline{Q}(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{M-1}], \mathbf{x}_f) &= V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{M-1}]) \\ &\quad + h(\mathbf{x}_M, \mathbf{x}_f), \mathbf{x}_M = \mathbf{x}(t_M) \end{aligned} \tag{7}$$

which is a lower bound on possible complete sequences of triangles, that extend from an incomplete sequence. This result is summed up in Lemma 3.

Assumption 2. *The heuristic function $h(\mathbf{x}_M, \mathbf{x}_f)$ is admissible. Hence the heuristic will always underestimate the true cost or value function for any feasible sequence of triangles $[\mathcal{T}_M, \dots, \mathcal{T}_N]$.*

$$h(\mathbf{x}_M, \mathbf{x}_f) \leq Q(\mathbf{x}_M, [\mathcal{T}_M, \dots, \mathcal{T}_N], \mathbf{x}_f),$$

Lemma 3. *Given Assumption 2 and a triangle sequence $[\mathcal{T}_0, \dots, \mathcal{T}_M, \dots, \mathcal{T}_N]$, we have the following lower bound on the trajectory cost:*

$$\begin{aligned} Q(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_N], \mathbf{x}_f) &\geq \underbrace{V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{M-1}]) + h(\mathbf{x}_M, \mathbf{x}_f)}_{:= \underline{Q}(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{M-1}], \mathbf{x}_f)} \end{aligned} \tag{8}$$

where $\mathbf{x}_M = \mathbf{x}(t_M)$ is the end of the optimal free endpoint trajectory given by $V(\mathbf{x}_0, [\mathcal{T}_0, \dots, \mathcal{T}_{M-1}])$.

Proof.

$$\begin{aligned}
 Q(\mathbf{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_N], \mathbf{x}_f) &= Q(\mathbf{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}], \mathbf{x}_M) \\
 &\quad + Q(\mathbf{x}_M, [\mathcal{T}_M, \dots \mathcal{T}_N], \mathbf{x}_f) \\
 &\geq V(\mathbf{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]) \\
 &\quad + Q(\mathbf{x}_M, [\mathcal{T}_M, \dots \mathcal{T}_N], \mathbf{x}_f) \\
 &\geq V(\mathbf{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}]) \\
 &\quad + h(\mathbf{x}_M, \mathbf{x}_f) \\
 &= \underline{Q}(\mathbf{x}_0, [\mathcal{T}_0, \dots \mathcal{T}_{M-1}], \mathbf{x}_f)
 \end{aligned}$$

□

Given the result from Lemma 3, where we have a lower bound $\underline{Q}(\cdot)$ for completing an incomplete sequence of triangles, we can use this to determine if completing an incomplete path will result in a complete sequence with a lower value $Q(\cdot)$, then some other completes sequence. This is summed up in Theorem 1.

Theorem 1. *Given a complete sequence of triangles $\mathcal{S}^* = [\mathcal{T}_0, \dots \mathcal{T}_N]$, and an incomplete sequence \mathcal{S}' satisfying:*

$$Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f) \leq \underline{Q}(\mathbf{x}_0, \mathcal{S}', \mathbf{x}_f), \tag{9}$$

Then completing the incomplete sequence \mathcal{S}' can not result in a trajectory with a lower value $Q(\cdot)$ then the sequence \mathcal{S}^ .*

Proof. From Lemma 3, we have that extending any incomplete sequence \mathcal{S}' to a complete sequence \mathcal{S} will result in a higher cost, i.e:

$$\underline{Q}(\mathbf{x}_0, \mathcal{S}', \mathbf{x}_f) \leq Q(\mathbf{x}_0, \mathcal{S}, \mathbf{x}_f).$$

Given the condition in (9), we get the following result:

$$\begin{aligned}
 Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f) \leq \underline{Q}(\mathbf{x}_0, \mathcal{S}', \mathbf{x}_f) &\Rightarrow \\
 Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f) &\leq Q(\mathbf{x}_0, \mathcal{S}, \mathbf{x}_f).
 \end{aligned}$$

This means that all sequences \mathcal{S} that can result from the incomplete sequences \mathcal{S}' will have higher cost then the optimal sequence \mathcal{S}^* if (9) holds. □

Using the refined trajectory cost $V(\cdot)$, heuristic admissible cost $h(\cdot)$ and the search termination conditions given Theorem 1, we can derive the complete trajectory planning Algorithm 1. Where at each iteration, the trajectory is expanded into the triangle that minimizes the cost lower bound $\underline{Q}(\cdot)$. Until a complete sequence of triangles \mathcal{S}^* is found, for which the termination condition in (9) is true for all sequences \mathcal{S}' , in the list of sequences to be searched (*open_list*). By ordering the (*open_list*) by the cost lower bound $\underline{Q}(\cdot)$, this reduces the termination to checking the termination condition (9) on only the first element in the (*open_list*). From Theorem 2 we can show that the

proposed algorithm will find the optimal sequence of triangles, and hence the globally optimal trajectory, under the assumption that the resulting optimization problem is convex. This is the case if the dynamical system results in convex constraints, as the spatial constraints will be convex due to the triangulation.

Lemma 4. *In Algorithm 1, the list of sequences to be searched (`open_list`) will always contain a sub-sequence \mathcal{S}' of any possible complete path \mathcal{S}*

Proof. Algorithm 1, changes the `open_list` by iterative removing incomplete sequences, and adding all feasible sequences that can be extended by one triangle from the sequence that is removed. Since any possible complete path must be extended from the sequence only containing the initial triangle \mathcal{T}_0 . Then the list of sequences to be searched (`open_list`) will always contain a sub-sequence of any possible complete path. \square

Theorem 2. *Algorithm 1 will find the optimal sequence of triangles \mathcal{S}^* , and hence the globally optimal trajectory.*

Proof. Given that Algorithm 1 terminates with the optimal sequence \mathcal{S}^* . If we assume there exists a better sequence $\tilde{\mathcal{S}}^*$. such that:

$$Q(\mathbf{x}_0, \tilde{\mathcal{S}}^*, \mathbf{x}_f) < Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f)$$

Then from Lemma 4, a sub-sequence $\tilde{\mathcal{S}}'$ of $\tilde{\mathcal{S}}^*$ must exist in the list of possible sequences to be extended (`open_list`). Given the result in Lemma 3 we get that:

$$\underline{Q}(\mathbf{x}_0, \tilde{\mathcal{S}}', \mathbf{x}_f) \leq Q(\mathbf{x}_0, \tilde{\mathcal{S}}^*, \mathbf{x}_f) < Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f).$$

This contradicts the termination condition in (9), and hence no sequence $\tilde{\mathcal{S}}^*$ that is better than \mathcal{S}^* can exist. \square

2.5 Implementation considerations

In order to implement the optimization problem given in (3) and (4), we need to formulate the constraint in (3c) and (4c) as a linear inequality constraint. The most straightforward way of doing this is to use the half-space representation of the triangle. Given a 2D triangle \mathcal{T}_i with vertices $\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, \mathbf{v}_{i,3}$, as illustrated in Figure 3, the half-space representation of a triangle gives a set of linear inequality constraints on the form:

$$\mathbf{A}_i \mathbf{p} \leq \mathbf{b}_i.$$

Where $\mathbf{A}_i \in \mathbb{R}^{3 \times 2}$ and $\mathbf{b}_i \in \mathbb{R}^{3 \times 1}$ is the matrix and vector making up the halfspace, and $\mathbf{p} = [x, y]^\top$ is a position. Using this, we can check if a position \mathbf{p} lies within the triangle \mathcal{T}_i , as follows:

$$\mathbf{A}_i \mathbf{p} \leq \mathbf{b}_i \quad \Leftrightarrow \quad \mathbf{p} \in \mathcal{T}_i. \quad (10)$$

Algorithm 1 Optimal trajectory planning. Note that evaluating $Q(\cdot)$ and $\underline{Q}(\cdot)$ involves solving the optimization problem in (3) and (4) using numerical optimization.

Require: Adjacency graph of triangulation, initial state \mathbf{x}_0 , and goal state \mathbf{x}_f .

```

 $\mathcal{S}^* = []$ 
 $\mathcal{S} = [\mathcal{T}_0]$  where  $\mathbf{x}_0 \in \mathcal{T}_0$ 
 $open\_list = \{\mathcal{S}\}$ 
while  $open\_list$  is not empty do
   $\mathcal{S} = \text{pop}$  sequence from  $open\_list$  with smallest  $\underline{Q}(\mathbf{x}_0, \mathcal{S}, \mathbf{x}_f)$ 
  if  $\mathcal{S}^*$  is not empty, and  $\underline{Q}(\mathbf{x}_0, \mathcal{S}, \mathbf{x}_f) \geq Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f)$  then
    return Optimal triangle sequence  $\mathcal{S}^*$ 
  end if
  for Triangle  $\mathcal{T}_n$  in  $neighbours(\mathcal{S})$  do
     $\mathcal{S}_n = \text{extend}(\mathcal{S}, \mathcal{T}_n)$ 
    if  $\mathbf{x}_f \in \mathcal{T}_n$  then
      if  $Q(\mathbf{x}_0, \mathcal{S}_n, \mathbf{x}_f) < Q(\mathbf{x}_0, \mathcal{S}^*, \mathbf{x}_f)$  then
         $\mathcal{S}^* = \mathcal{S}_n$ 
      end if
    else
      append  $\mathcal{S}_n$  to  $open\_list$ 
    end if
  end for
end while

```

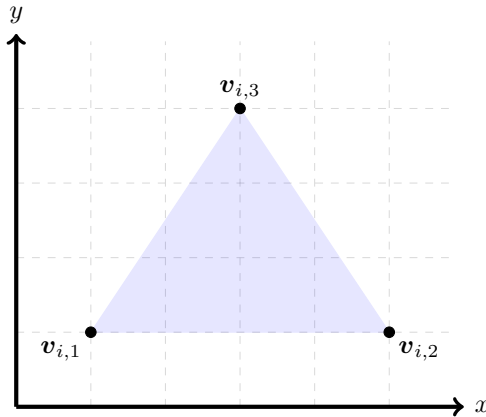


Figure 3: Triangle \mathcal{T}_i , with vertices $\mathbf{v}_{i,1}$, $\mathbf{v}_{i,2}$, $\mathbf{v}_{i,3}$

The matrix \mathbf{A}_i , and vector \mathbf{b}_i can be computed using the triangle vertices $\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, \mathbf{v}_{i,3}$ as follows:

$$\begin{aligned} \mathbf{A}_i &= \begin{bmatrix} (\mathbf{v}_{i,2} - \mathbf{v}_{i,1})^\top \mathbf{R}^\top \\ (\mathbf{v}_{i,3} - \mathbf{v}_{i,2})^\top \mathbf{R}^\top \\ (\mathbf{v}_{i,1} - \mathbf{v}_{i,3})^\top \mathbf{R}^\top \end{bmatrix} \\ \mathbf{b}_i &= \begin{bmatrix} (\mathbf{v}_{i,2} - \mathbf{v}_{i,1})^\top \mathbf{R}^\top \mathbf{v}_{i,1} \\ (\mathbf{v}_{i,3} - \mathbf{v}_{i,2})^\top \mathbf{R}^\top \mathbf{v}_{i,2} \\ (\mathbf{v}_{i,1} - \mathbf{v}_{i,3})^\top \mathbf{R}^\top \mathbf{v}_{i,3} \end{bmatrix} \end{aligned} \quad (11)$$

Where the matrix \mathbf{R} is given as the $\pm 90^\circ$ rotation matrix, when the triangle vertices are given in a in a clockwise/counter clockwise direction. In the example in Figure 3, the vertices are given in a counter clockwise direction, giving the following rotation matrix:

$$\mathbf{R} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

While the above linear inequality can be used to ensure the different path segments stay within the desired triangle, we propose a slight modification to this approach. The modification involves using a local triangle-centered coordinate system instead of a global coordinate system for optimizing the position within the triangle. Defining the following objects:

$$\begin{aligned} \mathbf{C}_i &= [\mathbf{v}_{i,2} - \mathbf{v}_{i,1}, \mathbf{v}_{i,3} - \mathbf{v}_{i,2}] \\ \mathbf{d}_i &= \mathbf{v}_{i,1}, \end{aligned} \quad (12)$$

we define the transformation between the position $\mathbf{p} = [x, y]^\top$ in the global coordinate system, and the position $\mathbf{p}' = [p'_1, p'_2]^\top$ in the local triangle coordinate system as follows:

$$\mathbf{p} = \mathbf{C}_i \mathbf{p}' + \mathbf{d}_i. \quad (13)$$

Using the triangle transformation in (13), the position \mathbf{p} is constrained within the triangle when the following inequality constraints are satisfied:

$$\begin{aligned} 0 &\leq \mathbf{p}' \leq 1 \\ p'_1 - p'_2 &\leq 0, \end{aligned} \quad (14)$$

which can be implemented directly into the optimization problem as the triangle constraints in equation (3c) and (4c). The reason for using this coordinate transformation is to help normalize the variables in the optimization problem as well as simplify the triangle constraints. This helps improve the conditioning of the optimization problem, and gives better performance when solving the problem.

Another consideration when solving (3) and (4), is how to perform the integration of the cost function (3a) and (4a), and system dynamics (3b) and (4b). In order to do this we propose using a multiple shooting collocation based scheme [39], for which the trajectory in each triangle is approximated by a polynomial of degree d . This results in an optimization problem, where the objective is to find a spline where each triangle contains a polynomial representing the trajectory through the triangle (Figure

1d), the trajectories are then constrained to being connected between neighbouring triangles (Figure 1e), while at the same time satisfy the system dynamics. It is worth noting that the trajectory within each triangle will differ in length due to the size and shape of the triangle. This means the a free time variable must be used for each triangle in order to ensure the trajectory is constrained within the triangle.

In the graph search phase, some additional assumptions were made, in order to prune and reduce the search space.

Assumption 3. *The optimal path will only pass through any given triangle \mathcal{T} once.*

Assumption 3, allows us to not extend a sequence of triangles into a given triangle if it already appears in the sequence. This results in a significantly smaller search space, when searching for the optimal triangle sequence. It should be noted that Assumption 3 is not strictly necessary, as the proposed method will in theory work without it. It does however significantly reduce the search space, and helps make the method computationally feasible. The downside of this assumption, is that the proposed planner will not allow for maneuvers which reenter triangles, which may be optimal for certain classes of problems.

Assumption 4. *If two initial starting points $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{T}$ are sufficiently close:*

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \leq \epsilon.$$

Then the optimal sequences of triangles \mathcal{S}^ to the goal will be the same for both trajectories, and the difference between values of the trajectories is bounded.*

$$\|Q(\mathbf{x}_1, \mathcal{S}^*, \mathbf{x}_f) - Q(\mathbf{x}_2, \mathcal{S}^*, \mathbf{x}_f)\| \leq \delta$$

Given two different triangle sequences \mathcal{S}_1 and \mathcal{S}_2 , that both end in the same triangle \mathcal{T} , and the same endpoints $\mathbf{x}_1 = \mathbf{x}_2$, $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{T}$, where:

$$V(\mathbf{x}_0, \mathcal{S}_1) \leq V(\mathbf{x}_0, \mathcal{S}_2),$$

we only need to continue the search from the sequence \mathcal{S}_1 , and hence can prune the sequence \mathcal{S}_2 . Using Assumption 4, we can extend the above argument to say that we can prune sequences if the states are sufficiently close. Unfortunately, computing the exact bounds would require completing the trajectory, which defeats the purpose of pruning. In stead we use the following heuristic for evaluating if two endpoints \mathbf{x}_1 and \mathbf{x}_2 are sufficiently close:

$$(\mathbf{x}_1 - \mathbf{x}_2)^\top \mathbf{W}(\mathbf{x}_1 - \mathbf{x}_2) \leq \epsilon, \quad \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{T}$$

where \mathbf{W} is a positive definite weighting matrix, and ϵ is a sufficiently small threshold. This is a relaxation of the exact condition for pruning, where $\mathbf{x}_1 = \mathbf{x}_2$, $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{T}$, and where the conditions are exactly the same in the limit as $\epsilon \rightarrow 0$. It should be noted that pruning potential sequences is not strictly necessary. It is however added

in order to further reduce the search space, and hence improve the computational complexity.

When running Algorithm 1, there may be certain triangle sequences for which no feasible trajectory can be found due to the constraints imposed by the vehicle dynamics, as well as additional state and input constraints. When an infeasible triangle sequence is found, no feasible path through the given sequence exists, and the sequence is discarded. In practise, numerical optimization tools are used to find the optimal trajectories and detect infeasible solutions. In certain situations, the numerical optimization tools may not return a feasible solution even though a feasible solution theoretically does exist. This can typically be mitigated using good initial guesses for the trajectory. For the proposed planner, the optimal trajectory from the previous triangle sequence can be used as one such initial guess. In addition to improving feasibility, using such an initial guess will typically also improve the computation time, as a good initial guess will result in fewer iterations when solving the optimization problem in (3) and (4).

Given algorithm 1, we can note that it is possible to parallelize the exploration of new triangle sequences. This is possible, as the exploration of possible sequences is not dependant on other sequences, however it requires some extra considerations in the termination criteria. For our implementation, this property was exploited in order to explore multiple sequences in parallel. It should be noted that if an exact heuristic function is known, the parallelization will not give a speedup, as the optimal sequence of triangles will always be the first to be explored. If however a poor heuristic function is used, parallelization will typically give a speedup, as it allows for multiple triangle sequences to be explored simultaneously.

3 Examples

In order to validate the method, we test it on a simple kinematic car model in a confined environment, and compare to a Rapidly-exploring Random Tree based approach. To further prove the versatility of the method we also show it on a test scenario in trajectory planning for marine vessels in the Trondheim fjord, for which we use it to plan trajectories that minimize time, distance as well as energy.

3.1 Trajectory planning for a simple kinematic car model

3.1.1 Simple kinematic car model

In order to verify the proposed method we will in this section show how it can be applied to planning distance optimal paths for a simple kinematic car model on the

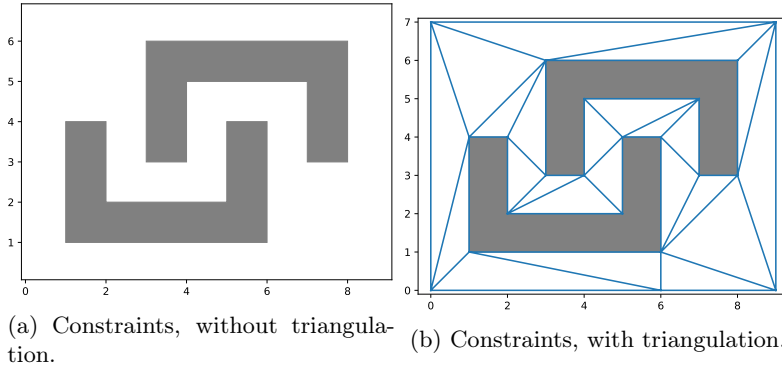


Figure 4: Polygonal spatial constraints, and the resulting CDT

form:

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} \cos(\psi)v \\ \sin(\psi)v \\ r \end{bmatrix}}_{f(\mathbf{x}, \mathbf{u})} \quad (15)$$

where x, y is the position, ψ is the heading, v is the velocity, and r is the turning rate. Using the constant speed $v = 1$, we are left with an under actuated system where the turning rate is the control variable $\mathbf{u} = r$. This type of model is often used robotics and control theory when planning paths for wheeled robots, airplanes and underwater vehicles. As the model offers a simple geometric approximation of the maneuvering capabilities of these types vehicles.

3.1.2 Spatial constraints

In order to validate the proposed method, the simple set of spatial constraints, seen in Figure 4a, were devised. Given the polygonal representation, the CDT was computed, giving the triangulation in Figure 4b

3.1.3 Optimization objective

The objective for the optimization problem, is to find the shortest path between two points. The instantaneous is then given by the path integral as follows:

$$\begin{aligned} J(\mathbf{x}, \mathbf{u}, t) &= \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \\ &= \sqrt{(\cos(\psi)v)^2 + (\sin(\psi)v)^2} \\ &= |v| \\ &= 1. \end{aligned} \tag{16}$$

It should be noted, that given a maximum turning rate, and the vehicle traveling at constant speed, the distance optimal path from one point to an other can be shown to be a Dubins path [21], which consists of straight lines and circles segments of maximum curvature.

3.1.4 Results

As the optimal path is known to be a Dubins path, a Dubins based RRT method [40] is used for comparison, as RRT based methods are the most commonly used approaches for motion planning for robotic applications when faced with spatial constraint. Given the spatial constraint in Figure 4, we get the resulting planned path in Figure 5. From the results we see that one of the major flaws of the Dubins based RRT method is that its performance is highly dependant on the randomly sampled nodes, which are used to select way points. For RRT, finding a feasible path is fairly quick, and it is possible to continue to optimize the path by generating new nodes. Further optimizing the path can often be very time consuming, as the RRT path can only guarantee converge to the optimal path as the number of sampled nodes approaches infinity [41]. For our proposed approach however, if a feasible path is found it is guaranteed to be optimal. This is verified in the results, where we can observe that our approach generates a path which is very similar to a Dubins path, and finds the shortest path that gets close to, but does not intersect the spatial constraints. In real time planning, where finding a feasible path in a short period of time is important, sampling based methods such as RRT will still be the better choice. However, if the goal is to find the optimal path in a finite amount of time, our proposed approach will be the better option, with our implementation of the optimization based planner using an average of 3.24 seconds for planning the optimal trajectory in Figure 5a.

3.2 Trajectory planning for an autonomous surface vessel

In the field of marine robotics, motion planning is an important problem, which has seen a lot of interest. Given the complex vessel dynamics, as well as complex

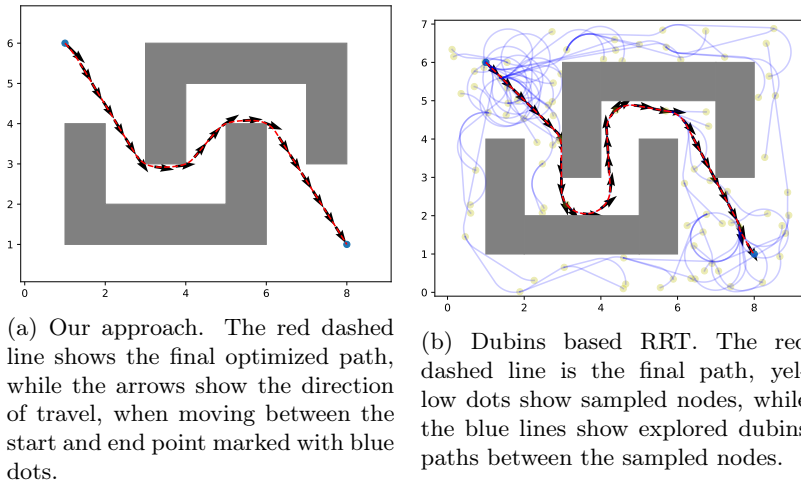


Figure 5: Results for path generated by our proposed approach and Dubins based RRT.

non-convex spatial constraints, the motion planning problem becomes very difficult. Because of this, most existing solutions heavily rely on simplifying the problem, this however results in loss of accuracy and optimality of the final solution. In this example we will show how our proposed planning algorithm can be used for optimal trajectory planning for an Autonomous Surface Vessel (ASV) in the Trondheim harbour.

3.2.1 Vessel model

As a model for the trajectory optimization, we will use a vessel model, where we assume the vessel moves on the ocean surface in a relatively large range of possible velocities. In addition to this, we assume that the effects of the roll and pitch motions of the vessel are negligible, and hence have little impact on the surge, sway, and yaw of the vessel. The mathematical model used to describe the system can then be kept reasonably simple by limiting it to the planar position and orientation of the vessel. The motion of a surface vessel can be represented by the pose vector $\boldsymbol{\eta} = [x, y, z_r, z_i]^\top \in \mathbb{R}^4$, and the velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$. Here, (x, y) describe the Cartesian position in the earth-fixed reference frame, (z_r, z_i) is a complex number of unit length $|z| = |z_r + i \cdot z_i| = 1$ describing the vessel orientation, where $\psi = \text{atan2}(z_i, z_r)$ is yaw angle, (u, v) is the body fixed linear velocities, and r is the yaw rate, an illustration is given in Figure 6. Using the notation in [42] we can describe a 3-DOF vessel model as follows

$$\dot{\boldsymbol{\eta}} = \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu}, \quad (17)$$

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}, \quad (18)$$

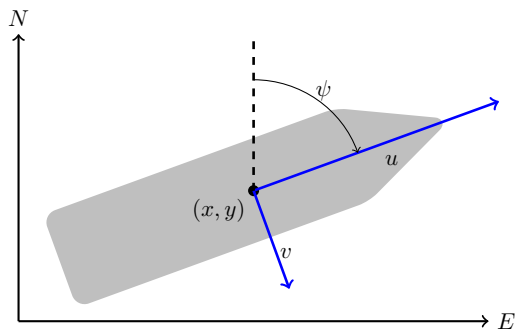


Figure 6: 3-DOF vessel centered at (x, y) , with surge velocity u , sway velocity v , heading ψ in a North-East-Down (NED) reference frame.

where $M, C(\boldsymbol{\nu}), D(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\boldsymbol{\tau} \in \mathbb{R}^3$ and $\mathbf{J}(\boldsymbol{\eta})$ are the inertia matrix, coriolis matrix, dampening matrix, control input vector, and transformation matrix respectively. The transformation matrix $\mathbf{J}(\boldsymbol{\eta})$ is given by

$$\mathbf{J}(\boldsymbol{\eta}) = \begin{bmatrix} z_r & -z_i & 0 \\ z_i & z_r & 0 \\ 0 & 0 & -z_i \\ 0 & 0 & z_r \end{bmatrix}, \quad (19)$$

and is the transformation from the body frame to the earth-fixed reference frame. Using the unit complex numbers in stead of a heading angle allows the the dynamics to avoid the angle wraparound problem, which avoids local optima when performing trajectory optimization. For the model dynamics $M, C(\boldsymbol{\nu}), D(\boldsymbol{\nu})$, parameters for a simplified model of the milliAmpere experimental platform was used, where:

$$\mathbf{M} = \begin{bmatrix} 2138 & 0 & 0 \\ 0 & 2528 & 0 \\ 0 & 0 & 3942 \end{bmatrix} \quad (20)$$

$$\mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} 10.3u + 114.6|u|u - 2528vr \\ 13.0v + 200.8|v|v + 2138ur \\ 201.0r + 424.1|r|r + 390uv \end{bmatrix}. \quad (21)$$

For the thrust configuration, one rotatable azimuth thruster is assumed, giving the following thrust vector:

$$\boldsymbol{\tau} = \begin{bmatrix} u_1 \cos(u_2) \\ u_1 \sin(u_2) \\ -2u_1 \sin(u_2) \end{bmatrix}, \quad (22)$$

Where $0 \leq u_1 \leq 400$ is the thruster force, and $-45^\circ \leq u_2 \leq 45^\circ$ the thruster angle.



(a) Map, without triangulation. (b) Map, with triangulation.

Figure 7: Map of the Trondheim fjord, based on polygons representing land masses.

3.2.2 Spatial constraints

Using a map where landmasses are represented by polygons, a CDT is created, where all edges of the polygons are treated as constraints. Doing this ensures that the resulting triangulation has triangles that do not intersect land. The resulting triangulation mesh is shown in Figure 7. While the whole map of the Trondheim fjord is used, for the example, only a small portion of the map was relevant as the start and goal positions were selected within the Trondheim harbour.

3.2.3 Optimization objective

Depending on the use-case, any optimization objective satisfying Assumption 1 can be selected. In this paper we will show three commonly used objectives, namely time minimization, distance minimization, and energy minimization.

Minimum time

In terms of instantaneous cost, the time minimization is the simplest optimization objective. where:

$$J(\mathbf{x}, \mathbf{u}, t) = 1. \tag{23}$$

This gives the path integral optimization problem as follows:

$$\int_{t_0}^{t_N} 1 dt. \tag{24}$$

Minimizing the above expression then equates to minimizing the total time, $t_N - t_0$, of the the trajectory, with boundary conditions given by the initial and final state.

For the heuristic function of the minimum time, we chose the time taken traveling in a straight line from the given state \mathbf{x}_N to the desired terminal state \mathbf{x}_f , at the

Publications

maximum vessel speed U_{\max} . Giving the following heuristic function:

$$h(\mathbf{x}_N, \mathbf{x}_f) = \frac{\sqrt{(x_N - x_f)^2 + (y_N - y_f)^2}}{U_{\max}}. \quad (25)$$

Intuitively, we can see that this is an admissible heuristic, as it represents the time of traveling the shortest possible path, at the highest speed possible, hence it will always underestimate the time of a feasible trajectory.

Minimum distance

In terms of minimizing distance, we can observe that the instantaneous cost of a trajectory given by the north, and east coordinates $x(t)$ and $y(t)$ respectively, is given as the instantaneous arc length:

$$\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}. \quad (26)$$

From the kinematics we note that the square of the instantaneous cost can be rewritten as:

$$\begin{aligned} \dot{x}^2 + \dot{y}^2 &= \cos(\psi)^2 u^2 + \sin(\psi)^2 v^2 - \cos(\psi) \sin(\psi) uv \\ &\quad + \sin(\psi)^2 u^2 + \cos(\psi)^2 v^2 + \cos(\psi) \sin(\psi) uv \\ &= (\cos(\psi)^2 + \sin(\psi)^2)(u^2 + v^2) \\ &= u^2 + v^2, \end{aligned} \quad (27)$$

giving the following instantaneous cost.

$$J(\mathbf{x}, \mathbf{u}, t) = \sqrt{u^2 + v^2} \quad (28)$$

This gives the path integral optimization problem as follows:

$$\int_{t_0}^{t_N} \sqrt{u^2 + v^2} dt. \quad (29)$$

Theoretically optimizing the above problem should give the shortest path, however for most optimization algorithms, the objective must be smooth and continuously differentiable, which is not the case when the square root is used. In order to ensure the function is continuously differentiable, a small positive number $\epsilon > 0$ is added, giving the following smooth approximation of the path integral:

$$\int_{t_0}^{t_N} \sqrt{u^2 + v^2 + \epsilon} dt. \quad (30)$$

For the heuristic function of the minimum distance, we simply chose the euclidean distance from the given state \mathbf{x}_N to the desired terminal state \mathbf{x}_f .

$$h(\mathbf{x}_N, \mathbf{x}_f) = \sqrt{(x_N - x_f)^2 + (y_N - y_f)^2} \quad (31)$$

Intuitively, we can see that this is an admissible heuristic, as represents the straight line path, which is the shortest possible path between two points. This means that it will always underestimate the length of a feasible trajectory.

Minimum energy

In many problems, it is often useful to minimize the energy usage. In the case of marine vessels, minimizing energy usage, equates to better fuel efficiency, and less pollution. For moving objects, the quantity of work over time (power) is integrated along the trajectory of the point of application of the force. This gives the instantaneous power as the scalar product of the force/torque and the linear/angular velocity.

$$\boldsymbol{\tau}^\top \boldsymbol{\nu} \tag{32}$$

In general, power regeneration and recapture is not possible for marine vessels, In order to account for this we instead use the absolute instantaneous power, giving the following instantaneous cost:

$$J(\mathbf{x}, \mathbf{u}, t) = |X \cdot u| + |Y \cdot v| + |N \cdot r|, \tag{33}$$

where the thrust vector is given as $\boldsymbol{\tau} = [X, Y, N]^\top$, and velocity vector is given as $\boldsymbol{\nu} = [u, v, r]$. This gives the path integral optimization problem as follows:

$$\int_{t_0}^{t_N} |X \cdot u| + |Y \cdot v| + |N \cdot r| dt. \tag{34}$$

Similarly to the minimum distance formulation, the absolute value is none smooth and the derivative not defined at 0, in order to avoid this problem, we again use an approximation of the absolute value giving the following path integral to be optimized.

$$\int_{t_0}^{t_N} \sqrt{(X \cdot u)^2 + \epsilon} + \sqrt{(Y \cdot v)^2 + \epsilon} + \sqrt{(N \cdot r)^2 + \epsilon} dt. \tag{35}$$

For the heuristic function of the minimum energy, it is difficult to find a good estimate for the cost to go from a given state \mathbf{x}_N to the desired terminal state \mathbf{x}_f . Hence the heuristic:

$$h(\mathbf{x}_N, \mathbf{x}_f) = 0 \tag{36}$$

is chosen. This is in general a poor estimate of the cost to go, and will result in a larger number of triangles being explored, but it is an admissible heuristic and hence satisfies Assumption 2.

3.2.4 Results

To visualize the proposed algorithm during the search phase, the value functions are shown in Figure 8. For the three different optimization objectives, the resulting

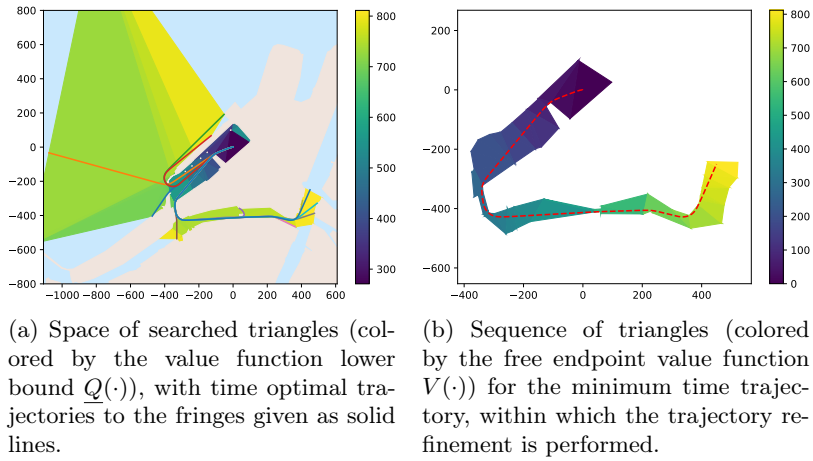


Figure 8: Search space and triangulation for minimum time trajectory.

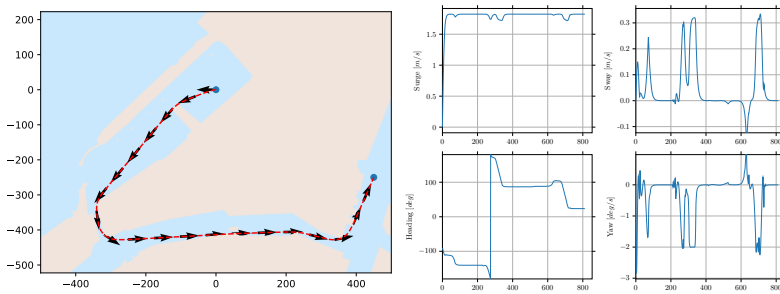


Figure 9: Minimum time path

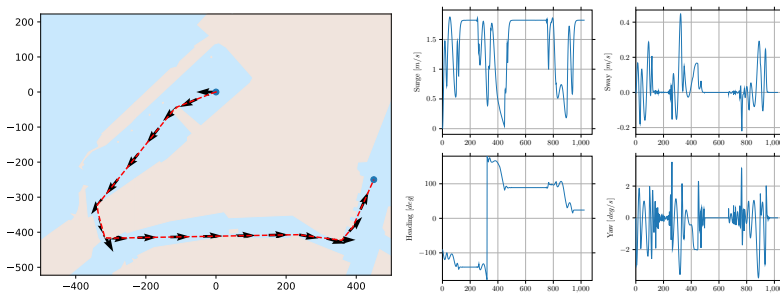


Figure 10: Minimum distance path

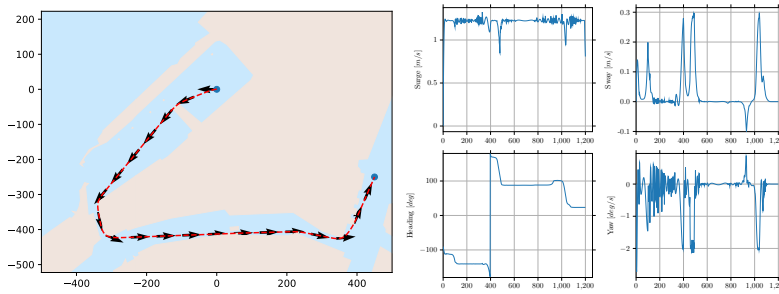


Figure 11: Minimum energy path

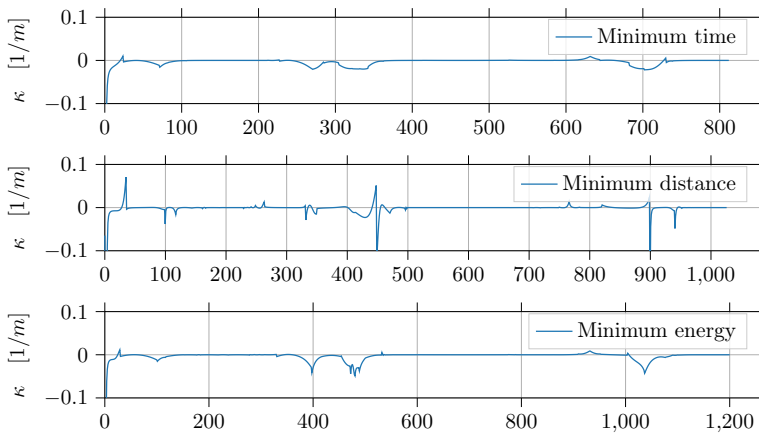


Figure 12: Trajectory curvature resulting from the different optimization objectives.

trajectories from the trajectory planner are given in Figures 9, 10 and 11 for the time, distance and energy minimization problems respectively. For the energy minimization problem, it is important to note that the any actuation of the control surfaces will result in energy being used, hence the optimal action would be to not move. In order to fix this, a terminal constraint was added on the time, in order to ensure the trajectory would be complete within 1200 seconds.

From the performance measure comparison in Table 1, we can see that the different optimization objectives perform as expected, as they each minimize their respective objectives. For the minimum time objective, we can see that the speed in the surge direction is close to the maximum for most of the duration of the trajectory, this is what results in the minimum time trajectory, but comes at the cost of a slightly longer trajectory in terms of distance, and a significantly larger energy consumption. For the minimum distance trajectory we see a more erratic behaviour, especially in the surge direction. This pattern of speeding up and slowing down, is what allows the vessel to take tight corners, and hence minimize the distance, however due to the trajectory dynamics, the resulting distance is only slightly shorter than that of the other two optimization objectives. For the minimum energy trajectory, the behaviour is similar to that of the minimum time objective, with the main difference being a lower surge speed. This behaviour is due to the nonlinear drag, which makes lower speeds more energy efficient.

A useful tool for evaluating a the feasibility of a trajectory, is the trajectory curvature κ .

$$\kappa = \frac{\dot{x} \cdot \ddot{y} - \dot{y} \cdot \ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \quad (37)$$

One of the reasons for the curvature being used as a way of evaluating trajectory feasibility, is that most vessels have a limit on the maximum possible path curvature. This has lead to the widespread use of Dubins paths [21] which consist of straight line segments and circle arcs with maximum curvature, giving path with piecewise constant curvature. These paths have been shown to be the shortest path for a vehicle that only travels forward, and has a constraint on max curvature. The Dubins path however does not consider the underlying system dynamics, hence a dubins path is no longer optimal once the dynamics are considered. This is illustrated in Figure 12 where the curavature is continuous, similar to [8]. From the curvature results it is worth noting the difference in curvature between the different optimization objectives. For the minimum time objective a higher speed is desired, hence the curvature is small allowing for taking turns at higher speeds. For the minimum distance trajectory, we can observe spikes of very high curvature, which is what we expect as the shortest path will consist only of straight line segments. For the minimum energy trajectory, we see similar results to that of the minimum time path, however the peak curvature is slightly higher, as a result of the velocities being lower.

For the implementation of Algorithm 1 used to solve the trajectory planning problem, we achieved the algorithm running time given in Table 2. The timing shows the results for running the algorithm sequentially, as well as the performance when running

Trajectory	Time [s]	Distance [m]	Energy [kJ]
Minimum Time	811.81	1460.97	584.82
Minimum Distance	1025.82	1450.58	484.70
Minimum Energy	1200.00	1456.20	269.96

Table 1: Performance measure

	Sequential	4 workers	8 workers
Minimum Time	4min 28s	6min 1s	7min 19s
Minimum Distance	6min 40s	6min 49s	8min 36s
Minimum Energy	18min 36s	15min 4s	13min 5s

Table 2: Time required for solving the different problems using the sequential approach, as well as 4 and 8 parallel workers.

the algorithm with 4 and 8 parallel workers. In theory, increasing the number of workers, should not lead to slower running times. In practise however, there is an overhead associated with each additional worker. This is reflected in the results for the minimum time and minimum distance objectives, where the sequential approach outperforms multiple workers. For the minimum energy approach however, we see that increasing the number of workers improves the solution time. This is due to the poor choice of heuristic function, resulting in having to search a larger part of the search space, and hence the ability to evaluate multiple sequences simultaneously, outweighs the overhead of having multiple workers. It should be noted that the timing result in Table 2, will vary greatly with implementation and hardware, and a more optimized implementation is likely to significantly improve the running time.

4 Conclusion

In this paper, we have proposed a method for planning and optimizing trajectories in an environment with static polygonal obstacles, and where the trajectories must be feasible with respect to model dynamics. Under some mild assumptions, we show that the method is able to plan globally optimal trajectories, even when faced with highly non-convex obstacles. The proposed method does however have some drawbacks. The main drawback being computational requirements, which is due to each iteration of the search phase requiring the solution of a numerical optimization problem. As well as the number of decision variables for the optimization problems increasing linearly with the number of triangles the trajectory passes through. Another important limitation of the proposed method is that the dynamics of the system is approximated by a single polynomial within each triangle, this can cause problems for large triangles and complex dynamical models, where the polynomial is not sufficiently rich to accurately capture the dynamics. Despite these limitations, the proposed method shows great promise based on simulation results. Offering great flexibility both in terms of

Publications

environment complexity, model complexity, as well as optimization objective.

For future work, one of the main concerns would be to improve the computational efficiency. Some potential methods for doing so, include fixing the trajectory after a certain number of triangles in order to reduce the number of decision variables at later stages, or developing better heuristics to reduce or limit the search space. Work can also be done on how to best select a numerical integration scheme to better balance accuracy, flexibility, and computational efficiency. Similarly, methods for further decomposing the triangulation may also be used to improve accuracy, especially in large triangles, or when performing complex maneuvers. It may also be interesting to add additional environmental disturbances to the problems. This would be especially useful in the case of vessel motion planning, where wind and current may greatly impact the performance.

References

- [1] Artur Wolek and Craig A Woolsey. “Model-based path planning”. In: *Sensing and Control for Autonomous Vehicles*. Springer, 2017, pp. 183–206.
- [2] Glenn Bitar, Morten Breivik, and Anastasios M Lekkas. “Energy-Optimized Path Planning for Autonomous Ferries”. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 389–394.
- [3] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [4] Marcelo Kallmann. “Path planning in triangulations”. In: *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*. 2005, pp. 49–54.
- [5] Peter E Hart, Nils J Nilsson, and Bertram 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.
- [6] Mauro Candeloro, Anastasios M Lekkas, and Asgeir J Sørensen. “A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels”. In: *Control Engineering Practice* 61 (2017), pp. 41–54.
- [7] Jerome Barraquand, Bruno Langlois, and J-C Latombe. “Numerical potential field techniques for robot path planning”. In: *IEEE transactions on systems, man, and cybernetics* 22.2 (1992), pp. 224–241.
- [8] Anastasios M Lekkas et al. “Continuous-Curvature Path Generation Using Fermat’s Spiral”. In: *Modeling, Identification and Control* 34.4 (2013), p. 183.
- [9] Paul Jacobs and John Canny. “Planning smooth paths for mobile robots”. In: *Nonholonomic Motion Planning*. Springer, 1993, pp. 271–342.
- [10] Sara Fleury et al. “Primitives for smoothing mobile robot trajectories”. In: *IEEE transactions on robotics and automation* 11.3 (1995), pp. 441–448.

- [11] Kevin Judd and Timothy McLain. “Spline based path planning for unmanned air vehicles”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2001, p. 4238.
- [12] Jia Pan et al. “Collision-free and curvature-continuous path smoothing in cluttered environments”. In: *Robotics: Science and Systems VII* 17 (2012), p. 233.
- [13] Anastasios M Lekkas and Thor I Fossen. “Integral LOS path following for curved paths based on a monotone cubic Hermite spline parametrization”. In: *IEEE Transactions on Control Systems Technology* 22.6 (2014), pp. 2287–2301.
- [14] Carlo L Bottasso, Domenico Leonello, and Barbara Savini. “Path planning for autonomous vehicles by trajectory smoothing using motion primitives”. In: *IEEE Transactions on Control Systems Technology* 16.6 (2008), pp. 1152–1168.
- [15] Lydia E Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.
- [16] Steven M LaValle. “Rapidly-exploring random trees: A new tool for path planning”. In: (1998).
- [17] Steven M LaValle and James J Kuffner Jr. “Randomized kinodynamic planning”. In: *The international journal of robotics research* 20.5 (2001), pp. 378–400.
- [18] Sertac Karaman and Emilio Frazzoli. “Incremental sampling-based algorithms for optimal motion planning”. In: *Robotics Science and Systems VI* 104.2 (2010).
- [19] Stefano Carpin and Gianluigi Pillonetto. “Motion planning using adaptive random walks”. In: *IEEE Transactions on Robotics* 21.1 (2005), pp. 129–136.
- [20] Stefano Carpin and Gianluigi Pillonetto. “Merging the adaptive random walks planner with the randomized potential field planner”. In: *Proceedings of the Fifth International Workshop on Robot Motion and Control, 2005. RoMoCo’05*. IEEE. 2005, pp. 151–156.
- [21] Lester 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), pp. 497–516.
- [22] James Reeds and Lawrence Shepp. “Optimal paths for a car that goes both forwards and backwards”. In: *Pacific journal of mathematics* 145.2 (1990), pp. 367–393.
- [23] Richard Bellman. “Dynamic programming”. In: *Science* 153.3731 (1966), pp. 34–37.
- [24] Russell Eberhart and James Kennedy. “Particle swarm optimization”. In: *Proceedings of the IEEE international conference on neural networks*. Vol. 4. Cite-seer. 1995, pp. 1942–1948.
- [25] James Kennedy and Russell C Eberhart. “A discrete binary version of the particle swarm algorithm”. In: *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*. Vol. 5. IEEE. 1997, pp. 4104–4108.


Publications

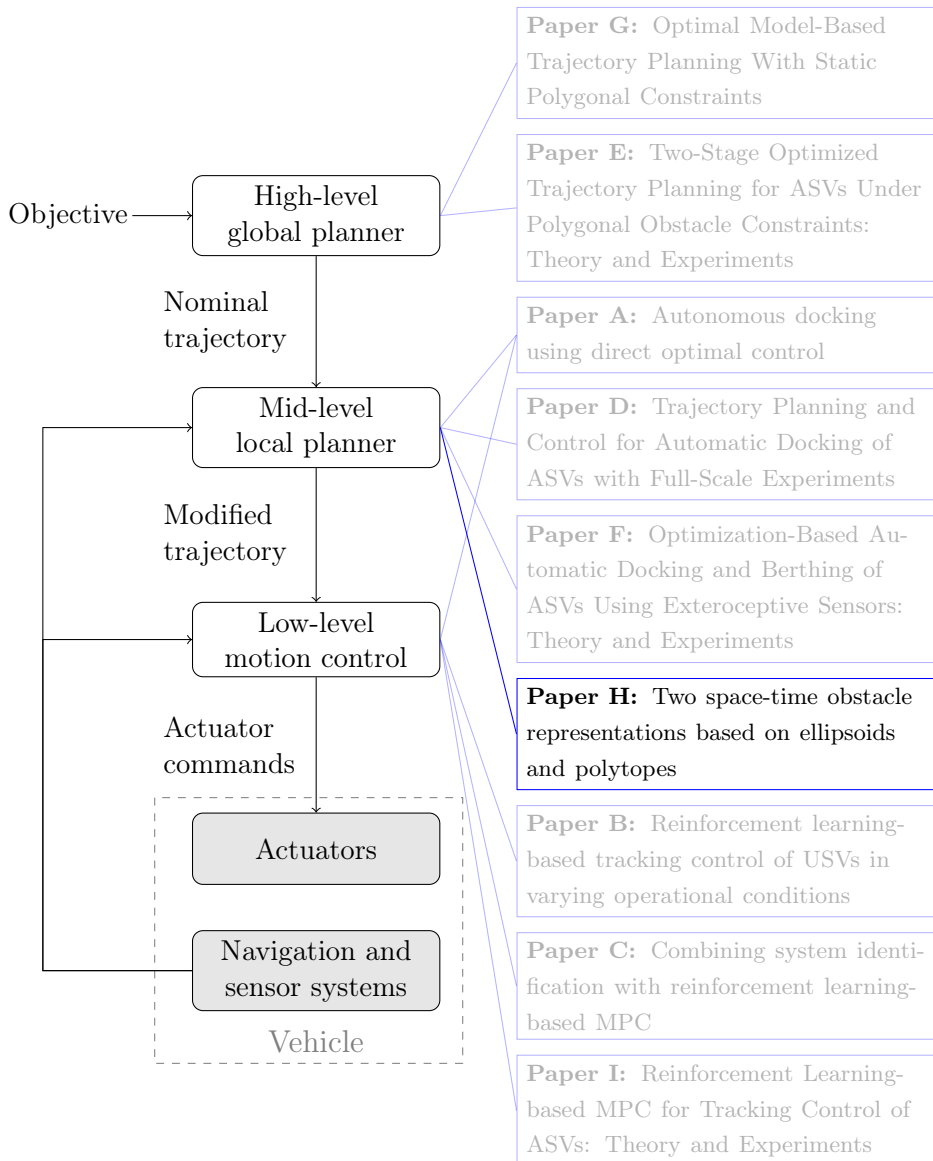
- [26] Hans Georg Bock and Karl-Josef Plitt. “A multiple shooting algorithm for direct solution of optimal control problems”. In: *IFAC Proceedings Volumes* 17.2 (1984), pp. 1603–1608.
- [27] Charles R Hargraves and Stephen W Paris. “Direct trajectory optimization using nonlinear programming and collocation”. In: *Journal of guidance, control, and dynamics* 10.4 (1987), pp. 338–342.
- [28] Fariba Fahroo and I Michael Ross. “Direct trajectory optimization by a Chebyshev pseudospectral method”. In: *Journal of Guidance, Control, and Dynamics* 25.1 (2002), pp. 160–166.
- [29] Oskar Ljungqvist et al. “Lattice-based motion planning for a general 2-trailer system”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 819–824.
- [30] Kristoffer Bergman et al. “An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 5283–5290.
- [31] Glenn Bitar et al. “Warm-started optimized trajectory planning for ASVs”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 308–314.
- [32] Lars Blackmore, Masahiro Ono, and Brian C Williams. “Chance-constrained optimal path planning with obstacles”. In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094.
- [33] Tobias Schoels et al. “CIAO*: MPC-based Safe Motion Planning in Predictable Dynamic Environments”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 6555–6562.
- [34] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Autonomous docking using direct optimal control”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 97–102.
- [35] Hongyang Yan et al. “Path planning based on constrained delaunay triangulation”. In: *2008 7th World Congress on Intelligent Control and Automation*. IEEE. 2008, pp. 5168–5173.
- [36] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. “Spline-based motion planning for autonomous guided vehicles in a dynamic environment”. In: *IEEE Transactions on Control Systems Technology* 26.6 (2017), pp. 2182–2189.
- [37] L Paul Chew. “Constrained delaunay triangulations”. In: *Algorithmica* 4.1-4 (1989), pp. 97–108.
- [38] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer-Verlag, 1985.
- [39] TH Tsang, DM Himmelblau, and Thomas F Edgar. “Optimal control via collocation and non-linear programming”. In: *International Journal of Control* 21.5 (1975), pp. 763–768.

- [40] Dino Živojević and Jasmin Velagić. “Path Planning for Mobile Robot using Dubins-curve based RRT Algorithm with Differential Constraints”. In: *2019 International Symposium ELMAR*. IEEE. 2019, pp. 139–142.
- [41] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.
- [42] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

H Two space-time obstacle representations based on ellipsoids and polytopes

Postprint of [25] **Andreas B Martinsen** and Anastasios M Lekkas. “Two space-time obstacle representations based on ellipsoids and polytopes”. In: *IEEE Access* 9 (2021). DOI: 10.1109/ACCESS.2021.3103323

©2021 Andreas B Martinsen and Anastasios M Lekkas. Reprinted and formatted to fit the thesis under the terms of the Creative Commons Attribution License 



Two space-time obstacle representations based on ellipsoids and polytopes

Andreas B. Martinsen¹ and Anastasios M. Lekkas^{1,2}

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

²Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway

Abstract: When operating autonomous surface vessels in uncertain environments with dynamic obstacles, planning safe trajectories and evaluating collision risk is key to navigating safely. In order to perform these tasks, it is important to have a computationally efficient and adaptable obstacle representation to allow for quick and robust predictions of the obstacle trajectory. This paper presents a novel space-time obstacle representation, which is able to predict the reachable set for a dynamic obstacle under uncertainty. This is done by projecting the area occupied by the obstacle forward in time, using a set of velocities representing the possible maneuvers that the obstacle may take. Under some mild assumptions, we show how the space-time obstacle can be implemented in a computationally efficient way, using both convex polytopes and ellipsoids. Additionally, we show how the space-time obstacle representation can be used for risk assessment, collision avoidance and trajectory planning for autonomous surface vessels.

Keywords: Autonomous surface vehicles, Collision avoidance, Marine vehicles, Motion planning, Optimal control, Trajectory optimization, Risk assessment

1 Introduction

With increasing interest in autonomy solutions in the maritime industry, it becomes increasingly important to develop robust and efficient methods for risk assessment and collision avoidance (COLAV). This is especially true for dynamic obstacles, for which accurate trajectory predictions is complicated by numerous uncertainties. A major component of developing robust and efficient methods, is the underlying obstacle representation. While the literature is for the most part concerned with COLAV and risk assessment methods, where the obstacle representation is chosen to fit the algorithm. The goal of this article is to create awareness around the representation, and showing some of the benefits of building a COLAV and risk assessment method around an obstacle representation, instead of the obstacle representation being built

around the method. In order for the obstacle representation to be practical, it needs to be able to capture the shape and movement of the obstacle in a way that is computationally efficient, allowing for real-time risk assessment, planning and decision making. Additionally the obstacle representation must be robust, allowing for both measurement uncertainties, as well as uncertainties in the obstacle behavior.

One of the first obstacle representations to be used for assessing collision risk is the closest point of approach (CPA) [1], which computes the distance and point in time when two vessels are at their closest, given that the vessels have a known constant velocity. CPA was initially developed to give human readable feedback to navigators on the risk associated with the speed and course of the vessel, but has more recently been incorporated into automated COLAV systems [2, 3]. Based on the same idea as CPA, the velocity obstacle (VO) representation [4, 5], computes the set of velocities which lead to a collision, i.e. giving a distance at the closest point of approach (dCPA) of zero, and a time of closest point of approach (tCPA) greater than zero. The VO approach has seen widespread use, as it allows for easily assessing if a given velocity vector is collision free. Additional extension to the VO representation allow for kinematic constraints, obstacle behaviour and uncertainty [6–8], with similar methods such as dynamic window (DW) methods, allowing for dynamic constraints [9]. While these obstacle representations are fairly computationally efficient, they come with some major drawbacks, mainly that they considering only a single maneuver, such as a constant velocity or turning rate. This makes the methods suitable for short term collision avoidance, but is only of limited use for long term planning and risk assessment of multiple complex maneuvers. In order to allow for longer term planning, a common approach is to use set based obstacle representation, where the set of points making up the obstacles is computed and projected forward in time. For surface vessels this is typically done using either circles and ellipsoids [10–14], or polytopes [15, 16]. While these methods allow for longer term planning, they usually assume that the obstacle velocity is exactly known over some prediction horizon, meaning that these methods are limited in terms of the robustness under both measurement and behaviour uncertainty of the obstacle. In order to account for obstacle uncertainty, the most accurate methods incorporate the obstacle uncertainty, this can be done by computing the reachable sets for the obstacle [17, 18], or using probabilistic methods [6, 19–22] for predicting the behaviour of the obstacles. These methods allow for accurate obstacle predictions, but are often less flexible and more computationally expensive than what is ideal for a general purpose obstacle representation.

In order to address some of the drawbacks of existing methods, we propose a novel space-time obstacle representation, which is able to predict the set of states which a dynamic obstacle may occupy, given uncertainty in both measurements, as well as the future behaviour of the obstacle. The proposed space-time representation is a set based representation, as the area occupied by the vessel is projected forward in time. Contrary to other set based approaches however, the proposed space-time representation uses a set of velocities representing the possible maneuvers that the obstacle may take. This ensures robustness to both measurement and behavioural uncertainty similarly to probabilistic methods. In addition to developing a theoretical

framework for the proposed space-time representation, we also show how a space-time obstacle can be efficiently implemented both as convex polytopes and ellipsoids, in addition to showing how the space-time obstacle may be used for both risk assessment and trajectory planning. The main contributions of this paper are as follows:

- The development of a novel space-time obstacle representation for predicting the possible future trajectories of an obstacle under uncertainty.
- Implementation of the space-time obstacle using both a convex polytope representation, and an ellipsoid representation.
- We provide examples of how the space-time representation can be used both for risk assessment and trajectory planning for surface vessels.

The rest of the paper is structured as follows: Section 2 introduces the space-time obstacle, and shows how it can be implemented using both polytopes and ellipsoids. Section 3 shows how the space-time obstacle can be used for both trajectory planning and risk assessment, and Section 4 concludes the paper.

2 Space-time obstacle representation

When performing obstacle avoidance, we need a way of representing the obstacle. This can be done by representing the obstacle as the set \mathcal{O} of all space-time coordinates (\mathbf{x}, t) that the obstacle can occupy. For static obstacles, the obstacle representation remains the same at for all time, however this is no longer the case when faced with dynamic obstacles. When planning safe trajectories, it is important to be able to account for the movement of the obstacle in order to safely avoid it. Given an obstacle \mathcal{O}_0 at time $t = 0$, with a velocity vector \mathbf{v} we can predict what the obstacle will look like in the future as:

$$\mathcal{O}_t = \mathcal{O}_0 + (\mathbf{v} \cdot t) \quad (1)$$

This will work in the case where the obstacle is deterministic, and we know its initial area \mathcal{O}_0 and future velocity \mathbf{v} . However in most real world applications, this is not the case, as we may only have noisy estimates of position and velocity, and the obstacle may speed up or slow down. In order to account for this, we propose using a set of feasible velocities \mathcal{V} , which represents the uncertainty about the measurement and behaviour of the obstacle. This gives the following obstacle prediction:

$$\mathcal{O}_t = \mathcal{O}_0 \oplus (\mathcal{V} \cdot t) \quad (2)$$

where \oplus denotes the *Minkowski sum*, i.e. the point-wise sum between two sets. This can be further generalized into space-time coordinates (\mathbf{x}, t) , as a robust space-time obstacle representation:

$$\mathcal{O} = \{(\mathbf{x}, t) \mid \mathbf{x} \in \mathcal{O}_t\} \quad (3)$$

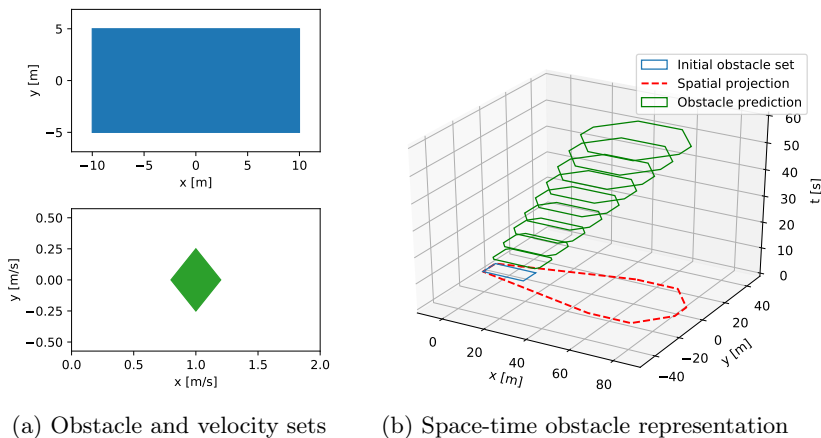


Figure 1: Given the obstacle and velocity sets in (a), we can compute the resulting space-time obstacle representation in (b). Where the blue lines represent the initial obstacle, green lines represent the obstacle at different times, and the red dashed line represents the spatial projection of the space-time obstacle.

Using this robust space-time obstacle representation, where the true obstacle lies within the initial obstacle set \mathcal{O}_0 , and the obstacle velocity lies within the velocity set \mathcal{V} , then space-time coordinates that do not fall within the set \mathcal{O} are guaranteed to be collision free.

2.1 Polytope space-time obstacle representation

One way of efficiently computing the robust space-time obstacle representation, is to use convex polytopes. Given a set of points $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n\}$ we can compute a convex polygon containing all the points as the convex hull of the points.

$$\text{Conv}(\mathcal{S}) \tag{4}$$

Using this we can define the obstacle and velocity set in terms of the convex hull of a finite set of points.

$$\mathcal{O}_0 = \text{Conv}(\{\mathbf{o}_1, \dots, \mathbf{o}_n\}), \quad \mathcal{V} = \text{Conv}(\{\mathbf{v}_1, \dots, \mathbf{v}_n\}) \tag{5}$$

A useful property of the convex hull is that the convex hull and Minkowski sum are commutative operations. This means that for the two sets \mathcal{O}_o and \mathcal{V} , the following equality holds:

$$\begin{aligned} \mathcal{O}_t &= \mathcal{O}_0 \oplus (\mathcal{V} \cdot t) \\ &= \text{Conv}(\{\mathbf{o}_1, \dots, \mathbf{o}_n\}) \oplus \text{Conv}(\{\mathbf{v}_1 \cdot t, \dots, \mathbf{v}_n \cdot t\}) \\ &= \text{Conv}(\{\mathbf{o}_1, \dots, \mathbf{o}_n\} \oplus \{\mathbf{v}_1 \cdot t, \dots, \mathbf{v}_n \cdot t\}), \end{aligned} \tag{6}$$

H. Two space-time obstacle representations based on ellipsoids ...

hence the resulting obstacle prediction remains convex [23]. In the case of a three dimensional space time, we can formulate the robust space-time obstacle representation in terms of the corresponding half-space representation:

$$\mathcal{O} = \left\{ (\mathbf{x}, t) \mid \mathbf{A}_o \begin{bmatrix} \mathbf{x} \\ t \end{bmatrix} \leq \mathbf{b}_o \right\}. \quad (7)$$

Given the extreme points (vertexes) of \mathcal{O}_t , in a counter clockwise order at arbitrarily chosen times t_1 and t_2 where $0 \leq t_1 < t_2$, as:

$$\begin{aligned} \text{Vertex}(\mathcal{O}_{t_1}) &= \{\mathbf{o}_{t_1,1}, \mathbf{o}_{t_1,2} \dots \mathbf{o}_{t_1,n}\} \\ \text{Vertex}(\mathcal{O}_{t_2}) &= \{\mathbf{o}_{t_2,1}, \mathbf{o}_{t_2,2} \dots \mathbf{o}_{t_2,n}\}, \end{aligned} \quad (8)$$

the half space representation is given by the following:

$$\begin{aligned} \mathbf{A}_{o,i} &= \begin{bmatrix} \mathbf{o}_{t_2,i+1} - \mathbf{o}_{t_2,i} \\ 0 \end{bmatrix} \times \begin{bmatrix} \mathbf{o}_{t_1,i} - \mathbf{o}_{t_2,i} \\ t_1 - t_2 \end{bmatrix} \\ \mathbf{b}_{o,i} &= \mathbf{A}_{o,i} \begin{bmatrix} \mathbf{o}_{t_2,1} \\ t_2 \end{bmatrix}, \end{aligned} \quad (9)$$

where $\mathbf{A}_{o,i}$ and $\mathbf{b}_{o,i}$ are the i -th rows of \mathbf{A}_o and \mathbf{b}_o respectively. We should also note that the \times operator denotes the cross product of two vectors, and $\mathbf{o}_{t_1,n+1} = \mathbf{o}_{t_1,1}$. Using the above space-time obstacle representation, it is straightforward to check if a given space-time coordinate (\mathbf{x}, t) will result in a collision with the obstacle. This is done simply by algebraically evaluating the matrix inequality given in (7), which has a computational complexity which grows linearly with the number of vertices in the initial obstacle set \mathcal{O}_0 and the velocity set \mathcal{V} .

Using the proposed polytope space-time obstacle representation, with the obstacle and velocity set given as follows:

$$\mathcal{O}_0 = \text{Conv} \left(\begin{bmatrix} -10 \\ -5 \end{bmatrix}, \begin{bmatrix} -10 \\ 5 \end{bmatrix}, \begin{bmatrix} 10 \\ 5 \end{bmatrix}, \begin{bmatrix} 10 \\ -5 \end{bmatrix} \right) \quad (10)$$

$$\mathcal{V} = \text{Conv} \left(\begin{bmatrix} 0.8 \\ 0.00 \end{bmatrix}, \begin{bmatrix} 1.0 \\ -0.25 \end{bmatrix}, \begin{bmatrix} 1.2 \\ 0.00 \end{bmatrix}, \begin{bmatrix} 1.0 \\ 0.25 \end{bmatrix} \right), \quad (11)$$

we get the space-time obstacle representation given in Figure 1. From the figure we see that the obstacle representation continues to grow with time. This is caused by the uncertainty in the velocity being compounded over time.

2.2 Ellipsoid space-time obstacle representation

The robust space-time obstacle representation may also be efficiently computed using an ellipsoidal set representation. We use the basic definition of an ellipsoid:

$$E(\mathbf{p}, \mathbf{Q}) = \{\mathbf{x} \in \mathbb{R}^n \mid (\mathbf{x} - \mathbf{p})^\top \mathbf{Q}^{-1} (\mathbf{x} - \mathbf{p}) \leq 1\}, \quad (12)$$

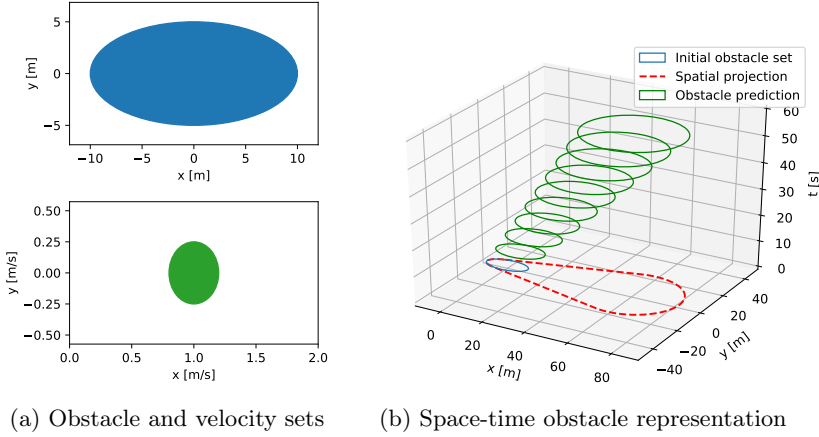


Figure 2: Given the obstacle and velocity sets in (a), we can compute the resulting space-time obstacle representation in (b). Where the blue lines represent the initial obstacle, green lines represent the obstacle at different times, and the red dashed line represents the spatial projection of the space-time obstacle.

where $\mathbf{p} \in \mathbb{R}^n$ is the ellipse center, and $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a positive definite shape matrix. In order to compute the robust space-time obstacle representation we must be able to compute the Minkowski sum $E(\mathbf{p}_1, \mathbf{Q}_1) \oplus E(\mathbf{p}_2, \mathbf{Q}_2)$ between two arbitrary ellipsoids. Unfortunately, the Minkowski sum of two ellipsoid is in general not an ellipsoid. However it is possible to formulate an ellipsoidal outer approximation:

$$E(\mathbf{p}_1, \mathbf{Q}_1) \oplus E(\mathbf{p}_2, \mathbf{Q}_2) \subset E(\mathbf{p}_1 + \mathbf{p}_2, (1 + c^{-1})\mathbf{Q}_1 + (1 + c)\mathbf{Q}_2) \quad \forall c > 0 \quad (13)$$

Moreover, the minimizer of the trace and hence the sum of the eigenvalues of the resulting symmetric shape matrix is analytically given as:

$$c = \sqrt{\frac{\text{Tr}(\mathbf{Q}_1)}{\text{Tr}(\mathbf{Q}_2)}} \quad (14)$$

Given an initial obstacle and velocity estimate as the ellipsoidal sets:

$$\mathcal{O}_0 = E(\mathbf{p}_o, \mathbf{Q}_o), \quad \mathcal{V} = E(\mathbf{p}_v, \mathbf{Q}_v), \quad (15)$$

where the scaled velocity ellipsoid $\mathcal{V} \cdot t$ is given as:

$$\mathcal{V} \cdot t = E(\mathbf{p}_v \cdot t, \mathbf{Q}_v \cdot t^2), \quad (16)$$

we can use (13) to formulate an outer approximation of the robust obstacle representation (2) as follows:

$$\mathcal{O}_t \subset E(\mathbf{p}_t, \mathbf{Q}_t) \quad (17)$$

H. Two space-time obstacle representations based on ellipsoids ...

where

$$\begin{aligned}
 \mathbf{p}_t &= \mathbf{p}_o + \mathbf{p}_v \cdot t \\
 \mathbf{Q}_t &= \left(1 + \frac{t}{d}\right) \mathbf{Q}_o + (t^2 + d \cdot t) \mathbf{Q}_v \\
 d &= \sqrt{\frac{\text{Tr}(\mathbf{Q}_o)}{\text{Tr}(\mathbf{Q}_v)}}
 \end{aligned} \tag{18}$$

Using this ellipsoidal outer approximation, we define the ellipsoidal robust space-time obstacle representation as follows:

$$\mathcal{O} = \{(\mathbf{x}, t) \mid (\mathbf{x} - \mathbf{p}_t)^\top \mathbf{Q}_t^{-1} (\mathbf{x} - \mathbf{p}_t) \leq 1\}. \tag{19}$$

We can note that using the above space-time obstacle representation, it is straightforward to check if a given space-time coordinate (\mathbf{x}, t) will result in a collision with the obstacle, as it simply involves algebraically evaluating the inequality in (19), which has a constant computation time.

Using the proposed ellipsoid space-time obstacle representation, with the obstacle and velocity set given as follows:

$$\mathcal{O}_0 = E \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 10^2 & 0 \\ 0 & 5^2 \end{bmatrix} \right) \tag{20}$$

$$\mathcal{V} = E \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.2^2 & 0 \\ 0 & 0.25^2 \end{bmatrix} \right), \tag{21}$$

we get the space-time obstacle representation given in Figure 2. Similarly to the polytope representation, we see that the obstacle representation continues to grow with time in order to account for the increasing uncertainty.

2.3 Space-time obstacle risk assessment

The robust space-time obstacle representations presented in the previous sections, show how we can compute a space-time volume which the obstacle may occupy, given an initial obstacle area and a set of obstacle velocities. For some applications, this may be overly restrictive, as the true obstacle will take only one velocity within the set of possible velocities. In this case it may be more useful to reason about the risk associated with the space-time obstacle. One way of reasoning about the risk is to consider the size of the velocity set \mathcal{V} . Writing the velocity uncertainty as:

$$\mathcal{V} = \mathbf{v} + \alpha \cdot \mathcal{V}_0 \tag{22}$$

where \mathbf{v} is the geometric center of the velocity set \mathcal{V} , \mathcal{V}_0 is the velocity uncertainty with geometric center at the origin, and $\alpha \in [0, 1]$ is the velocity uncertainty scaling. Using this we can define the scaled obstacle prediction as:

$$\mathcal{O}_{t,\alpha} = \mathcal{O}_0 \oplus (\mathbf{v} + \alpha \cdot \mathcal{V}_0) \cdot t, \tag{23}$$

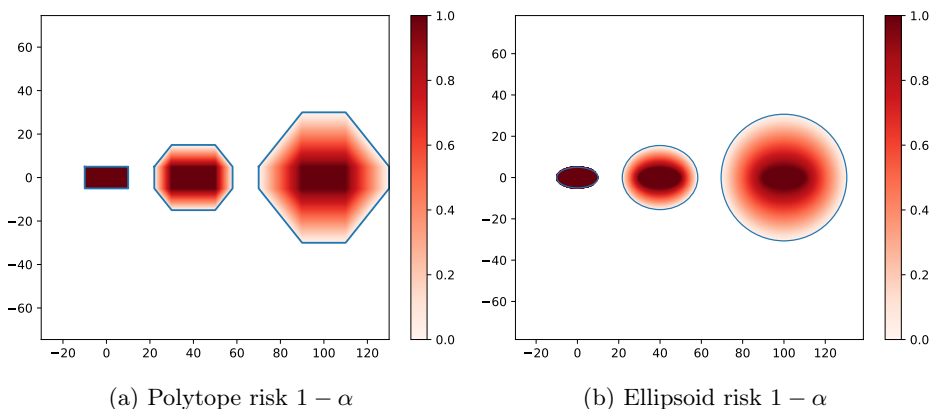


Figure 3: Risk for the polytope and ellipsoid space-time obstacle at 0, 40 and 100 seconds.

and the scaled space-time obstacle as:

$$\mathcal{O}_\alpha = \{(\mathbf{x}, t) \mid \mathbf{x} \in \mathcal{O}_{t,\alpha}\}. \quad (24)$$

Using the above formulation, we can represent the risk as $1 - \alpha$ where α is chosen as the minimum scaling for which a space-time coordinate (\mathbf{x}, t) is within the space-time obstacle \mathcal{O}_α . This can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & \alpha \\ \text{s.t.} \quad & (\mathbf{x}, t) \in \mathcal{O}_\alpha \\ & \alpha \in [0, 1] \end{aligned} \quad (25)$$

Solving the above optimization problem is in general quite computationally expensive, however utilizing the properties of the polytope and ellipsoid representation, the constrained optimization problem above, has a closed form solution in the case of the polytope, and can be transformed to a unconstrained optimization problem for the ellipsoid. For the examples given in Figure 1 and 2, we get the risk seen in Figure 3. It should be noted that this measure of risk is not a measure of probability, but rather represents the degree of uncertainty in the set of velocities that the obstacle can take.

3 Application

In this section we will show how the proposed space-time obstacle representation can be used for risk assessment, collision avoidance (COLAV) and trajectory planning for autonomous surface vessels (ASVs).

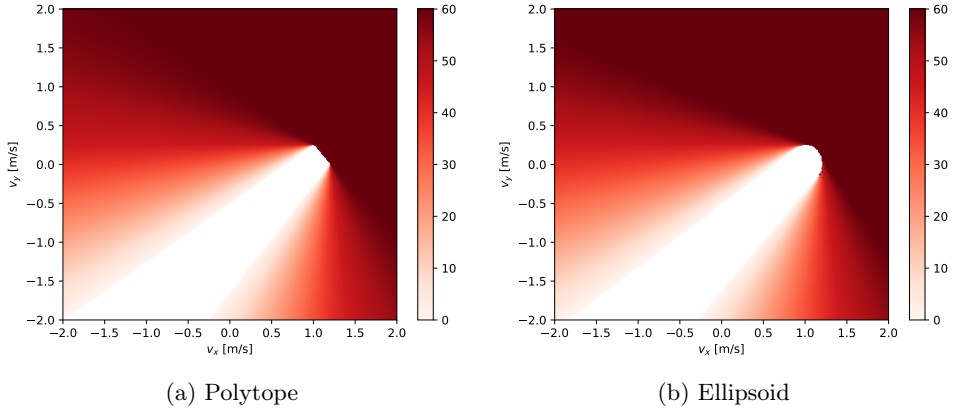


Figure 4: Distance to obstacle at closest point of approach (dCPA) shaded, and velocity obstacle (VO) in white.

3.1 VO and CPA conversion

Some of the most common COLAV methods used today rely on evaluating multiple candidate velocities using either closest point of approach (CPA), or velocity obstacles (VO). Given a candidate velocity \mathbf{v} and an initial position \mathbf{x}_0 , making up the straight line trajectory $\mathbf{x}_v(t) = \mathbf{x}_0 + \mathbf{v} \cdot t$, we can evaluate the CPA and VO by finding the time that minimizes the distance between the $\mathbf{x}_v(t)$ and a point (\mathbf{x}, t) in the space-time obstacle. This can be formulated as the following optimization problem:

$$\begin{aligned}
 & \min_{\mathbf{x}, t} \|\mathbf{x}_v(t) - \mathbf{x}\|^2 \\
 & \text{s.t. } (\mathbf{x}, t) \in \mathcal{O}, \\
 & \quad t \geq 0.
 \end{aligned} \tag{26}$$

Given the solution (\mathbf{x}, t) of the optimization problem, t is the tCPA, $\mathbf{x}_v(t)$ is the closest point of approach, and $\|\mathbf{x}_v(t) - \mathbf{x}\|$ is the dCPA. If the distance $\|\mathbf{x}_v(t) - \mathbf{x}\|$ at the CPA is zero, then the candidate velocity lies within the VO, and is considered unsafe. We can note that the optimization problem in (26) is a quadratic programming problem for the polytope representation, and a nonlinear programming problem for the elliptical obstacle representation.

For the space-time obstacles in Figure 1 and 2, and an initial position $\mathbf{x}_0 = [50, 50]^\top$, we get the CPA and VO seen in Figure 4. Using this, automatic COLAV can be performed by choosing a velocity outside of the VO. Additionally, collisions can be avoided with a specified margin, by choosing a velocity with a large enough dCPA.

3.2 Simple path-time planner

In some circumstances, a preplanned path may be given, and the goal during transit, is to follow the path as closely as possible. When dynamic obstacles are introduced, the trajectory planning problem is reduced to safely regulating the velocity along the preplanned path in order to perform COLAV. This problem is particularly interesting in confined waters, where we often find predetermined shipping lanes for larger vessels, and set routes for regularly scheduled traffic such as ferries. One such path-time decomposition approach to COLAV, called path-velocity decomposition, was first introduced in [24], where a path-time obstacle representation was used together with graph search methods in order to plan collision free trajectories following predetermined paths. Since then the method has been used in a number of applications, including COLAV for small urban passenger ferries in confined waters [25]. In this section we will show how our proposed space-time obstacle representation, is a generalisation of path-time coordinates used in [24], and how the space-time obstacle representation can be used to easily introduce uncertainty and risk into the path-time decomposition approach.

3.2.1 Space-time to path-time projection

Given a space-time obstacle, the corresponding path-time obstacle is given as the projection of the space-time obstacle along the desired path. In the case of a polytopic space time obstacle on the form:

$$\mathcal{O} = \left\{ (\mathbf{x}, t) \mid \mathbf{A}_o \begin{bmatrix} \mathbf{x} \\ t \end{bmatrix} \leq \mathbf{b}_o \right\}, \quad (27)$$

and a straight line path on the form:

$$\mathbf{x}(s) = \mathbf{x}_0 + \mathbf{n} \cdot s, \quad (28)$$

where \mathbf{x}_0 is the initial position, \mathbf{n} , where $\|\mathbf{n}\| = 1$, is the path direction, and s is the distance along the path. We can compute the path-time obstacle \mathcal{O}_p as the projection of the space-time obstacle \mathcal{O} as follows:

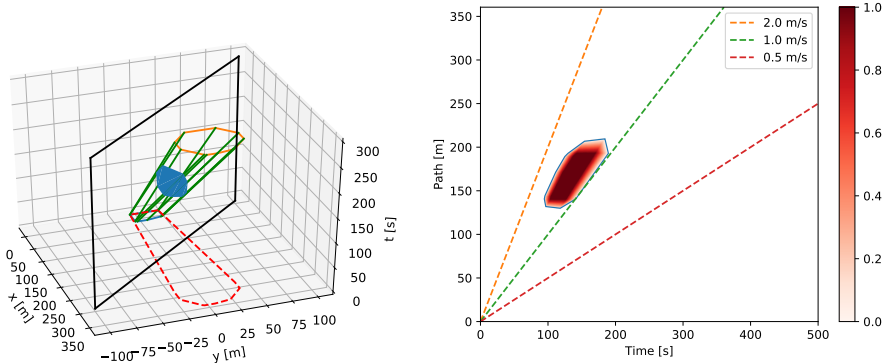
$$\mathcal{O}_p = \left\{ (s, t) \mid \mathbf{A}_o \begin{bmatrix} \mathbf{n} & \mathbf{0} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} \leq \mathbf{b}_o - \mathbf{A}_o \begin{bmatrix} \mathbf{x}_0 \\ t_0 \end{bmatrix} \right\}. \quad (29)$$

Given the space-time obstacle and path in Figure 5a, the path projection gives the path-time diagram in Figure 5b.

3.2.2 Example

Given a desired path, which is collision free with respect to static obstacles such as a land, we can combine the proposed space-time obstacle representation, and the

H. Two space-time obstacle representations based on ellipsoids ...



(a) Space-time obstacle projected onto the path in black results in the path-time obstacle in blue.

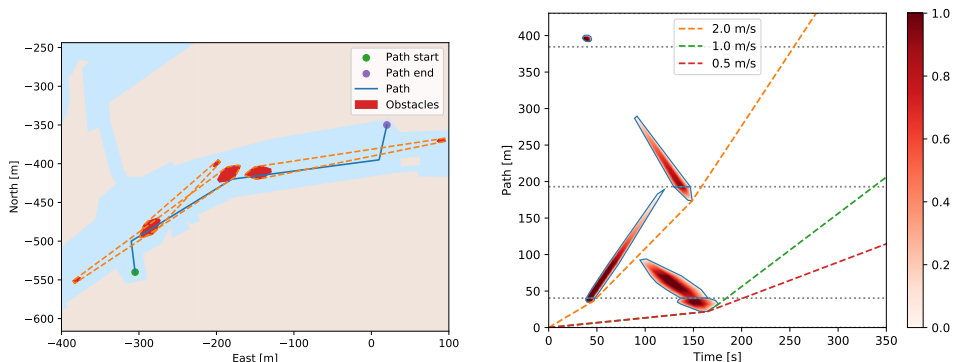
(b) Path-time obstacle, where constant velocity lines that do not intersect the obstacle are considered to be safe.

Figure 5: Example of how a polytopic space-time obstacle can be projected along a desired path (a), and the resulting path-time diagram (b). Note that straight lines in the path-time diagram correspond to constant velocities along the path.

path-time planning approach from [24], in order to plan an optimal velocity profile along the predetermined desired path, which ensures COLAV with respect to dynamic space-time obstacles. The resulting planning algorithm can be described as follows:

1. From obstacle tracks, compute the space-time representation of the vessel, predicting the obstacle movement and uncertainty into the future (Figure 6a).
2. Project the space-time obstacles onto the predetermined desired path, giving a path-time obstacle diagram as seen in Figure 6b.
3. Using a graph search algorithm such as Dijkstra [26] or A* [27] with a given cost function, plan a sequence of constant velocities (straight lines on the path-time diagram in Figure 6b), between vertices of the path-time obstacles, which do not intersect with the path-time obstacles. The planned sequence of velocities then give an optimal collision free trajectory.

In Figure 6, we show a simple scenario from the Trondheim harbour, where three dynamic obstacles moving with different velocity uncertainties. Using time as the optimization objective with three different maximum velocities, we get the time optimal trajectories seen in Figure 6b. This planning method can be further generalized to allow for switching between multiple paths, similar to [25], and can be modified to allow for trajectories with none-zero risk, in order to allow for planning more optimal trajectories at the cost of higher risk.



(a) Confined waters with dynamic obstacles at 0 seconds and predicted obstacles at 120 seconds. The goal is to find a safe velocity profile along the path.

(b) Path-time diagram, with collision free trajectories for different maximum velocities. The dotted lines mark transitions between different straight line path segments.

Figure 6: Example of how the path-time obstacles can be used for planning safe trajectories under obstacle uncertainty. Given the desired path and obstacles predicted motion in (a), we get the path-time diagram and planned velocity profiles in (b).

3.3 Simple space-time planner

For the path-time planning approach in the previous section, the trajectory is restricted to lie on a predetermined path. In many situations, this may be overly restrictive, and allowing for free movement is the better option. This includes following the International Regulations for Preventing Collisions at Sea (COLREGs), where clear maneuvers conveying the vessel intentions is required, and situations where evasive action is needed to avoid collision.

3.3.1 Dubins trajectory planner

Based on the path-time planner in the previous section, it is possible to generalize the approach into a space-time planner, where the goal is to plan a collision free trajectory that avoid intersecting the space-time obstacle. This can be easily done by using sampling based methods such as probabilistic roadmaps (PRM) [28] and rapidly-exploring random tree (RRT) [29]. For our implementation however, we utilize the geometry of the space-time obstacle itself, and plan a collision free trajectory by connecting trajectory segments along the edges of the space-time obstacle representation. In order to ensure that the path is feasible with respect to the maximum turning rate of the vessel, we use Dubins paths [30], which consist of straight line segments and circular arcs of a maximum curvature. Using a graph search algorithm such as Dijkstra or A* with a given cost function, an optimal path can be found by

connecting the edges of the space-time obstacles with Dubins paths.

3.3.2 Example

Given a desired trajectory in an environment with dynamic obstacles, we can use the space-time obstacle representation in order to plan an optimal trajectory, which is collision free, and dynamically feasible. The resulting planning algorithm can be described as follows:

1. From obstacle tracks, compute the space-time representation of the vessel, predicting the obstacle movement and uncertainty into the future.
2. Using a graph search algorithm such as Dijkstra or A* with a given cost function, plan a sequence of connected Dubins paths (constant curvature circle segments and straight lines, see Figure 7), between edges of the space-time obstacles, which do not intersect with the space-time obstacles. The planned connected Dubins path then gives an optimal collision free trajectory.

In order to implement the planner, we chose to use the objective of finding the path that minimizes the squared space-time error between a desired trajectory and the planned trajectory, defined as:

$$\int_0^1 (x - x_d)^2 + (y - y_d)^2 + q \cdot (t - t_d)^2 ds, \quad (30)$$

where the trajectory and desired trajectory are functions of the path variable $s \in [0, 1]$, and q is a weight factor for weighting the time versus position error. Using this cost function is useful, as it encourages the planned trajectory to follow the desired trajectory, making evasive maneuvers that keep the vessel as close to the desired trajectory as possible. In order to discretize the search space and make the planner computationally feasible, a finite number of vessel velocities were considered, and the space-time obstacle intersections, were computed based on the finite velocities. For the implementation, we designed the initial obstacle with a large forbidden region in front and to the right of the obstacle vessel, making COLREGs compliant trajectories optimal in terms of the planning objective given by the space-time error (30). Running the planner for overtaking, head on, stand on and give way scenarios, we got the results seen in Figure 8. From the results, we see that the planned trajectory initially follows the desired trajectory, before taking an evasive maneuver, ensuring COLAV, while adhering to the COLREGs. We can note that in the stand on scenario, the planned trajectory keeps the desired course and speed initially, as is expected in a stand on situation, however due to the uncertainty in the future obstacle position, maintaining the course and speed can in the worst case scenario lead to a collision, and the vessel must deviate from the desired path in order to ensure collision avoidance. For a physical implementation, the planner can be run iteratively, in order to replan the trajectory, as new information about the space-time obstacle becomes available, decreasing the obstacle uncertainty and hence improving the planned trajectories.

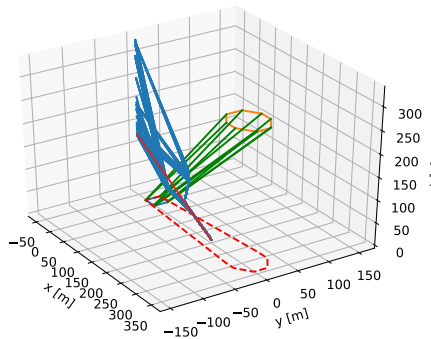


Figure 7: Possible collision free trajectories found during planning step.

3.4 Optimization based planner

A common approach for planning optimal trajectories, is to formulate the problem as an model based optimization problem, and solving it using numerical optimization. For these methods, ellipsoids are commonly used for representing obstacles [11–13], as they the ellipsoid representation is computationally cheap, and simple to implement into a numerical optimization problem. In this section we will show how the ellipsoid space-time obstacle can be used together with numerical optimization in order to plan optimal collision free trajectories.

3.4.1 Optimal control problem

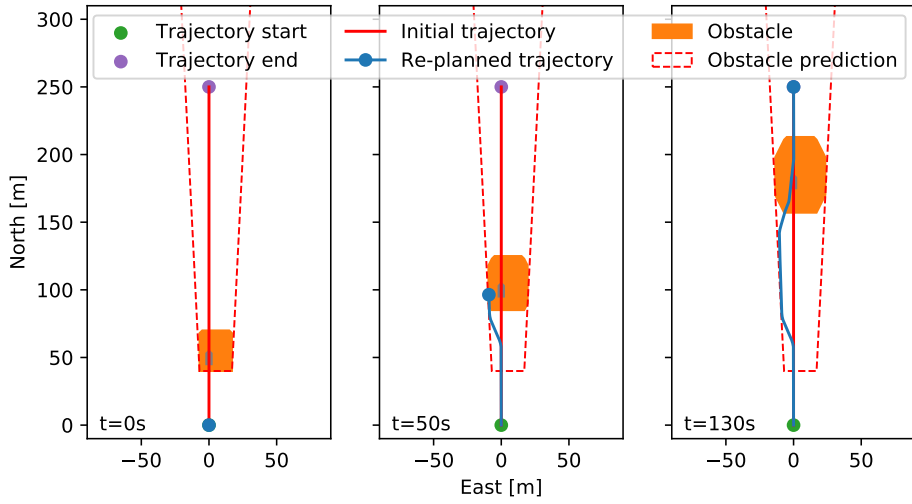
Given a cost function $J(\mathbf{x}, \mathbf{u})$ and a continuous time model of the vessel on the form:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \tag{31}$$

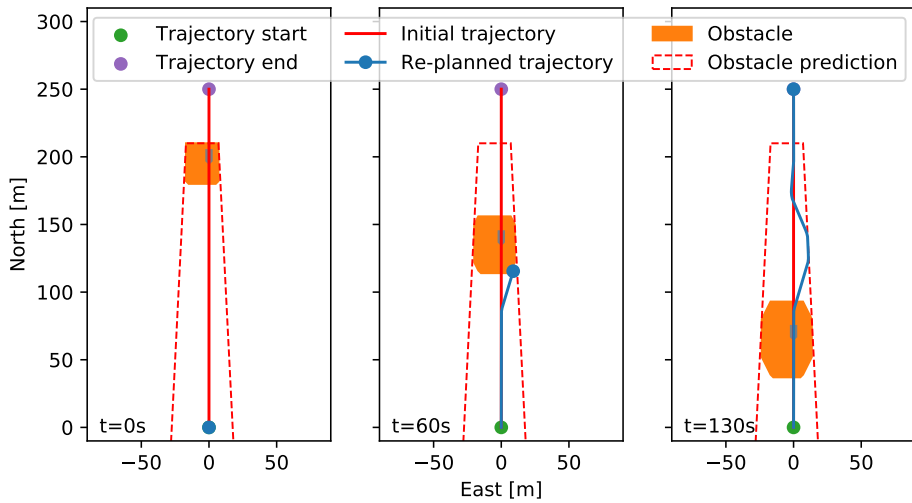
where \mathbf{x} are the vessel states, including position heading and velocity, and \mathbf{u} are the control surface. We can formulate an optimal control problem, which is to find the trajectory $\mathbf{x}(t)$ and controls $\mathbf{u}(t)$ which minimize the cost $J(\cdot)$ and are dynamically feasible with respect to the continuous time model over a time interval $[0, T]$. In order to account for the ellipsoidal obstacle, we can directly use the obstacle constraint from (19), giving the following optimal control problem:

$$\begin{aligned} \min_{\mathbf{x}(t), \mathbf{u}(t)} \quad & \int_0^T J(\mathbf{x}(t), \mathbf{u}(t)) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \\ & (\mathbf{x}(t) - \mathbf{p}_t)^\top \mathbf{Q}_t^{-1} (\mathbf{x}(t) - \mathbf{p}_t) \geq 1 \\ & \mathbf{x}(0) = \mathbf{x}_0. \end{aligned} \tag{32}$$

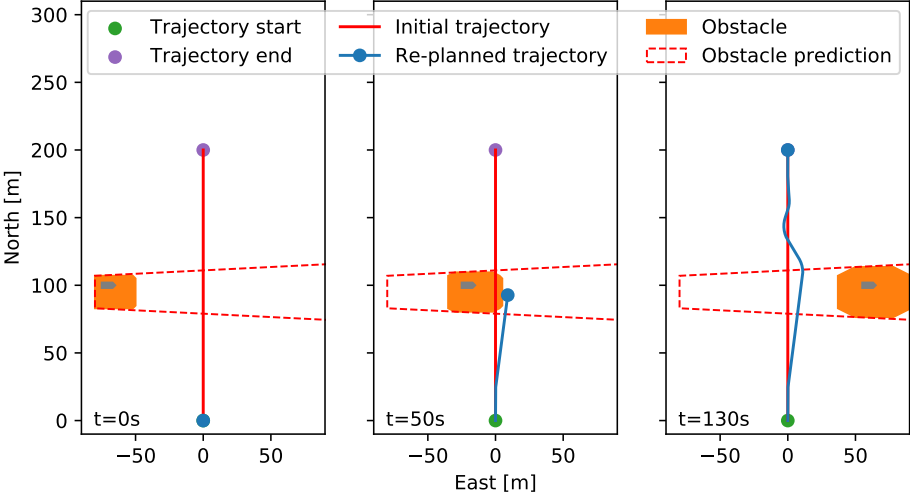
H. Two space-time obstacle representations based on ellipsoids ...



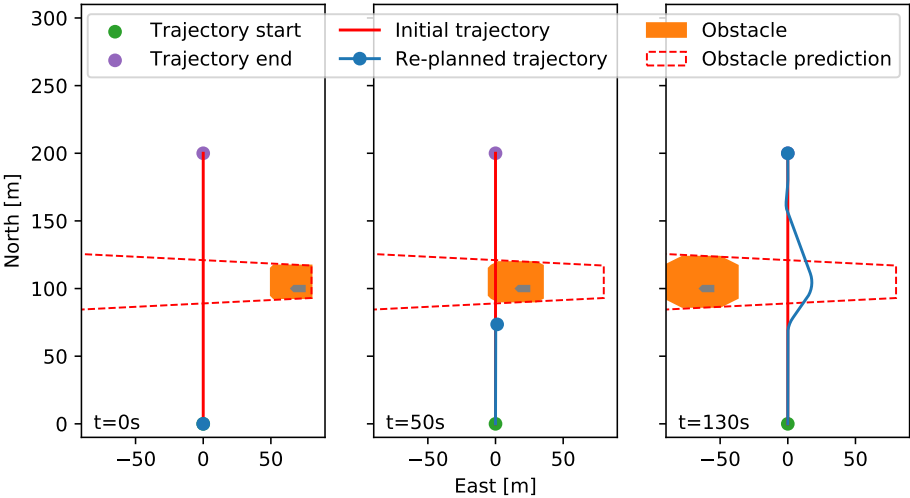
(a) Overtaking.



(b) Head on.



(c) Stand on.



(d) Give way.

Figure 8: Planned trajectories for different scenarios when using the Dublins trajectory planner. Note the obstacle asymmetry which ensures COLREGs compliance.

3.4.2 Example

In order to test the optimization based planner, we used the Cybership II model (see Appendix A) together with a quadratic cost function, and three dynamic obstacles. Using a collocation based transcription method to convert the optimization problem into an nonlinear programming problem, we got the optimal trajectory seen in Figure 9. From the results, we see that using the space-time obstacles, the planned trajectory is able to keep clear of the different obstacles in a way that accounts for the uncertainty obstacle uncertainty over time. This is a conservative strategy, but guarantees COLAV under obstacle uncertainty.

For more complex scenarios, it is possible to use other optimization objectives, such as time, energy, and distance [31]. It is also possible to include risk in the objective or constraints, allowing for a certain amount of risk to be taken when planning the optimal trajectory. This type of planner is also possible to implement as nonlinear model predictive control scheme by re-planning the trajectory at each time step, allowing for the planner to incorporate less conservative obstacle estimates as they become available over time.

4 Conclusion

We have presented a novel space-time obstacle representation, which can be used to predict the reachable set of dynamic obstacles under uncertainty. Additionally, we have shown how the proposed space-time representation can be efficiently computed using a convex polygon half-space representation, as well as an ellipsoid representation. Finally, we demonstrated how the space-time obstacle representation can be used for risk assessment, collision avoidance and planning for surface vessels in various environments with uncertain dynamic obstacles.

Based on the example applications we have demonstrated how the proposed space-time obstacle representation offers a flexible framework for representing and predicting obstacle trajectories in way that is computationally efficient. For future work it would be interesting to look at the possibility of extending the method to allow for time varying velocity uncertainties. Using the space-time obstacle representation together with other trajectory planning and collision avoidance methods would also be interesting, as well as further studying how to best represent the initial obstacle shapes in order to promote COLREGs compliance.

References

- [1] J. D. Luse. “Collision avoidance systems and the rules of the nautical road”. In: *NAVIGATION, Journal of the Institute of Navigation* 19.1 (1972), pp. 80–88.

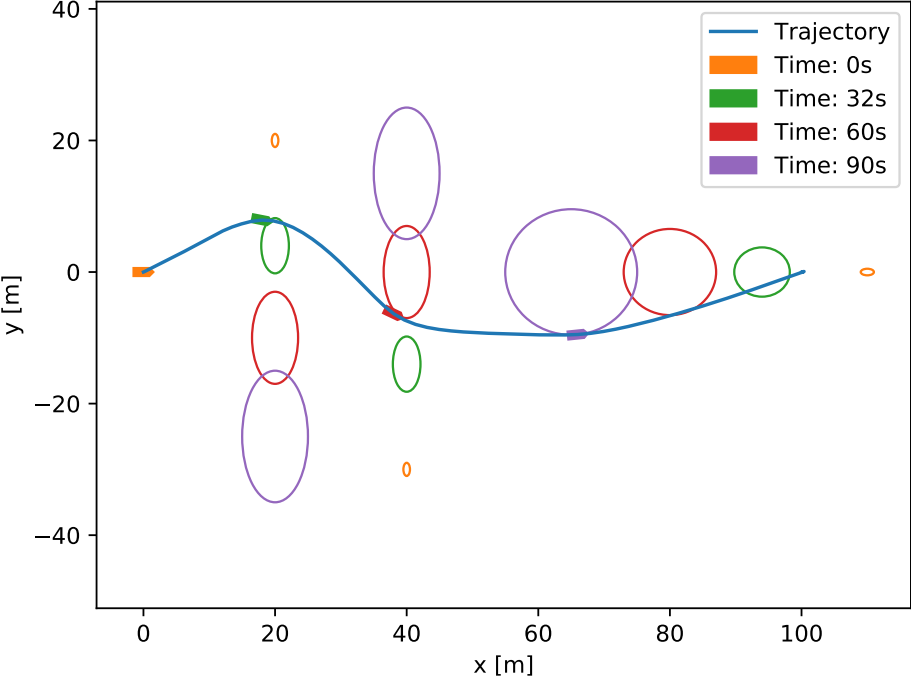


Figure 9: Optimal trajectory, given dynamic obstacles with uncertainty. The four timestamps show the vessel at the initial position, as well as the closest point of approach to each of the three obstacles.

- [2] Michael R Benjamin et al. “Navigation of unmanned marine vehicles in accordance with the rules of the road”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. IEEE, 2006, pp. 3581–3587.
- [3] D Kwame Minde Kufoalor, Edmund Førland Brekke, and Tor Arne Johansen. “Proactive collision avoidance for ASVs using a dynamic reciprocal velocity obstacles method”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2402–2409.
- [4] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using the relative velocity paradigm”. In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 560–565.
- [5] Animesh Chakravarthy and Debasish Ghose. “Obstacle avoidance in a dynamic environment: A collision cone approach”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 28.5 (1998), pp. 562–574.
- [6] Boris Kluge and Erwin Prassler. “Reflective navigation: Individual behaviors and group behaviors”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*. Vol. 4. IEEE, 2004, pp. 4172–4177.
- [7] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. “Generalized velocity obstacles”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 5573–5578.
- [8] Jamie Snape et al. “The hybrid reciprocal velocity obstacle”. In: *IEEE Transactions on Robotics* 27.4 (2011), pp. 696–706.
- [9] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33.
- [10] Y Yavin, C Frangos, and T Miloh. “Computation of feasible control trajectories for the navigation of a ship around an obstacle in the presence of a sea current”. In: *Mathematical and computer modelling* 21.3 (1995), pp. 99–117.
- [11] Bjørn-Olav H Eriksen and Morten Breivik. “MPC-based mid-level collision avoidance for ASVs using nonlinear programming”. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 766–772.
- [12] Glenn Bitar et al. “Warm-started optimized trajectory planning for ASVs”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 308–314.
- [13] Glenn Bitar et al. “Energy-optimized hybrid collision avoidance for ASVs”. In: *18th European Control Conference (ECC)*. IEEE, 2019, pp. 2522–2529.
- [14] Signe Moe, Kristin Y Pettersen, and Jan Tommy Gravdahl. “Set-based collision avoidance applications to robotic systems”. In: *Mechatronics* 69 (2020), p. 102399.
- [15] Michael Erdmann and Tomas Lozano-Perez. “On multiple moving objects”. In: *Algorithmica* 2.1 (1987), pp. 477–521.

Publications

- [16] Jacoby Larson, Michael Bruch, and John Ebken. “Autonomous navigation and obstacle avoidance for unmanned surface vehicles”. In: *Unmanned systems technology VIII*. Vol. 6230. SPIE, 2006, pp. 53–64.
- [17] Yonghoon Cho and Jinwhan Kim. “Collision probability assessment between surface ships considering maneuver intentions”. In: *OCEANS 2017-Aberdeen*. IEEE, 2017, pp. 1–5.
- [18] Yucong Lin and Srikanth Saripalli. “Collision avoidance for UAVs using reachable sets”. In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2015, pp. 226–235.
- [19] Maruthi R Akella and Kyle T Alfriend. “Probability of collision between space objects”. In: *Journal of Guidance, Control, and Dynamics* 23.5 (2000), pp. 769–772.
- [20] Jeonghong Park and Jinwhan Kim. “Predictive evaluation of ship collision risk using the concept of probability flow”. In: *IEEE Journal of Oceanic Engineering* 42.4 (2016), pp. 836–845.
- [21] Christopher T Shelton and John L Junkins. “Probability of collision between space objects including model uncertainty”. In: *Acta Astronautica* 155 (2019), pp. 462–471.
- [22] Trym Tengedal, Edmund F Brekke, and Tor A Johansen. “On collision risk assessment for autonomous ships using scenario-based-MPC”. In: *IFAC World Congress*. 2020.
- [23] Marc Van Kreveld et al. *Computational geometry algorithms and applications*. Springer, 2000.
- [24] Kamal Kant and Steven W Zucker. “Toward efficient trajectory planning: The path-velocity decomposition”. In: *The international journal of robotics research* 5.3 (1986), pp. 72–89.
- [25] Emil H. Thyri, Morten Breivik, and Anastasios M. Lekkas. “A Path-Velocity Decomposition Approach to Collision Avoidance for Autonomous Passenger Ferries in Confined Waters”. In: *IFAC-PapersOnLine* 53.2 (2020).
- [26] Edsger W Dijkstra et al. “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1 (1959), pp. 269–271.
- [27] Peter E Hart, Nils J Nilsson, and Bertram 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.
- [28] Lydia E Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.
- [29] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [30] Lester 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), pp. 497–516.

- [31] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Optimal model-based trajectory planning with static polygonal constraints”. In: *arXiv preprint arXiv:2010.14428* (2020).

A Cybership II model

Cybership II is a 1 : 70 scale model supply ship. The length of the ship is 1.3 m and the weight about 24 kg. The maximum actuated surge force is 2N, the maximum sway force is 1.5N and the maximum yaw moment is 1.5Nm. Given the pose $\boldsymbol{\eta} = [x, y, \psi]^\top$ in terms of the position (x, y) and heading ψ , velocity $\boldsymbol{\nu} = [u, v, r]^\top$ in surge, sway and yaw, the Cybership II can be modeled as follows:

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\boldsymbol{\nu}} \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \begin{bmatrix} \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \underbrace{-\mathbf{M}^{-1}(\mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{\tau})}_{f(\boldsymbol{x}, \boldsymbol{\nu})} \end{bmatrix},$$

where the inertia matrix, Coriolis matrix, damping matrix, and transformation matrix are given as:

$$\mathbf{M} = \begin{bmatrix} 25.8 & 0 & 0 \\ 0 & 33.8 & 1.0115 \\ 0 & 1.0115 & 2.76 \end{bmatrix}$$

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -33.8v - 1.0115r \\ 0 & 0 & 25.8u \\ 33.8v + 1.0115r & -25.8u & 0 \end{bmatrix}$$

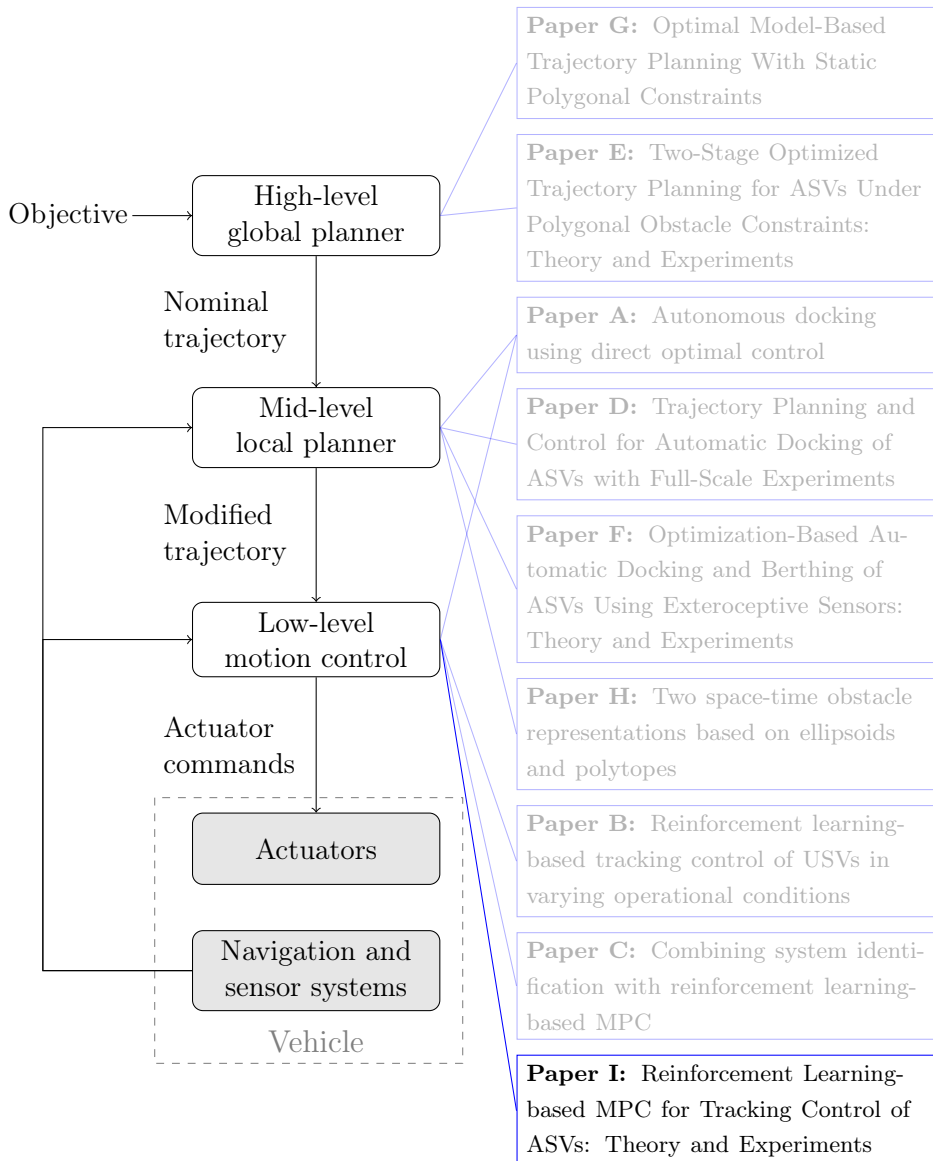
$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} 0.72 + 1.33|u| & 0 & 0 \\ 0 & 0.86 + 36.28|v| & -0.11 \\ 0 & -0.11 - 5.04|v| & 0.5 \end{bmatrix}$$

$$\mathbf{J}(\boldsymbol{\eta}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

I Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments

Preprint of [24] **Andreas B Martinsen**, Anastasios M Lekkas, and Sebastien Gros. “Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments”. In: *Review* (2021)

©2021 Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. Reprinted and formatted to fit the thesis with permission from Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros.



Reinforcement Learning-based NMPC for Tracking Control of ASVs: Theory and Experiments

Andreas B. Martinsen¹, Anastasios M. Lekkas^{1,2}, and Sébastien Gros¹

¹Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

²Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway

Abstract: We present a reinforcement learning-based (RL) model predictive control (MPC) method for trajectory tracking of surface vessels. The proposed method uses an MPC controller in order to perform both trajectory tracking and control allocation in real-time, while simultaneously learning to optimize the closed loop performance by using RL and system identification (SYSID) in order to tune the controller parameters. The efficiency of the method is evaluated by performing simulations on the unmanned surface vehicle (USV) *ReVolt*, as well as simulations and sea trials on the autonomous urban passengers ferry *milliAmpere*. Our results demonstrate that the proposed method is able to outperform other state of the art methods both in tracking performance, as well as energy efficiency.

Keywords: Dynamic positioning, Model predictive control, Optimal control, Reinforcement learning, Surface vessels, System identification, Trajectory tracking

1 Introduction

In recent years we have seen a growing interest in developing methods for automatic and autonomous marine operations, with applications such as surveying and mapping, surveillance, and transportation, being of interest both for commercial and government use. This has led to the need for robust high precision motion control systems, for performing operations such as docking and berthing [1], trajectory tracking [2], and collision avoidance [3].

Efficient control system design for marine vessels poses a number of challenges including the development of accurate mathematical models to describe complex vessel dynamics, the estimation of hydrodynamic coefficients that can vary significantly during operation, and the unpredictable nature of the marine environment. Consequently, extensive research has taken place in the past using ideas from almost all branches of control engineering. Linear, nonlinear, stochastic, optimal, intelligent, fuzzy, and

Publications

adaptive control, to name a few, approaches have been developed and tested via simulations and field trials [4–12]. In order to simplify the control design process, a common approach is to choose control strategies based on operating conditions. This has led to station keeping and dynamic positioning (DP) controllers for low speed maneuvers, and path following or trajectory tracking controllers for higher speeds and transit. However, using this approach has the drawback of requiring multiple controllers and/or models with different properties. In order to achieve performance diversity with conventional methods, the two most common approaches are to design multiple controllers and switch between them, or to use adaptive control methods. To this end, research effort has been dedicated to developing methods for learning the vessel model and model parameters by, for instance, using parameter estimation, system identification or adaptive control [13–19]. In most of these works, model-based approaches exploiting knowledge on hydrodynamics and the laws of motion were considered.

Reinforcement learning (RL) is a subfield of machine learning (ML) which tackles the problem of optimal sequential decision making under uncertainty. The roots of RL can be traced back to the Artificial Intelligence (AI) community in the 60's [20, 21]. Since then the field has come a long way, evolving in several directions to become one of the most active research areas at the intersection of machine learning, artificial intelligence, neural network and control theory. Contrary to other machine learning methods, RL does not rely on prerecorded datasets, but rather learns by following a trial and error process, from which it receives evaluative feedback. Similarly to optimal control, this feedback comes in the form of a hand-engineered reward or cost function, which assigns a reward, or penalty, to the actions that result in desired, or undesired, outcomes, respectively. Given the reward or cost function, the job of the RL algorithm is to find a state-action mapping, known as the policy (the analog of a controller, in control engineering terminology), that optimizes the reward or cost given the problem constraints and uncertainties. To sum up, RL algorithms learn through feedback from the reward function, using trial and error in order to learn a policy that optimizes the given reward. In recent years, RL has also been shown to be as useful as an adaptive control approach for marine vehicles [22–26].

Nonlinear model predictive control (MPC) is a popular approach for optimizing the closed loop performance of complex systems subject to constraints, which includes trajectory tracking and control of surface vessels [27–30]. MPC works by solving an optimal control problem (OCP) at each control interval in order to find an optimal policy. The optimal control problem seeks to minimize the sum of stage costs over a horizon, provided a model of the system and the current observed state. While MPC is a well-studied approach, and an extensive literature exists on analysing its properties [31, 32], the closed loop performance heavily relies on the accuracy of the underlying system model, which naturally presents challenges when significant unmodeled uncertainties are present.

In this work, we propose a model based RL approach for trajectory tracking of surface vessels. The approach builds on the work in [19], and extends it to use a nonlinear MPC

(NMPC) in order to perform the trajectory tracking in combination with control allocation. In order to optimize performance, the NMPC and model parameters are updated using RL [33] and system identification (SYSID) [34]. This allows the proposed method to compensate for model mismatch and environmental forces, with a focus on optimizing the closed loop performance of the trajectory tracking controller, rather than simply fitting the MPC model to the real system dynamics. In order to run the proposed control scheme in real-time, we implemented it using advanced-step NMPC (asNMPC). Additionally, simulations as well as sea trials were performed on the unmanned surface vehicle (USV) *ReVolt*, and the autonomous urban passengers ferry *milliAmpere*. The main contributions of this work are:

- A NMPC-based controller which combines trajectory tracking and control allocation for surface vessels (Section 2.2).
- The addition of RL and SYSID to the NMPC, in order to update the controller on-line. Making the controller able to compensate for model mismatch and environmental forces, and optimize the closed loop performance (Section 2.3).
- An implementation of the method using asNMPC, allowing for the controller to run in real-time (Section 3.1).
- Simulation study on two different vessel models, demonstrating how the approach outperforms our previous method from [19], and a traditional PID based controller (Section 4).
- Experimental results on an autonomous urban passenger ferry, demonstrating that NMPC based tracking control for surface vessels is real-time feasible, and is able to outperform a traditional PID based controller (Section 4).

The rest of the article is structured as follows. In Section 2 we show how reinforcement learning-based NMPC can be used for trajectory tracking for surface vessels. In Section 3 we outline the implementation of the proposed control scheme. Section 4 discuss the simulation and experimental results, while Section 5 concludes the paper.

2 Reinforcement learning-based trajectory tracking NMPC

In this section, we will outline the proposed control scheme, as well as providing background on modeling of surface vessels, reinforcement learning-based NMPC and system identification.

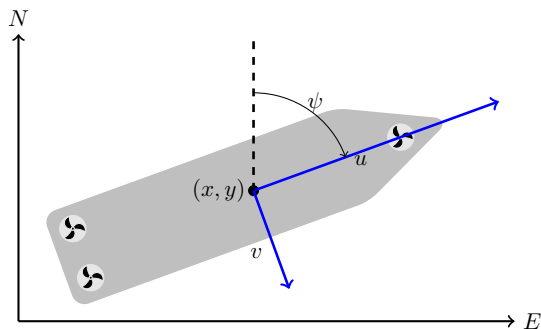


Figure 1: 3-DOF vessel centered at (x, y) in the North-East-Down (NED) reference frame, with surge velocity u , sway velocity v and heading ψ .

2.1 Modeling of surface vessels

In order to accurately perform trajectory tracking, it is important to have a good model of the system we want to control. In this section we will provide background on how to model a surface vessel, including kinematics, dynamics, and thrusters. We will also show how the model can be made parametric, in a way that keeps the parameters linear in the model. This is useful, as it gives some nice properties when learning the model parameters.

2.1.1 Kinematics and dynamics

For control purposes, it is beneficial to keep the vessel model reasonably simple, this can be done by limiting the degrees of freedom, 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 $SO(n) = \{\mathbf{R} | \mathbf{R} \in \mathbb{R}^{n \times n}, \mathbf{R}^\top \mathbf{R} = \mathbf{R} \mathbf{R}^\top = \mathbf{I}, \det(\mathbf{R}) = 1\}$ 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, $\mathbf{p} = [x, y]^\top$ 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 1. Using the notation from [35], a 3-DOF vessel can be modeled as follows

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu}, \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau}_{\text{Thrust}} + \boldsymbol{\tau}_{\text{Env}}, \end{aligned} \tag{1}$$

where $\mathbf{M} \in \mathbb{R}^{3 \times 3}$, $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3 \times 3}$, $\boldsymbol{\tau}_{\text{Thrust}} \in \mathbb{R}^3$, $\boldsymbol{\tau}_{\text{Env}} \in \mathbb{R}^3$ and $\mathbf{J}(\boldsymbol{\eta}) \in SO(3)$ are the inertia matrix, damping matrix, Coriolis matrix, thruster forces, environmental forces and transformation matrix respectively. The transformation

matrix $\mathbf{J}(\boldsymbol{\eta}) \in SO(3)$ is given by

$$\mathbf{J}(\boldsymbol{\eta}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

and is the rotation from the body frame to the earth-fixed North East Down (NED) reference frame. For notational brevity, we can express the vessel dynamics in (1), in terms of the implicit continuous time dynamics:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}} - \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{\tau}_{\text{Thrust}} - \boldsymbol{\tau}_{\text{Env}} \end{bmatrix} = 0. \quad (3)$$

2.1.2 Thrust configuration

In order to find the thrust vector $\boldsymbol{\tau}_{\text{Thrust}}$ we must consider the mapping between the actuators present on the vessel, and how they translate into the surge, sway and yaw forces and moments acting on the vessel. This mapping can be represented by the thrust configuration matrix $\mathbf{T}(\boldsymbol{\alpha}) \in \mathbb{R}^{3 \times n_{\text{thrusters}}}$ which maps the thrust \mathbf{f} from each thruster into the surge, sway and yaw forces and moments in the body frame of the vessel given the thruster azimuth angles $\boldsymbol{\alpha}$.

$$\boldsymbol{\tau}_{\text{Thrust}} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{f} \quad (4)$$

Each column $\mathbf{T}_i(\alpha_i)$ in $\mathbf{T}(\boldsymbol{\alpha})$ gives the configuration of the forces and moments of a thruster i as follows:

$$\mathbf{T}_i(\boldsymbol{\alpha})\mathbf{f}_i = \begin{bmatrix} F_x \\ F_y \\ F_y l_x - F_x l_y \end{bmatrix} = \begin{bmatrix} f_i \cos(\alpha_i) \\ f_i \sin(\alpha_i) \\ f_i(l_x \sin(\alpha_i) - l_y \cos(\alpha_i)) \end{bmatrix} \quad (5)$$

where α_i is the orientation of the thruster in the body frame, and f_i is the force it produces. Selecting the orientation $\boldsymbol{\alpha}$ and force \mathbf{f} of the thrusters in order to generate the desired force $\boldsymbol{\tau}$ is called the *thrust allocation* problem. While there are numerous ways of solving the thrust allocation problem [36], for our purpose we want to include the thrust allocation as part of the optimization for performing path tracking. This allows us to take into account physical thruster constraints such as force saturation and feasible azimuth sectors:

$$\begin{aligned} \underline{\alpha}_i &\leq \alpha_i \leq \bar{\alpha}_i \\ \underline{f}_i &\leq f_i \leq \bar{f}_i, \end{aligned}$$

without limiting the trajectory tracking control scheme to fully-actuated vessels. We may additionally take into account thruster dynamics, this is especially useful for azimuth thrusters, where the rotation of the truster from one orientation to an other can be quite slow. Given a setpoint for the azimuth angle $\boldsymbol{\alpha}_s$ and thrust force \mathbf{f}_s , we can express the thruster dynamics as follows:

$$\begin{aligned} \dot{\boldsymbol{\alpha}} &= \mathbf{f}_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}, \boldsymbol{\alpha}_s) \\ \dot{\mathbf{f}} &= \mathbf{f}_{\mathbf{f}}(\mathbf{f}, \mathbf{f}_s) \end{aligned} \quad (6)$$

Adding the thruster dynamics we get the following continuous time implicit model dynamics.

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\eta}} - \mathbf{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{T}(\boldsymbol{\alpha})\mathbf{f} - \boldsymbol{\tau}_{\text{Env}} \\ \dot{\boldsymbol{\alpha}} = f_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}, \boldsymbol{\alpha}_s) \\ \dot{\mathbf{f}} = f_{\mathbf{f}}(\mathbf{f}, \mathbf{f}_s) \end{bmatrix}}_{f(\hat{\mathbf{x}}, \mathbf{x}, \mathbf{u})} = 0, \quad (7)$$

where the state \mathbf{x} consists of the pose $\boldsymbol{\eta}$, velocity $\boldsymbol{\nu}$, thruster force \mathbf{f} and azimuth angles $\boldsymbol{\alpha}$. And the control inputs \mathbf{u} are given in terms of the thrust forces setpoint \mathbf{f}_s and azimuth angle setpoint $\boldsymbol{\alpha}_s$.

2.1.3 Parametric model

While the model structure for a surface vessel is well known, estimate the model parameters can be quite difficult. For our approach we try to make as few assumptions on the parameters of the vessel model as possible, and use online learning in order to model the vessel based on gathered data. For this we assume that we know the model structure as given in (7), but that the model parameters in the inertia matrix \mathbf{M} , coriolis matrix \mathbf{C} and damping matrix \mathbf{D} are unknown. Assuming the vessel has port starboard symmetry, from [35] we get the following structure:

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & 0 & 0 \\ 0 & m_{2,2} & m_{2,3} \\ 0 & m_{2,3} & m_{3,3} \end{bmatrix}, \quad (8)$$

$$\mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m_{2,2} \cdot v - m_{2,3} \cdot r \\ 0 & 0 & m_{1,1} \cdot u \\ m_{2,2} \cdot v + m_{2,3} \cdot r & m_{1,1} \cdot u & 0 \end{bmatrix}, \quad (9)$$

$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} -X_u - X_{|u|u} \cdot |u| & 0 & 0 \\ 0 & -Y_v - Y_{|v|v} \cdot |v| - Y_{|r|v} \cdot |r| & -Y_r - Y_{|v|r} \cdot |v| - Y_{|r|r} \cdot |r| \\ 0 & -N_v - N_{|v|v} \cdot |v| - N_{|r|v} \cdot |r| & -N_r - N_{|v|r} \cdot |v| - N_{|r|r} \cdot |r| \end{bmatrix}, \quad (10)$$

where $m_{1,1}$, $m_{2,2}$, $m_{2,3}$, $m_{2,3}$, $m_{3,3}$ are the mass and added mass in the inertia matrix and X_u , $X_{|u|u}$, Y_v , $Y_{|v|v}$, $Y_{|r|v}$, Y_r , $Y_{|v|r}$, $Y_{|r|r}$, N_v , $N_{|v|v}$, $N_{|r|v}$, N_r , $N_{|v|r}$, $N_{|r|r}$ are the linear and nonlinear dampening terms. For the damping matrix $\mathbf{D}(\boldsymbol{\nu})$, both linear and nonlinear terms are included. The linear terms are important for low speed maneuvering and station keeping, while ensuring the velocity converges exponentially to zero. The nonlinear terms are required as they dominate at higher velocities. This ensures that the model is able to handle a wide range of velocities. To more accurately capture the dynamics, it is possible to include higher order terms, however this also increases the complexity of the model, and may in some cases lead to overfitting of the model.

In addition to learning the vessel dynamics, it is also useful to compensate for environmental forces, such as wind and current. This can be done by modeling the

environmental forces as bias vector $\mathbf{w} \in \mathbb{R}^3$, which can be learned online together with the model parameters. Assuming that \mathbf{w} is given in the the NED frame, the resulting force in the body frame can then be modeled as follows:

$$\boldsymbol{\tau}_{\text{Env}} = \mathbf{W}\mathbf{J}^\top(\boldsymbol{\eta})\mathbf{w}, \quad (11)$$

where \mathbf{W} is a weighting matrix representing how environmental forces act on the different dimensions of the vessel. In this approach, we choose a weighting based on the cross sectional area of the vessel:

$$\mathbf{W} = \text{diag}([w, l, 1]^\top)$$

where l and w are the length and width of the vessel respectively, note that for better accuracy, a possibly state dependant \mathbf{W} based on the hull geometry may be used instead of the length and width.

Choosing the parameters $\boldsymbol{\theta}_{\text{model}}$ as the vector of model parameters from the mass matrix $(m_{1,1}, m_{1,2}, m_{2,2}, m_{2,3}, m_{3,3})$, dampening matrix $(X_u, Y_v, Y_r, N_v, N_r, X_{|u|u}, Y_{|v|v}, Y_{|v|r}, Y_{|r|v}, Y_{|r|r}, N_{|v|v}, N_{|v|r}, N_{|r|v}, N_{|r|r})$ and environmental forces (w_1, w_2, w_3) , we get a parametric continuous time model, which is linear in the parameters, on the following form:

$$f_{c,\theta}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}) = 0. \quad (12)$$

2.1.4 Parametric discrete time model

In order to use the vessel model for control, we discretize the continuous time dynamics in (7). While many options for discretizing the continuous time model exist, we chose to use the forward Euler integration for two main reasons.

- Forward Euler results in a discretization that is computationally cheap to evaluate. This is important when used in a real time NMPC setting, where increased model complexity results in increased evaluation time.
- Forward Euler preserves the linear in the parameters model structure from the the continuous time model. This is a highly desired property when performing parameter updates.

Given a sampling time T_s , the discretization resulting from the forward Euler method is given by replacing the derivative of the state $\dot{\mathbf{x}}$ with the approximation $\frac{\mathbf{x}^+ - \mathbf{x}}{T_s}$, where \mathbf{x}^+ is the state at the next timestep. Using this we can formulate an implicit discrete time model on the form:

$$f_{d,\theta}(\mathbf{x}^+, \mathbf{x}, \mathbf{u}) = f_{c,\theta}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}) \Big|_{\dot{\mathbf{x}} = \frac{\mathbf{x}^+ - \mathbf{x}}{T_s}} = 0. \quad (13)$$

2.2 Trajectory tracking NMPC

For trajectory tracking control, the objective is to find a control policy which is able to make the vessel converge to a desired trajectory $\mathbf{x}_d(t)$. For a vessel with the dynamics given in (13), the control policy is the mapping from the vessel position $\mathbf{x}(t)$ and desired trajectory $\mathbf{x}_d(t)$, to the individual thruster force \mathbf{f}_d and azimuth α_d setpoints. Formulating this as an OCP, we get the following:

$$\min_{\mathbf{x}, \boldsymbol{\alpha}, \mathbf{f}} \quad \lambda_{\boldsymbol{\theta}}(\mathbf{x}_0, \mathbf{x}_{d,0}) + \sum_{i=0}^{N-1} \gamma^i L(\mathbf{x}_i, \mathbf{x}_{d,i}, \mathbf{u}_i) + \gamma^N V_{\boldsymbol{\theta}}^f(\mathbf{x}_N, \mathbf{x}_{d,N}) \quad (14a)$$

$$\text{s.t.} \quad f_{d,\boldsymbol{\theta}}(\mathbf{x}_{i+1}, \mathbf{x}_i, \mathbf{u}_i) = 0, \quad (14b)$$

$$\underline{\mathbf{f}} \leq \mathbf{f}_{s,i} \leq \bar{\mathbf{f}}, \quad (14c)$$

$$\underline{\boldsymbol{\alpha}} \leq \boldsymbol{\alpha}_{s,i} \leq \bar{\boldsymbol{\alpha}}, \quad (14d)$$

$$\mathbf{x}_0 = \mathbf{s}. \quad (14e)$$

Due to the nonlinear nature of the kinematics and dynamics of the vessel model, as well as the nonlinear cost function, we should note that the above OCP is a nonlinear optimization problem, and can be solved using a NMPC. The goal is to minimize the objective function (14a), consisting of an initial cost $\lambda_{\boldsymbol{\theta}}(\cdot)$, a discounted stage cost $L(\cdot)$ over a horizon N and a terminal cost $V_{\boldsymbol{\theta}}^f(\cdot)$, subject to vessel dynamics (14b), force (14c) and azimuth (14d) bounds and vessel initial condition (14e). In this formulation we can note that the initial cost $\lambda_{\boldsymbol{\theta}}(\mathbf{x}_0, \mathbf{x}_{d,0})$ does not effect the solution of the OCP, but is used when performing the reinforcement learning. We should also note that discounting the stage cost with a discount factor $\gamma < 1$, ensures that the cumulative cost converges to a finite value over the infinite horizon, allowing us to approximate it with the terminal cost.

2.2.1 Cost function

In order to perform trajectory tracking, we need to formulate a cost function (14a) which ensures that the NMPC performs the desired trajectory tracking behaviour. For the stage cost the following cost function was chosen:

$$\begin{aligned} L(\mathbf{x}, \mathbf{x}_d, \mathbf{u}) &= q_{x,y} \cdot c_{x,y}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) \\ &+ q_{\psi} \cdot c_{\psi}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) \\ &+ (\boldsymbol{\nu} - \boldsymbol{\nu}_d)^\top \mathbf{Q} (\boldsymbol{\nu} - \boldsymbol{\nu}_d) \\ &+ \boldsymbol{\alpha}^\top \mathbf{R}_{\alpha} \boldsymbol{\alpha} + \mathbf{f}^\top \mathbf{R}_f \mathbf{f} \end{aligned} \quad (15)$$

The stage cost in (15) uses a quadratic penalty on velocity and control actions with weight matrices \mathbf{Q} , \mathbf{R}_{α} and \mathbf{R}_f . The position cost $c_{x,y}(\boldsymbol{\eta}, \boldsymbol{\eta}_d)$, weighted by $q_{x,y}$, is

chosen as a pseudo-Huber function, penalizing the difference between the vessel pose $\boldsymbol{\eta}$ and the desired pose $\boldsymbol{\eta}_d$, and is given as follows:

$$c_{x,y}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) = \delta^2 \left(\sqrt{1 + \frac{(x - x_d)^2 + (y - y_d)^2}{\delta^2}} - 1 \right). \quad (16)$$

Using a pseudo-Huber cost, provides a quadratic penalty when the quadrature position error is small and linear when the position error is large. This helps with numerical stability, as well as performance when large position errors are observed [37, 38]. For the heading cost function $c_\psi(\boldsymbol{\eta}, \boldsymbol{\eta}_d)$, weighted by q_ψ , the following was chosen:

$$c_\psi(\boldsymbol{\eta}, \boldsymbol{\eta}_d) = \frac{1 - \cos(\psi - \psi_d)}{2}, \quad (17)$$

as it avoids the problem of heading wraparound.

For the parametric initial cost $\lambda_\theta(\mathbf{x}_0, \mathbf{x}_{d,0})$, a simple bias term was chosen, giving the following:

$$\lambda_\theta(\mathbf{x}_0, \mathbf{x}_{d,0}) = \theta_\lambda.$$

Similarly, the parametric terminal cost approximation $V_\theta^f(\cdot)$ was chosen as a quadratic cost as follows:

$$V_\theta^f(\mathbf{x}_N, \mathbf{x}_{d,N}) = (\mathbf{x}_N - \mathbf{x}_{d,N})^\top \text{diag}(\boldsymbol{\theta}_V)(\mathbf{x}_N - \mathbf{x}_{d,N}),$$

with the parameters θ_λ and $\boldsymbol{\theta}_V$ being learned through reinforcement learning.

2.3 Reinforcement Learning-based NMPC

Given the model parameters $\boldsymbol{\theta}_{\text{model}}$ and cost function parameters θ_λ and $\boldsymbol{\theta}_V$ the goal is to learn the the parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_{\text{model}}, \theta_\lambda, \boldsymbol{\theta}_V),$$

based on data gathered on line, in a way that optimizes the closed loop performance of the NMPC given by (14). In recent works, such as [33, 34, 39], this has been done by allowing RL to use a NMPC as a function approximator. This combines the benefits of data-driven optimization from RL with the tools available for analysing and certifying the closed loop performance of NMPC. For our implementation, we will use the approach in [34], where the RL-based NMPC is combined with system identification (SYSID) in a way that minimizes plant model mismatch while optimizing the closed loop performance of the NMPC. In the next subsections we will show how this approach can be applied to the trajectory tracking problem in (14).

2.3.1 Value functions and policy

Given the parametric optimization problem (14), we define the parametric action-value function as:

$$Q_{\theta}(\mathbf{s}, \mathbf{a}) = \min_{\mathbf{x}, \mathbf{u}} \quad (14a) \tag{18a}$$

$$\text{s.t.} \quad (14b) - (14d), \tag{18b}$$

$$\mathbf{x}_0 = \mathbf{s}, \tag{18c}$$

$$\mathbf{u}_0 = \mathbf{a}. \tag{18d}$$

This action-value function $Q_{\theta}(\mathbf{s}, \mathbf{a})$ approximates the expected cumulative discounted cost when taking an action \mathbf{a} in a state \mathbf{s} . Using the action-value function $Q_{\theta}(\mathbf{s}, \mathbf{a})$, we can express the state-value function $V_{\theta}(\mathbf{s})$ and policy $\pi_{\theta}(\mathbf{s})$ as follows:

$$V_{\theta}(\mathbf{s}) = \min_{\mathbf{a}} Q_{\theta}(\mathbf{s}, \mathbf{a}), \tag{19a}$$

$$\pi_{\theta}(\mathbf{s}) = \arg \min_{\mathbf{a}} Q_{\theta}(\mathbf{s}, \mathbf{a}), \tag{19b}$$

where the the policy $\pi_{\theta}(\mathbf{s})$ approximates the optimal action for a state \mathbf{s} , and the state-value function $V_{\theta}(\mathbf{s})$ approximates the expected cumulative discounted cost under the policy.

2.3.2 Q-Learning

The goal of RL is to find the parameters θ that maximize the closed loop performance under the policy $\pi_{\theta}(\mathbf{s})$. While a number of different approaches exist, we will focus on the classical Q-Learning method [40]. In Q-Learning the goal is to find the parameterization which best fits the action-value function to the observed data. Given an observed transition $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ Q-Learning can be performed by minimizing the temporal-difference error:

$$\delta_t = y_t - Q_{\theta}(\mathbf{s}_t, \mathbf{a}_t), \tag{20}$$

where $y_t = L(\mathbf{x}_t, \mathbf{x}_{d,t}, \mathbf{u}_t) + \gamma V_{\theta}(\mathbf{x}_{t+1})$ is the fixed target value. Defining the squared temporal-difference error as the minimization objective, and assuming that the target value is independent of the parameterization θ , we get the semi-gradient update [20]:

$$\theta \leftarrow \theta + \beta_Q \delta \nabla_{\theta} Q_{\theta}(\mathbf{x}_t, \mathbf{u}_t), \tag{21}$$

where $\beta_Q > 0$ is the step-size or learning rate. For the classical semi-gradient Q-learning scheme given in (21), a second order method can be implemented by using quasi-Newton steps instead of gradient steps. This results in the following update law:

$$\theta \leftarrow \theta + \beta_Q \underbrace{\delta \mathbf{H}_Q^{-1} \nabla_{\theta} Q_{\theta}(\mathbf{x}_t, \mathbf{u}_t)}_{:= \Delta \theta_Q}, \tag{22}$$

where $\mathbf{H}_Q = \nabla_{\boldsymbol{\theta}}^2 (y_t - Q_{\boldsymbol{\theta}}(\mathbf{x}_t, \mathbf{u}_t))^2$ is the Hessian of the error between the targets and the action-value function. It is also possible to further generalize this method for a batch of transitions, resulting in a nonlinear least squares problem [34].

2.3.3 System Identification

In addition to learning the parameters $\boldsymbol{\theta}$ from Q-Learning, it is also possible to learn the parameters associated to the MPC model using SYSID. One such approach is the Prediction Error Method (PEM) where the objective is to minimize the difference between the observed state and the predicted state given the observed transition $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$. For a parametric model approximation of the form:

$$f_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}_t, \mathbf{u}_t) = 0$$

the prediction error \mathbf{e}_t between the parametric model and the observed state can then be expressed as follows:

$$\mathbf{e}_t = f_{\boldsymbol{\theta}}(\mathbf{x}_{t+1}, \mathbf{x}_t, \mathbf{u}_t).$$

In the simplest case, where the state vector \mathbf{x} is fully observable, PEM can be performed by minimizing the squared error $\|\mathbf{e}_t\|^2$ between the observed state, and the predicted state. This optimization problem can be tackled via gradient descent, giving the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta_f \nabla_{\boldsymbol{\theta}} \mathbf{e}_t^{\top} \mathbf{e}_t,$$

where β_f is the learning rate. Similarly to the RL objective, we can use Quasi newton steps:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta_f \underbrace{\mathbf{H}_f^{-1} \nabla_{\boldsymbol{\theta}} \mathbf{e}_t^{\top} \mathbf{e}_t}_{:= \Delta \boldsymbol{\theta}_f}, \quad (23)$$

where \mathbf{H}_f is the hessian of the squared prediction error. This can be further generalized for a batch of transitions [34]. We can also note that if the model is linear in the parameters, which is the case for the vessel model in (13), this becomes a linear least squares problem, for which the global minimum can be found by taking the a full newton step i.e. choosing $\beta_f = 1$.

2.3.4 Parameter update law

When updating the parameters $\boldsymbol{\theta}$ it is possible to combine both Q-Learnig and SYSID. The simplest approach is to directly combine the steps from both the Q-Learning and SYSID. Using the second order update laws in (22) and (23), with the parameter updates $\Delta \boldsymbol{\theta}_Q$ and $\Delta \boldsymbol{\theta}_f$ respectively, we get the following:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_Q \Delta \boldsymbol{\theta}_Q + \beta_f \Delta \boldsymbol{\theta}_f. \quad (24)$$

Here the step-lengths β_Q and β_f can be thought of as the weighting between the Q-Learning and SYSID respectively. However, the end goal is arguably to maximize the

closed-loop performance of the MPC scheme rather than minimizing the prediction error of the model, hence if the two objectives are competing, the RL objective should be prioritized. In order to prioritize the RL objective we use instead the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_Q \Delta \boldsymbol{\theta}_Q + \beta_f \mathbf{P} \Delta \boldsymbol{\theta}_f, \quad (25)$$

where \mathbf{P} is a projection matrix. As proposed in [34], we can choose the projection matrix as the direction in the parameter space for which the RL objective is the least sensitive, i.e. direction of the smallest eigenvalues of \mathbf{H}_Q . This allows for minimizing the prediction error of the model with SYSID, while prioritizing the RL objective of optimizing the closed loop performance of the MPC.

3 Implementation

In order to test the proposed RL base trajectory tracking NMPC, we performed tests on two different vessels. In this section we will introduce the two platforms, and discuss how we implemented the proposed RL based trajectory tracking control method in a way that allowed for it to be used in real time.

3.1 Advanced-step Nonlinear Model Predictive Control

While a number of NMPC schemes exist, that guarantee closed loop stability [31], the necessary on-line computation time is typically not taken into account. Even though recent hardware and software developments have lead to faster and more efficient numerical solution methods for open-loop optimal control, the solution time is often significant in the context of closed-loop control. The resulting delay caused by solving the NMPC problem online can often lead to degraded performance, or in some cases even instability of the closed loop system. In order to account for the computational delay, a number of methods have been proposed [41–44]. One such approach is the advanced-step NMPC (asNMPC) [44], where the idea is to first use the current state measurement and control action to predict the state one step into the future and then solve the corresponding NMPC problem in advance. Transcribing the OCP in (14) into standard form nonlinear programming problem (NLP):

$$\begin{aligned} \min_{\mathbf{w}} \quad & \phi(\mathbf{w}, \mathbf{p}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{w}, \mathbf{p}) = 0 \\ & \mathbf{h}(\mathbf{w}, \mathbf{p}) \leq 0, \end{aligned} \quad (26)$$

where \mathbf{w} are the decision variables, and \mathbf{p} are the parameters, chosen to be the initial state \mathbf{s} . This gives the First-Order Necessary Conditions of a primal dual interior

point method as follows:

$$\mathbf{r}(\mathbf{z}, \mathbf{p}) = \begin{bmatrix} \nabla_{\mathbf{w}}\phi + \nabla_{\mathbf{w}}\mathbf{g}\boldsymbol{\lambda} + \nabla_{\mathbf{w}}\mathbf{h}\boldsymbol{\mu} \\ \mathbf{g} \\ \text{diag}(\boldsymbol{\mu})\mathbf{h} + \tau \end{bmatrix} = 0 \quad (27)$$

where $\mathbf{z} = [\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}]$ are the primal-dual variables, and τ is a constraint relaxation parameter. If the Linear Independence Constraint Qualification (LICQ) and Second Order Sufficient Conditions (SOSC) hold [45] for the NLP in (26), then the Implicit Function Theorem (IFT) guarantees that:

$$\frac{\partial \mathbf{r}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{p}} + \frac{\partial \mathbf{r}}{\partial \mathbf{p}} = 0. \quad (28)$$

Solving the NLP in (26) for a parameterization \mathbf{p}_0 , with the solution of the primal-dual variables \mathbf{z}_0 , we can construct the following linear predictor.

$$\mathbf{z} = \mathbf{z}_0 + \frac{\partial \mathbf{z}}{\partial \mathbf{p}}(\mathbf{p} - \mathbf{p}_0) \quad (29)$$

Using the first order predictor on the NLP resulting from an NMPC problem, with the initial state \mathbf{x}_0 chosen as the parameter vector $\mathbf{p} = \mathbf{x}_0$, the asNMPC can be summarized as follows:

- In the background between time step t and $t + 1$:
 - Predict the state $\hat{\mathbf{x}}_{t+1}$ using forward simulation.
 - Solve the NMPC problem with $\mathbf{p}_0 = \hat{\mathbf{x}}_{t+1}$, to get the primal dual variables \mathbf{z}_0 .
 - Compute the parameter sensitivity $\frac{\partial \mathbf{z}}{\partial \mathbf{p}}$ using the IFT (28).
- On-line at time step $t + 1$:
 - Obtain the true state of the system \mathbf{x}_{t+1} from sensor measurements.
 - Use the linear predictor (29) to find the approximate solution \mathbf{z} of the NLP.
 - Extract the first control input \mathbf{u}_{t+1} from the approximate solution \mathbf{z} .
 - Apply the control input \mathbf{u}_{t+1} to the plant, and return to the background step.

The above asNMPC algorithm will then yield an approximate control law with a minimal delay between the measurement of the state and the application of the control input. This allows us to approximately solve the NMPC problem in (14) in real time, making the the proposed control scheme feasible for use on physical platforms. It should however be noted that this is still a computationally demanding control architecture, and requires that the OCP can be solved within one time step. In general, this requirement will limit the length of the prediction horizon of the asNMPC, and the complexity of the parameterized model.



Figure 2: The ReVolt test platform is a 1 : 20 scale model of a autonomous concept vessel with two fully rotatable azimuth thrusters in the stern and one fully rotatable azimuth thruster in the bow.

3.2 Experimental Platforms

In order to test the proposed method, simulations studies were performed on two different platforms, with additional full scale experiments carried out on one of them. In the next sections we will introduce the two platforms, namely *ReVolt* and *milliAmpere*, and discuss how the proposed RL based trajectory tracking NMPC was implemented.

3.2.1 ReVolt platform

The *ReVolt*, shown in Figure 2, is a 1 : 20 scale model of a autonomous concept vessel developed and built by DNV GL in collaboration with NTNU. The 3 meter long and 0.72 meter wide model, weighs approximately 257 kg, and has three fully rotatable azimuth thrusters for propulsion. The thrust configuration seen in Figure 2, consists of two identical stern thrusters, and one slightly less powerful bow thruster, giving the vessel a total combined engine power of 360 W and a top speed of 2 knots (approximately 1 m/s).

For simulating the *ReVolt* an accurate Digital Twin, developed by DNV GL, was used. The Digital Twin is based on a full 6DOF model, with parameters identified through tow-tank experiments, as well as frequency domain analysis of a 3D model of the vessel hull. The Digital Twin allowed for rapidly testing how the proposed control scheme performed under ideal conditions, as well as under different environmental conditions, including disturbances from wind, waves and ocean currents.

The trajectory tracking controller for the *ReVolt* was implemented as an asNMPC, solving the OCP in (14) in each sampling interval. Due to the vessel having three fully rotatable azimuth thrusters with relatively fast dynamics, the thruster dynamics were not modeled, and an additional singularity avoidance penalty (30) was added to



Figure 3: The MilliAmpere test platform has two fully rotatable azimuth thrusters along the centerline of the vessel.

the cost function, in order to encourage nonsingular thrust configurations [46].

$$\frac{\rho}{\epsilon + \det(\mathbf{T}(\boldsymbol{\alpha})\mathbf{T}^\top(\boldsymbol{\alpha}))} \quad (30)$$

Using the solution of the advanced step prediction, the thrust and azimuth angle commands were directly applied to the vessel thrusters. The sampling time of the controller was chosen as $T_s = 0.2\text{s}$ (5Hz) giving enough time to solve the NMPC, while still being fast enough to stabilize the system. In order to learn the parameters $\boldsymbol{\theta}$ on-line, the update law in (25) was used on a batch of $M = 1$ samples which were recorded on-line. A list of parameter values used for the NMPC implementation is given in Table 3.

3.2.2 milliAmpere platform

The *milliAmpere*, shown in Figure 3, is an experimental autonomous urban passenger ferry which has been in development at the Norwegian University of Science and Technology (NTNU) since 2017. *milliAmpere* has served as a platform for testing and developing autonomous technology, including software, sensor arrays, as well as hardware solutions. The platform is 5 meters long and 2.8 meters wide, with a symmetric footprint. It has two fully rotatable azimuth thrusters mounted along the center line of the vessel, giving it a top speed of 5 knots (approximately 2.5 m/s).

For simulating the *milliAmpere*, a nonlinear 3DOF model of the vessel was used together with models of the thruster and azimuth dynamics, with the model parameters being identified through experiments.

The trajectory tracking controller for the *milliAmpere* was implemented as an asNMPC, solving the OCP in (14). Due to slow azimuth thruster rotation, the azimuth dynamics were also included in the vessel model, allowing for the NMPC to account

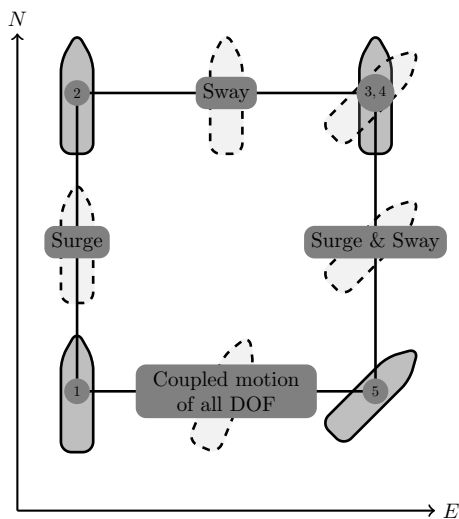


Figure 4: Illustration of the four corner DP test, used for testing trajectory tracking performance in individual as well as coupled degrees of freedom.

for the the dynamics when planning the control actions. Similar to the *ReVolt* implementation, the sampling time for the *milliAmpere* controller was chosen as $T_s = 0.2s$ (5Hz) giving enough time to solve the NMPC, while still being fast enough to stabilize the system. The parameter update law (25) was also used on the *milliAmpere*, on a batch of $M = 10$ samples recorded on-line. This allowed the controller to continuously adapt to the changing conditions. A list of parameter values used for the NMPC implementation is given in Table 4.

4 Results

In this section we present the results from simulations on the *ReVolt* test platform (Figure 2), as well as simulations and sea trials on the *milliAmpere* test platform (Figure 3).

4.1 Four corner DP test

In order to evaluate the trajectory tracking capabilities of the proposed control method, the four corner test seen in Figure 4 is used. This test is used in order to evaluate the trajectory tracking capabilities of the vessel for individual degrees of freedom, as well as the coupled motion of all degrees of freedom. The four corner test starts with the vessel pointing north 0° , then performs the following commands:

I. Reinforcement Learning-based MPC for Tracking Control ...

Table 1: List test scenarios for *ReVolt* (only simulations).

Scenario	Description
<i>R1</i>	Simulation of baseline Adaptive dynamic programming (ADP) method from [19]
<i>R2</i>	Simulation of RL-based NMPC without online learning
<i>R3</i>	Simulation of RL-based NMPC with online learning
<i>R4</i>	Simulation of RL-based NMPC with online learning and 3m/s wind from the north
<i>R5</i>	Simulation of RL-based NMPC with online learning and 0.1m/s current from the west

1. Move l meters due north, this tests the surge motion of the vessel.
2. Move l meters due east, this tests the sway motion of the vessel.
3. Rotate to a heading of 45° while keeping the same position, this tests the yaw motion of the vessel.
4. Move l meters due south while keeping the same heading, this tests the coupled surge and sway motion of the vessel.
5. Move l meters due west while rotating to a heading of 0° , this tests coupled motion of all degrees of freedom.

For the four corner test we chose the side length l to be 5 meters, for the *ReVolt*, and l to be 10 meters for the *milliAmpere*. Each maneuver is given 60 seconds to complete, and reference filter is used to generate a continuous trajectory between the commanded maneuvers.

In order to evaluate the performance of the dynamic positioning, we use the Integral Absolute Error (IAE) given in (31).

$$IAE(t) = \int_0^t \sqrt{(\boldsymbol{\eta} - \boldsymbol{\eta}_d)^\top \mathbf{W}_{IAE}^{-1} (\boldsymbol{\eta} - \boldsymbol{\eta}_d)} dt \quad (31)$$

Where \mathbf{W}_{IAE} is a weighting factor, which is chosen to normalize the pose between ± 5 meters in north and east direction, and $\pm 50^\circ$ in heading, giving the following.

$$\mathbf{W}_{IAE} = \begin{bmatrix} 5^2 & 0 & 0 \\ 0 & 5^2 & 0 \\ 0 & 0 & 50^2 \end{bmatrix}$$

4.2 Results ReVolt (Simulations Only)

For the *ReVolt* platform, validation of the proposed trajectory tracking controller was performed in simulations for the five different scenarios given in Table 1. Performing

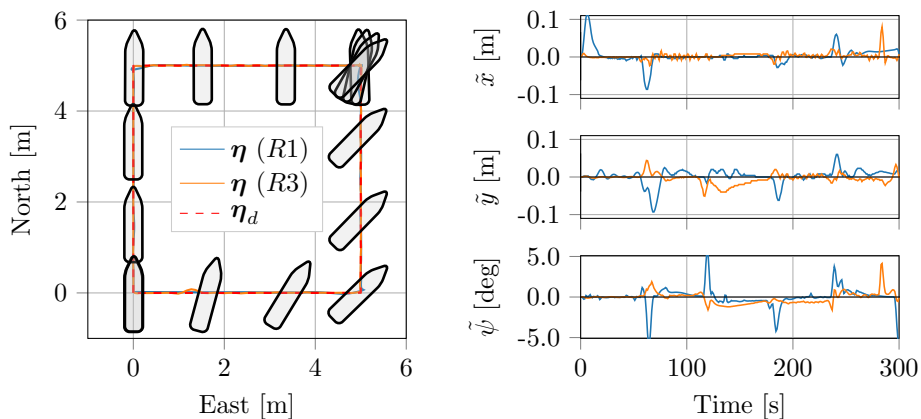


Figure 5: Simulation results for *ReVolt* with online learning (*R3*), and baseline (*R1*)

the four corner test, we got the results seen in Figure 5, with the IAE performance seen in Figure 6.

From the trajectory and tracking error for our method (*R3*) seen in Figure 5, we can observe that the positioning error is less than 10cm in both the North and East direction, while the heading is within 5° of the desired heading. Compared to the baseline Adaptive Dynamic Programming (ADP) method (*R1*), our method does not have the same spikes when transitioning between maneuvers. This is likely due to the NMPC planning ahead for the maneuver changes, while the baseline approach is having to react to them as they happen. The added prediction horizon of the NMPC is a definite advantage of the proposed approach, however it does come at the cost of computational complexity. While the proposed control scheme is limited to a 5Hz update rate due to the time it takes to solve the OCP, the ADP based solution is easily able to run at 10Hz.

From the IEA performance in Figure 6, we see how the proposed controller performs in different conditions, with and without parameter updates, as well as the performance compared to a baseline ADP approach. Looking at the performance of our method without online learning (*R2*), compared to our method with online learning (*R3*), we see a significant difference in performance. This is due to model mismatch between the initial model used in the OCP and the Simulator. Using online parameter updates, allows the model and performance of the NMPC to be improved based on gathered data, and results in a significant performance boost. It is also worth noting the performance difference between the baseline approach (*R1*), and our method with online learning (*R3*). For the baseline approach (*R1*), we see large increases in IAE when transitioning between the different maneuvers, while for our approach (*R3*), these increases are less prevalent. This is due to the RL-based NMPC being able to take into account the trajectory over future time horizon, as well as

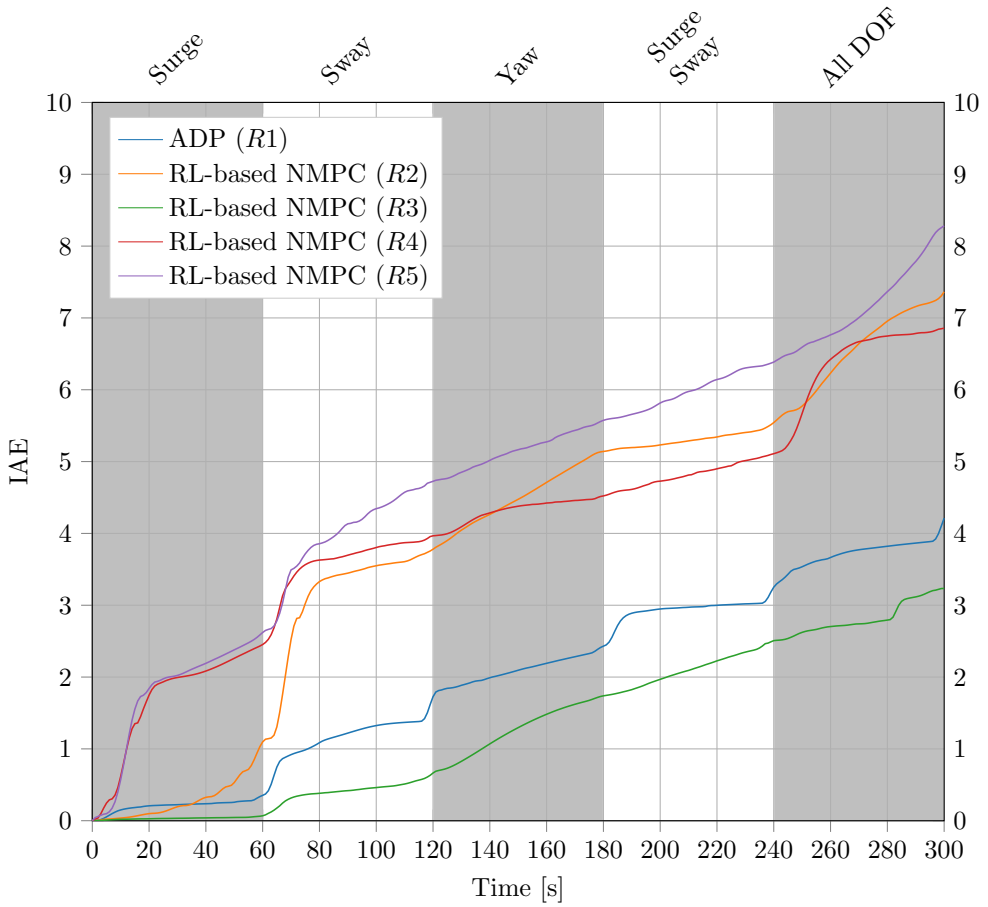


Figure 6: Integral Absolute Error (IAE) for *ReVolt*, the white and gray bands show the different phases of the four corner test.

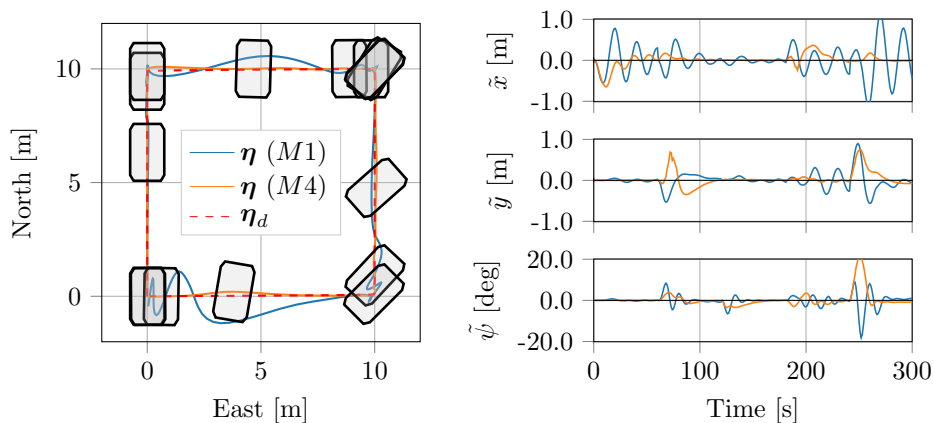


Figure 7: Simulation results for *milliAmpere* with online learning (*M4*), and baseline (*M1*).

Table 2: List test scenarios for *milliAmpere* (simulations and sea trials).

Scenario	Description
<i>M1</i>	Simulation of baseline PID based DP method from [47]
<i>M2</i>	Sea trials of baseline PID based DP method from [47]
<i>M3</i>	Simulation of RL-based NMPC without online learning
<i>M4</i>	Simulation of RL-based NMPC with online learning
<i>M5</i>	Sea trials of RL-based NMPC with online learning

including the thrust allocation in the OCP, allowing for more accurately planning and performing the maneuvers. Looking at our method when subject to external disturbances in terms of wind (*R4*) and current (*R5*), we see an initial increase in the IAE before the performance starts to stabilize, with a slope similar to that of trained RL-based NMPC. This behaviour is expected, as the initial increase happens since the controller is not aware of the disturbance, and flattens out as the controller learns how to compensate for the disturbance as a constant force and torque in the NED frame (11).

4.3 Results *milliAmpere* (Simulations and Sea Trials)

For the *milliAmpere* platform, validation of the proposed trajectory tracking controller was performed for five different scenarios, Table 2, including both simulations as well as sea trials. Performing the four corner test, we got the simulation results seen in Figure 7, the experimental results seen in Figure 8 and 11. The performance in terms of the IAE is given in Figure 9, and the performance in terms of power consumption

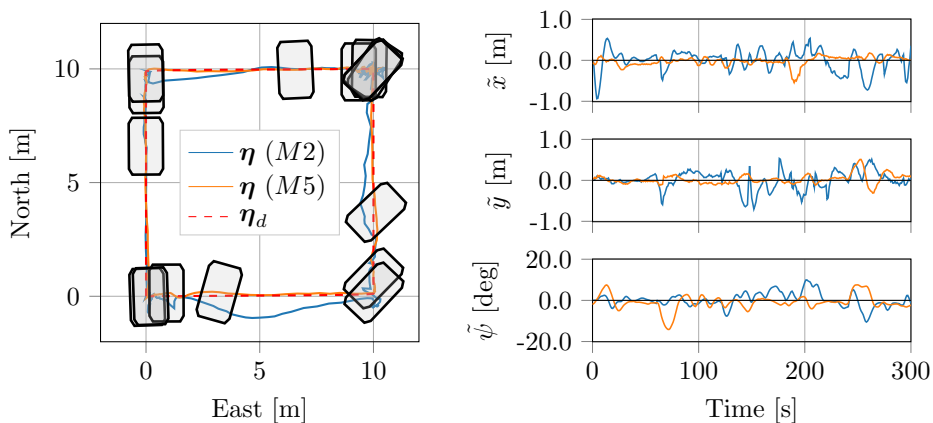


Figure 8: Sea trial results for *milliAmpere* with online learning ($M5$), and baseline ($M2$).

is shown in Figure 10.

Based on the results from our method in Figure 7 and 8, we see good tracking performance, with similar results for both the simulations ($M4$) and the sea trials ($M5$). From the tracking error we see that the trajectory is well within one meter, with most of the major tracking errors happening after command changes. This is mostly due to the vessel not being able to accelerate fast enough to follow the reference trajectory when switching between the different poses. For the heading error, we see a maximum error of about 20° . This is a relatively large error, and is the largest contributor to the IAE as can be seen in Figure 9. By choosing the weighting between the position error $q_{x,y}$ and the heading error q_ψ it is possible to change the priority between heading and position error, with our main focus being on the position error when choosing the parameters.

In order to evaluate our proposed control scheme, we performed the same simulations ($M1$) and sea trials ($M2$) using a standard Proportional Integral Derivative (PID) based DP controller with an optimization based control allocation scheme [47]. It should be noted that the PID controller was not tuned to optimize any performance measure, however it still provides a good benchmark. Compared to our approach, the PID based method has slightly less heading error, while our approach has significantly lower position error as can be seen in Figure 7 and 8, as well as in the IAE in Figure 9. Our approach also has about half the power consumption when performing the maneuver compared with the PID based DP controller, as seen in Figure 10. This is likely due to the PID based DP controller relying on an aggressive control allocation method, while our approach integrates the control allocation into the optimization problem, allowing it to consume less power while still being able to perform

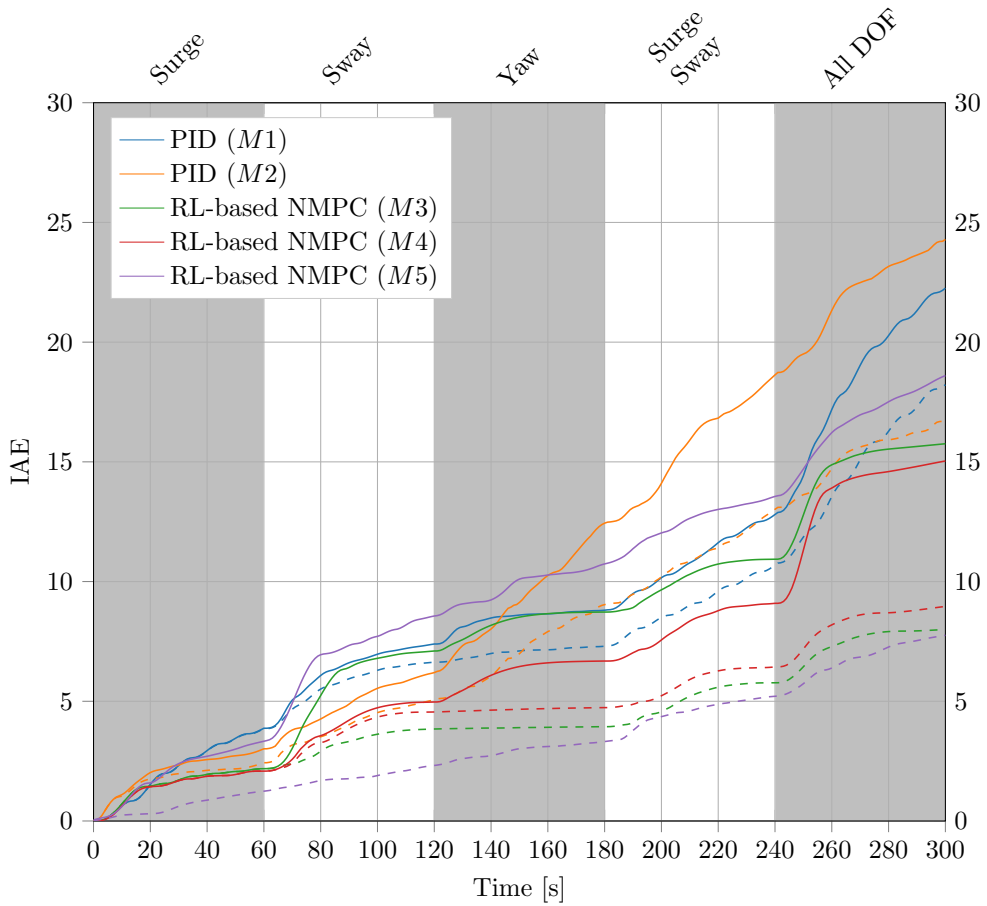


Figure 9: Integral Absolute Error (IAE) for *milliAmpere*, the white and gray bands show the different phases of the four corner test. The solid lines show the IAE with the heading error, while the dashed lines show the IAE without the heading error.

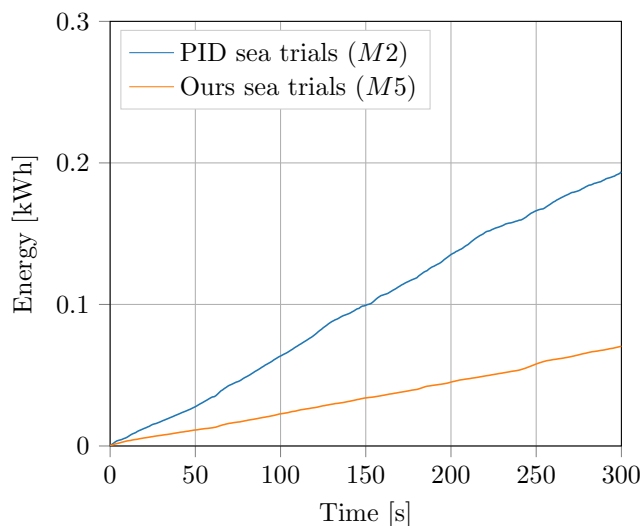


Figure 10: Power consumption for the Dynamic Positioning performed on *milliAmpere*.

the desired maneuvers. The integration of control allocation and thruster dynamics into OCP is a definite advantage of the proposed control scheme, as it significantly reduces the mismatch between desired and actual thrust, allowing for more accurate maneuvering. It should however be noted that the addition of control allocation and thruster dynamics does increase the model complexity, and hence the computational complexity when solving the OCP.

During the experiments, an unexpected azimuth failure occurred on one of the thrusters, but the proposed method was able to compensate and still perform the desired maneuver by learning a new model of the vessel which compensated for the azimuth failure. These additional results are shown in Appendix B.

5 Conclusion

We have presented a method for trajectory tracking control of surface vessels using a RL-based NMPC. The proposed method performs optimal tracking control, as well as control allocation by considering both the dynamics and kinematics of the vessel and the actuators. In order to account for model mismatch, and external disturbances, the proposed method uses RL and SYSID in order to update the model and NMPC on line and improve the closed loop performance. In order to run the method in real time, asNMPC was used. This reduced the computational delay of solving the optimization problem in each sampling interval, making the method real time feasi-

ble. It should however be noted that the proposed control architecture is still more computationally demanding than more traditional methods, which can be considered a drawback of the proposed method. This added complexity does however come with a trade off in terms of optimality and the ability to include complex truster dynamics and physical constraints into the OCP, resulting in better tracking performance than other traditional methods. This is further improved by using RL and SYSID to learn and identify disturbances and modelling errors in order to optimize the closed loop performance.

Based on both simulations and sea trials on both the *ReVolt* and *milliAmpere* platforms we have shown the flexibility of the proposed method. The experimental results also show how using a NMPC for handling both tracking and control allocation, allows the controller to account for the performance of the vessel over a prediction horizon. This makes the controller preemptive, leading to better tracking performance and less power consumption compared with other methods. The addition of an RL and SYSID update law allows for the control scheme to adapt to the environmental disturbances, and model mismatch in a way that optimizes the closed loop performance of the proposed control scheme. These benefits do however come with some drawbacks, including the computational complexity of the solving OCP, and robustness of the RL and SYSID update laws with respect to disturbances and measurement noise.

For future work, we would like to look more into how to perform more robust and safe RL and SYSID parameter updates, as care must be taken in order to avoid problematic parameter updates caused by for example noisy and inaccurate measurements. This is mostly a problem when running on a physical platform, and for us it was solved using batches of transitions, and sufficiently small learning rate. For future research it would also be interesting to look at different vessel models, including under-actuated vessels. It would also be interesting to look at other problems than tracking, such as for example planning and docking, where the problems have economic cost functions and additional constraints that need to be taken into account. An additional area of potential research is to optimize the implementation of the control scheme in order to improve the computation time. With dedicated hardware, and a dedicated real time NMPC implementations it should be possible to make to significantly improve computation time, allowing for the proposed method to be used on systems requiring faster update rates, more complex models and with a longer prediction horizon.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 3: List of parameter values for *ReVolt*.

Symbol	Value	Description
$q_{x,y}$	30	Huber penalty weight
q_ψ	50	Heading penalty weight
Q	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$	Velocity weight matrix
R_α	$\begin{bmatrix} 10^{-2} & 0 & 0 \\ 0 & 10^{-2} & 0 \\ 0 & 0 & 10^{-2} \end{bmatrix}$	Azimuth angle weight matrix
R_f	$\begin{bmatrix} 10^{-1} & 0 & 0 \\ 0 & 10^{-1} & 0 \\ 0 & 0 & 10^{-1} \end{bmatrix}$	Thruster force weight matrix
ρ	10^{-5}	Singular value penalty weight
ϵ	10^{-3}	Singular value penalty offset
δ	10	Huber penalty slope
T_s	0.2s	Time step
N	50	OCP shooting intervals
M	1	Batch size

A Controller parameters

The parameter values for the *ReVolt* and *milliAmpere* the MPC implementation are given in Table 3 and 4 respectively.

B Thruster failure results

During the *milliAmpere* sea trials, an unexpected azimuth failure occurred during one of the tests, see Figure 11. The proposed controller was however still able to perform the desired maneuver, and learn how to compensate for the failure.

References

- [1] Andreas B Martinsen et al. "Optimization-Based Automatic Docking and Berthing of ASVs Using Exteroceptive Sensors: Theory and Experiments". In: *IEEE Access* 8 (2020), pp. 204974–204986.
- [2] Glenn Bitar et al. "Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments". In: *IEEE Access* 8 (2020), pp. 199953–199969.

Table 4: List of parameter values for *milliAmpere*.

Symbol	Value	Description
$q_{x,y}$	20	Huber penalty weight
q_ψ	50	Heading penalty weight
\mathbf{Q}	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 50 \end{bmatrix}$	Velocity weight matrix
\mathbf{R}_α	$\begin{bmatrix} 10^{-1} & 0 \\ 0 & 10^{-1} \end{bmatrix}$	Azimuth angle weight matrix
\mathbf{R}_f	$\begin{bmatrix} 10^{-5} & 0 \\ 0 & 10^{-5} \end{bmatrix}$	Thruster force weight matrix
δ	10	Huber penalty slope
T_s	0.2s	Time step
N	50	OCP shooting intervals
M	10	Batch size

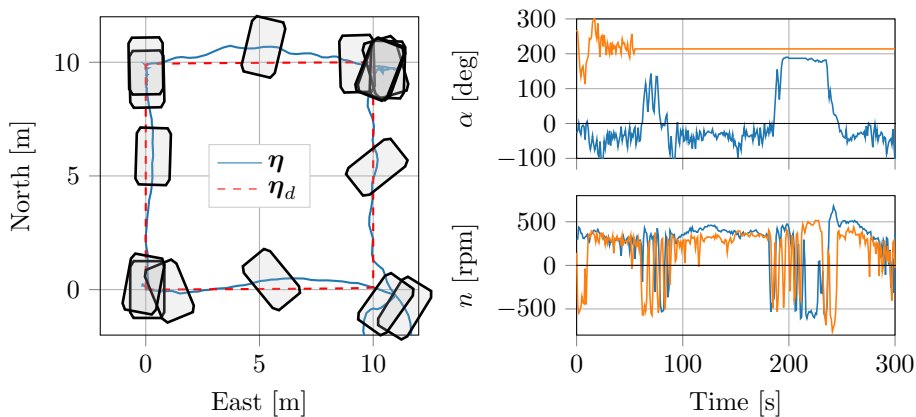


Figure 11: Sea trial results for *milliAmpere*, with azimuth failure at 50 seconds

- [3] Bjørn-Olav Holtung Eriksen. “Collision Avoidance and Motion Control for Autonomous Surface Vehicles”. PhD thesis. Trondheim: Norwegian University of Science and Technology (NTNU), 2019.
- [4] Thor I Fossen and Aslaug Grovlen. “Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping”. In: *IEEE transactions on control systems technology* 6.1 (1998), pp. 121–128.
- [5] Kristin Y Pettersen and Henk Nijmeijer. “Underactuated ship tracking control: theory and experiments”. In: *International Journal of Control* 74.14 (2001), pp. 1435–1446.
- [6] Roger Skjetne and Thor I Fossen. “Nonlinear maneuvering and control of ships”. In: *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No. 01CH37295)*. Vol. 3. IEEE, 2001, pp. 1808–1815.
- [7] Erjen Lefeber, Kristin Ytterstad Pettersen, and Henk Nijmeijer. “Tracking control of an underactuated ship”. In: *IEEE transactions on control systems technology* 11.1 (2003), pp. 52–61.
- [8] Khac Duc Do, Zhong-Ping Jiang, and Jie Pan. “Robust adaptive path following of underactuated ships”. In: *Automatica* 40.6 (2004), pp. 929–944.
- [9] A Pedro Aguiar and António M Pascoal. “Dynamic positioning and way-point tracking of underactuated AUVs in the presence of ocean currents”. In: *International Journal of Control* 80.7 (2007), pp. 1092–1108.
- [10] Massimo Caccia et al. “Basic navigation, guidance and control of an unmanned surface vehicle”. In: *Autonomous Robots* 25.4 (2008), pp. 349–365.
- [11] Marco Bibuli et al. “Path-following algorithms and experiments for an unmanned surface vehicle”. In: *Journal of Field Robotics* 26.8 (2009), pp. 669–688.
- [12] Yaseen Adnan Ahmed and Kazuhiko Hasegawa. “Fuzzy reasoned waypoint controller for automatic ship guidance”. In: *IFAC-PapersOnLine* 49.23 (2016), pp. 604–609.
- [13] Wei-Yuan Hwang. “Application of system identification to ship maneuvering”. PhD thesis. Massachusetts Institute of Technology, 1980.
- [14] Roland S Burns. “The use of artificial neural networks for the intelligent optimal control of surface ships”. In: *IEEE Journal of Oceanic Engineering* 20.1 (1995), pp. 65–72.
- [15] A Pedro Aguiar and Joao P Hespanha. “Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty”. In: *IEEE transactions on automatic control* 52.8 (2007), pp. 1362–1379.
- [16] Gollavelli Rajesh and Subrata K Bhattacharyya. “System identification for nonlinear maneuvering of large tankers using artificial neural network”. In: *Applied Ocean Research* 30.4 (2008), pp. 256–263.

Publications

- [17] Ning Wang et al. “Adaptive robust online constructive fuzzy control of a complex surface vehicle system”. In: *IEEE Transactions on Cybernetics* 46.7 (2015), pp. 1511–1523.
- [18] Wilmer Ariza Ramirez et al. “Non-parametric dynamic system identification of ships using multi-output Gaussian Processes”. In: *Ocean Engineering* 166 (2018), pp. 26–36.
- [19] Andreas Bell Martinsen et al. “Reinforcement learning-based tracking control of USVs in varying operational conditions”. In: *Frontiers in Robotics and AI* 7 (2020), p. 32.
- [20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [21] Dimitri P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.
- [22] Rushikesh Kamalapurkar et al. *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Springer, 2018.
- [23] Andreas B. Martinsen and Anastasios M. Lekkas. “Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models”. In: *OCEANS 2018 MTS/IEEE Charleston*. IEEE, 2018, pp. 1–8.
- [24] Andreas B Martinsen and Anastasios M Lekkas. “Straight-Path Following for Underactuated Marine Vessels using Deep Reinforcement Learning”. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 329–334.
- [25] Eivind Meyer et al. “COLREG-Compliant Collision Avoidance for Unmanned Surface Vehicle Using Deep Reinforcement Learning”. In: *IEEE Access* 8 (2020), pp. 165344–165364.
- [26] Ning Wang et al. “Reinforcement Learning-Based Optimal Tracking Control of an Unknown Unmanned Surface Vehicle”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).
- [27] Zhen Li and Jing Sun. “Disturbance Compensating Model Predictive Control With Application to Ship Heading Control”. In: *IEEE transactions on control systems technology* 20.1 (2011), pp. 257–265.
- [28] Huarong Zheng, Rudy R Negenborn, and Gabriel Lodewijks. “Trajectory tracking of autonomous vessels using model predictive control”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 8812–8818.
- [29] Chenguang Liu et al. “Trajectory tracking control for underactuated surface vessels based on nonlinear Model Predictive Control”. In: *International Conference on Computational Logistics*. Springer, 2015, pp. 166–180.
- [30] Aleksander Veksler et al. “Dynamic positioning with model predictive control”. In: *IEEE Transactions on Control Systems Technology* 24.4 (2016), pp. 1340–1353.
- [31] David Q Mayne et al. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814.

- [32] James B Rawlings and Rishi Amrit. “Optimizing Process Economic Performance Using Model Predictive Control”. In: *Nonlinear model predictive control*. Springer, 2009, pp. 119–138.
- [33] Sébastien Gros and Mario Zanon. “Data-driven economic NMPC using reinforcement learning”. In: *IEEE Transactions on Automatic Control* 65.2 (2019), pp. 636–648.
- [34] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Combining system identification with reinforcement learning-based MPC”. In: *arXiv preprint arXiv:2004.03265* (2020).
- [35] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.
- [36] Tor A Johansen and Thor I Fossen. “Control allocation—A survey”. In: *Automatica* 49.5 (2013), pp. 1087–1103.
- [37] Sébastien Gros and Mario Zanon. “Penalty Functions for Handling Large Deviation of Quadrature States in NMPC”. In: *IEEE Transactions on Automatic Control* 62.8 (2017), pp. 3848–3860.
- [38] Sébastien Gros and Moritz Diehl. “NMPC based on Huber Penalty Functions to Handle Large Deviations of Quadrature States”. In: *2013 American Control Conference*. IEEE, 2013, pp. 3159–3164.
- [39] Mario Zanon and Sébastien Gros. “Safe reinforcement learning using robust MPC”. In: *IEEE Transactions on Automatic Control* (2020).
- [40] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. PhD thesis. King’s College, Cambridge, 1989.
- [41] W-H Chen, Donald J Ballance, and John O’Reilly. “Model predictive control of nonlinear systems: computational burden and stability”. In: *IEE Proceedings-Control Theory and Applications* 147.4 (2000), pp. 387–394.
- [42] Rolf Findeisen and Frank Allgöwer. “Computational delay in nonlinear model predictive control”. In: *IFAC Proceedings Volumes* 37.1 (2004), pp. 427–432.
- [43] Moritz Diehl, Hans Georg Bock, and Johannes P Schlöder. “A real-time iteration scheme for nonlinear optimization in optimal feedback control”. In: *SIAM Journal on control and optimization* 43.5 (2005), pp. 1714–1736.
- [44] Victor M Zavala and Lorenz T Biegler. “The advanced-step NMPC controller: Optimality, stability and robustness”. In: *Automatica* 45.1 (2009), pp. 86–93.
- [45] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [46] Tor Arne Johansen, Thor I Fossen, and Stig P Berge. “Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming”. In: *IEEE Transactions on Control Systems Technology* 12.1 (2004), pp. 211–216.

Publications

- [47] Tobias R. Torben, Astrid H. Brodtkorb, and Asgeir J. Sørensen. “Control allocation for double-ended ferries with full-scale experimental results”. In: *Proceedings of the 12th IFAC CAMS, Daejeon, South Korea*. 2019, pp. 45–50.

References

- [1] John Gray Landels. *Engineering in the Ancient World*. Univ of California Press, 2000.
- [2] James Clerk Maxwell. “I. On governors”. In: *Proceedings of the Royal Society of London* 16 (1868), pp. 270–283. DOI: 10.1098/rsp1.1867.0055.
- [3] S Bennett. *A history of control engineering 1800-1930*. eng. Vol. 8. IEE control engineering series. London: Peter Peregrinus on behalf of the Institution of Electrical Engineers, 1979.
- [4] Nicolas Minorsky. “Directional stability of automatically steered bodies”. In: *Journal of the American Society for Naval Engineers* 34.2 (1922), pp. 280–309.
- [5] Stuart Bennett. “Nicholas Minorsky and the automatic steering of ships”. In: *IEEE Control Systems Magazine* 4.4 (1984), pp. 10–15. DOI: 10.1109/MCS.1984.1104827.
- [6] Rasheed Hussain and Sherali Zeadally. “Autonomous cars: Research results, issues, and future challenges”. In: *IEEE Communications Surveys & Tutorials* 21.2 (2018), pp. 1275–1313. DOI: 10.1109/COMST.2018.2869360.
- [7] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. “A survey of autonomous driving: Common practices and emerging technologies”. In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149.
- [8] Marius Thoresen, Solveig Bruvoll, and Martin Syre Wiig. *An automatic route planning service for unmanned surface vehicles*. Tech. rep. Norwegian Defence Research Establishment, 2019.

- [9] Rachel Courtland. *DARPA's Self-Driving Submarine Hunter Steers Like a Human*. 2016. URL: <https://spectrum.ieee.org/automan/robotics/military-robots/darpa-actuv-self-driving-submarine-hunter-steers-like-a-human> (visited on 03/03/2021).
- [10] Rolls-Royce. *Rolls-Royce and Finferries demonstrate world's first Fully Autonomous Ferry*. 2018. URL: <https://www.rolls-royce.com/media/press-releases/2018/03-12-2018-rr-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry.aspx> (visited on 03/03/2021).
- [11] Alexander Farnsworth. *Look, Ma, No Hands! Auto-docking ferry successfully tested in Norway*. 2018. URL: <https://www.wartsila.com/insights/article/look-ma-no-hands-auto-docking-ferry-successfully-tested-in-norway> (visited on 03/03/2021).
- [12] Kongsberg. *Automatic ferry enters regular service following world-first crossing with passengers onboard*. 2020. URL: <https://www.kongsberg.com/maritime/about-us/news-and-media/news-archive/2020/first-adaptive-transit-on-bastofosen-vi/> (visited on 03/03/2021).
- [13] DNV GL. *The ReVolt A new inspirational ship concept*. 2015. URL: <https://www.dnv.com/technology-innovation/revolt/index.html> (visited on 03/03/2021).
- [14] Yara. *The ReVolt A new inspirational ship concept*. 2017. URL: <https://www.yara.com/news-and-media/press-kits/yara-birkeland-press-kit/> (visited on 03/03/2021).
- [15] Kongsberg. *Kongsberg Maritime and Massterly to equip and operate two zero-emission autonomous vessels for ASKO*. 2020. URL: <https://www.kongsberg.com/maritime/about-us/news-and-media/news-archive/2020/zero-emission-autonomous-vessels/> (visited on 03/03/2021).
- [16] Unni Skoglund. *Førerløse ferger kan erstatte gangbruer*. 2018. URL: <https://gemini.no/2018/06/forerlose-ferger-kan-erstatte-gangbruer/> (visited on 03/03/2021).
- [17] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. "Autonomous docking using direct optimal control". In: *IFAC-PapersOnLine* 52.21 (2019), pp. 97–102. DOI: 10.1016/j.ifacol.2019.12.290.

-
- [18] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Combining system identification with reinforcement learning-based MPC”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 8130–8135. DOI: 10.1016/j.ifacol.2020.12.2294.
- [19] Glenn Bitar, Andreas B Martinsen, Anastasios M Lekkas, and Morten Breivik. “Trajectory Planning and Control for Automatic Docking of ASVs with Full-Scale Experiments”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 14488–14494. DOI: 10.1016/j.ifacol.2020.12.1451.
- [20] Andreas Bell Martinsen, Anastasios Lekkas, Sébastien Gros, Jon Arne Glomsrud, and Tom Arne Pedersen. “Reinforcement learning-based tracking control of USVs in varying operational conditions”. In: *Frontiers in Robotics and AI* 7 (2020), p. 32. DOI: 10.3389/frobt.2020.00032.
- [21] Glenn Bitar, Andreas B Martinsen, Anastasios M Lekkas, and Morten 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.
- [22] Andreas B Martinsen, Glenn Bitar, Anastasios M Lekkas, and Sébastien 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.
- [23] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Optimal Model-Based Trajectory Planning With Static Polygonal Constraints”. In: *IEEE Transactions on Control Systems Technology* 29.5 (2021). DOI: 10.1109/TCST.2021.3094617.
- [24] Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. “Reinforcement Learning-based MPC for Tracking Control of ASVs: Theory and Experiments”. In: *Review* (2021).
- [25] Andreas B Martinsen and Anastasios M Lekkas. “Two space-time obstacle representations based on ellipsoids and polytopes”. In: *IEEE Access* 9 (2021). DOI: 10.1109/ACCESS.2021.3103323.
- [26] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

- [27] Bjørn-Olav Holtung Eriksen. “Collision Avoidance and Motion Control for Autonomous Surface Vehicles”. PhD thesis. Trondheim: Norwegian University of Science and Technology (NTNU), 2019. URL: <http://hdl.handle.net/11250/2616394>.
- [28] Glenn Bitar. “Optimization-based Trajectory Planning and Automatic Docking for Autonomous Ferries”. PhD thesis. Trondheim: Norwegian University of Science and Technology (NTNU), 2021. URL: <https://hdl.handle.net/11250/2732513>.
- [29] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
- [30] Artur Wolek and Craig A Woolsey. “Model-based path planning”. In: *Sensing and Control for Autonomous Vehicles*. Springer, 2017, pp. 183–206. DOI: 10.1007/978-3-319-55372-6_9.
- [31] Glenn Bitar, Morten Breivik, and Anastasios M Lekkas. “Energy-Optimized Path Planning for Autonomous Ferries”. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 389–394. DOI: 10.1016/j.ifacol.2018.09.456.
- [32] Edsger W Dijkstra et al. “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1 (1959), pp. 269–271.
- [33] Peter E Hart, Nils J Nilsson, and Bertram 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.
- [34] Marcelo Kallmann. “Path Planning in Triangulations”. In: *Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games*. 2005, pp. 49–54.
- [35] Mauro Candeloro, Anastasios M Lekkas, and Asgeir 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.
- [36] Jerome Barraquand, Bruno Langlois, and J-C Latombe. “Numerical potential field techniques for robot path planning”. In: *IEEE transactions on systems, man, and cybernetics* 22.2 (1992), pp. 224–241. DOI: 10.1109/ICAR.1991.240539.
- [37] Anastasios M Lekkas, Andreas Reason Dahl, Morten Breivik, and Thor I Fossen. “Continuous-Curvature Path Generation Using Fermat’s Spiral”. In: *Modeling, Identification and Control* 34.4 (2013), p. 183. DOI: 10.4173/mic.2013.4.3.

-
- [38] Paul Jacobs and John Canny. “Planning Smooth Paths for Mobile Robots”. In: *Nonholonomic Motion Planning*. Springer, 1993, pp. 271–342. DOI: 10.1007/978-1-4615-3176-0_8.
- [39] Sara Fleury, Philippe Soueres, J-P Laumond, and Raja Chatila. “Primitives for smoothing mobile robot trajectories”. In: *IEEE transactions on robotics and automation* 11.3 (1995), pp. 441–448. DOI: 10.1109/70.388788.
- [40] Kevin Judd and Timothy McLain. “Spline based path planning for unmanned air vehicles”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2001, p. 4238. DOI: 10.2514/6.2001-4238.
- [41] Jia Pan, Liangjun Zhang, Dinesh Manocha, and UC Hill. “Collision-Free and Curvature-Continuous Path Smoothing in Cluttered Environments”. In: *Robotics: Science and Systems VII* 17 (2012), p. 233.
- [42] Anastasios M Lekkas and Thor I Fossen. “Integral LOS Path Following for Curved Paths Based on a Monotone Cubic Hermite Spline Parametrization”. In: *IEEE Transactions on Control Systems Technology* 22.6 (2014), pp. 2287–2301. DOI: 10.1109/TCST.2014.2306774.
- [43] Carlo L Bottasso, Domenico Leonello, and Barbara Savini. “Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives”. In: *IEEE Transactions on Control Systems Technology* 16.6 (2008), pp. 1152–1168. DOI: 10.1109/TCST.2008.917870.
- [44] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark 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.
- [45] Steven M LaValle. “Rapidly-Exploring Random Trees: A New Tool for Path Planning”. In: (1998).
- [46] Steven M LaValle and James J Kuffner Jr. “Randomized Kinodynamic Planning”. In: *The international journal of robotics research* 20.5 (2001), pp. 378–400. DOI: 10.1177/02783640122067453.
- [47] S. Karaman and E. Frazzoli. “Incremental Sampling-based Algorithms for Optimal Motion Planning”. In: *Proceedings of Robotics: Science and Systems*. 2010. DOI: 10.15607/RSS.2010.VI.034.
- [48] Stefano Carpin and Gianluigi Pillonetto. “Motion planning using adaptive random walks”. In: *IEEE Transactions on Robotics* 21.1 (2005), pp. 129–136. DOI: 10.1109/TR0.2004.833790.

- [49] Stefano Carpin and Gianluigi Pillonetto. “Merging the adaptive random walks planner with the randomized potential field planner”. In: *Proceedings of the Fifth International Workshop on Robot Motion and Control, 2005. RoMoCo'05*. IEEE, 2005, pp. 151–156. DOI: 10.1109/ROMOCO.2005.201416.
- [50] Lester 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), pp. 497–516. DOI: 10.2307/2372560.
- [51] James Reeds and Lawrence Shepp. “Optimal paths for a car that goes both forwards and backwards”. In: *Pacific journal of mathematics* 145.2 (1990), pp. 367–393.
- [52] Russell Eberhart and James Kennedy. “Particle swarm optimization”. In: *Proceedings of the IEEE international conference on neural networks*. Vol. 4. 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.
- [53] James Kennedy and Russell C Eberhart. “A discrete binary version of the particle swarm algorithm”. In: *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*. Vol. 5. IEEE, 1997, pp. 4104–4108. DOI: 10.1109/ICSMC.1997.637339.
- [54] Hans Georg Bock and Karl-Josef Plitt. “A multiple shooting algorithm for direct solution of optimal control problems”. In: *IFAC Proceedings Volumes* 17.2 (1984), pp. 1603–1608. DOI: 10.1016/S1474-6670(17)61205-9.
- [55] Charles R Hargraves and Stephen W Paris. “Direct trajectory optimization using nonlinear programming and collocation”. In: *Journal of guidance, control, and dynamics* 10.4 (1987), pp. 338–342. DOI: 10.2514/3.20223.
- [56] Fariba Fahroo and I Michael Ross. “Direct Trajectory Optimization by a Chebyshev Pseudospectral Method”. In: *Journal of Guidance, Control, and Dynamics* 25.1 (2002), pp. 160–166. DOI: 10.2514/2.4862.
- [57] Glenn Bitar, Vegard N Vestad, Anastasios M Lekkas, and Morten Breivik. “Warm-Started Optimized Trajectory Planning for ASVs”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 308–314. DOI: 10.1016/j.ifacol.2019.12.325.

-
- [58] Kristoffer Bergman, Oskar Ljungqvist, Jonas Linder, and Daniel Axelhill. “An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 5283–5290. DOI: 10.1109/CDC42340.2020.9303746.
- [59] J. D. Luse. “Collision avoidance systems and the rules of the nautical road”. In: *NAVIGATION, Journal of the Institute of Navigation* 19.1 (1972), pp. 80–88. DOI: 10.1002/j.2161-4296.1972.tb00129.x.
- [60] Lou Tychonievich et al. “A Maneuvering-Board Approach to Path Planning with Moving Obstacles”. In: *Proceedings of the 11th international joint conference on Artificial intelligence-Volume 2*. 1989, pp. 1017–1021.
- [61] Paolo Fiorini and Zvi Shiller. “Motion planning in dynamic environments using the relative velocity paradigm”. In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 560–565. DOI: 10.1109/ROBOT.1993.292038.
- [62] Animesh Chakravarthy and Debasish Ghose. “Obstacle avoidance in a dynamic environment: A collision cone approach”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 28.5 (1998), pp. 562–574. DOI: 10.1109/3468.709600.
- [63] Boris Kluge and Erwin Prassler. “Reflective navigation: Individual behaviors and group behaviors”. In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 4. IEEE, 2004, pp. 4172–4177. DOI: 10.1109/ROBOT.2004.1308926.
- [64] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. “Generalized velocity obstacles”. In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 5573–5578. DOI: 10.1109/IRoS.2009.5354175.
- [65] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. “The hybrid reciprocal velocity obstacle”. In: *IEEE Transactions on Robotics* 27.4 (2011), pp. 696–706. DOI: 10.1109/TR0.2011.2120810.
- [66] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. “The dynamic window approach to collision avoidance”. In: *IEEE Robotics & Automation Magazine* 4.1 (1997), pp. 23–33. DOI: 10.1109/100.580977.
- [67] Oussama Khatib. “Real-time obstacle avoidance for manipulators and mobile robots”. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

- [68] Michael R Benjamin and Joseph A Curcio. “COLREGS-based navigation of autonomous marine vehicles”. In: *2004 IEEE/OES Autonomous Underwater Vehicles*. IEEE. 2004, pp. 32–39. DOI: 10.1109/AUV.2004.1431190.
- [69] Signe Moe and Kristin Y Pettersen. “Set-based Line-of-Sight (LOS) path following with collision avoidance for underactuated unmanned surface vessel”. In: *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE. 2016, pp. 402–409. DOI: 10.1109/MED.2016.7535964.
- [70] Emil H Thyri et al. “Reactive collision avoidance for ASVs based on control barrier functions”. In: *2020 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE. 2020, pp. 380–387. DOI: 10.1109/CCTA41146.2020.9206340.
- [71] AW Merz and JS Karmarkar. “Collision Avoidance Systems and Optimal Turn Manoeuvres”. In: *The Journal of Navigation* 29.2 (1976), pp. 160–174. DOI: 10.1017/S0373463300030150.
- [72] Cornelis de Wit and J Oppe. “Optimal Collision Avoidance in Unconfined Waters”. In: *Navigation* 26.4 (1979), pp. 296–303. DOI: 10.1002/j.2161-4296.1979.tb01389.x.
- [73] Y Yavin, C Frangos, and T Miloh. “Computation of feasible control trajectories for the navigation of a ship around an obstacle in the presence of a sea current”. In: *Mathematical and computer modelling* 21.3 (1995), pp. 99–117. DOI: 10.1016/0895-7177(94)00218-D.
- [74] A Miele, T Wang, CS Chao, and JB Dabney. “Optimal Control of a Ship for Collision Avoidance Maneuvers”. In: *Journal of optimization theory and applications* 103.3 (1999), pp. 495–519. DOI: 10.1023/A:1021775722287.
- [75] Hao-Tien Lewis Chiang and Lydia Tapia. “COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2024–2031. DOI: 10.1109/LRA.2018.2801881.
- [76] Bjørn-Olav H Eriksen, Morten Breivik, Erik F Wilthil, Andreas L Flåten, and Edmund 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.

- [77] Kazuhiko Hasegawa and T Fukutomi. “On Harbour Manoeuvring and Neural Control System for Berthing with Tug Operation”. In: *Proc. 3rd International Conference on Manoeuvring and Control of Marine Craft*. 1994, pp. 197–210.
- [78] Hideki Kawai Van Phuoc Bui, Young Bok Kim, and Kwon Soon Lee. “A ship berthing system design with four tug boats”. In: *Journal of Mechanical Science and Technology* 25.5 (2011), pp. 1257–1264. DOI: 10.1007/s12206-011-0215-4.
- [79] Phuoc Van Bui and Young Bok Kim. “Development of Constrained Control Allocation for Ship Berthing by Using Autonomous Tugboats”. In: *International Journal of Control, Automation and Systems* 9.6 (2011), pp. 1203–1208. DOI: 10.1007/s12555-011-0622-4.
- [80] Van Luong Tran and Namkyun 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.
- [81] Joohyun Woo and Nakwan Kim. “Vector Field based Guidance Method for Docking of an Unmanned Surface Vehicle”. In: *The Twelfth ISOPE Pacific/Asia Offshore Mechanics Symposium*. International Society of Offshore and Polar Engineers. 2016.
- [82] 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: *1993 American Control Conference*. IEEE, 1993, pp. 580–584. DOI: 10.23919/ACC.1993.4792923.
- [83] Ken Teo, Benjamin Goh, and Oh Kwee 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.
- [84] Van-Suong Nguyen and Nam-Kyun 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.
- [85] Hiroyuki Yamato. “Automatic Berthing by the Neural Controller”. In: *Proc. Of Ninth Ship Control Systems Symposium*. Vol. 3. 1990, pp. 3183–3201.
- [86] K Hasegawa. “Automatic Berthing Control System Using Network and Knowledgebase”. In: *Journal of the Society of Naval Architects of Japan* 220 (1993), pp. 135–143.

- [87] Yao Zhang, Grant E Hearn, and Pratyush Sen. “A Multivariable Neural Controller for Automatic Ship Berthing”. In: *IEEE Control Systems Magazine* 17.4 (1997), pp. 31–45. DOI: 10.1109/37.608535.
- [88] Namkyun Im and Kazuhiko Hasegawa. “A Study on Automatic Ship Berthing Using Parallel Neural Controller”. In: *Journal of the Kansai Society of Naval Architects, Japan* 2001.236 (2001), pp. 65–70.
- [89] Yaseen Adnan Ahmed and Kazuhiko 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.
- [90] Nam-Kyun Im and Van-Suong Nguyen. “Artificial neural network controller for automatic ship berthing using head-up coordinate system”. In: *International Journal of Naval Architecture and Ocean Engineering* 10.3 (2018), pp. 235–249. DOI: 10.1016/j.ijnaoe.2017.08.003.
- [91] Van-Cuong Do Van-Suong Nguyen and Nam-Kyun Im. “Development of Automatic Ship Berthing System Using Artificial Neural Network and Distance Measurement System”. In: *International journal of Fuzzy logic and Intelligent systems* 18.1 (2018), pp. 41–49. DOI: 10.5391/IJFIS.2018.18.1.41.
- [92] Daesoo Lee, Seung-Jae Lee, and Seo Yu-Jeong. “Application of Recent Developments in Deep Learning to ANN-based Automatic Berthing Systems”. In: *International Journal of Engineering and Technology Innovation* 10.1 (2019), p. 75. DOI: 10.46604/ijeti.2020.4354.
- [93] Naoki Mizuno and Ryo Kuboshima. “Implementation and Evaluation of Non-linear Optimal Feedback Control for Ship’s Automatic Berthing by Recurrent Neural Network”. In: *IFAC-PapersOnLine* 52.21 (2019), pp. 91–96. DOI: 10.1016/j.ifacol.2019.12.289.
- [94] Ella-Lovise Hammervold Rørvik. “Automatic Docking of an Autonomous Surface Vessel”. MA thesis. Trondheim, Norway: Norwegian University of Science and Technology (NTNU), 2020. URL: <https://hdl.handle.net/11250/2656724>.
- [95] Hiroyuki Yamato, Takeo Koyama, and Takehiko Nakagawa. “Automatic Berthing Using the Expert System”. In: *IFAC Proceedings Volumes* 25.3 (1992), pp. 173–184. DOI: 10.1016/S1474-6670(17)50288-8.

-
- [96] Kouichi Shouji, Kohei Ohtsu, and Sumitoshi Mizoguchi. “An Automatic Berthing Study by Optimal Control Techniques”. In: *IFAC Proceedings Volumes* 25.3 (1992), pp. 185–194. DOI: 10.1016/S1474-6670(17)50289-X.
- [97] Karim Djouani and Yskandar 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.
- [98] K Ohtsu, K Shoji, and T Okazaki. “Minimum-time maneuvering of a ship, with wind disturbances”. In: *Control Engineering Practice* 4.3 (1996), pp. 385–392. DOI: 10.1016/0967-0661(96)00016-0.
- [99] Tadatsugi Okazaki, Kohei Ohtsu, and Naoki Mizuno. “A Study of Minimum Time Berthing Solutions”. In: *IFAC Proceedings Volumes* 33.21 (2000), pp. 135–139. DOI: 10.1016/S1474-6670(17)37064-7.
- [100] Naoki Mizuno, Yosuke Uchida, and Tadatsugi Okazaki. “Quasi Real-Time Optimal Control Scheme for Automatic Berthing”. In: *IFAC-PapersOnLine* 48.16 (2015), pp. 305–312. DOI: 10.1016/j.ifacol.2015.10.297.
- [101] Mikkel Cornelius Nielsen, Tor Arne Johansen, and Mogens Blanke. “Cooperative Rendezvous and Docking for Underwater Robots using Model Predictive Control and Dual Decomposition”. In: *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 14–19. DOI: 10.23919/ECC.2018.8550366.
- [102] N Mizuno, T Kita, and T Ishikawa. “A New Solving Method for Non-Linear Optimal Control Problem and Its Application to Automatic Berthing Problem”. In: *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 2183–2188. DOI: 10.1109/IECON.2018.8592735.
- [103] Shijie Li, Jialun Liu, Rudy R Negenborn, and Qing 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.
- [104] Tadao Takai and Kohei Ohtsu. “Automatic Berthing Experiments Using "Shioji-Maru"”. In: *The Journal of Japan Institute of Navigation* 83 (1990), pp. 267–276. DOI: 10.9749/jin.83.267.

- [105] L. Gucma et al. “Laser docking system integrated with pilot navigation support system. Background to high precision, fast and reliable vessel docking”. In: *17th Saint Petersburg International Conference on Integrated Navigation Systems, ICINS 2010 - Proceedings* (2010), pp. 268–275.
- [106] Marko Perkovic, Maciej Gucma, Blaz Luin, Lucjan Gucma, and Tanja Brcko. “Accommodating larger container vessels using an integrated laser system for approach and berthing”. In: *Microprocessors and Microsystems* 52 (2017), pp. 106–116. DOI: 10.1016/j.micpro.2017.05.015.
- [107] Leonard I Schiff and M Grimprich. “Automatic steering of ships by proportional control”. In: *Society of Naval Architects and Marine Engineers-Transactions* 57 (1949).
- [108] Kensaku Nomoto, Kenshi Taguchi, Keinosuke Honda, and Susumu Hirano. “On the steering qualities of ships”. In: *Journal of Zosen Kiokai* 1956.99 (1956), pp. 75–82. DOI: 10.2534/jjasnaoe1952.1956.99_75.
- [109] Takeo Koyama. “On the Optimum Automatic Steeing System of Ships at Sea”. In: *Journal of Zosen Kiokai* 1967.122 (1967), pp. 18–35. DOI: 10.2534/jjasnaoe1952.1967.122_18.
- [110] N.H. Norrbin. “On the Added Resistance due to Steering on a Straight Course”. In: *Proceedings of the 13th ITTC, Berlin, Hamburg, Germany*. 1972.
- [111] J Van Amerongen and H.R. van Nauta Lemke. *Optimum Steering of Ships with an Adaptive Autopilot*. Tech. rep. Delft University of Technology, 1978.
- [112] Claes G Källström, Karl Johan Åström, NE Thorell, J Eriksson, and L Sten. “Adaptive autopilots for tankers”. In: *Automatica* 15.3 (1979), pp. 241–254. DOI: 10.1016/0005-1098(79)90042-6.
- [113] Job Van Amerongen. “Adaptive steering of ships—A model reference approach”. In: *Automatica* 20.1 (1984), pp. 3–14. DOI: 10.1016/0005-1098(84)90060-8.
- [114] T Holzhüter and H Strauch. “A commercial adaptive autopilot for ships: design and operational experiences”. In: *IFAC Proceedings Volumes* 20.5 (1987), pp. 233–238. DOI: 10.1016/S1474-6670(17)55207-6.
- [115] J Van Amerongen and H.R. van Nauta Lemke. “Criteria for Optimum Steering of Ships”. In: *Proceedings of the Symposium on Ship Steering Automatic Control, Genoa, Italy*. 1980.

- [116] DC Donha, DS Desanj, MR Katebi, and MJ Grimble. “H- ∞ adaptive controllers for auto-pilot applications”. In: *International journal of adaptive control and signal processing* 12.8 (1998), pp. 623–648. DOI: 10.1002/(SICI)1099-1115(199812)12:8<623::AID-ACS517>3.0.CO;2-2.
- [117] Kazuhiko Hasegawa, Akihiko Kouzuki, Tooru Muramatsu, Hirofumi Komine, and Yuuji Watabe. “Ship Auto-navigation Fuzzy Expert System (SAFES)”. In: *Journal of the Society of Naval Architects of Japan* 1989.166 (1989), pp. 445–452. DOI: 10.2534/jjasnaoe1968.1989.166_445.
- [118] Thor I. Fossen and Marit Paulsen. “Adaptive feedback linearization applied to steering of ships”. In: *Modeling, Identification and Control* 14.4 (1993), pp. 229–237. DOI: 10.4173/mic.1993.4.4.
- [119] R Sutton, GN Roberts, and SDH Taylor. “Tuning fuzzy ship autopilots using artificial neural networks”. In: *Transactions of the Institute of Measurement and Control* 19.2 (1997), pp. 94–106. DOI: 10.1177/014233129701900204.
- [120] Thor I. Fossen and Jan P. Strand. “Tutorial on nonlinear backstepping: Applications to ship control”. In: *Modeling, Identification and Control* 20.2 (1999), pp. 83–135. DOI: 10.4173/mic.1999.2.3.
- [121] C.Y. Tzeng. “Optimal Control of a Ship for a Course-Changing Maneuver”. In: *Journal of Optimization Theory and Applications* 97.2 (1998), pp. 281–297. DOI: 10.1023/A:1022674516570.
- [122] Zhen Li and Jing Sun. “Disturbance Compensating Model Predictive Control With Application to Ship Heading Control”. In: *IEEE transactions on control systems technology* 20.1 (2011), pp. 257–265. DOI: 10.1109/TCST.2011.2106212.
- [123] Jacques Harbonn et al. “The Terebel Dynamic Positioning System-Results of Five Years of Field Work and Experiments”. In: *Offshore Technology Conference*. Offshore Technology Conference. 1971. DOI: 10.4043/1499-MS.
- [124] John S Sargent and Paul N Cowgill. “Design Considerations for Dynamically Positioned Utility Vessels”. In: *Offshore Technology Conference*. Offshore Technology Conference. 1976. DOI: 10.4043/2633-MS.
- [125] JG Balchen, NA Jenssen, and S Sælid. “Dynamic positioning using Kalman filtering and optimal control theory”. In: *IFAC/IFIP symposium on automation in offshore oil field operation, Bergen, Norway*. North-Holland Publishing Company, 1976, pp. 183–186.

- [126] Jens G Balchen, Nils A Jenssen, Eldar Mathisen, and Steinar Saelid. “Dynamic positioning of floating vessels based on Kalman filtering and optimal control”. In: *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*. IEEE, 1980, pp. 852–864. DOI: 10.1109/CDC.1980.271924.
- [127] MJ Grimble, RJ Patton, and DA Wise. “The design of dynamic ship positioning control systems using stochastic optimal control theory”. In: *Optimal Control Applications and Methods 1.2* (1980), pp. 167–202. DOI: 10.1002/oca.4660010207.
- [128] Asgeir J Sørensen, Svein I Sagatun, and Thor I Fossen. “Design of a Dynamic Positioning System Using Model-Based Control”. In: *IFAC Proceedings Volumes 28.2* (1995), pp. 16–26. DOI: 10.1016/S1474-6670(17)51646-8.
- [129] M.R. Katebi, M.J. Grimble, and Y. Zhang. “H-∞ robust control design for dynamic ship positioning”. In: *IEE Proceedings-Control Theory and Applications 144.2* (1997), pp. 110–120. DOI: 10.1049/ip-cta:19971030.
- [130] Thor I Fossen and Svein P Berge. “Nonlinear vectorial backstepping design for global exponential tracking of marine vessels in the presence of actuator dynamics”. In: *Proceedings of the 36th IEEE Conference on Decision and Control*. Vol. 5. IEEE. 1997, pp. 4237–4242. DOI: 10.1109/CDC.1997.649499.
- [131] Thor I Fossen and Aslaug Grovlen. “Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping”. In: *IEEE transactions on control systems technology 6.1* (1998), pp. 121–128. DOI: 10.1109/87.654882.
- [132] Kristin Y Pettersen and Henk Nijmeijer. “Underactuated ship tracking control: theory and experiments”. In: *International Journal of Control 74.14* (2001), pp. 1435–1446. DOI: 10.1080/00207170110072039.
- [133] J-M Godhavn, Thor I Fossen, and Svein P Berge. “Non-linear and adaptive backstepping designs for tracking control of ships”. In: *International Journal of Adaptive Control and Signal Processing 12.8* (1998), pp. 649–670. DOI: 10.1002/(SICI)1099-1115(199812)12:8<649::AID-ACS515>3.0.CO;2-P.
- [134] Aleksander Veksler, Tor Arne Johansen, Francesco Borrelli, and Bjørnar Realfsen. “Dynamic positioning with model predictive control”. In: *IEEE Transactions on Control Systems Technology 24.4* (2016), pp. 1340–1353. DOI: 10.1109/TCST.2015.2497280.

-
- [135] Rushikesh Kamalapurkar, Patrick Walters, Joel Rosenfeld, and Warren Dixon. *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Springer, 2018. DOI: 10.1007/978-3-319-78384-0.
- [136] T Holzhüter. “A High Precision Track Controller for Ships”. In: *IFAC Proceedings Volumes* 23.8 (1990), pp. 425–430. DOI: 10.1016/S1474-6670(17)51861-3.
- [137] Anthony J Healey, DB Marco, et al. “Slow Speed Flight Control Of Autonomous Underwater Vehicles: Experimental Results With NPS AUV II”. In: *Proc. ISOPE*. 1992, pp. 523–532.
- [138] Anthony J Healey and David Lienard. “Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles”. In: *IEEE journal of Oceanic Engineering* 18.3 (1993), pp. 327–339. DOI: 10.1109/JOE.1993.236372.
- [139] Roland S Burns. “The use of artificial neural networks for the intelligent optimal control of surface ships”. In: *IEEE Journal of Oceanic Engineering* 20.1 (1995), pp. 65–72. DOI: 10.1109/48.380245.
- [140] J-M Godhavn. “Nonlinear tracking of underactuated surface vessels”. In: *Proceedings of 35th IEEE Conference on Decision and Control*. Vol. 1. IEEE, 1996, pp. 975–980. DOI: 10.1109/CDC.1996.574608.
- [141] Knut Eilif Husa and Thor I Fossen. “Backstepping Designs for Nonlinear Way-Point Tracking of Ships”. In: *IFAC Proceedings Volumes* 30.22 (1997), pp. 111–116. DOI: 10.1016/S1474-6670(17)46499-8.
- [142] Roger Skjetne and Thor I Fossen. “Nonlinear maneuvering and control of ships”. In: *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No. 01CH37295)*. Vol. 3. IEEE, 2001, pp. 1808–1815. DOI: 10.1109/OCEANS.2001.968121.
- [143] Svein P Berge, Kohei Ohtsu, and Thor I Fossen. “Nonlinear control of ships minimizing the position tracking errors”. In: *IFAC Proceedings Volumes* 31.30 (1998), pp. 129–134. DOI: 10.1016/S1474-6670(17)38429-X.
- [144] Zhong-Ping Jiang. “Global tracking control of underactuated ships by Lyapunov’s direct method”. In: *Automatica* 38.2 (2002), pp. 301–309. DOI: 10.1016/S0005-1098(01)00199-6.
- [145] Thor I Fossen, Morten Breivik, and Roger Skjetne. “Line-of-sight path following of underactuated marine craft”. In: *IFAC proceedings volumes* 36.21 (2003), pp. 211–216. DOI: 10.1016/S1474-6670(17)37809-6.

- [146] Khac Duc Do, Zhong-Ping Jiang, and Jie Pan. “Robust adaptive path following of underactuated ships”. In: *Automatica* 40.6 (2004), pp. 929–944. DOI: 10.1016/j.automatica.2004.01.021.
- [147] A Pedro Aguiar and Joao P Hespanha. “Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty”. In: *IEEE transactions on automatic control* 52.8 (2007), pp. 1362–1379. DOI: 10.1109/TAC.2007.902731.
- [148] Huarong Zheng, Rudy R Negenborn, and Gabriel Lodewijks. “Trajectory tracking of autonomous vessels using model predictive control”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 8812–8818. DOI: 10.3182/20140824-6-ZA-1003.00767.
- [149] Chenguang Liu, Huarong Zheng, Rudy R Negenborn, Xiumin Chu, and Le Wang. “Trajectory tracking control for underactuated surface vessels based on nonlinear Model Predictive Control”. In: *International Conference on Computational Logistics*. Springer, 2015, pp. 166–180. DOI: 10.1007/978-3-319-24264-4_12.
- [150] Ning Wang, Meng Joo Er, Jing-Chao Sun, and Yan-Cheng Liu. “Adaptive robust online constructive fuzzy control of a complex surface vehicle system”. In: *IEEE Transactions on Cybernetics* 46.7 (2015), pp. 1511–1523. DOI: 10.1109/TCYB.2015.2451116.
- [151] Yaseen Adnan Ahmed and Kazuhiko Hasegawa. “Fuzzy reasoned way-point controller for automatic ship guidance”. In: *IFAC-PapersOnLine* 49.23 (2016), pp. 604–609. DOI: 10.1016/j.ifacol.2016.10.501.
- [152] Ning Wang, Shun-Feng Su, Jianchuan Yin, Zhongjiu Zheng, and Meng Joo Er. “Global asymptotic model-free trajectory-independent tracking control of an uncertain marine vehicle: An adaptive universe-based fuzzy control approach”. In: *IEEE Transactions on Fuzzy Systems* 26.3 (2017), pp. 1613–1625. DOI: 10.1109/TFUZZ.2017.2737405.
- [153] Andreas Bell Martinsen. “End-to-end training for path following and control of marine vehicles”. MA thesis. Norwegian University of Science and Technology, 2018. URL: <http://hdl.handle.net/11250/2559484>.
- [154] Andreas B Martinsen and Anastasios M Lekkas. “Straight-path following for underactuated marine vessels using deep reinforcement learning”. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 329–334. DOI: 10.1016/j.ifacol.2018.09.502.

-
- [155] Andreas B Martinsen and Anastasios M Lekkas. “Curved path following with deep reinforcement learning: Results from three vessel models”. In: *OCEANS 2018 MTS/IEEE Charleston*. IEEE, 2018, pp. 1–8. DOI: 10.1109/OCEANS.2018.8604829.
- [156] Thor I Fossen. “A survey on Nonlinear Ship Control: from Theory to Practice”. In: *IFAC Proceedings Volumes* 33.21 (2000), pp. 1–16. DOI: 10.1016/S1474-6670(17)37044-1.
- [157] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Sokaert. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814. DOI: 10.1016/S0005-1098(99)00214-9.
- [158] James B Rawlings and Rishi Amrit. “Optimizing Process Economic Performance Using Model Predictive Control”. In: *Nonlinear model predictive control*. Springer, 2009, pp. 119–138. DOI: 10.1007/978-3-642-01094-1_10.
- [159] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [160] Dimitri P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.
- [161] Mauro Candeloro, Anastasios M Lekkas, Asgeir J Sørensen, and Thor I Fossen. “Continuous Curvature Path Planning using Voronoi diagrams and Fermat’s spirals”. In: *IFAC Proceedings Volumes* 46.33 (2013), pp. 132–137. DOI: 10.3182/20130918-4-JP-3022.00064.
- [162] Glenn Bitar, Bjørn-Olav H Eriksen, Anastasios M Lekkas, and Morten Breivik. “Energy-optimized hybrid collision avoidance for ASVs”. In: *18th European Control Conference (ECC)*. IEEE, 2019, pp. 2522–2529. DOI: 10.23919/ECC.2019.8795645.
- [163] Xiaojing Zhang, Alexander Liniger, Atsushi Sakai, and Francesco Borrelli. “Autonomous Parking Using Optimization-Based Collision Avoidance”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332. DOI: 10.1109/cdc.2018.8619433.
- [164] Oskar Ljungqvist, Niclas Evestedt, Marcello Cirillo, Daniel Axehill, and Olov Holmer. “Lattice-based motion planning for a general 2-trailer system”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 819–824. DOI: 10.1109/IVS.2017.7995817.

- [165] Lars Blackmore, Masahiro Ono, and Brian C Williams. “Chance-Constrained Optimal Path Planning With Obstacles”. In: *IEEE Transactions on Robotics* 27.6 (2011), pp. 1080–1094. DOI: 10.1109/TR0.2011.2161160.
- [166] Tobias Schoels et al. “CIAO*: MPC-based Safe Motion Planning in Predictable Dynamic Environments”. In: *IFAC-PapersOnLine* 53.2 (2020).
- [167] Hongyang Yan, Huifang Wang, Yangzhou Chen, and Guiping Dai. “Path planning based on Constrained Delaunay Triangulation”. In: *2008 7th World Congress on Intelligent Control and Automation*. IEEE, 2008, pp. 5168–5173. DOI: 10.1109/WCICA.2008.4593771.
- [168] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. “Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment”. In: *IEEE Transactions on Control Systems Technology* 26.6 (2017), pp. 2182–2189. DOI: 10.1109/TCST.2017.2739706.
- [169] Margarita V Sotnikova and Evgeny I Veremey. “Dynamic Positioning Based on Nonlinear MPC”. In: *IFAC Proceedings Volumes* 46.33 (2013), pp. 37–42. DOI: 10.3182/20130918-4-JP-3022.00058.
- [170] Tor Arne Johansen, Thor I Fossen, and Stig P Berge. “Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming”. In: *IEEE Transactions on Control Systems Technology* 12.1 (2004), pp. 211–216. DOI: 10.1109/TCST.2003.821952.
- [171] Patrick Walters, Rushikesh Kamalapurkar, Forrest Voight, Eric M Schwartz, and Warren E Dixon. “Online approximate optimal station keeping of a marine craft in the presence of an irrotational current”. In: *IEEE Transactions on Robotics* 34.2 (2018), pp. 486–496. DOI: 10.1109/TR0.2018.2791600.
- [172] Sébastien Gros and Mario Zanon. “Data-Driven Economic NMPC Using Reinforcement Learning”. In: *IEEE Transactions on Automatic Control* 65.2 (2019), pp. 636–648. DOI: 10.1109/TAC.2019.2913768.
- [173] Mario Zanon and Sébastien Gros. “Safe Reinforcement Learning Using Robust MPC”. In: *IEEE Transactions on Automatic Control* (2020). DOI: 10.1109/TAC.2020.3024161.

ISBN 978-82-326-6068-1 (printed ver.)
ISBN 978-82-326-5438-3 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)



NTNU

Norwegian University of
Science and Technology