Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Caroline Fleischer

Optimal Path-Planning on a Bio-Inspired Neural Network Guidance Model for Autonomous Surface Vessels

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
June 2020

NTNU
Norwegian University of
Science and Technology

Caroline Fleischer

Optimal Path-Planning on a Bio-Inspired Neural Network Guidance Model for Autonomous Surface Vessels

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
June 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER OF TECHNOLOGY THESIS DEFINITION (30 SP)

Name of the candidate:	Fleischer, Caroline Sophie Röhm
Field of study:	Marine control engineering
Thesis title (Norwegian):	Optimal baneplanlegging på et bio-inspirert nevralt nettverk guidance-modell for autonome skip
Thesis title (English):	Optimal path-planning on a bio-inspired neural network guidance model for autonomous surface vessels

Background

For autonomous ships, the ability to reactively plan and replan a path to avoid collision during maneuvering and voyaging is crucial. For this purpose, bio-inspired neural networks (BINN) are explored to serve as a guidance model for online and stepwise path-planning. These are discrete, topologically organized networks that contain information of "targets" and "obstacles" and are updated dynamically as the external environment changes. Typically, the optimal path-planning used in BINN applications are very simplistic. The objective of this thesis is to study optimal path-planning techniques that will improve the resulting path on a BINN model, in order for it to be feasible for a ship as well as minimizing certain cost. For this task, both model-predictive control (MPC) with mixed-integer programming (MIP) and the Hybrid A* search algorithm are relevant methods to be explored.

An autonomous system is to be developed for testing and verification of the path-planning algorithms through simulations and in MC-lab experiments – if available. In such a system, it is the responsibility of a cognitive machine pilot to sense and interpret the ambient conditions with high certainty, plan a path and speed, navigate, and execute the plan by maneuvering the ship from departure to destination without human intervention. This involves understanding of:

- The ship dynamics (inertial delays, responses to currents, wind, and propulsion, etc.),
- path feasibility requirements and safety maneuvers (crash stop, turning circle, etc.) for the ship,
- how to maneuver safely and optimally in waves and currents,
- in-voyage path and speed decision making to satisfy local and global objectives, and
- the sensors and monitoring variables for data analysis to ensure situational awareness.

Not all these aspects are objectives in this thesis. The following list details the work tasks:

Work description

1. Perform a background and literature review to provide information and relevant references on:
   - BINN as a dynamic guidance model.
   - Basic graph theory for efficient computations on a network of neurons.
   - Methods for optimal path-planning (MIP, MPC, Hybrid A*).
   - Dynamic obstacle avoidance compliant with relevant ship rules and regulations.
   - MC-Lab and the CyberShip Enterprise 1 (CSE1) model.

   Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the literature study and project assignment.

2. Work together with other students on preparing CSE1 for testing and experimentation in the MC-Lab. Formulate the experimental problem, including a description of setup, the vessel and its equipment, dynamic models, operation workspace, test scenarios, and specific assumptions.

3. Propose one or several optimal path-planning techniques (incl. Hybrid A* and/or MPC) to be used on a discrete BINN model. Explain the objective functions and constraints used in each method, and how these handle vessel constraints and act on the information provided by neural activities in the BINN. Specify a few interesting topographical landscapes, containing some obstacles, and find an optimal path from an initial position to one or several targets. Present the results.

4. Introduce relevant ship rules to the guidance model (e.g. COLREGs). Simulate the path-planning algorithms in different scenarios (i.e., an obstacle moving in different directions with different speeds) by using a receding horizon scheme (i.e., for each BINN evolution, implement only the next optimal waypoint, and repeat the process until the target is reached). Present the results.

5. Implement and present a path-generation algorithm (e.g., Bézier curves) that ensures a smooth (C^3) curve in the connection points. The next waypoint can be assumed to be determined by the path-planner at some point along the current path segment, with enough time to replan and compute next segment. Derive the heading and speed reference along the path.

6. Modify the guidance system by, for example, selecting 2, 3, or more waypoints forward for each run of the optimization, instead of only the “next waypoint”. For all cases, suggest and justify a common key performance indicator (KPI) to evaluate path performance with. Evaluate this for each path, and tabulate and discuss the results.

7. Set up an autonomous system, with your developed functions, for simulation and control of CSE1. System modules include the path-planner and path-generator, an observer, and a motion controller. Simulate some scenarios for the transit phase of a voyage on an appropriate simulation model, and thereafter test the system on CSE1 in the MC-Lab. Present the implementation and results.

Specifications

Every weekend throughout the project period, the candidate shall send a status email to the supervisor and co-advisors, providing two brief bulleted lists: 1) work done recent week, and 2) work planned to be done next week.

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, problem, design, results, and critical assessments. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 70 A4-pages, 100 B5-pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the elements: Title page, abstract, acknowledgement, project definition, list of symbols and acronyms, table of contents, introduction (project motivation, objectives, scope and delimitations), background/literature review, problem formulation, method, results, conclusions with recommendations for further work, references, and optional appendices. Figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct, which is taken very seriously by the university and cause consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis description shall be included after the title page. Computer code, pictures, videos, dataseries, etc., shall be included electronically with the report.

Start date:	15 January, 2020	**Due date:**	As specified by the administration.

Supervisor:	Roger Skjetne
Co-advisor(s):	Zhengru Ren, Mathias Marley, and Einar Ueland

Trondheim, 23.03.2020

Roger Skjetne, Supervisor

Abstract

Intelligent path-planning for surface vessels is crucial when enhancing their level of autonomy – or self-government. Path-planning is typically based on a guidance model, that replicates the operational environment according to a priori knowledge and perceptual information. Changes in the model prompt replanning of the path, which is intrinsic to collision avoidance and the execution of a safe voyage. A bio-inspired neural network proposed by Yang and Meng (2001) was explored to serve as the guidance model. The model seems promising in how information can be fused into a neural activity landscape. A downside is related to the steepness of the landscape, yielding minimal differences in the activities of neurons associated with free space, and thus increasing the difficulty of determining the better waypoints.

Waypoints were determined by a path search strategy. A mixed-integer program was formulated, but did not produce practicable outputs. Nevertheless, to promote system reactivity, the receding horizon approach was proceeded with and combined with a hybrid-state A* path search. Advantages of the hybrid A* algorithm include that it considers the continuous nature of the search space as well as nonholonomic constraints of the vehicle. Expansions of the search tree were performed either with straight lines of a predefined length or with arcs based on the vessel's minimum turning circle, and according to an heuristic estimate of the distance to the target. The latter was either calculated as the Euclidean distance, or a modification based on neural activities and with a penalty on changes in search direction. Straight-line expansions together with the former heuristics finished in the shortest time, but the latter heuristics generally yielded paths with less turns. Although any combination of heuristics and type of expansion finished within reasonable computation times, it might worsen in congested environments. Then, enhancing the heuristic estimate by, for instance, assigning it to the maximum of several admissible estimates is a possible way forward.

Predictable behavior of the autonomous surface vessel in encounter situations was sought by complying with the rules of the sea (COLREGs). A rectangular avoidance region, that adapted to the distance and bearing between the encountering vessels, was defined. The vessel under control was repelled from moving inside the region by manipulating neural activities of the neurons within the region. Although the method proved effective in different encounter situations, a severe shortcoming is that the vessel might get trapped inside the avoidance region – leading to potential hazardous situations. A redefinition of the region is required, possibly with another geometric shape, so that the vessel at least can enter a minimum risk condition as defined in the guidelines of DNV GL (DNV GL, 2018).

The path planner was integrated as part of a guidance, navigation, and control system. A path generator using Bézier curves, together with the path planner, comprised the guidance module. Based on the planned WPs, the path generator was responsible for defining a continuous path, taking inherent limitations of the vessel into account. A backstepping maneuvering controller then computed the forces and moment to be produced by the propulsion system in order to follow the path. Accurate estimates of the vessel's position and heading were provided by a nonlinear passive observer, even when measurements were corrupted by noise, and sent to the controller and path planner. The path planner proved to be responsive to inputs and changes in the environment, and the straight-line expansion was probably the most promising considering computation time and consistent outputs in the form of equidistant WPs.

Sammendrag

Planlegging av en trygg og kollisjonsfri bane er avgjørende når graden av autonomi i skip øker. Det gjøres med utgangspunkt i en modell av omgivelsene, som her har vært i form av et bio-inspirert nevralt nettverk. En strategi for å finne en mulig bane i dette landskapet er hybrid A*. Denne algoritmen tar hensyn til at området skipet beveger seg i er kontinuerlig, samt fartøyets iboende begrensninger i form av svingradius. To ulike metoder for å danne søketreet ble utviklet: Enten ved bruk av rette linjestykker med fast lengde, eller med buesegmenter gitt av fartøyets minste oppnåelige svingradius. I tillegg ble to kostfunksjoner evaluert, hvorav en er den Euklidiske avstanden til mål, mens den andre var justert for nevral aktivitet og inkluderte en straff på endring i søkeretning. Hybrid A* algoritmen ble testet gjennom simuleringer, og bruk av rette linjestykker sammen med førstnevnte kostfunksjon viste seg å være mest effektiv med tanke på antall noder som evalueres ved hvert søk etter en bane.

Trygg ferdsel og forutsigbare handlinger i situasjoner med møtende trafikk sikres ved å få skipet til å følge konvensjonen om internasjonale regler til forebygging av sammenstøt på sjøen, COLREG. Strategien var å definere et område framfor det møtende skipet som ikke skulle krysses av skipet under kontroll. Området ble beregnet utfra relativ vinkel og avstand mellom skipene, slik at manøvrer passende til den gjeldende situasjonen ble igangsatt. Simuleringer viste at strategien fungerte i ulike situasjoner, men en alvorlig svakhet er at skipet kan bli fanget inne i området som skal unngås, som igjen kan resultere i potensielt farlige situasjoner. En mulig løsning er å benytte en annen geometrisk form enn et rektangel og i det minste sikre at skipet har en mulig utvei.

Baneplanleggeren ble integrert som en del av et helhetlig autonomt system. Det inkluderte en banegenerator, en estimator, samt et kontrollsystem. Utfra veipunktene gitt av baneplanleggeren, var banegeneratoren ansvarlig for å definere en sammenhengende og glatt bane tatt i betraktning fartøyets iboende begrensninger. Til dette ble Bézier-kurver benyttet. Kontrollsystemet var ansvarlig for at skipet faktisk fulgte banen, ved å beregne nødvendige krefter og moment som propulsjonssystemet måtte generere. Estimatoren viste seg å gi nøyaktige estimater av skipsposisjonen, selv med støyete målinger, og disse ble benyttet av baneplanleggeren. Hybrid A* med rette linjestykker er trolig den mest lovende metoden i et slikt system, ettersom den er forutsigbar i det at den gir veipunkter med konstant avstand fra hverandre.

Preface

Inspired by the emerging projects on maritime autonomy, and the all-electric container ship *Yara Birkeland* in particular, I chose the specialization of marine cybernetics three years ago. Now, I conclude five years at the Norwegian University of Science and Technology with this dissertation on path-planning for autonomous vessels. I would like to thank my supervisor Prof. Roger Skjetne for the interesting topic and for feedback throughout the semester, as well as for sharing material and papers from the latest research in the field. Being on the finish line of earning a master's degree, I am more than ever convinced by Albert Einstein in that "The more I learn, the more I realize how much I don't know", and I am eager to continue to explore what is unknown to me.

Due to the COVID-19 pandemic and subsequent lockdown of educational institutions in Norway on March 12th (Helsedirektoratet, 2020), physical experiments could unfortunately not be carried out in the Marine Cybernetics Laboratory (MC-Lab). Experimental testing was eventually substituted by simulation studies. Mathias Marley has been a sparring partner when it comes to implementation and simulation of the path-planning system, and has provided me with interesting material to study. He and Zhengru Ren have also commented the writing of this thesis, and I owe them both sincere thanks.

Prior to the lockdown, a significant amount of time was committed to preparations of experiments and the test platform named CyberShip Enterprise I (CSEI). Nevertheless, the hours spent in the MC-Lab together with fellow master's students Elias Gauslaa and Jakob S. Jensen were not all wasted: It resulted in the revision and update of the CSEI Handbook (NTNU, 2020), including a new tuning of the servos controlling the propellers. This will hopefully be of benefit to students coming after us. Thanks goes to Torgeir Wahl and Einar S. Ueland for helping us with laboratorial issues, and to Elias and Jakob for our teamwork both inside and outside the laboratory.

The last half of a year has undoubtedly been challenging for the majority of society and did not turn out as we had planned for. I am truly glad for the memorable time while we were still allowed to work together in office A2.019. The struggles of frontline health workers and those infected by the coronavirus I can only imagine, but I have realized that the importance of everyman's mental health should not be understated either. Special thanks goes to my family, who urged me to come home after some lonesome weeks in lockdown and put up with me for two months. Furthermore, I am more than grateful for the encouraging words and high fives along the way.

Caroline Fleischer
Trondheim, June 2020

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AIS	Automatic Identification System, p. 32
ASV	autonomous surface vehicle, p. 9
BINN	bio-inspired neural network, p. 5
CLF	control Lyapunov function, p. 50
COLAV	collision avoidance, p. 11
COLREGs	International Regulations for Preventing Collisions at Sea, p. 6
CPU	central processing unit, p. 42
CSEI	CyberShip Enterprise I, p. 54
DOF	degree of freedom, p. 9
DP	dynamic positioning, p. 10
ECDIS	electronic chart display and information system, p. 31
GNC	guidance, navigation and control, p. 12
GNSS	global navigation satellite system, p. 13
GW	give-way, p. 40
HO	head-on, p. 40
IMO	International Maritime Organization, p. 1
KPI	key performance indicator, p. 7
LOS	line-of-sight, p. 27
LP	linear program, p. 28
MI(L)P	mixed-integer (linear) programming, p. 7
MRC	minimum risk condition, p. 31
MSC	Maritime Safety Committee, p. 31
NE	the North-East frame, p. 10

OS	ownship, p. 40
$\mathrm{OT}_{\mathrm{p/s}}$	overtaking on port/starboard side, p. 40
QP	quadratic program, p. 28
RRT	rapidly-exploring random trees, p. 4
SA	situational awareness, p. 11
SO	stand-on, p. 40
TS	target ship, p. 40
WP	waypoint, p. 2

Symbols

β	angle defining straight-line motion primitives, p. 37
$\boldsymbol{\eta}$	vehicle position and orientation with respect to an inertial frame, p. 10
$\boldsymbol{\nu}$	linear and angular velocities in a body-fixed frame, p. 10
$\boldsymbol{b}$	bias, p. 13
$\boldsymbol{p}$	position in the state space, p. 11
$\boldsymbol{x}, \boldsymbol{x}^*$	vector of real decision variables, optimal solution, p. 27
$\boldsymbol{y}$	vector of integer decision variables, p. 30
$\boldsymbol{z}^\psi$	unit velocity vector, p. 38
Δp	distance between two encountering vessels in $\{n\}$, p. 41
ϵ_{ss}	Condition for steady-state neural activity, p. 35
κ	large positive constant defining external input to the BINN, p. 22
λ	penalty factor on changes in heading, p. 23
$\mathbb{R}^n$	Euclidean space of dimension n, p. 23
$\mathcal{R}_d$	detection region, p. 11
$\mathcal{R}_i$	receptive field, p. 22
μ	constant determining activity propagation among neurons, p. 22
Ω	set of feasible solutions, p. 27
ω_{max}	system bandwidth, p. 63
ψ	vessel heading, p. 10
σ	standard deviation, p. 63
θ	central angle of the kinematic motion primitives, p. 37
ζ	breadth of free corridor enclosing two WPs, p. 46

A	passive decay rate, p. 22
a	slope of a straight-line motion primitive, p. 37
$A(i,j)$	adjacency matrix, p. 23
B	upper bound on neural activity, p. 22
$c_i(\,\cdot\,)$	equality ($i \in \mathcal{E}$) and inequality ($i \in \mathcal{I}$) constraints, p. 27
D	lower bound on neural activity, p. 22
d	length of a motion primitive, p. 36
d_{ij}	distance between two neurons i and j (metric), p. 22
dX, dY	Length of the grid cells along the x- and y-axis, p. 35
E	set of edges of graph G, p. 24
$e(t)$, $s(t)$	cross-track error, along-track error, p. 52
$f(\,\cdot\,)$	objective function, p. 27
G	graph data structure, p. 24
g	gravitational acceleration, p. 55
$g(n)$	cost of the optimal path from the start node n_s to a current node n, p. 26
$h(n)$	cost of the optimal path from current node n to the target n_t, p. 26
I_i	external input to the ith neuron, p. 22
k	number of neighboring neurons, p. 22
L	vessel length, p. 55
n, n_c, n_t	current node, child node, target node, p. 26
N_{ss}	size of the search space for optimal Bézier-path generation, p. 49
r_0	radius defining the set of neighboring neurons, p. 22
r_k	radius of acceptance at waypoint k, p. 52
R_{min}	minimum turning radius of the vessel, p. 37
$s,\ s'$	path parameter, path parameter along a line segment, p. 15
t_c	correlation time, p. 63
U	vessel speed in the absence of current, p. 10
U_{ref}	commanded speed, p. 15
V	set of vertices of graph G, p. 24
$V_P,\ E_P$	ordered set of waypoints, set of edges connecting the WPs, p. 14
w_{ij}	connection weight between neuron i and j, p. 22
z_i	neural activity of the ith neuron, p. 22

Chapter 1

Introduction

Autonomy is a term of ancient Greek origin and literally translates as 'self-government'. In the context of vehicle control, autonomy refers to the degree to which decisions about future actions can be made without the intervention from a human operator. For the time being, the International Maritime Organization (IMO) distinguishes between four levels of autonomy, ranging from a ship with some automated processes and decision support to a fully autonomous ship (IMO, n.d.). The self-governing abilities are dependent on the information available to the vehicle, regarding its position and surroundings, as well as the required actuator control to accomplish a certain mission. This chapter gives a brief discussion on aspects that motivate the development of autonomous ships, and, from a technical perspective, what remains to be done for them to become a reality. The contribution and an outline of this thesis is also presented.

1.1 Motivation

Technologies enabling autonomous sailing are improving rapidly, leveraging advances in the fields of computer and materials sciences and improved connectivity at sea. The invention of a device for the automatic steering of a ship – an autopilot – dates back to the early twentieth century (Fossen, 2011, p. 231). Since then, the ship autopilot has become more sophisticated, and with the recent advent of automated docking systems (Wärtsilä, 2018; Kongsberg, 2020) fully automatic fjord crossings, although still supervised, are being executed. However, unmanned shipping is not far away, being anticipated in the Oslofjord within two years (Kongsberg, n.d.). In the mean time, there are still some challenges to be properly addressed – among others related to the use of smart sensors and the reliability of situational awareness systems, cybersecurity and robustness of software, and national and international legislation.

Solving encounter situations by machines is a critical part of autonomous systems. In fact, as a majority of ship collision and grounding incidents are imputed to human errors (Chauvin et al., 2013; DNV GL, 2015), increasing the level of autonomy is expected to reduce the number of accidents and help human operators make more informed decisions. Furthermore, autonomous sailing has the potential to impact fuel economy, travel time and congestion.

Unmanned vessels, in particular, can be designed more efficiently as there will be no need for accommodation and safety equipment for a human crew, and they may keep slower cruise speeds to save fuel (Wright, 2020). The planning of an efficient and safe voyage can be divided into three main layers (Huang et al., 2020):

1. A route, or voyage, plan,
2. a planned path, and
3. reactive replanning of the path.

The route plan is drawn on a large scale map and describes the entire voyage – berth to berth, considering all navigational information available (e.g. weather forecasts and traffic). Path-planning is the problem of finding a collision-free path on a local map while underway. A path is established by defining waypoints (WPs), headings and safe speeds. The feasibility of the path depends on the accuracy of the map and the complexity of the surrounding environment.

Sailing through congested and dynamic environments requires for intelligent path-planning algorithms. Non-static environments are characterized by information that is incomplete and uncertain. This motivates the search for a path-planner which reactively adjusts the path to perceptual information and at the same time considers the characteristics and limitations of the vehicle itself. The latter is necessary to ensure that the vehicle is actually capable of following the planned path. Commonly, vehicle dynamics are first taken into account in the subsequent path-following problem (Kamal, Gu, and Postlethwaite, 2005). However, introducing the vehicle characteristics at an earlier stage has the potential of yielding a more suitable and optimal path. The hybrid A* algorithm seems promising in this regard. Moreover, optimality can be pursued by searching for a path that minimizes certain objectives, such as fuel consumption. This brings up mathematical programming and model predictive control (MPC) as relevant methods to explore.

1.2 State of the art in the industry and academia

The world's first unmanned system was, in fact, a radio-controlled surface vessel. Tesla (1898) patented the technology for his "teleautomaton" and demonstrated the miniature boat publicly in Madison Square Garden (Nikola Tesla Museum, n.d.). Yet, up until recently, autonomous ships have not received as much attention as other autonomous systems (Savitz et al., 2013). Today, Tesla's name is rather associated with the American electric-automobile manufacturer and their ambitions to commercialize driverless cars (The Tesla Team, 2016). Technology advancements in modern cars, comprising adaptive cruise control, self-parking capabilities and driving assistance systems (e.g. Mercedes-Benz, n.d.; Ford, n.d.), as well as extensive testing of fully-autonomous automobiles, such as Waymo's (n.d.) self-driving car, has encouraged the maritime industry to proceed towards unmanned ships (Kühner, 2017).

Norway, being a maritime nation, aims to be at the forefront in the development of autonomous ships. With designated test beds for autonomous vessels in the Trondheimsfjord, Storfjord, and Oslofjord (NFD, 2019), there are several initiatives both in industry and academia. Among others, Autoferry and Autosea are two research projects concerned with autonomous all-electric passenger ferries, and sensor fusion and collision avoidance (NTNU, n.d.). A commercial counterpart, receiving considerable media attention, is the all-electric shortsea container ship *Yara Birkeland* (Kongsberg, n.d.). The vessel is expected to be launched this year, and gradually move from manned to fully autonomous operation within two years. Overseas, joint effort by industry and academia, under the US Office of Naval Research, already resulted in a fully autonomous voyage of the 40-meter-long *Sea Hunter*, from California to Hawaii and back, in 2019. In the UK, simulators of autonomous vessel control and ship maneuvering by L3 ASV and BMT, respectively, are merged to evaluate multi-vessel conflicts and the coexistence of autonomous and conventional manned ships in shared waters (Wright, 2020). These are only a few of the ongoing research and development projects on autonomous ships in Norway and abroad.

An extensive study of technological, economic, social and regulatory factors essential for realizing autonomous ships was conducted by the Advanced Waterborne Applications Initiative (AAWA) in 2016. The initiative was led by academic researchers at some of Finland's leading universities together with members of the maritime clusters, including Rolls-Royce and DNV GL. From a technical point of view, Poikonen et al. (2016) points out that ships are not confined to narrow roads and operate at relatively slow speeds as opposed to cars. At the same time, ships have a large inertia and struggle in taking sharp turns or stop quickly. Therefore kinematic (e.g. turning radius) and dynamic constraints (e.g. speed limits) of the vessel have to be considered when planning a path. Furthermore, processing of sensor data and maintaining an updated map of the environment will be critical for the planning and reactive replanning of a path. Herein, prediction of future positions of moving obstacles is essential. According to Jokioinen (2016), sensor technology needed for autonomous and remotely operated ships already exists. The challenge lies in an proper fusion of different types of radars and visual sensors, and, further, to use this information to generate a feasible path. The latter will be addressed in this thesis.

(a) Autoferry. *Courtesy of Dragland (n.d.).*

(b) Yara Birkeland. *Courtesy of Yara (2019).*

(c) Sea Hunter. *Courtesy of Williams (2016).*

1.3 Literature review

As indicated in Section 1.2, modern automobiles are generally more advanced in autonomous operations than present-day ships. Studies on autonomous ground – as well as on underwater and airborne – vehicles can provide valuable references for the path-planning problem. Recent reviews on path-planning methods within these fields are presented by Campbell et al. (2020), Panda et al. (2020), and Aggarwal and Kumar (2020), respectively. However, experiences and results from other fields may not be directly transferable to the maritime domain (Wright, 2020; Chen, Hopman, and Negenborn, 2018; Poikonen et al., 2016). Among others, the large inertia of vessels and hydrodynamic impacts yield slower acceleration and deceleration compared to other vehicles. Furthermore, environmental forces from waves, wind, and currents significantly affect ship motions, and increase the level of uncertainty in motion control. Nevertheless, the path-planning algorithms presented for maritime craft are mostly based on similar principles as for ground and airborne vehicles.

Path-planning, in general, is the task of finding a feasible path for a vehicle to move from one location to another. A formal definition of the path-planning problem is given by Petereit et al. (2012): “Given a map, find the cost-optimal path from the vehicle’s current location to a goal region subject to the constraints given by the vehicle kinematics and the terrain”. In maritime applications the ‘terrain’, or the surrounding environment, will comprise oceans, seas, bays, estuaries, islands and other major water bodies, as well as the airspace above. There are mainly two types of planning problems, illustrated in Figure 1.3.1: *point-to-point* problems and *coverage* problems (Beard and McLain, 2012). The objective of the former is to plan a path passing through one or a few target points, whereas the second type of problems aim at finding a path such that all unobstructed points in a certain region are covered. If the region is equal to the entire workspace, the problem is referred to as *complete coverage.* The two types of planning problems can also be combined, such as in Scibilia, Jørgensen, and Skjetne (2012) where an AUV transits through an obstructed area before performing a complete coverage. Coverage problems can be regarded as a special case of point-to-point paths, and for this reason the latter will be in focus.

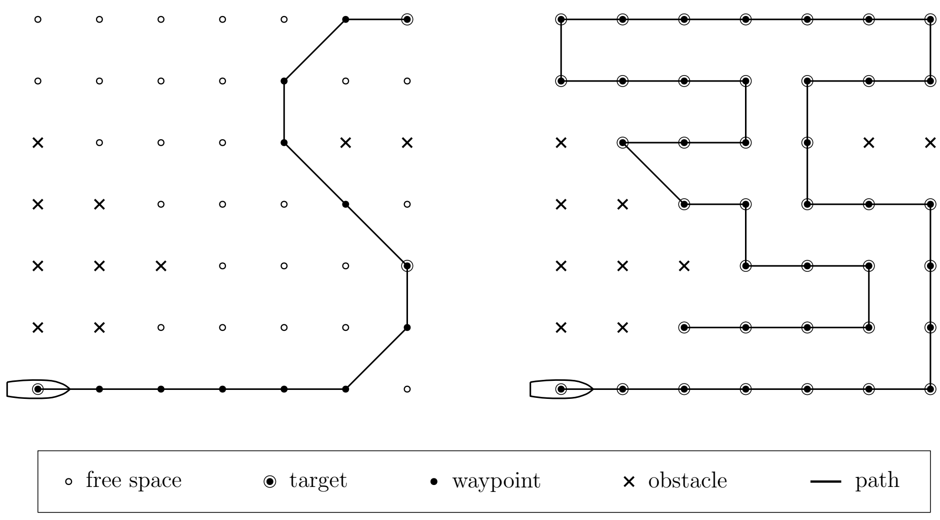


Figure 1.3.1: Illustration of a point-to-point problem to the left and a (complete) coverage problem to the right.

Commonly, path-planning approaches are divided into *deliberative* and *reactive* systems (Beard and McLain, 2012), along with hybrid architectures seeking to benefit from both (Eriksen and Breivik, 2017). The former, also known as *global* path-planning, computes explicit paths based on a priori information of the environment. Methods herein include the construction of roadmaps from visibility graphs and Voronoi diagrams (e.g. Šeda, 2007), sampling-based algorithms such as rapidly-exploring random trees (RRT) (Beard and McLain, 2012, pp. 212-219), and grid-based algorithms like the A* search algorithm in Hart, Nilsson, and Raphael (1968). Highly dynamic environments, where information is incomplete and uncertain, require reactive or *local* path-planning. Examples of such methods are genetic algorithms (e.g. Hu and Yang, 2004), potential field methods originating from Khatib (1985), the dynamic window approach in Fox, Burgard, and Thrun (1997), and receding-horizon optimization (e.g. Schouwenaars et al., 2001). These online algorithms typically use a minimum of computations leading to sub-optimal solutions. Furthermore, sensors are needed to detect unknown obstacles for the replanning of the path. Subjected to a continuous flow of information and time-varying constraints, the path-planning problem becomes rather complex.

Due to complexity, path-planning algorithms may turn into computationally expensive optimization problems. Complexity stems both from a large set of constraints, and the fact that these constraints might be highly nonlinear. The latter suggests that linear approximations will not yield acceptable results. Even a single nonlinear constraint in the program to be solved increases the difficulty in obtaining a solution. According to Reif and Sharir (1985), path-planning among a finite number of moving obstacles is NP-hard. A problem is said to be NP (nondeterministic polynomial) if a solution can be randomly guessed and verified in polynomial time, and being NP-hard, it is at least as hard as any NP problem (Weisstein, n.d.[b]). An intrinsic challenge of the path-planning problem is related to the nonconvex nature of the search space. For instance, a vehicle moving in a two-dimensional space can choose to pass on either side of an obstacle (Richards, Feron, et al., 2002). Nevertheless, planning algorithms have to be computationally efficient and respect hardware limitations of the on-board computer as they are executed on a regular basis in an outer feedback loop. Regular execution of the algorithm is necessary to enable replanning of the path.

Besides the computational complexity of path-planning algorithms, another challenge lies in the continuous nature of the search space. The vessel surroundings are typically partitioned into a regular grid, where the cell size depends on whether a deliberative or a reactive system is concerned and the time needed to perform an avoidance maneuver or a crash-stop. A coarse partitioning yields few nodes to optimize on, and hence shorter computation times, but at the cost of how reactive the system will be. Thus, in narrow and densely obstructed waters, a fine grid partition is preferable, whereas during transit in open waters a coarser grid may be applied. Alternatives to the regular grid include polygonal maps, navigation meshes and Voronoi graphs, but these are all discrete representations of the environment and produce non-smooth paths that generally do not satisfy *nonholonomic*[1] constraints of the vehicle. This shortcoming was addressed by Richards, Sharma, and Ward (2004) through the hybrid A* algorithm. It uses a heuristic search in continuous coordinates to produce a path that respects the vehicle's turning radius. Nonholonomic constraints can also be incorporated into mathematical programming, in which optimization within a continuous search space is possible. Therefore, both hybrid A* and mathematical optimization seem promising for the path-planning for an ASV.

Grid-based methods, such as the A* search algorithm, are traditionally applied to costmap representations of the environment. Costs are assigned to each cell in the grid, and can, for instance, be used to maximize the distance to obstacles (Jaillet, Cortes, and Simeon, 2008). Occupancy grids are considered the ancestor of modern costmaps: Each cell maintains a stochastic estimate of its occupancy state based on sensor readings (Elfes, 1989). Disadvantages of both costmaps and occupancy grids include that the knowledge acquired in generating one map cannot be reused or generalized. Moreover, costs have a single interpretation, and storing all data in one map can be problematic when evaluating new observations. For this reason, multiple and layered costmaps have been used in some applications (e.g. Ferguson and Likhachev, 2008; Lu, Hershberger, and Smart, 2014), but the layers need to be managed dynamically to provide the right context. Another way of representing the vehicle surroundings is a bio-inspired neural network (BINN), as proposed in Yang and Meng (2001).

[1]A *holonomic* vehicle has constraints on configurations only, and can immediately move in any direction with any orientation. *Nonholonomic* vehicles have additional constraints imposed on velocities and accelerations: A car, for instance, cannot move directly into a parking spot on its side, but has to drive backwards and forwards in order to turn.

It is a topological map where obstacles and targets can be identified as maxima and minima. Advantages of using BINNs as a guidance model include their ability to take and assimilate information about the environment, both known a priori and perceived underway. Here, the BINN will be explored and replace the conventional costmap, with neural activities serving as costs in the path search.

Replanning strategies to avoid collisions while underway are crucial to guarantee feasibility of the path at all times. A comprehensive study on state-of-the-art collision avoidance methods is presented in Huang et al. (2020). Replanning requires for a measure of collision risk, and according to Huang et al. (2020) the risk is either determined by experts or through a simplified collision model. The former strategy is often defined in terms of risk indicators with a preset threshold, whereas the latter can be calculated probabilities of collision based on uncertainties in sensor information, environmental disturbances, and reaction times. In any case, accurate motion prediction of other traffic participants is needed. Physics-based methods predict the motions based on physical laws while ignoring control inputs or treating maneuvers as white noise. Hence, if the other vessel alters its course, the predictions will be inaccurate. Interaction-aware methods are probably the most accurate; utilizing communications between vessel to negotiate, broadcast, or exchange intentions and path information (e.g. Porathe, 2017; Chen, Hopman, and Negenborn, 2018). However, smaller crafts like leisure and fishing boats mostly do not have the needed equipment for route exchange. Moreover, the information received from other vehicles might be erroneous. Therefore, maneuver-based methods will be in focus here. These methods consider the navigational intention of other vessels, either learned or estimated from historical traffic data and conventions for vessel encounters.

A challenge in avoiding collisions is the interaction between the vehicle and obstacles. For navigation among pedestrians, Ferrer and Sanfeliu (2014) presents a variant of an RRT in which motion of obstacles and the robot are jointly forward simulated to identify trajectories that minimize social work. At sea, predictable behavior can be obtained by obeying rules and regulations. The International Regulations for Preventing Collisions at Sea (COLREGs), published in 1972 (IMO, 1972), dictate appropriate actions by motorized vessels in different encounter situations. These are frequently integrated in path-planning algorithms, and strategies to achieve COLREGs-compliant behavior include the extension of bounding boxes around obstacles (e.g. Chiang and Tapia, 2018; Song et al., 2019) and vector-based methods (e.g. Johansen, Perez, and Cristofaro, 2016; Zaccone, Martelli, and Figari, 2019). Song et al. (2019) argue that at some occasions violation of COLREGs might be acceptable: For instance, if other vessels do not comply with their obligations, then hazard levels might increase if the ASV blindly follows the rules. This line of thought is also seen in the simulation-based optimization approach in Johansen, Perez, and Cristofaro (2016), where the selected control is a compromise between COLREGs-compliance and acceptable levels of grounding and collision hazards. Furthermore, several rules may be activated simultaneously in real-world scenarios (Huang et al., 2020). Hence, COLREGs will be regarded as soft constraints[2] in the search for a feasible path.

[2]Soft constraints can be violated, but incurs a penalty, whereas hard constraint must be satisfied by any feasible solution.

Compatibility with other autonomy-enabling subsystems is essential for the path-planning unit to be practical. A path is typically defined in terms of WPs, being intermediate points along the desired trajectory. Combining these WPs with straight-line segments generally gives a nonsmooth path, violating the nonholonomic constraints of the vehicle. Therefore, the WPs produced by the path planner should be processed by a path manager (Beard and McLain, 2012), hereinafter referred to as the path generator. The path generator defines a continuous path by, for instance, constructing a Dubins path (Dubins, 1957), or by path-smoothing using Bézier curves (Choi, Curry, and Elkaim, 2008) or non-parametric interpolation (Dolgov et al., 2008). This requires for the output from the path planner to be readable by the path generator and contain all necessary information to generate a smooth and traversable path (e.g. constraints on curvature, desired speed, and clearance to obstacles). Moreover, a continuous path definition is worthless if there exists no motion control system that can transform the desired vehicle state into actuator inputs. Therefore, the synergy between modules constituting the autonomous system, and in particular the responsibility of the path-planning module in such a system, will be examined.

1.4 Objectives

The master's thesis is a continuation of the specialization project written during the fall of 2019. Then, some algorithms for optimal path-planning were evaluated through simulations on a kinematic vessel model. Results therefrom indicated that the hybrid A* search could be a possible way forward to obtain a path that accounts for constraints set by the vehicle itself and defined in a continuous search space. In that respect, also MPC with mixed-integer programming (MIP) yielded promising results, although a reformulation is needed to make it computationally less expensive and to take the continuous nature of the search space into account. Furthermore, the algorithms have to produce paths that are compliant with rules and regulations, and the path planner must work as part of and together with other modules in a guidance, navigation and control system. These considerations will be the focus in this thesis.

1.5 Contribution

The contribution of the thesis is a brief review of methods for optimal path-planning for autonomous surface vessels, and the development and implementation of a path planner based on a hybrid-state A* path search algorithm and an anti-collision strategy to obtain COLREGs-compliant behavior. Information from the dynamic and unstructured environment is captured in a guidance model in the form of a BINN. The proposed path search algorithm utilize two different strategies for expanding the search tree: Either by straight-lines or by arcs related to the minimum turn circle of the vehicle. Feasibility and performance of the path planner was examined through simulations in MATLAB and Simulink, and evaluated against selected key performance indicators (KPIs). The Simulink model comprised a simulation model of a real model boat, a Bézier-based path generator, a backstepping maneuvering controller, and a nonlinear passive observer. Details on the implementation as well as the choice of KPIs are elaborated on in this report.

1.6 Outline

The remainder of this report is structured as follows: **Chapter 2** presents the problem and the scope of study. Thereafter, mathematical models of the system are given in **Chapter 3**, and a theoretical background on path-planning and the proposed designs follow in **Chapter 4**. The sections on BINN and MPC, in particular, are to a large extent based on the project thesis. Performance measures as basis for the evaluation and comparison of the path-planning algorithms are also specified in Chapter 4. **Chapter 5** focuses on implementation details of the modules in the autonomous system architecture other than the path planner, and **Chapter 6** presents some simulation cases and results therefrom. A discussion of each simulation case is provided, and strengths and shortcomings of the algorithms as well as future directions are pointed out. Concluding remarks and suggestions for further work follow in **Chapter 7**.

Chapter 2

Problem formulation

2.1 System description

Consider a vessel operating on the water surface with no or reduced attention from a bridge crew, commonly referred to as an autonomous surface vehicle (ASV). In general, the configuration of a marine craft moving freely in three dimensions can be completely defined by a total of three translational and three rotational components. These 6 components constitute the degrees of freedom (DOFs) of the craft, and are depicted in Figure 2.1.1.

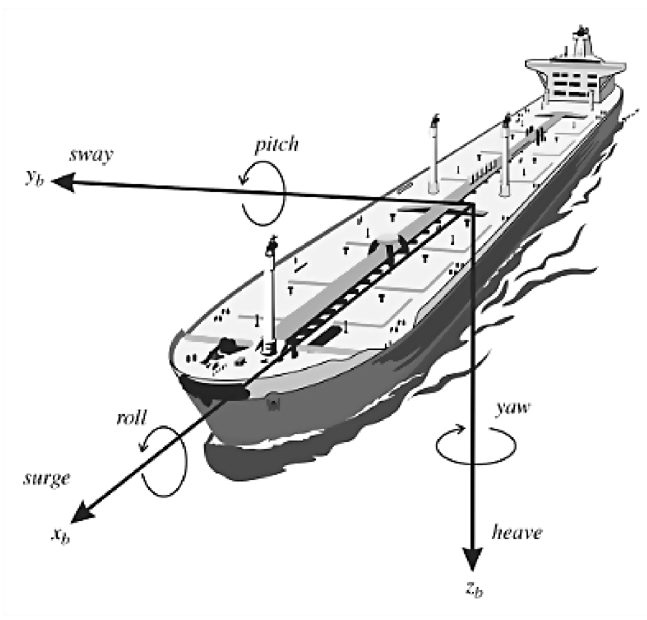


Figure 2.1.1: Motions in 6 DOFs. *Courtesy of Fossen, 2020.*

The ASV is assumed to be metacentric stable. This implies that restoring forces will counteract perturbations away from the equilibrium points in heave, roll and pitch (Fossen, 2020, p. 68). The equilibrium points correspond to zero inclination in roll and pitch and is approximated to lie at the mean water surface in heave. Thus, it is reasonable to neglect motions other than those in the horizontal plane, and only consider surge, sway, and yaw dynamics.

A 3DOF kinematic representation of the ASV is given by (Fossen, 2020, p. 43)

$$\dot{x} = u\cos\psi - v\sin\psi, \tag{2.1.1a}$$
$$\dot{y} = u\sin\psi + v\cos\psi, \tag{2.1.1b}$$
$$\dot{\psi} = r. \tag{2.1.1c}$$

According the notation of SNAME (1950), $\boldsymbol{\eta} := [x, y, \psi]^\top$ denotes the vehicle position and orientation with respect to an inertial frame, and $\boldsymbol{\nu} := [u, v, r]^\top$ are the 3DOF linear and angular velocities in the body-fixed frame. It is assumed that the ASV is at least fully actuated in the horizontal plane, implying that forces and moment can be produced independently in surge, sway and yaw. Then, the vessel is capable of controlling all three DOFs by use of its propulsion system in what is called dynamic positioning (DP) (Fossen, 2011, p. 286).

Two different reference frames – an inertial and the body-fixed frame – are introduced to express the generalized coordinates $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$. For local navigation, within a smaller geographical area about 10km × 10km, it is acceptable to assume that the North-East (NE) frame $\{n\}$ is inertial (Fossen, 2020, p. 18, 41). The NE coordinate system spans an Earth-fixed tangent plane, aligned such that the x_i- and y_i-axis point to the North and East, respectively. The body-fixed frame $\{b\}$ has its origin at the center of gravity of the vehicle, with the x_b-axis pointing out of the bow and the y_b-axis directed towards starboard (Fossen, 2020, p. 16–19). The reference frames $\{n\}$ and $\{b\}$ and the generalized position $\boldsymbol{\eta}$ and velocities $\boldsymbol{\nu}$ are illustrated in Figure 2.1.2.

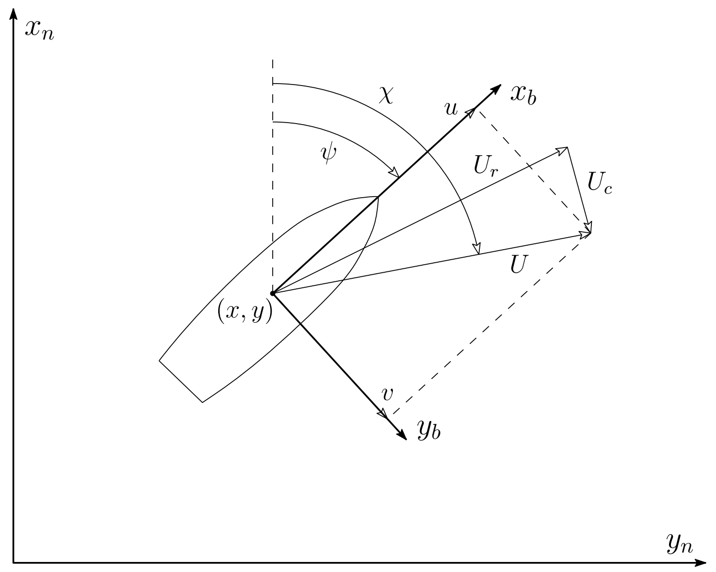


Figure 2.1.2: Illustration of the coordinate systems $\{n\}$ and $\{b\}$ and the generalized coordinates in 3DOF. *Adapted from Fossen (2020, Fig. 2.8).*

The focus in this study will be on the transit phase of a voyage. This phase is characterized by passage on relatively open waters at close to a constant speed. A reasonable assumption is that the speed in surge will be much larger than that in sway, so that the total speed $U \approx u$. Further, if ocean currents are negligible so that $U_c \approx 0$, the course angle χ will coincide with the heading of the vessel ψ. Then, the kinematics (2.1.1) can be rewritten as

$$\dot{\boldsymbol{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = U \begin{bmatrix} \cos\psi \\ \sin\psi \end{bmatrix}, \tag{2.1.2a}$$

$$\dot{\psi} = r. \tag{2.1.2b}$$

The ASV will operate in a dynamic and unstructured environment. Since there will not necessarily be officers on watch, changes in the surroundings are detected by a situational awareness (SA) system of sensors mounted on board (e.g. lidar, radar, and cameras). It is assumed that the position, velocity and direction of movement of obstacles can be accurately identified within a detection region $\mathcal{R}_d$ at all times. Based on this information, future positions of the obstacles can be predicted. Adopting the definition of collision avoidance (COLAV) from Huang et al. (2020), *motion prediction* is the first step in determining an evasive action.

> ***Definition (Collision Avoidance, COLAV)*** *Collision avoidance is the process in which one ship (manned or unmanned) departs from its planned trajectory to avoid a potential undesired physical contact at a certain time in the future.*

Following motion prediction are the steps of *conflict detection* and *conflict resolution.* As a conflict is detected by the SA system, a measure is needed to quantify the collision risk and determine when to take action. On a fully autonomous vehicle, the conflict is then to be resolved by machines. This involves the tasks of deciding on appropriate actions to avoid collision and to replan the vehicle's path. The actions should be in agreement with relevant rules and regulations.

As mentioned in Section 1.3, rules on the duties of a vessel in a vessel-to-vessel encounter are stated in COLREGs. Pertinent rules to a COLAV system are found in Part II, Section B, and include (IMO, 1972):

> **Rule 13 – Overtaking situation.** A vessel coming up from a direction larger than 22.5° abaft of another vessel, is considered an overtaking vessel. The overtaking vessel is obliged to keep out of the way of the vessel being overtaken, regardless of any subsequent changes in bearing between the two vessels.

> **Rule 14 – Head-on situation.** If two power-driven vessels are approaching each other on reciprocal or near reciprocal collision courses, each vessel shall alter course to starboard so that they pass on the port side of the other.

> **Rule 15 – Crossing situation.** In a crossing situation between two power-driven vessels that involves risk of collision, the vessel with the other on her starboard side shall keep out of the way, and, as far as possible, avoid passing ahead of the other vessel.

> **Rule 17 – Action by stand-on vessel.** The vessel not expected to keep out of the way shall maintain her course and speed. If the give-way vessel is not taking appropriate action or if collision cannot be avoided by the action of the give-way vessel alone, then the stand-on vessel shall take action that best aids in preventing collision. Obligations of the give-way vessel are not reversed by this rule.

The encounter situations are exemplified in Figure 2.1.3. By Rule 8, any alterations in speed and/or course shall be made in ample time and be large enough to be readily detected by the other vessel. Furthermore, a safe distance should be kept at all times, and Rule 13 demands every vessel to proceed with a safe speed so that appropriate and effective actions are executable to the prevailing circumstances and conditions. The first step in determining the appropriate action is hence to identify the rules that apply to the situation at issue.

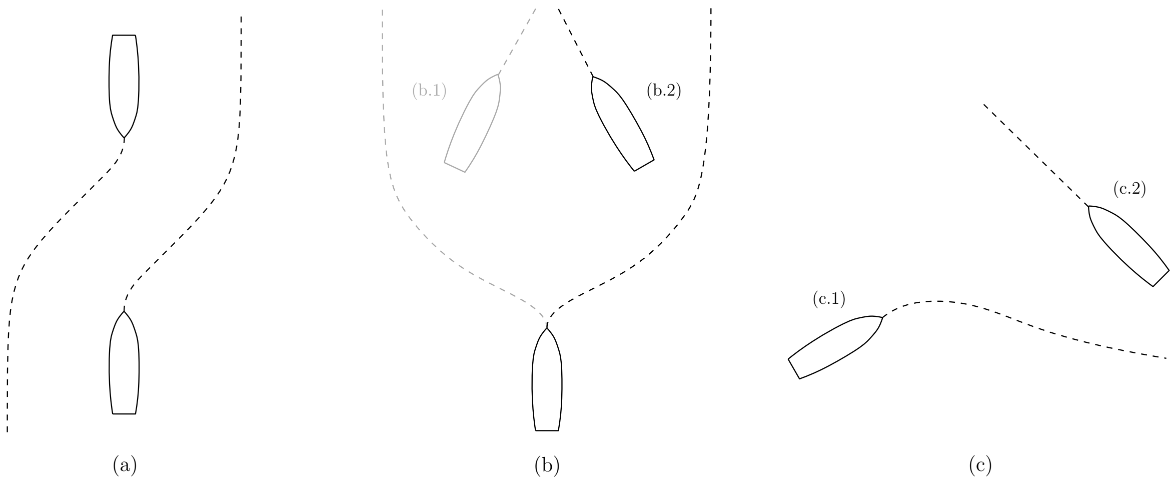


Figure 2.1.3: Encounter situations and appropriate maneuvers as defined by COLREGs: (a) Head-on; (b) overtaking on (b.1) starboard and (b.2) port side; (c) crossing situation in which (c.1) is the give-way vessel and (c.2) is the stand-on vessel.

2.2 Problem statement

On an autonomous vehicle, a guidance, navigation and control (GNC) system is responsible for collision prevention. The guidance system detects and solves conflicts by deciding when and how to take evasive actions. It may be divided into a path planner and a path generator, providing a discrete and continuous definition of the path, respectively. The navigation part of the GNC system offers information to support the guidance system, and the control system is responsible for implementing the planned actions (Huang et al., 2020). An illustration of the system architecture on board the ASV is shown in Figure 2.2.1. More tightly coupling of the GNC blocks might improve performance, but from an industrial point of view modularity is preferred as it facilitates software updates of single blocks (Fossen, 2011, p. 232).

Consider the horizontal NE-frame and an ASV in an arbitrary, but known, initial position. The vessel is requested to move to a fixed and known final position, or several consecutive target positions. Given the kinematic model of the vehicle (2.1.2), a static map of the area of interest, and the availability of accurate perceptual data within a certain region around the vessel, the problem is to develop a GNC system for maneuvering control of the ASV. The ASV is expected to follow a path that connects the initial and final position(s) with a desired speed profile along the path. Since the GNC system operates online, the underlying algorithms have to be computationally efficient and respect hardware limitations of the computer on board. The responsibilities of each GNC module are presented next. In this study, the emphasis is put on the guidance system, and more specifically on the path planner.

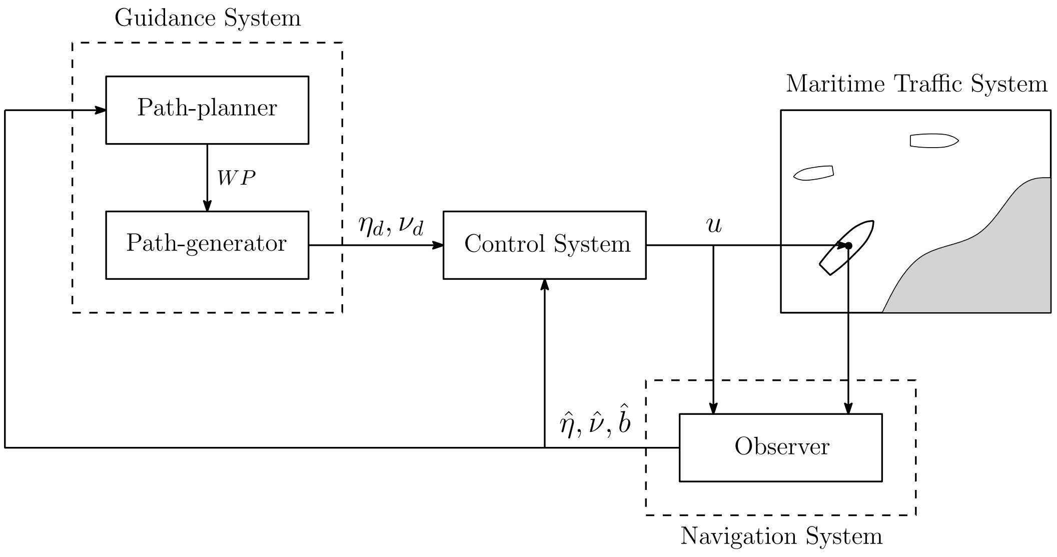


Figure 2.2.1: System architecture on board the unmanned vehicle. *Adapted from Beard and McLain, 2012; Huang et al., 2020.*

2.2.1 Navigation system

Navigation is composed by the Latin words for 'ship' and 'to drive', respectively *navis* and *agere*. It comprises the task of directing a vehicle by determining its position, course, and distance traveled, as well as its velocity and acceleration. Hence, a navigation system forms the foundation in the planning and execution of a safe and timely operation (Fossen, 2011, p. 233). This is achieved by processing data from a global navigation satellite system (GNSS) and motion sensors such as accelerometers and gyroscopes. A signal processing unit is usually responsible for monitoring and evaluating the quality of the measurement signals, before an observer performs noise filtering, prediction, and reconstruction of unmeasured states. The aim of the navigation system is thus to provide accurate estimates of the ASV states.

Define the estimation error as

$$\boldsymbol{e}(t) = \begin{bmatrix} \tilde{\boldsymbol{\eta}}(t) \\ \tilde{\boldsymbol{\nu}}(t) \\ \tilde{\boldsymbol{b}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\eta}(t) - \hat{\boldsymbol{\eta}}(t) \\ \boldsymbol{\nu}(t) - \hat{\boldsymbol{\nu}}(t) \\ \boldsymbol{b}(t) - \hat{\boldsymbol{b}}(t) \end{bmatrix}, \tag{2.2.1}$$

with $\hat{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{\nu}}$ being the estimated generalized coordinates and velocities, and where the bias $\boldsymbol{b}$ represents unmodeled dynamics and external disturbances and $\hat{\boldsymbol{b}}$ is the corresponding estimate. Then, an observer shall be designed such that the equilibrium $[\tilde{\boldsymbol{\eta}}, \tilde{\boldsymbol{\nu}}, \tilde{\boldsymbol{b}}]^\top = \boldsymbol{0}$ is asymptotically stable, or equivalently

$$\lim_{t \to \infty} |\boldsymbol{e}(t)| = 0, \tag{2.2.2}$$

where $|\cdot| := |\cdot|_2$ denotes the second, or Euclidean, vector norm. As shown in Figure 2.2.1, the state estimates $[\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\nu}}, \hat{\boldsymbol{b}}]^\top$ are forwarded to the guidance and control systems.

2.2.2 Guidance system

Guidance refers to the system that continuously computes the reference position, heading, velocity and acceleration of the ASV (Fossen, 2011, p. 233). The reference signals are passed to the motion control system, as illustrated in Figure 2.2.1. Input from the navigation module to the guidance module comprise motion estimates and external data, such as weather information. Here, the guidance system is divided into a path planner and a path generator.

Path planner

The path planner is responsible for determining WPs that define the desired path for the ASV to follow. Each WP represents a unique position in the NE-frame, given as *Cartesian* coordinates $\boldsymbol{p} = [x, y]^\top$. Constraints on placing the WPs include both feasibility and optimality considerations. Feasibility is related to the path being traversable for the vessel, by means of avoiding obstacles and taking inherent limitations of the vessel itself into account. The latter, for instance, includes bounds on its rate of turn and hence place restrictions on path curvature. Optimality is evaluated against certain criteria. Relevant measures to be minimized could be the path length, the time of travel, and the accumulated heading change along the path.

In general, the path-planning problem can be stated in terms of a graph representation $G(V, E)$ of the environment (see Section 4.1.1). Let V be the set of all feasible positions in the horizontal plane, called nodes or vertices, and E be the set of all traversable connections between nodes. Further, let $T \subseteq V$ be the set of nodes marked as targets. Each target node corresponds to a position in the plane where the path is required to pass through. Then, the path-planning problem is to define a walk in G; that is a sequence of WPs given by the ordered set

$$V_P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \dots\}, \qquad T \subseteq V_P \subseteq V, \quad E_P \subseteq E. \tag{2.2.3}$$

The set of WPs V_P must necessarily contain all target nodes, and the edges that connects two consecutive WPs in V_P have to be traversable and hence be contained in E. Figure 2.2.2 illustrates the problem and sets of nodes and edges.

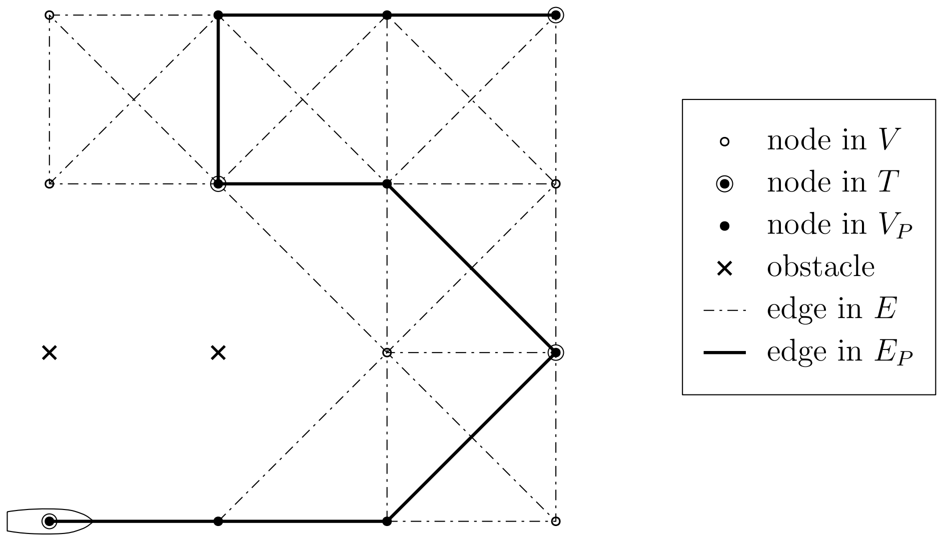


Figure 2.2.2: Diagram illustrating the path-planning problem. For clarity, nodes in T and V_P are also contained in V, and the edges in E_P form a subset of E, as stated in (2.2.3).

Replanning

The graph $G(V, E)$ will be updated continuously based on perceptual information from the SA system. If objects are detected along the planned path, the WPs in V_P have to be reevaluated to avoid collision and to keep an appropriate safety margin when passing the obstacle. This is achieved by modifying the sets of feasible nodes V and edges E in the graph. Replanning of the path will be based on the perceived position, velocity and direction of movement of the obstacle. Perceptual information is assumed to be known and accurate within a detection region $\mathcal{R}_d$ around the ASV. Furthermore, the replanned path should trigger COLAV maneuvers in compliance with the relevant COLREGs rules stated in Section 2.1.

Path generator

Given a list of WPs, the path generator is responsible for uniting these points into a feasible and continuous path. The path is expressed as a hybrid[1] parametric curve given by

$$\boldsymbol{\eta}_d(s) = \begin{bmatrix} x_d(s) \\ y_d(s) \\ \psi_d(s) \end{bmatrix}, \qquad s = s' + l - 1,\ s' \in [0,1],\ l \in \{1, ..., m\} \tag{2.2.4}$$

where s is the path parameter, m is number of line segments in E_P, and s' is the parameter along a line segment. A certain degree of parametric continuity along the path is sought to ensure a smooth transfer between line segments. This implies that the path (2.2.4) must be sufficiently differentiable.

Along with the parameterized path and heading curve (2.2.4), the output of the path-generator contains a speed profile $v_d(s,t)$. The speed profile dictates the speed along the path. By differentiating the position curve in (2.2.4) with respect to time and assigning it to the commanded speed $U_{ref}(t)$, as in

$$|\dot{\boldsymbol{p}}_d(s)| = \sqrt{x_d^s(s)^2 \dot{s}^2 + y_d^s(s)^2 \dot{s}^2} = |\boldsymbol{p}_d^s(s)|\, \dot{s} = U_{ref}(t), \tag{2.2.5}$$

then the speed profile along the path can be defined as

$$v_d(s,t) := \frac{U_{ref}(t)}{|\boldsymbol{p}_d^s(s)|}, \tag{2.2.6a}$$

$$v_d^s(s,t) = -U_{ref}(t) \frac{\boldsymbol{p}_d^s(s)^\top \boldsymbol{p}_d^{2s}(s)}{|\boldsymbol{p}_d^s(s)|^3}. \tag{2.2.6b}$$

Since this study is limited to the transit phase of a voyage, only forward motion is considered. This implies that $U_{ref}(t) \geq 0$ and hence $v_d \geq 0$.

2.2.3 Control system

The motion controller determines necessary control forces and moments that the vessel must produce in order to meet a certain control objective (Fossen, 2011, p. 233). In accordance with the *maneuvering* problem formulation by Skjetne (2005), the control objective comprises two tasks. The primary task is to follow the path given by the path generator: With a parameterized definition of the path (2.2.4), the path-following problem can be stated as

$$\lim_{t \to \infty} |\eta(t) - \eta_d(s,t)| = 0. \tag{2.2.7}$$

[1]A mix of both a continuous and a discrete parameterization (Skjetne, 2005).

As soon the path is reached, the ASV will be forced to stay on it for all future time. Secondary to the geometric task of path-following is the dynamic task of satisfying the desired speed assignment (2.2.5) along the path. This is achieved by forcing the time derivative of the path parameter $\dot{s}$ to converge to $v_d(s,t)$, according to

$$\lim_{t\to\infty} |\dot{s}(t) - v_d(s,t)| = 0. \tag{2.2.8}$$

The responsibility of the control system also includes control allocation. Solving the maneuvering problem (2.2.7) and (2.2.8) with an appropriate control law, yields a 3DOF generalized control vector $\boldsymbol{\tau} = [X, Y, N]^\top$ with the needed forces in surge and sway, and moment in yaw. Control allocation is then the task of computing individual inputs $\boldsymbol{u} \in \mathbb{R}^r$ to each of the r actuators in order to produce the commanded $\boldsymbol{\tau}$. This is a model-based optimization problem, which in its simplest form is unconstrained and given by (Fossen, 2011, p. 398–410)

$$\begin{aligned} &\min_{\boldsymbol{f}} \; \boldsymbol{f}^\top \boldsymbol{W} \boldsymbol{f}, \\ &\text{subject to: } \boldsymbol{\tau} - \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{f} = \boldsymbol{0} \end{aligned} \tag{2.2.9}$$

where $\boldsymbol{W}$ is a matrix weighting the control forces $\boldsymbol{f}$, and $\boldsymbol{T}(\boldsymbol{\alpha})$ describes the configuration of actuators with angles $\boldsymbol{\alpha}$. Control forces are related to the inputs $\boldsymbol{u}$ through a diagonal force coefficient matrix $\boldsymbol{K}$, that is $\boldsymbol{f} = \boldsymbol{K}\boldsymbol{u}$. The optimization problem (2.2.9) can be augmented to include physical limitations, such as input amplitude and rate saturation.

2.2.4 Assumptions and delimitation

The delimitation of the study and underlying assumptions in the derivation of a GNC system for the ASV can be summarized as follows:

- Operations are assumed to be performed in calm waters, such that the influence of waves and ocean currents is negligible (or captured by a bias $\boldsymbol{b}$).
- Only the transit phase of a voyage is considered. This phase is characterized by forward motion with the surge velocity u markedly larger than in sway, so that the total speed $U \approx u$. Accordingly, the commanded speed is assumed to satisfy $U_{ref} \geq 0$.
- The ASV is sufficiently metacentric stable so that motions in heave, pitch and roll are neglected. Hence, a 3DOF horizontal plane model, comprising surge, sway and yaw dynamics, is used.
- The ASV is at least fully actuated in the horizontal plane, so that it can control all 3DOFs independently and enter a DP mode.
- The ASV is a rigid body, thus ignoring forces acting between individual mass components. This is a prerequisite in the derivation of the equations of motions of the ASV (Fossen, 2020, p. 53).
- Operations are conducted within a limited region on the Earth's surface, and the NE-frame $\{n\}$ is inertial. Consequently, forces on the ASV due to the Earth's rotation are neglected (Fossen, 2020, p. 53).
- The position, velocity and direction of movement of obstacles are accurately perceived by the ASV within a region of detection $\mathcal{R}_d$.

Chapter 3

Vessel modeling

The first step in designing a GNC system for an ASV is to model the physical system by nonlinear differential equations of motion. The physical system comprises the ASV, including actuators and sensors, while operating in its environment. Even though approximations and simplifications are needed at this step, the aim is to capture in mathematics all key characteristics of the physical system (Beard and McLain, 2012, p. 5). The result is a high-fidelity *simulation model* used for computer simulations. From the more complex simulation model, one may derive simpler models for the purpose of designing controller and navigation systems. Figure 3.0.1 illustrates the three types of models and how they interact. These are the *control design* and *observer design models*, in addition to the simulation model. A presentation of the sets of equations constituting each model is given in this chapter.

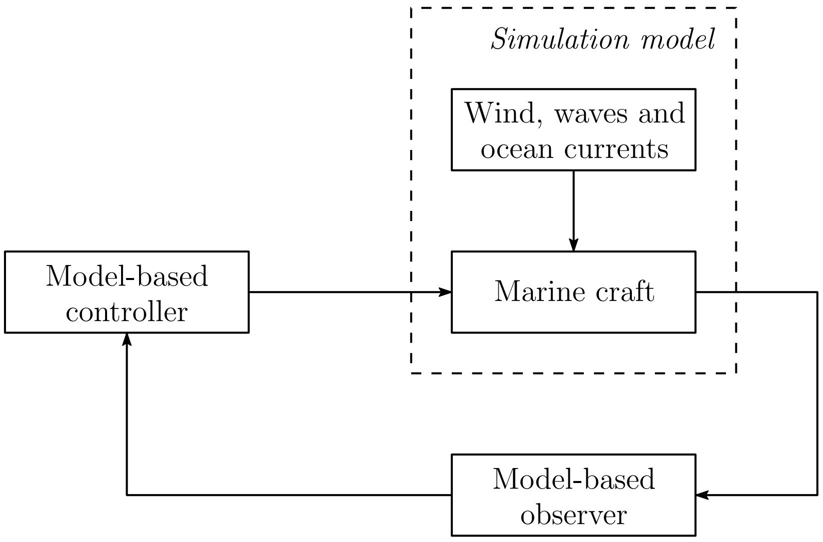


Figure 3.0.1: Models for guidance, navigation and control. *Adopted from Fossen (2020).*

3.1 Simulation model

3.1.1 6DOF nonlinear maneuvering model

A simulation model is of high fidelity and captures all essential characteristics of the system. The main purpose of such a model is to reproduce the real dynamics as accurately as possible (Sørensen, 2018, p. 102). It comprises the craft dynamics, propulsion system, as well as environmental impacts by wind, waves, and ocean currents. For a marine craft, the general maneuvering model can be represented in 6DOF with 12 ordinary differential equations on the following compact vector form (Fossen, 2020, ch. 6)

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{3.1.1a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave}. \tag{3.1.1b}$$

Here, impacts from wind and waves are added as external forces $\boldsymbol{\tau}_{wind}$ and $\boldsymbol{\tau}_{wave}$. Although environmental forces generally are highly nonlinear and multiplicative to the dynamic equations of motion, the *superposition property*[1] is valid for most marine control applications (Fossen, 2020, ch. 10). Wind loads are caused by the movement of air relative to the Earth's surface, and can be represented in terms of speed, angle of attack, and dynamic pressure. Wave-induced forces arise due to a pressure change on the hull surface, and can be divided into first- and second-order components. The former yield zero-mean oscillatory perturbations that vary linearly with wave elevation, referred to as wave-frequency motions, whereas low-frequency motions are induced by second-order slowly-varying wave forces. Due to the high inertia of craft and considering power consumption and potential wear of the actuators, only slowly-varying mean wind and wave forces and moments need to be compensated for by the propulsion system.

Ocean currents are incorporated through the relative velocity between the body and the fluid:

$$\boldsymbol{\nu}_r := \boldsymbol{\nu} - \boldsymbol{\nu}_c, \tag{3.1.2}$$

where the 6DOF body-fixed velocity vector is given by $\boldsymbol{\nu} = [u, v, w, p, q, r]^\top$ and surface currents are assumed to be irrotational with velocity $\boldsymbol{\nu}_c = [u_c, v_c, w_c, 0, 0, 0]^\top$ in $\{b\}$. The first three components of the velocity vectors represent linear velocities in surge, sway, and heave, and the last three components are angular rates in roll, pitch, and yaw. Transformation from $\{b\}$ to $\{n\}$ is achieved through the 6DOF rotation matrix $\boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})$. The Euler angles $\boldsymbol{\Theta} = [\phi, \theta, \psi]^\top$ specify the orientation of the body in $\{n\}$, and, together with the vehicle position $\boldsymbol{p} = [x, y, z]^\top$ in $\{n\}$, make the 6DOF position and orientation vector $\boldsymbol{\eta} = [\boldsymbol{p}^\top, \boldsymbol{\Theta}^\top]^\top$. System inertia and added mass is given by the matrix $\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A$ in (3.1.1). Furthermore, $\boldsymbol{C}(\boldsymbol{\nu}_r) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}_r) + \boldsymbol{C}_A(\boldsymbol{\nu}_r)$ is the Coriolis-centripetal matrix, $\boldsymbol{D}(\boldsymbol{\nu}_r)$ is the damping matrix, and $\boldsymbol{g}(\boldsymbol{\eta})$ is a vector of gravitational and buoyancy forces and moments. Propulsion loads $\boldsymbol{\tau} \in \mathbb{R}^6$ for m actuators are defined by

$$\boldsymbol{\tau} = \boldsymbol{\tau}_1 + \boldsymbol{\tau}_2 + \cdots + \boldsymbol{\tau}_m, \tag{3.1.3}$$

and for each $i \in \{1, \dots, m\}$, $\boldsymbol{\tau}_i$ can be split into forces $\mathcal{F}_i$ and moments $\mathcal{M}_i$ according to

$$\boldsymbol{\tau}_i := \begin{bmatrix} \mathcal{F}_i \\ \mathcal{M}_i \end{bmatrix}, \quad \mathcal{F}_i := \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} [N], \quad \mathcal{M}_i := \begin{bmatrix} K_i \\ M_i \\ N_i \end{bmatrix} [Nm]. \tag{3.1.4}$$

[1] The sum of responses caused by each disturbance individually.

Being a maneuvering model, (3.1.1) is valid for a ship moving in calm water at constant positive speed U. As opposed to seakeeping theory which deals with ships motions at constant speed in waves, maneuvering theory does not consider *fluid memory effects*[2] and approximate the hydrodynamic coefficients by constant values. In fact, when stabilizing motions in surge, sway, and yaw through feedback control, the natural frequencies will be about $100 - 200s$ corresponding to $0.03 - 0.10$rad/s (Fossen, 2020, p. 131). Being close to the zero wave excitation frequency, the coefficients can be approximated at the single frequency $\boldsymbol{\omega} = \mathbf{0}$. Thus, since this study is limited to surface vessel (ref. Section 2.2.4), a 3DOF maneuvering model is used as the simulation model.

3.1.2 3DOF nonlinear maneuvering model

Considering a surface vessel, as described in Chapter 2, the maneuvering model (3.1.1) can be reduced to the nonlinear model in surge, sway, and yaw given by (Fossen, 2020, p. 146)

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{3.1.5a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}. \tag{3.1.5b}$$

When comparing (3.1.5) with (3.1.1), it is evident that environmental forces are assumed to be negligible, that is $\boldsymbol{\nu}_c = 0$ and $\boldsymbol{\tau}_{wind} = \boldsymbol{\tau}_{wave} = 0$. Modeling of environmental forces is beyond the scope of this thesis, and the interested reader is referred to e.g. Fossen (2020, ch. 10). The kinematic relationship (3.1.5a), describing the geometrical aspects of motion, is recognized as a compact vector notation of (2.1.1).

3.2 Control design model

The control design model, being a simplified version of the simulation model (3.1.5), is used in the design of the motion control system. Only the main physical properties are captured in this low-fidelity representation, so as to simplify the control design and respect on-board hardware limitations and time constraints. Control design models may also be used in theoretical analyses of system stability and limitations (Sørensen, 2018, p. 102).

Consistent with maneuvering theory, presuming that the ASV operates at constant speed U in calm waters, a linear control model in 3DOFs is given by (Fossen, 2020, p. 168)

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(t)\boldsymbol{\nu}, \tag{3.2.1a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{R}(t)^T\boldsymbol{b} + \boldsymbol{\tau}, \tag{3.2.1b}$$

$$\dot{\boldsymbol{b}} = \mathbf{0}. \tag{3.2.1c}$$

The nonlinear Coriolis-centripetal and damping forces of (3.1.5) are linearized about the sway and yaw velocities $v = r = 0$ and the cruise speed $u \approx U$. Linear damping is a good assumption for low-speed applications, and the model (3.2.1) is valid for stationkeeping and low-speed maneuvering up to approximately 2m/s (Fossen, 2020, p. 163). The linear model (3.2.1) is derived under the assumption that measurements of the heading angle $\psi = \psi(t)$ are accurate, so that

$$\boldsymbol{R}(\psi(t)) := \boldsymbol{R}(t), \tag{3.2.2}$$

[2] A term capturing the fact that waves on the free surface due to vessel motions will persist at all subsequent times (Fossen, 2020, p. 115).

thereby removing the kinematic nonlinearity introduced by the rotation matrix in (3.1.5a). Furthermore, a linear relationship between the generalized control force vector $\boldsymbol{\tau}$ and control inputs $\boldsymbol{u}$ is assumed, so that

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{K}\boldsymbol{u} = \boldsymbol{B}\boldsymbol{u}, \tag{3.2.3}$$

where $\boldsymbol{B} = \boldsymbol{T}\boldsymbol{K}$ is the input matrix and the thrust configuration matrix $\boldsymbol{T} = \boldsymbol{T}(\boldsymbol{\alpha}_0)$ is constant. If the vessel has any rotatable actuators, an extended input vector $\boldsymbol{u}_e$ and the corresponding extended input matrix $\boldsymbol{B}_e$ is used to satisfy the linear relationship (3.2.3) (see Fossen, 2011, pp. 402-403).

3.3 Observer design model

Similar to the control design model, the observer design model is a simplified version of the simulation model (Fossen, 2020, p. 6). However, the observer design model is different from the control design model in that emphasis is put on the additional dynamics of sensors and navigation systems and disturbances. In particular, filtering of measurement noise, *dead-reckoning*[3] in failure situations, and motion prediction are tasks for the observer. Model-based observers typically include a disturbance model, responsible of estimating wind, waves and ocean current forces by treating these as colored noise.

A 3DOF stochastic observer design model for low-speed applications is given by (Fossen, 2011, p. 306)

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_w\boldsymbol{\xi} + \boldsymbol{E}_w\boldsymbol{\omega}_w, \tag{3.3.1a}$$
$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{3.3.1b}$$
$$\dot{\boldsymbol{b}} = \boldsymbol{\omega}_b, \tag{3.3.1c}$$
$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{R}(\psi)^T\boldsymbol{b} + \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\omega}_\nu, \tag{3.3.1d}$$
$$\boldsymbol{y} = \boldsymbol{\eta} + \boldsymbol{C}_w\boldsymbol{\xi} + \boldsymbol{v}_y, \tag{3.3.1e}$$

where $\boldsymbol{\omega}_w$, $\boldsymbol{\omega}_b$, and $\boldsymbol{\omega}_\nu$ denote white process noise, and $\boldsymbol{v}_y$ is white measurement noise. GNSS and compass measurements are denoted $\boldsymbol{y}$, and include both low-frequency and wave-frequency components. Effects of surface currents and nonlinear dynamics are lumped into a slowly varying bias $\boldsymbol{b}$, typically modeled by an integration of white noise (3.3.1c) known as a Wiener process or a random walk. Oscillatory wave responses due to 1st order force components, mentioned in Section 3.1, are removed by the *wave filter* (3.3.1a), with $\boldsymbol{A}_w$, $\boldsymbol{E}_w$, and $\boldsymbol{C}_w$ being constant matrices describing the sea state (Fossen, 2020, p. 254). If environmental forces and system noise is ignored or captured by a constant bias, $\dot{\boldsymbol{b}} = \boldsymbol{0}$, as assumed in Section 2.2.4, the observer design model (3.3.1) reduces to (3.2.1). Therefore, the control design model and observer design model will be equivalent in this study.

[3]Observer estimates are used for feedback if measurement signals are lost (Sørensen, 2018, p. 75).

Chapter 4

Path-planning

Autonomy of mobile robots, in general, is enabled by (Petereit et al., 2012):

- The use of sensors for the robot to locate itself and process its environment,
- the construction and update of a map of its environment, and
- the planning of future motions towards a particular target.

In other words, autonomous sailing relies on an intelligent and interconnected GNC system. This chapter is dedicated to the path-planning unit: First, a theoretical background on the chosen methods is provided, followed by details on the design and implementation. As stated in Chapter 1, path-planning is the task of tracing out a feasible and optimal path for the ASV from a starting point, typically its current position, to a desired final destination. The path search techniques examined here were selected for their compatibility with grid maps, and hence with the BINN guidance model, as well as their capability of finding WPs in a continuous space. A drawback of a path definition based on WPs is the discontinuity of the first derivative at the WPs when linked together by straight line segments. However, it will be the task of the path generator (see Section 5.2) to post-process the planned sequence of WPs and generate a smooth and feasible trajectory for the vessel.

4.1 Theoretical background

The path-planning problem has been subject to research across industries for several decades, as briefly covered in Sections 1.2 and 1.3. In the following, the BINN guidance model of Yang and Meng (2001) is presented, as well as the hybrid-state A* search algorithm from Richards, Sharma, and Ward (2004). The latter is an enhancement of the established and widely applied A* search algorithm by Hart, Nilsson, and Raphael (1968). Furthermore, applications of and theory behind MPC and mathematical programming is explored. It is based on the compendium written by Foss and Heirung (2016) as well as the work in, among others, Schouwenaars et al. (2001) and Chen, Hopman, and Negenborn (2018). In Section 1.3 compliance with rules and regulations is emphasized as a way to obtain predictable actions. Therefore, a brief presentation of rules applicable to ASVs, although still in their infancy, concludes this section.

4.1.1 BINN as a guidance model

A dynamic guidance model, containing information about the external environment, is a prerequisite for ASVs to reactively plan and replan a path to follow during maneuvering and voyaging. The BINN derived in Yang and Meng (2001) can be such a model. It is a topologically organized map within a finite dimensional state space, such as the Cartesian grid. A position in the grid is related to a specific neuron in the map. Each neuron is connected to a set of neighboring neurons which constitutes its receptive field $\mathcal{R}_i$, and it responds to stimulus within $\mathcal{R}_i$ only. Targets and obstacles can be recognized as peaks and valleys in the neural activity landscape by properly defining external inputs and internal neural connections. Yang and Meng (2001) proved stability and convergence of the BINN with Lyapunov stability theory. That is, the dynamics of the network will eventually arrive at an equilibrium state.

The dynamic neural activity landscape is based on the biophysical model of current flow through a nerve fiber membrane in Hodgkin and Huxley (1952). Neural activity z_i of the ith neuron with k neighboring neurons is calculated according to

$$\frac{dz_i}{dt} = -Az_i + (B - z_i)\Big([I_i]^+ + \sum_{j=1}^{k} w_{ij}[z_j]^+\Big) - (D + z_i)[I_i]^-. \tag{4.1.1}$$

Here, A denotes the passive decay rate, B and D are the upper and lower bounds on neural activity, and $[a]^{\pm} = \max\{\pm a, 0\}$ specify excitatory and inhibitory inputs, respectively. Furthermore, external inputs to the ith neuron are given according to

$$I_i = \begin{cases} \kappa, & \text{if there is a target} \\ -\kappa, & \text{if there is an obstacle} \\ 0, & \text{otherwise} \end{cases} \tag{4.1.2}$$

with $\kappa \gg B$ being a very large positive constant. The strength of the connection between two neurons i and j is captured by the weight

$$w_{ij} = f(d_{ij}), \tag{4.1.3}$$

where d_{ij} is the distance, or metric in mathematical terms (see e.g. Weisstein (n.d.[a])), between two neurons i and j, and f is a monotonically decreasing function

$$f(a) = \begin{cases} \mu/a, & \text{if } 0 < a < r_0, \\ 0, & \text{if } a \geq r_0. \end{cases} \tag{4.1.4}$$

Activity propagation among neurons is determined by a constant μ, and r_0 denotes the radius of $\mathcal{R}_i$. The target is rendered globally attractive as positive neural activity propagates through the entire state space over neural connections. Conversely, obstacles repel only locally to prevent collision (Yang and Meng, 2001). This is evident from (4.1.1), as excitatory inputs stem from both targets, $[I_i]^+$, and lateral connections to neighboring neurons, $\sum_{j=1}^{k} w_{ij}[x_j]^+$, whereas the inhibitory inputs exclusively come from obstacles via $[I_i]^-$.

A path can be obtained by climbing the BINN landscape according to the steepest ascent rule. Then each move will be in the direction along which the neural activity increases the most. Equivalently, if formulated as a minimization problem with targets identified as minima of the landscape, the next move would be along the steepest descent direction (Nocedal and Wright, 2006). Following the definition given by Yang and Meng (2001), with targets as peaks in the BINN landscape, the next vehicle position in $\mathbb{R}^2$ is determined from

$$\boldsymbol{p}_n \Leftarrow z_n = \max\{z_j,\ j = 1, 2, \ldots, k\}. \tag{4.1.5}$$

When arriving at the next position $\boldsymbol{p}_n$, this becomes the current position $\boldsymbol{p}_c$. If the neural activity of all neighboring neurons is not larger than that of the current position, the vehicle will not move. An example of a BINN landscape and the corresponding path of steepest ascent (4.1.5) is displayed in Figure 4.1.1.

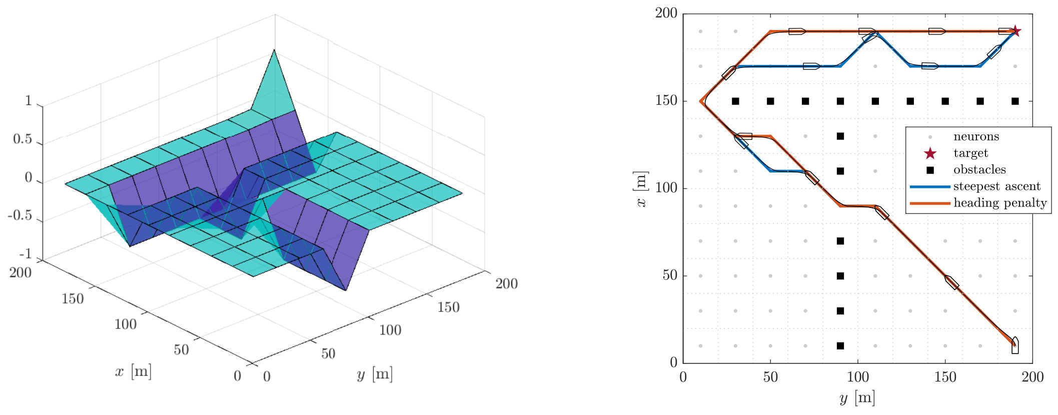


Figure 4.1.1: BINN landscape and the paths found by traversing the landscape according to the steepest ascent rule and the method in Scibilia, Jørgensen, and Skjetne (2012).

An enhanced method for traversing the neural activity landscape is formulated in Scibilia, Jørgensen, and Skjetne (2012). The steepest ascent rule (4.1.5) is augmented with a penalty term on changes in navigation direction to eliminate unnecessary turns in the path. Mathematically, the next position $\boldsymbol{p}_n$ is determined by evaluating the weighted dynamic activity of neighboring neurons,

$$\boldsymbol{p}_n \Leftarrow z_n = \max_{z_j : \boldsymbol{p}_j \in A(c,j)} \Big\{ \Big(1 - \lambda \frac{\mathsf{diff}(\psi_c - \psi_j)}{\pi}\Big) z_j \Big\}. \tag{4.1.6}$$

The navigation direction is represented by the vehicle heading ψ with the operator $\mathsf{diff} : [0, 2\pi) \times [0, 2\pi) \to [0, \pi]$ returning the smallest angle difference, and $\lambda > 0$ is a tuning parameter defining the magnitude of penalty on heading change. If the heading at a neighboring position ψ_j is equal to the current heading ψ_c, there will be no reduction in the corresponding neural activity z_j, and thereby (4.1.6) favors neighbors straight ahead. Neighboring neurons j to the present position $\boldsymbol{p}_c$ are provided by an adjacency matrix $A(c, j)$ as defined in the following section.

Graph theory for efficient computations on neural networks

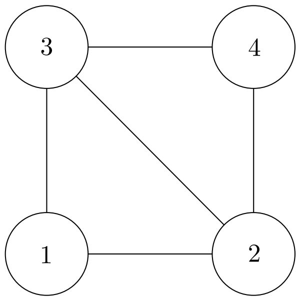
A *graph* is a suitable data structure to represent the network of neurons in a BINN model. Planning a path through the BINN requires for a systematic definition of the set of neighbors that are within reach. Formally, a graph can be represented by the pair (Trudeau, 1993):

$$G = (V, E), \tag{4.1.7}$$

where V is the node – or vertex – set of graph G, and E is its set of edges. A node can exist without incident edges, whereas an edge must have one node at each end. Applied to the path planning problem, a node represents a geometric location in space (e.g. a position in the Cartesian grid), and an edge connects two nodes if a node is visible and approachable from the other. Thus, two nodes connected by an edge form a pair of neighbors.

A graph can be depicted as a diagram. Yet, according to Trudeau (1993), the diagram has properties beyond those related the graph, and the fact that it is a plane surface is such an incidental feature. In a diagram, nodes are typically identified as dots, whereas an edge is a line connecting two nodes. By evaluating which nodes that are directly coupled together by an edge, one may establish an adjacency matrix indicating the pairs of neighboring nodes in the graph. An element (i, j) in this matrix takes the value 1 if node i and node j are adjacent, otherwise 0. Figure 4.1.2 displays a graph to the left and the corresponding adjacency matrix to the right. Since the graph is undirected in this case, the resulting adjacency matrix is symmetric.

Costs can be assigned to both nodes and edges in the graph. These represent a penalty on visiting or approaching a node, respectively. For instance, there is a cost in the form of neural activity assigned to each node in the BINN model described in Section 4.1.1. A path is obtained by traversing the graph along edges that give the minimum or maximum total cost, depending on whether the path-planning problem is formulated as a minimization or maximization problem. Thus, for the particular landscape in Figure 4.1.1 where minima indicate the locations of obstacles and maxima represent targets, the optimal path has the maximum total cost. In graph theory, the ordered sequence of adjacent nodes that define the path is termed a *walk*.



$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 4.1.2: A graph and its adjacency matrix. *Adapted from Nordstoga (2019).*

4.1.2 Hybrid A*

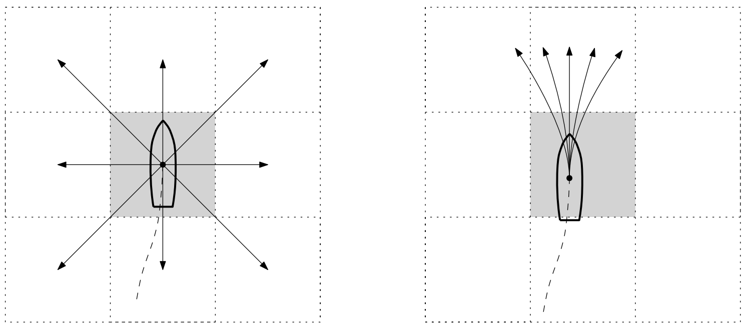
A popular method for path-planning in a grid is the A* search algorithm proposed by Hart, Nilsson, and Raphael (1968). It is an informed search algorithm that finds the optimal path – if one exists – by use of *heuristics*. Heuristics can be seen as a "rule of thumb", or a strategy based on experience, to shorten the search time. It relies on a priori knowledge about the problem domain, and is used by the A* algorithm to systematically explore nodes in the grid to obtain a minimum cost path. This way, a heuristic approach is incorporated into a formal mathematical graph search strategy. According to Hart, Nilsson, and Raphael (1968), A* is an optimal algorithm in the sense that the least number of nodes are expanded in order to guarantee that the path is minimum cost.

Although the A* algorithm produces a minimum cost path, it is rarely the true optimal solution in a continuous workspace. The traditional A* algorithm is constrained to grid edges and allows for the vehicle to turn on the spot – thereby violating nonholonmic constraints, if any. Moreover, the resulting paths are generally non-smooth. There have been several efforts to circumvent these shortcomings, and real-world implementations commonly support any-angle pathing. Any-angle algorithms interleave the A* search and path smoothing, and include Block A*, Field D*, and Theta* (see Nash and Koenig, 2013). Another method taking the continuous nature of the search space into account is the hybrid-state A* algorithm in Richards, Sharma, and Ward (2004), hereinafter referred to as hybrid A*. It was demonstrated in obstacle avoidance problems for aircraft, and has been adopted by Dolgov et al. (2008) and Petereit et al. (2012), among others, to plan paths for nonholonomic mobile outdoor robots.

Hybrid A* is constructed similarly as the traditional A* algorithm. Both algorithms require that the workspace is discretized into connected, and often equisized, cells. The key difference between traditional A* and the hybrid version is how nodes are expanded in the search for a feasible path: A* is restricted to search among nodes located at the cell centers, whereas hybrid A* utilizes predefined motion primitives to obtain nodes, or WPs, in continuous space. Motion primitives are the smallest entities of a path, defined such that (Petereit et al., 2012):

- The traveled distance is sufficient to leave the current grid cell,
- the curvature does not violate a maximum turning angle, and
- the change in heading is a multiple of the discretization size in the heading dimension.

Figure 4.1.3 illustrates the difference between the traditional A* and the hybrid A* methods in terms of possible moves and position of nodes within a cell.



(a) conventional A* (b) hybrid A*

Figure 4.1.3: Possible moves for the A* search methods. *Adopted from Petereit et al. (2012).*

Nodes are organized in two sets named *CLOSED* and *OPEN*. In a hybrid A* path search, each node, or end position of a motion primitive, is stored alongside the grid cell in which the node is located. The closed set contains all grid cells with nodes that have been expanded, or those occupied by obstacles. Motion primitives from expansions of different nodes may end in the same cell. Therefore, the grid cells are separately closed for each possible heading. In the special case of obstacles, the obstructed nodes are closed for all headings in the discrete set. The open set contains all nodes that have been encountered but not yet expanded. At each iteration, a node from the open set is examined and expanded with the predefined motion primitives. Resulting child nodes n_c that are clear of obstacles are added to the open set, and the parent node is moved to the closed set. The procedure is repeated until the target is within a radius of acceptance. Due to the definition of motion primitives, a hybrid A* search will generally not end at the exact target state.

As for the A* algorithm, nodes in the open set are expanded based on their cost. The cost of all nodes in the open set is evaluated, and the node n of the least cost is expanded and moved to the closed set. The total cost of the path from n to the goal is given by

$$f(n) = g(n) + h(n), \tag{4.1.8}$$

where $g(n)$ and $h(n)$ denote the actual cost along an optimal path from the start node n_s to n and from n to the target n_t, respectively. An estimate of the former is typically the smallest cost so far discovered to node n, whereas information from the problem domain is needed to accurately estimate the latter. When expanding a node, a pointer links the child nodes to its parent. Eventually, as the target is reached, the path can be reconstructed by tracking parent nodes – starting at the node evaluated last and ending at the initial node. The general procedure is given in Algorithm 4.1.1. Note that the hybrid A* yields a suboptimal path by definition, as it searches the continuous space using predefined motion primitives, but according to Richards, Sharma, and Ward (2004) it will be close to the optimum.

Algorithm 4.1.1: General A* search algorithm

```
initialize: $CLOSED \leftarrow \emptyset$, $f(n_s) \leftarrow h(n_s)$, $OPEN \leftarrow n_s$
while $OPEN \neq \emptyset$ do
    $n \leftarrow$ getMinCost($OPEN$)                    // Find node with lowest $f(n)$
    if targetReached($n_t, n$) then
        return getPath($n$)                          // Extract the shortest path
    end
    for $n_c \leftarrow$ getChild($n$) do
        $cost \leftarrow g(n) +$ dist($n, n_c$)              // Distance from $n_s$ to $n_c$
        if $cost < g(n_c)$ then
            $g(n_c) \leftarrow cost$
            $h(n_c) \leftarrow$ dist($n_c, n_t$)             // Estimated distance to $n_t$
            $n_p \leftarrow n$                           // Pointer to parent node
            if $n_c \notin OPEN$ then
                $OPEN \leftarrow n_c$
            end
        end
    end
    $CLOSED \leftarrow n$
end
```

Search time is minimized by selecting proper heuristics. The special case with zero heuristic cost, $h(n) = 0$, is equivalent to Dijkstra's (1959) algorithm. Then, any node is assumed to be equally far away from the target as only $g(n)$ is decisive in the path search. In general, considerably fewer nodes are expanded by using an *admissible* heuristic function, such as the Euclidean distance,

$$h(n) = \sqrt{(x_{n_t} - x_n)^2 + (y_{n_t} - y_n)^2} = \sqrt{\Delta x^2 + \Delta y^2}, \tag{4.1.9}$$

between the position of node n denoted by (x_n, y_n) and the target position (x_{n_t}, y_{n_t}). A heuristic function is said to be admissible if it is optimistic and never overestimates the path cost to the target, or, in other words, if the estimated heuristics is any lower bound of the true $h(n)$. The computational effort in calculating a nonlinear cost, such as (4.1.9), can be significant compared to the cost of expanding some extra nodes. An alternative is $\frac{1}{2}(\Delta x + \Delta y)$, which is strictly less than (4.1.9) and hence admissible, but at the expense of using less knowledge about the problem domain. Thus, the choice of $h(n)$ will be a compromise between admissibility, heuristic effectiveness, and computational efficiency (Hart, Nilsson, and Raphael, 1968).

4.1.3 Mathematical optimization and MPC

There are several applications of MPC to the control problem of a GNC system. For instance, Oh and Sun (2010) presents an MPC design that merges line-of-sight (LOS) guidance and path-following control for surface vessels, Li et al. (2016) proposes an MPC scheme incorporating neural-dynamic optimization for trajectory tracking of nonholonomic mobile robots, and in Chen, Hopman, and Negenborn (2018) an MPC-based approach is applied to coordinate and control a train formation of ASVs. Yet, MPC seems less explored for path-planning, but there are some applications to COLAV problems. Bousson (2008), as an example, presents an MPC-based method to compute collision-free trajectories for aircraft in a given control area. Optimization is performed according to priority indices, favoring aircraft closest to their destination. Recent maritime applications of MPC are found in Eriksen and Breivik (2017) and Hagen et al. (2018). The proposed MPC-based anti-collision algorithm is a supplement to already existing GNC systems on ASVs, and is shown to handle both static and dynamic obstacles. The inherent design flexibility of the mathematical programming framework, accepting both dynamic models and a diverse set of operational constraints, makes it advantageous to other COLAV methods. Moreover, Chen, Hopman, and Negenborn (2018) argues that the receding horizon scheme facilitates early detection of conflicts. Nevertheless, if the set of constraints gets large and with a nonlinear model, MPC may become computationally inefficient for the purpose of path-planning.

Mathematical programming

A mathematical program is, in general, composed of three elements (Foss and Heirung, 2016):

- A scalar objective function, $f(\,\cdot\,)$, to be minimized or maximized,
- decision variables denoted by a vector $\boldsymbol{x}$, and
- equality and inequality constraints, $c_i(\,\cdot\,)$.

The program can be formulated in terms of these three components as

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{4.1.10a}$$

$$\text{subject to} \quad c_i(\boldsymbol{x}) = 0, \quad i \in \mathcal{E} \tag{4.1.10b}$$

$$c_i(\boldsymbol{x}) \geq 0, \quad i \in \mathcal{I}, \tag{4.1.10c}$$

where $\mathcal{E}$ and $\mathcal{I}$ are the index sets for the equality and inequality constraints, respectively. A maximization problem $\max -f(\boldsymbol{x})$ is equivalent to (4.1.10), and yields the exact same solution apart from the opposite sign in the objective function value. All feasible solutions to (4.1.10) can be expressed in terms of a set Ω given by

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid (c_i(\boldsymbol{x}) = 0, i \in \mathcal{E}) \wedge (c_i(\boldsymbol{x}) \geq 0, i \in \mathcal{I})\}. \tag{4.1.11}$$

The objective is to find the global minimum $\boldsymbol{x}^*$ within (4.1.11), according to

$$\boldsymbol{x}^* \in \Omega, \qquad \text{if} \quad f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega. \tag{4.1.12}$$

Yet, no solution (4.1.12) to a program (4.1.10) will be universally optimal, but is conditional on the objective function. Hence, $f(\,\cdot\,)$ must be carefully selected. Moreover, in the search for a solution one might get trapped in a local optimum. A local minimum $\boldsymbol{x}'$ is defined by

$$f(\boldsymbol{x}') \leq f(\boldsymbol{x}), \quad \boldsymbol{x} \in \left|\boldsymbol{x} - \boldsymbol{x}'\right| < \epsilon, \tag{4.1.13}$$

that is, a minimum within some neighborhood of radius ϵ. It is generally easier to search for a local rather than a global optimum. The former coincides with the latter in convex problems, and for this reason mathematical programs are preferred to be convex. Problem (4.1.10) is said to be convex if both of the following conditions are satisfied:

- The objective function $f(\,\cdot\,)$ is a convex function, and
- the feasible set Ω is a convex set.

A set is said to be convex if any straight line connecting two arbitrary points in the set is exclusively inside the set, and a function is said to be convex if its epigraph is convex (Nocedal and Wright, 2006, p. 8). The two conditions are illustrated in Figure 4.1.4.

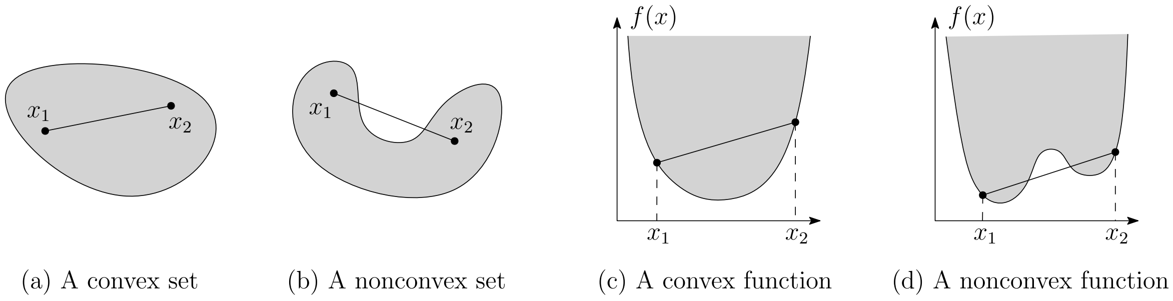


Figure 4.1.4: A comparison of convex and nonconvex sets and functions. *Adopted from Foss and Heirung (2016).*

Linear programs (LPs) are a special class of convex optimization problems. They are characterized by a linear objective function and linear constraints:

$$\min_{\boldsymbol{z} \in \mathbb{R}^n} d^\top \boldsymbol{x} \tag{4.1.14a}$$

$$\text{subject to} \qquad a_i^\top \boldsymbol{x} - b = 0, \quad i \in \mathcal{E} \tag{4.1.14b}$$

$$a_i^\top \boldsymbol{x} - b \geq 0, \quad i \in \mathcal{I}. \tag{4.1.14c}$$

LPs are often preferred over nonlinear programs since they are generally easier to solve, for instance by the well-known simplex method (Nocedal and Wright, 2006, p. 355), and they have an intuitive graphical interpretation. Nevertheless, quadratic programs (QPs) seem to be more widely employed. They differ from the LP formulation (4.1.14) in that the objective function (4.1.14a) is replaced by a quadratic cost $\boldsymbol{x}^\top \boldsymbol{Q} \boldsymbol{x} + d^\top \boldsymbol{x}$. A QP is convex if the Hessian matrix $\boldsymbol{Q}$ is positive semidefinite (i.e. $\boldsymbol{Q} \succeq 0$), and will then typically be similar in difficulty to an LP (Nocedal and Wright, 2006, p. 449).

Model predictive control

Dynamic optimization is required for systems characterized by frequent changes over time. MPC merges dynamic optimization and control through the explicit use of a model to predict future system outputs and by solving a program (4.1.10) to obtain optimal input commands. As illustrated in Figure 4.1.5, open-loop optimization is combined with a feedback loop to account for model errors and dynamics occurring in between two time instants. The open-loop optimization program is initialized to the current state of the system $\boldsymbol{x}_t$ and solved at each sampling instant within a finite horizon N_h. Thus, an optimal sequence of inputs is obtained for the entire horizon, but only the first input $\boldsymbol{u}_t$ is applied to the system. The steps of state feedback MPC are outlined in Algorithm 4.1.2. In practice, only an estimate $\hat{\boldsymbol{x}}_t$ of the current state will be available, and the procedure is then referred to as output feedback MPC. The estimate is based on measured data $(\boldsymbol{u}_t, \boldsymbol{y}_t)$ up until time t.

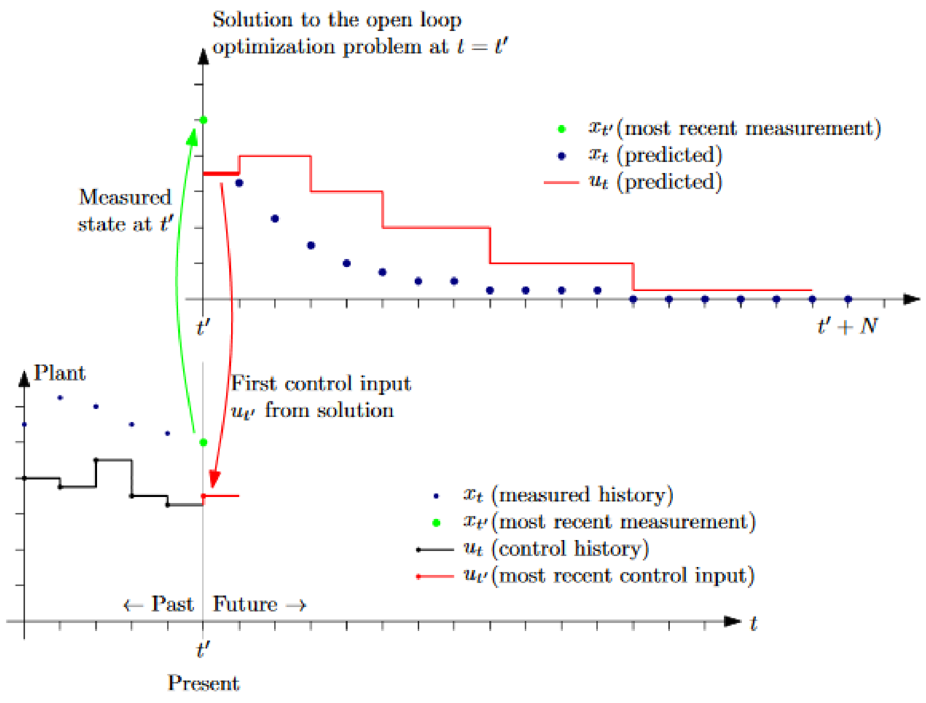

Solution to the open loop optimization problem at $t = t'$
● $x_{t'}$ (most recent measurement)
● x_t (predicted)
— u_t (predicted)
Measured state at t'
t'
$t' + N$
Plant
First control input $u_{t'}$ from solution
● x_t (measured history)
● $x_{t'}$ (most recent measurement)
— u_t (control history)
— $u_{t'}$ (most recent control input)
← Past
Future →
t
t'
Present

Figure 4.1.5: Visualization of the MPC procedure with input blocking. *Adopted from Foss and Heirung (2016).*

Computers operate at discrete time instants, and hence require discrete-time optimization programs. Therefore, the system model is sampled at equidistant points in time:

$$\boldsymbol{x}_{t+1} = g(\boldsymbol{x}_t, \boldsymbol{u}_t), \quad \boldsymbol{x}_t \in \mathbb{R}^{n_x}, \quad \boldsymbol{u}_t \in \mathbb{R}^{n_u}. \tag{4.1.15}$$

The discrete-time model (4.1.15), together with initial conditions of the system, are introduced as equality constraints to the MPC optimization program (4.1.10). Limits on actuators and states constitute the inequality constraints.

In an MPC scheme, the frequently used quadratic objective function is typically on the form

$$f = \sum_{t=0}^{N-1} (\boldsymbol{x}_{t+1} - \boldsymbol{x}_{t+1}^{ref})^\top \boldsymbol{Q}_{t+1} (\boldsymbol{x}_{t+1} - \boldsymbol{x}_{t+1}^{ref}) + \boldsymbol{u}_t \boldsymbol{R}_t \boldsymbol{u}_t, \tag{4.1.16}$$

where $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ are positive semi-definite weighting matrices and $\boldsymbol{x}_t^{ref}$ is the reference trajectory. Alternatives to the quadratic cost (4.1.16) include the use of first norms, that is the sum of absolute values. A program with a 1-norm objective function, subject to linear constraints, can be transformed into a linear program by adding binary decision variables. Then, being a linear program with both real and integer decision variables, it is referred to as a mixed-integer linear program (MILP). MIPs, in general, seem suitable for waypoint path-planning, and especially if the search space is discontinuous. Therefore, a brief introduction to MIPs and some applications to the path-planning problem are presented next.

Algorithm 4.1.2: State feedback MPC. *Courtesy of Foss and Heirung (2016).*

```
1 for $t' = 0, 1, 2, \dots$ do
2     Get the current state $\boldsymbol{x}_t$.
3     Solve the dynamic optimization problem, initialized to $\boldsymbol{x}_t$, over a horizon $N_h$.
4     Apply the first control move $\boldsymbol{u}_t$ from the solution above.
5 end
```

Mixed-integer programming

The general formulation of a MIP is obtained by altering (4.1.10) to include integer variables $\boldsymbol{y}$. The resulting program is then given by

$$\min_{\boldsymbol{x}\in\mathbb{R}^n,\ \boldsymbol{y}\in\mathbb{Z}^q} f(\boldsymbol{x}, \boldsymbol{y}) \tag{4.1.17a}$$

$$\text{subject to} \quad c_i(\boldsymbol{x}, \boldsymbol{y}) = 0, \quad i \in \mathcal{E} \tag{4.1.17b}$$

$$c_i(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \quad i \in \mathcal{I}. \tag{4.1.17c}$$

Integer variables are defined in terms of a vector $\boldsymbol{y} \in \mathbb{Z}^q$, where q denotes its dimension. Since the domain $\mathbb{Z}^q$ is disconnected, (4.1.17) will always be a non-convex problem (Foss and Heirung, 2016). Nevertheless, discrete variables are convenient to describe discrete decisions variables such as a sequence of waypoints.

There are some applications of MILP, the linear version of (4.1.17), to path-planning problems. In Schouwenaars et al. (2001) and Richards, Feron, et al. (2002), a program (4.1.17) with a 1-norm objective function is transformed it into a MILP by introducing slack variables and additional constraints. The objective is to generate fuel-optimal paths for multiple vehicles, avoiding both static and moving obstacles. To solve the MILP, it is implemented in the framework of the mathematical programming language AMPL and passed to the CPLEX solver. CPLEX is applicable to LPs and QPs, as well as their equivalents in terms of MIPs. In the case of large, nonlinear programs, Knitro is a suitable solver. Both solvers utilize a *branch-and-bound*[1] approach to obtain a solution.

[1]A technique for solving integer programming problems, by successive evaluation of "relaxations" in which integer variables are fixed or integrality constraints are temporarily ignored (Nocedal and Wright, 2006, p. 6).

4.1.4 Rules-compliant collision avoidance

As mentioned in Section 1.3, predictable behavior of an autonomous system may be achieved by complying with rules and regulations. In fact, rules are the core of a decisive system. IMO, being an organization under the United Nations responsible for the regulatory framework for international shipping, has lead the way in establishing regulations for ASVs. As part of the 98th through 101st sessions of the Maritime Safety Committee (MSC) there has been conducted a regulatory scoping exercise to identify regulations that already apply and areas that might require new regulations. At the latest session, the Committee approved interim guidelines for ASV trials that treat the authorization of participating ships, qualifications of personnel involved, and the risk associated with the trials, among others (IMO, 2019). The IMO scoping exercise was expected to be concluded at the MSC in May 2020 (IMO, n.d.).

To accommodate the introduction of autonomous technologies on national and regional levels, *classification societies*[2] have established guidelines for such novel systems and operational concepts. The directions in DNV GL (2018), among others, were developed to support both actors in the industry and regulatory bodies in documenting and assuring safe implementations. These are founded on the requirement for autonomous systems and functions to be at least as safe as operations of conventional vessels. In that regard, a minimum risk condition (MRC) to maintain a safe state is defined: When experiencing situations beyond normal operation, either due to deteriorating weather conditions or by system failures such as loss of propulsion, the vessel shall enter a state posing the least risk to life, property and the environment (DNV GL, 2018, pp.18-21). Included in the guidelines are methods, technical requirements, principles and acceptance criteria related to autonomous and remotely operated ships.

COLREGs apply to all vessels on high seas and connected waters, even vessels confined to waters under the jurisdiction of one coastal state. Although COLREGs were developed for manned ships, several rules can be adopted by an autonomous collision avoidance systems (Johansen, Perez, and Cristofaro, 2016), including those listed in Section 2.1. Yet, as pointed out in Porathe (2017), the ambiguities of COLREGs constitute a challenge when it comes to implementation of the rules in anti-collision algorithms. For instance, there are no precise definitions of the terms "safe speed" and "safe distance" in Rule 6 and 8, respectively. The former is simply declared a speed at which the vessel is able to deviate from the planned path or to stop in time to avoid collision, whereas no interpretation is given for the latter term. Obedience to COLREGs demands for a SA system that analyzes the traffic situation, environmental conditions, and area of navigation in relation to maneuvering characteristics of the vessel and hazards along the planned path (DNV GL, 2018, p.64). Robust processing of information from navigational sensors is critical to provide anti-collision algorithms with accurate data. Specifically, decision support systems required by DNV GL (2018, pp.105-107) on board an ASV include:

- A combination of detection technologies for safe maneuvering, such as video surveillance, radar, and laser-based systems, as well as a sound reception system.
- An approved system for accurate reading of electronic navigational charts to plan and execute a safe voyage, such as electronic chart display and information systems (ECDIS).

[2]Classification societies establish technical standards for the design, construction and maintenance of ships, and certify that a vessel and its systems are in accordance with the rules (DNV GL, 2018, p. 11).

- An automatic identification system (AIS) interconnected with radar(s) and ECDIS to assist in obstacle detection and classification.
- A system for observing local weather and vessel monitoring in terms of ship movements and hull stress.
- Speed logs to continuously measure the speed and distance through water as well as speed over ground in both longitudinal and transversal directions.
- A minimum of two separate systems for continuous determination of the vessel heading relative to the geographic north.

In line with the delimitation and assumptions in Section 2.2.4, it is beyond the scope of this thesis to go into further details about such systems.

With regard to the planning of a safe voyage, DNV GL (2018, pp. 53, 60–63) considers both the route plan and the replanning of a path. A route plan shall be based on all pertinent information, including adequate charts and up-to-date nautical publications concerning navigational limitations and hazards, and be validated against general grounding avoidance criteria and feasibility with regard to environmental conditions and expected traffic. The route plan includes specifications of the demands for fuel, water, lubricants, chemicals, supplies, and other. The execution of substantial deviations from the planned route while underway shall be succeeded by the planning and validation of an amended route. Furthermore, actions to avoid collision must be compliant with COLREGs. Particularly relevant to the development of anti-collision algorithms, in addition to incorporating COLREGs, is the instruction to immediately bring the vessel to an MRC if the navigation situation exceeds the level of complexity the system can handle. According to DNV GL (2018, p. 63), parameters important to navigational systems include the number of objects to relate to, the transparency of planned actions, as well as the robustness of COLREGs-compliant algorithms.

In the literature, there are several approaches to COLREGs-compliant collision avoidance. Methods of extending bounding boxes around obstacles (Chiang and Tapia, 2018; Song et al., 2019), or based on vector algebra (Johansen, Perez, and Cristofaro, 2016; Zaccone, Martelli, and Figari, 2019) are mentioned in Section 1.3. The strategy in Thyri et al. (2020), about to be published, is instead to divide the NE-plane in two domains by a straight line based on the type of encounter as defined by COLREGs. A control barrier function is applied to ensure that the domain of the encountered ship is not entered, and to obtain predictable and effective maneuvers by the vessel under control. Thyri et al. (2020), and several others including Eriksen, Bitar, et al. (2020), use a geometrical interpretation scheme proposed in Tam and Bucknall (2010) to determine COLREGs-compliant actions. Therein, encounter types are categorized based on relative position and heading between the obstacle and the vessel under control. The scheme is adapted in the design of a COLREGs-compliant anti-collision strategy in this thesis as well.

4.2 Design and implementation

The path planner is composed of several units: A guidance model, an anti-collsion strategy, and a path search algorithm. The former provides a representation of the environment and is in the form of a BINN. External inputs to the BINN are associated with the presence of obstacles and a target, and define the shape of the neural activity landscape. Neural activities are adjusted according to the anti-collision strategy to stimulate for COLREGs-compliant behavior. The strategy is based on the scheme in Tam and Bucknall (2010) to identify the encounter type and the appropriate actions. Thereafter, paths are obtained by applying either a hybrid A* algorithm or a MIP in an iterative manner. The repetitive call to the path-planning algorithm enables replanning of the path based on updates of the BINN. Implementation details related to each unit and the modeling of dynamic obstacles is treated in the following. Also, measures to evaluate the performance of the path-planning algorithms are presented.

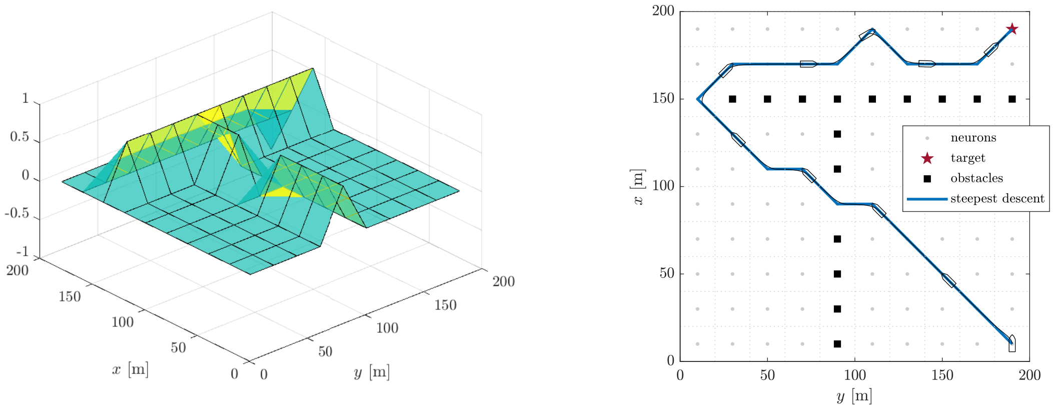


Figure 4.2.1: BINN landscape and the path obtained by traversing the landscape according to the steepest descent rule.

4.2.1 The BINN guidance model

A keystone of the path-planning algorithms presented in this thesis is a dynamic guidance model for stepwise path-planning. The BINN in Yang and Meng (2001) is explored for this purpose. Contrary to the case presented in Section 4.1.1, a minimization rather than a maximization problem is considered. This implies that targets and obstacles are recognized as minima and maxima, respectively. Thus, the vessel will traverse the landscape analogous to a ball rolling down a hill, before coming to rest at the bottom of a valley and thereby minimizing its potential energy. A minimization formulation can be obtained by modifying the shunting model (4.1.1) according to

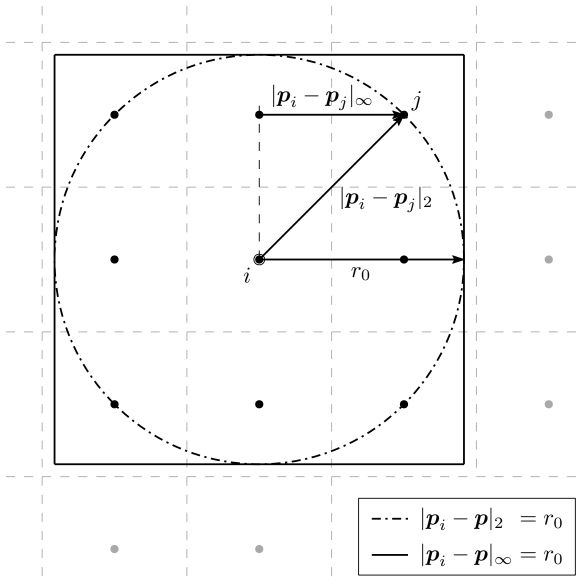
$$\frac{dz_i}{dt} = -Az_i - (D + z_i)\Big([I_i]^+ + \sum_{j=1}^{k} w_{ij}[z_j]^+\Big) + (B - z_i)[I_i]^-, \tag{4.2.1}$$

with external inputs defined as

$$I_i = \begin{cases} \kappa, & \text{if there is an obstacle} \\ -\kappa, & \text{if there is a target} \\ 0, & \text{otherwise.} \end{cases} \tag{4.2.2}$$

The equivalent BINN landscape to the example in Figure 4.1.1 in terms of the minimization formulation (4.2.1), and the path found by traversing the landscape according to a steepest *descent* rule are depicted in Figure 4.2.1. As anticipated, the resulting path is identical for both the minimization and maximization problem. Since the former is convention in mathematical optimization and due to the intuitive rolling ball analogy, the minimization formulation (4.2.1) is used. This implies that a feasible solution of minimum neural activity will be the optimum.

The neurons forming the BINN are structured in a graph. As argued in Section 4.1.1, this structure allows for a systematic definition of neighboring neurons in terms of an adjacency matrix. Moreover, the graph can be depicted in a diagram, which is easily linked to the Cartesian plane. Since the transit phase of an operation is considered, the grid partitioning was set to be of an order of magnitude similar to the vessel length. A downside of using such a coarse partitioning is that obstacles that are clearly smaller than the area covered by the grid cell, mark the entire cell as occupied. That said, the environment is typically more spacious in transit than, for instance, in harbors, and hence one may get away with a coarser partitioning. A coarse partitioning is also computationally less demanding, but at the expense of poorer reactive abilities.



$|\boldsymbol{p}_i - \boldsymbol{p}_j|_\infty$
j
$|\boldsymbol{p}_i - \boldsymbol{p}_j|_2$
i
r_0
$|\boldsymbol{p}_i - \boldsymbol{p}|_2 = r_0$
$|\boldsymbol{p}_i - \boldsymbol{p}|_\infty = r_0$

Figure 4.2.2: Neighbors to neuron i and the distance between neuron i and j according to the second and infinity norms.

In Yang and Meng (2001) the Euclidean, or second, vector norm is used to describe the distance d_{ij} between two neurons i and j. Here, another valid metric is applied: The infinity norm is generally defined by (Weisstein, n.d.[c])

$$|\boldsymbol{x}|_\infty := \lim_{p\to\infty} \left(\sum_i |x_i|^p \right)^{\frac{1}{p}} = \max_i |x_i| \tag{4.2.3}$$

for an n-dimensional vector $\boldsymbol{x} \in \mathbb{R}^n$. Applied to the Cartesian plane, the distance function can thence be calculated as

$$d_{ij} = \left|\boldsymbol{p}_i - \boldsymbol{p}_j\right|_\infty , \tag{4.2.4}$$

with $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ denoting the positions of the two neurons. Figure 4.2.2 illustrates the difference between the two distance functions based on the 2-norm and the ∞-norm. Furthermore, the receptive fields $\mathcal{R}_i$ of neuron i when the ∞-norm versus the 2-norm is applied with the same radius r_0 are delineated in Figure 4.2.2. Note that in order to enclose all connections to neurons in the closest vicinity of neuron i, the radius of $\mathcal{R}_i$ must satisfy $r_0 \geq \max\{dX, dY\}$ with the ∞-norm, as against the larger radius $r_0 \geq \sqrt{dX^2 + dY^2}$ when using the 2-norm. Here, dX and dY represent the grid cell dimensions along the x- and y-axis, respectively. In relation with the connection weights defined by (4.1.3) and (4.1.4), it is apparent from Figure 4.2.2 that the ∞-norm distance function will weight all neighboring neurons equally, whereas the 2-norm yields a larger distance d_{ij} for diagonal connections and hence weight these less than horizontal and vertical ones. In other words, the choice of distance function will influence how neural activity propagates through the landscape.

Different search strategies are applied to the neural activity landscape in order to establish feasible paths. One such strategy is the hybrid A* algorithm presented in Section 4.2.2. Contrary to the simple steepest descent method, nonholonomic constraints are accounted for with this a strategy. Furthermore, a MIP was designed to minimize neural activity as well as heading changes. Proper weighting of the two objectives should stimulate the construction of a path that circumvents obstacles while respecting turn constraints of the vessel. The BINN parameters were tuned to obtain a suitable shape of the neural activity landscape and ease the search for a feasible path, with values given in Yang and Meng (2001) and Scibilia, Jørgensen, and Skjetne (2012) as a starting point. As noted in Yang and Meng (2001), the shape of the BINN landscape is almost solely determined by the passive decay rate A and the neural connection weight μ. BINN parameter values used throughout this thesis are listed in Table 4.2.1, and the path search strategies are explained in detail in Sections 4.2.2 and 4.2.3.

Table 4.2.1: BINN parameters

Parameter	Value [-]
κ	10
r_0	$2\max(dX, dY)$
A	10
B	1.0
D	1.0
μ	10
ϵ_{ss}	1e−10

Serving as a dynamic model, the BINN is updated with external inputs on regular basis. The termination criterion when evolving the BINN is based on the fact that the BINN converges to an equilibrium state (see Section 4.1.1). If the neural activity of all neurons in the network change less in absolute value than a small constant ϵ_{ss} between iterations, the BINN is considered to have reached steady-state and the iterative process is terminated. The BINN is not updated continuously, but rather at discrete points in time. To simplify the path-planning process, the environment is considered *quasi-static* during each search for the next WP. In a quasi-static system, dynamic effects are negligible, so that the system is evaluated as if it was static. As a consequence, the time step between each update of the BINN must be sufficiently small so that changes in the environment are properly captured by the BINN.

4.2.2 Hybrid A* implementation

The development of a hybrid A* algorithm was based on the A* implementation by Premakumar (2016) found on the *MathWorks File Exchange* and the theory presented in Section 4.1.2. A traversability graph is generated by evaluating the current quasi-static neural activity landscape: Peaks in the landscape are identified as obstacles and the node with the lowest neural activity is set as the target. Specifically, the target position is given by

$$\boldsymbol{p}_{n_t} \Leftarrow \min\{z_j,\ j = 1, 2, \dots, N_n\}, \tag{4.2.5}$$

where N_n is the total number of cells in the search space, and obstructed cells are defined by the discrete nodes satisfying

$$n_j \Leftarrow \{j \in \{1, 2, \dots, N_n\} : z_j > 0\}\,. \tag{4.2.6}$$

The latter set is closed for all possible discrete headings, as explained in Section 4.1.2. Remaining nodes represent free space. Nodes in the free space are expanded with motion primitives based on their total cost (4.1.8). Two different heuristic cost estimates were tested, and two methods of generating motion primitives were implemented. These are presented in the following.

Heuristics based on the Euclidean distance and a modification, taking the neural activities into account, are applied. First, the heuristic cost $h(n)$ is estimated by calculating the distance (4.1.9) from the current expanded node to the target. This choice will undoubtedly be an underestimate of the true $h(n)$ since it ignores the presence of obstacles and does not reflect nonholonomic constraints. Hence, this estimate of $h(n)$ provides an admissible heuristic as explained in Section 4.1.2. Second, (4.1.9) was augmented with a factor based on the neural activity associated with the cell in which the currently expanded node and the target node are located. Since the neural activity z_i of neuron i is bounded by -1 and 1 (see Table 4.2.1), the factor was set to $\frac{1}{2}(1 + z_i)$. Adding unity makes sure that the factor will always be positive, whereas dividing by 2 ensures that the heuristic will never be greater than (4.1.9), and hence still yield an admissible algorithm. Note that the neurons are associated with the cells, so that several nodes defined in the continuous space may share the same neural activity but their cost differ in the distance to the target. To shorten the search time, nonholonomic constraints are taken into consideration by adding a cost on direction changes along the path and the progression towards the target. However, the introduction of such costs must be made with caution since it may come at the expense of heuristic admissibility, as discussed in Section 4.1.2.

Motion primitives are either chosen to be straight line segments, or computed based on kinematic considerations. In either case the motion primitives have to satisfy the conditions stated in Section 4.1.2. A total of three motion primitives are generated at each node, spanning a feasible region that the vessel is able to reach. One of the motion primitives is directed straight ahead, whereas the remaining two are tilted to the port and starboard side. For the straight-line motion primitives, their length d is set to be slightly larger than the cell diagonal, that is

$$d = \sqrt{dX^2 + dY^2} + 1\text{e}{-5}. \tag{4.2.7}$$

Thus, the the motion primitive is guaranteed to leave the current cell. A strategy is needed to check whether the straight-line motion primitive crosses an obstructed cell. The implemented strategy is based on the Bresenham algorithm (see Flanagan, n.d.), which is typically used to determine the pixels indicating a slanted line on a computer screen. First the slope of the line segment is computed according to

$$a = \frac{x_c - x_p}{y_c - y_p}, \tag{4.2.8}$$

where the subscripts p and c denote parent and child, respectively. If $|a| \geq 1$, then x increases the most and is said to be the major dimension, and vice versa. With x being the major dimension, a point on the line segment lying within a neighboring cell will be given by

$$x_m = x_c - \text{sign}(x_c - x_p)dX, \tag{4.2.9a}$$

$$y_m = -\frac{1}{a}(x_c - x_m) + y_c, \tag{4.2.9b}$$

and similarly for y, but with the inverse of the slope (i.e $\frac{1}{a}$). By checking whether the cell within which (x_m, y_m) lies is obstructed or not, one can either validate the feasibility of the motion primitive or discard it. The angle of a straight-line motion primitive relative to another is set to be a multiple of the heading discretization step and denoted by an angle β. Constraints on path curvature can be handled through a proper choice for β.

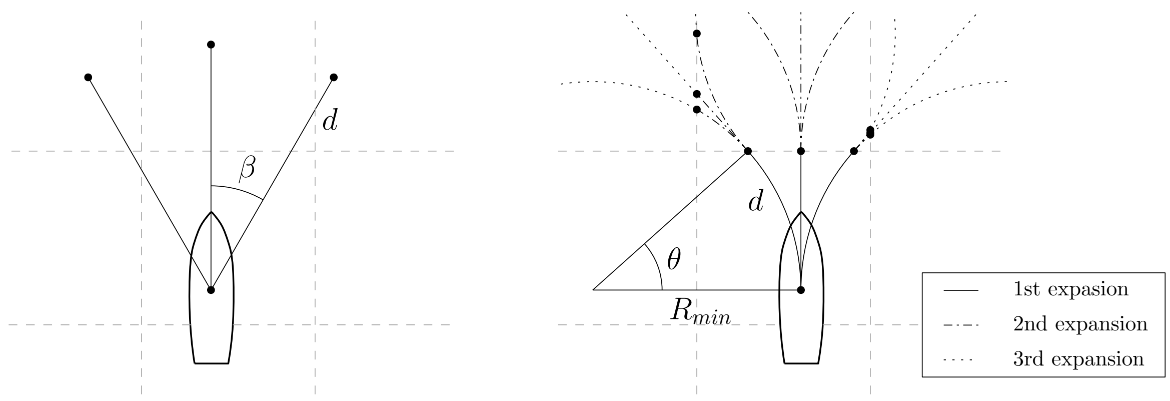


Figure 4.2.3: Motion primitives in the form of straight-line segments (left) and based on kinematic considerations (right).

On the contrary, kinematics-based motion primitives are constructed so that the turning ability of the vessel is considered in the first place. An arc based on the vessel's minimum turning radius R_{min} is drawn by incrementally increasing the central angle θ until an adjacent cell is encountered. The endpoint of a kinematic motion primitive is obtained from

$$\boldsymbol{p}_{end} = \boldsymbol{p}_{center} + R_{min} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \tag{4.2.10}$$

where $\boldsymbol{p}_{center} \in \mathbb{R}^2$ is the center of the turn circle determined by

$$\boldsymbol{p}_{center} = \boldsymbol{p}_0 + R_{min} \begin{bmatrix} \cos\left(\psi + \frac{\pi}{2}\right) \\ \sin\left(\psi + \frac{\pi}{2}\right) \end{bmatrix}. \tag{4.2.11}$$

Here, $\boldsymbol{p}_0$ denotes the position of the node being expanded. The motion primitive going straight ahead is also calculated from (4.2.10) and (4.2.11) by assigning a sufficiently large value to the turn radius R. Since R is either given by the vessel's minimum turn radius or a very large value for constructing the straight-line motion primitive, it is important to choose an appropriate increment of θ so that the corresponding increase in the length d is not too large. Therefore, the implemented increment of θ depends on R and is given by

$$d\theta = \frac{dX}{12\,|R|}. \tag{4.2.12}$$

The total length of the kinematic-based motion primitives will generally vary, and can be calculated from (Barile and Weisstein, n.d.)

$$d = R_{min}\theta, \tag{4.2.13}$$

given that θ is measured in radians. Since the endpoints (4.2.10) of all motion primitives are equally feasible, two successive WPs that are sufficiently close to another are merged together. Otherwise, undesired behavior might occur if the vessel is commanded towards a WP in its close vicinity. The heading along the kinematic motion primitive is related to θ according to $\psi = \theta + \frac{\pi}{2}$. It will not necessarily be within the discrete set of heading angles, and is therefore rounded to its nearest discrete value. The difference between the two methods of generating motion primitives is illustrated in Figure 4.2.3.

4.2.3 A mixed-integer program for path-planning

An attempt was made in formulating a MIP for the purpose of planning a path for an ASV from its current position to a target. The path was supposed to avoid obstacles and take inherent limitations of the vessel into account. The optimization program is based on the kinematic equations (2.1.2), which can be rewritten in terms of the unit velocity vector $\boldsymbol{z}^{\psi} := [\cos\psi,\ \sin\psi]^{\top}$ as

$$\dot{\boldsymbol{p}} = U\boldsymbol{z}^{\psi}, \tag{4.2.14a}$$

$$\dot{\boldsymbol{z}}^{\psi} = rS\boldsymbol{z}^{\psi}, \quad S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{4.2.14b}$$

Properties related to $\boldsymbol{z}^{\psi}$ being a *unit* vector include

$$z_i^{\psi} \in [-1,\ 1],\ i = 1, 2, \tag{4.2.15a}$$

$$\left|\boldsymbol{z}^{\psi}\right| = 1, \tag{4.2.15b}$$

and it is hence given by the unit circle. The second norm (4.2.15b) poses a nonlinear constraint on $\boldsymbol{z}^{\psi}$, but linear bounds can be established by using the 1st norm, generally given by (Weisstein, n.d.[c])

$$|\boldsymbol{x}|_1 := \sum_i |x_i|, \tag{4.2.16}$$

and the infinity norm (4.2.3). Linear constraints on $\boldsymbol{z}^{\psi}$ on $\boldsymbol{z}^{\psi}$ in terms of (4.2.16) and (4.2.3) can then be derived as

$$-z_1^{\psi} - z_2^{\psi} \leq -1 + My_1, \tag{4.2.17a}$$

$$-z_1^{\psi} + z_2^{\psi} \leq -1 + My_2, \tag{4.2.17b}$$

$$z_1^{\psi} + z_2^{\psi} \leq -1 + My_3, \tag{4.2.17c}$$

$$z_1^{\psi} - z_2^{\psi} \leq -1 + My_4, \tag{4.2.17d}$$

$$\sum_i y_i \leq 3, \quad y_i \in \{0, 1\} \tag{4.2.17e}$$

$$-1 \leq z_i^{\psi} \leq 1, \quad i = 1, 2, \tag{4.2.17f}$$

where $M \gg 1$ and together with (4.2.17e) ensures that at least one of the constraints (4.2.17a) through (4.2.17d) is satisfied. A graphical interpretation of the constraints (4.2.17) is the shaded area in Figure 4.2.4, and it can be easily verified that the unit circle constraint is relaxed to become $0.7 \le \left|\boldsymbol{z}^{\psi}\right| \le 1.4$.

The MIP is hence formulated as

$$
\begin{aligned}
\min \sum_{n_k \in V} y(k, n_k) Z(k, n_k) + \sum_{k=1}^{N_h} \lambda \Delta \psi_k \qquad & (4.2.18a)\\
\text{subject to} \qquad x_k = x_{k-1} + d z^{\psi}_{1,k-1}, \qquad & (4.2.18b)\\
y_k = y_{k-1} + d z^{\psi}_{1,k-1}, \qquad & (4.2.18c)\\
n_k \Leftarrow \boldsymbol{p}_{n_k} = \left(\lceil x_k \rceil, \lceil y_k \rceil\right), \qquad & (4.2.18d)\\
\Delta z^{\psi}_{2,k} \le \Delta z^{\psi}_{1,k}, \qquad & (4.2.18e)\\
\text{and} \quad (4.2.17). \qquad &
\end{aligned}
$$

in which the objective is to minimize neural activity, thereby searching for a path that circumvents obstacles and proceeds towards the target, and to avoid unnecessary changes in heading along the path similar to the heading penalty in Scibilia, Jørgensen, and Skjetne (2012). Proper weighting of the two objectives should thus stimulate the construction of a path that progresses towards the target, while circumventing obstacles and respecting turn constraints of the vessel. The Δ in front of a variable indicates the absolute difference in time for that variable, i.e. for the heading it is defined as $\Delta\psi_k := |\psi_k - \psi_{k-1}|$. Note that the ceil functions $\lceil\,\cdot\,\rceil$ can be reformulated as linear constraints. Nevertheless, the neural activity matrix $Z(\,\cdot\,,\,\cdot\,)$ imposes a nonlinearity to the program and hence increases the difficulty to find a solution, as discussed in Section 4.1.3.

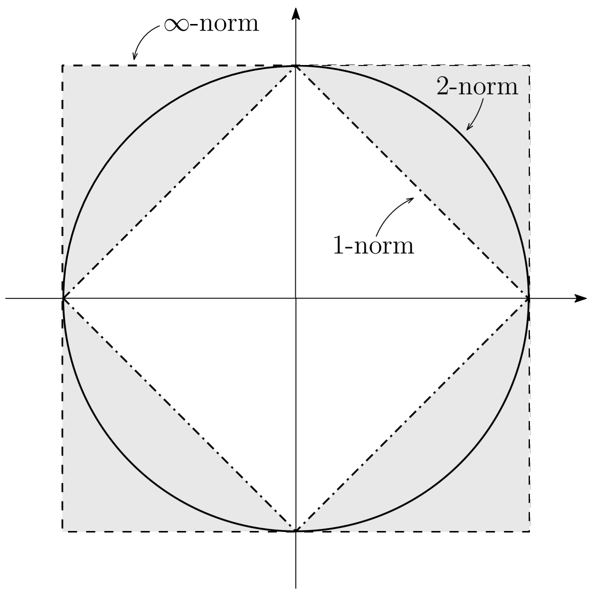


Figure 4.2.4: The first, second, and infinity unit norms.

4.2.4 Obstacle modeling

In addition to stationary obstacles, a virtual obstacle with forward speed is introduced to simulate a dynamic environment. In literature, a moving obstacles is typically referred to as a target ship (TS) (Huang et al., 2020). Here, the TS is conservatively modeled as a disk defined by a radius and center coordinates, similar to the definition in Wiig, Pettersen, and Savkin (2017). It is primarily assumed to move along a straight path at a constant speed, although this might be a poor and potentially dangerous assumption in some cases (Johansen, Perez, and Cristofaro, 2016). Therefore, to test the robustness of the COLREGs-compliant path-planning algorithms, random motions of the TS are simulated as well. The random and highly oscillating motions can be thought to represent position data from detection sensors, such as radars, being contaminated by noise. To circumvent inevitable collision states, two prerequisites for the simulations are adopted from Chen, Hopman, and Negenborn (2018) consistent with the assumptions in Section 2.2.4:

- The initial state of the ASV is feasible, and
- the first time an obstacle is detected, it will be avoidable.

In other words, an obstacle will be detected in ample time to take appropriate action to avoid collision.

4.2.5 Collision avoidance compliant with COLREGs

Actions by the ship under control, commonly named the ownship (OS) (Huang et al., 2020), are determined based on the type of encounter. As stated in Section 4.1.4, COLREGs apply to all vessels and are supported by guidelines for ASVs. Therefore, relevant rules from COLREGs will be the core of the anti-collision algorithm. Five situations are identified from the COLREGs rules listed in Section 2.1, namely:

- stand-on (SO),
- give-way (GW),
- head-on (HO), and
- overtaking on port ($\mathrm{OT_p}$) or starboard side ($\mathrm{OT_s}$).

In a crossing situation, the SO vessel is expected to keep its course and speed, whereas the GW vessel shall steer clear, and preferably in back, of the other vessel. If two vessels are approaching each other HO, both are obliged to alter course to their starboard side. Lastly, in an OT situation, the vessel coming up from behind of another vessel shall keep out of the way, and, accordingly, pass either on port or starboard side. Actions pertinent to the five encounter situations are illustrated in Figure 2.1.3. The type of encounter can be determined from the relative angle between the two ships,

$$\psi_{rel} = \psi_2 - \psi_1, \tag{4.2.19}$$

given that ψ_1 is the heading of the OS. With ψ_{rel} being calculated, the corresponding encounter type can be read off the pie diagram in Figure 4.2.5. Adopted from Tam and Bucknall (2010), and in analogy with the COLREGs definition of an OT situation, a HO scenario is considered if the TS is approaching the fore of the OS at an angle less than $22.5° = \frac{\pi}{8}$[rad]. Such a wide HO-region was recommended in order to enhance the robustness of the geometrical COLREGs interpretation scheme.

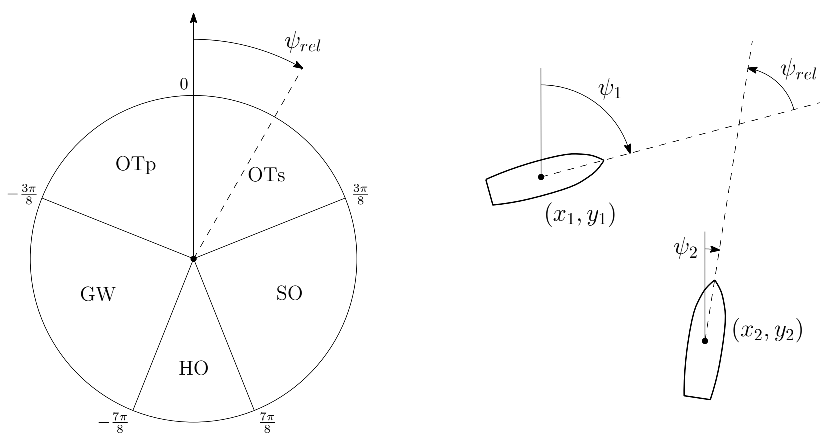


Figure 4.2.5: COLREGs-compliant action by the ownship when encountering another power-driven vessels at relative angle ψ_{rel}. *Adapted from Tam and Bucknall (2010).*

An adaptive *avoidance region* is generated in order to realize the appropriate actions by the OS. For instance, in a HO situation, it will cover an area in front of the TS and be extended to its starboard side. As the name suggests, the avoidance region should ideally neither be entered nor crossed by the OS. “Adaptive” indicates that the region is adjusted according to the sensed velocity of the approaching TS and the distance between the two vessels relative to the OS body-fixed frame. The latter is given by

$$\Delta p_1 = R(\psi_1)^\top \Delta p, \tag{4.2.20}$$

where the distance between the vessels in $\{n\}$, $\Delta p = [x_2 - x_1, y_2 - y_1]^\top \in \mathbb{R}^2$, is rotated to the OS frame by its heading ψ_1 through the transformation matrix $R(\psi_1)$. Similar to the approach in Chiang and Tapia (2018), the avoidance region will be considered a virtual obstacle of rectangular shape. Nodes located within the avoidance region are classified as obstacles, even if no physical obstacle is present, and the associated neurons receive a large positive external input κ consistent with (4.2.2). Consequently, the OS is repelled from moving into the avoidance region, and an appropriate definition of the region should trigger maneuvers compliant with COLREGs.

Once an encounter type is identified as active and appropriate actions are initiated, the OS should adhere to the rules that apply to that type of encounter until past and clear of the TS. In other words, subsequent changes in bearing between the two vessels should not change the encounter type. In that regard, evaluation of ψ_{rel} alone might be insufficient. Particularly, if the relative angle is close to a borderline between two encounter types there might be a rapid and constant change of the active encounter type, with inconsistent actions by the OS as a result. A possible solution is a state machine as presented in Eriksen, Bitar, et al. (2020, ch. 4.2.1): Transitions between encounter types have to go through a safe state, that is a state where COLREGs does not enforce any rules with respect to the TS. However, it is also pointed out in Eriksen, Bitar, et al. (2020, ch. 4.2.1), that a direct transition between encounter types might be appropriate in some cases. For instance, a direct transition from HO to an emergency state, that is a state covering situations in which the TS behaves unpredictably or is in proximity of the OS, could be effective to reduce the risk of collision and proceed to an MRC. The problem of adhering to one encounter type is not explored further in this thesis, but is left to future work.

4.2.6 Key performance indicators

Comparison of the different path-planning algorithms requires for some performance measures. KPIs are quantifiable measures that can demonstrate the effectiveness of the path-planning algorithms with respect to both the objectives and constraints of the problem. However, the choice of KPIs will influence the evaluation of the algorithms, and therefore the metrics should be chosen so that important aspects and requirements of the path-planning problem are evaluated. Here, three evaluation criteria were used in assessing the path-planning output: efficiency, optimality, and feasibility.

Efficiency is measured in terms of computation time. There are two options: Either assessing the CPU time or the elapsed time. The former is the amount of time a central processing unit (CPU) needs to execute a program, whereas the latter is the total duration of a task and includes waiting for input/output, etc. Although the planning algorithms have to respect the onboard processing capacity, as argued in Section 1.3, the elapsed time will probably be the bottleneck and is assessed here. In a dynamic environment, the ASV will acquire information through its sensors, and the path planning algorithm is executed on regular basis to adjust the path to the new knowledge. Therefore, it is important that the execution of the algorithm is relatively fast.

As well as being efficient, the algorithm should produce a path that is optimal according to some measure. Path length is a widely used criteria and is straightforward to calculate. Other evaluation criteria could be energy demands or work, and success rate in terms of avoiding obstacles or the distance to obstacles. The former criterion would depend on the fuel-consumption characteristics of the specific vessel, whereas path length is an vessel independent measure. The success rate, that is the fraction of collision-free voyages among a number of attempts, was deemed redundant since the resulting paths steered clear of obstacles in the environments simulated here. Moreover, assuming that obstacles are modeled with an additional margin, every unobstructed node will be equally traversable. Therefore the path length was deemed the most suitable measure of optimality.

Last, but not least, the planned path must necessarily be feasible. For a marine vehicle, changes in heading are undesired due to its large inertia, as discussed throughout Chapter 1. Contrary to a vacuum cleaner robot, a vessel moves in a fluid of significantly higher density and is not capable of turning on the spot. It has restrictions on turn radius and turn rate. Here, feasibility is evaluated based on changes in heading along the path, since it is included as an objective to be minimized in the proposed path-planning algorithms. The feasibility measure was quantified by computing the accumulated changes in heading $\sum |\Delta\psi|$ along the final path.

Chapter 5

Observer and path-following designs

The path-planning subsystem will be integrated and tested as part of a GNC system, as defined in Section 2.2. Herein, guidance comprises both path-planning and the generation of a continuous path from the planned WPs, of which the latter subsystem has not yet been addressed. In addition to guidance, a navigation system, in the form of an observer that estimates the vessel position, velocity, and the bias, as well as a control system for path-following is needed. This chapter provides a description of the design and implementation of a nonlinear passive observer, a path generator based on Bézier curves, and a DP backstepping controller. A simple LOS path-following law, used to quickly test and verify the path-planning algorithms, is also presented. Since the university and its laboratories were closed after the outbreak of COVID-19 in Norway and no physical experiments could be conducted, details on control allocation are left to Appendix C.

5.1 Observer

A nonlinear passive observer is introduced to reconstruct the state of the vessel $[\boldsymbol{\eta}, \boldsymbol{\nu}, \boldsymbol{b}]^\top$ from measurements. Passivity arguments simplify the tuning process significantly compared to the tuning of relatively large sets of parameters when using observer backstepping or Kalman filter-based designs. Moreover, the nonlinear passive observer guarantees global convergence of estimation errors – including bias terms – to zero (Fossen, 2011, p. 311), thereby solving the observer problem (2.2.2). Therefore, a passive observer design is applied to provide the state estimates $[\hat{\boldsymbol{\eta}}, \hat{\boldsymbol{\nu}}, \hat{\boldsymbol{b}}]^\top$.

According to Fossen (2011, p. 311), a passive observer is derived under the following two conditions:

1. Zero-mean Gaussian white noise terms are omitted, $\boldsymbol{w} = \boldsymbol{0}$ and $\boldsymbol{v} = \boldsymbol{0}$. Otherwise, the error dynamics will be uniformly ultimately bounded rather than globally converging according to Lyapunov function analysis.
2. Heading measurements are assumed to be accurate, so that $\boldsymbol{R}(y_3) = \boldsymbol{R}(\psi)$ and hence the heading measurement satisfies $y_3 = \psi + \psi_w \approx \psi$. The assumption is acceptable since the magnitude of wave-induced yaw disturbances will typically be less than 5° in extreme weather conditions and less than 1° in normal conditions.

Simulations reported in Fossen (2011, p. 319) demonstrate that all state estimates converge to their true values also with nonzero stochastic noise terms $\boldsymbol{v}$ and $\boldsymbol{w}$. Moreover, the second condition is equivalent to (3.2.2), and hence consistent with the linear maneuvering model (3.2.1).

The observer equations are obtained by copying the dynamics of the maneuvering model (3.2.1) and adding injection terms according to (Værnø and Skjetne, 2017)

$$\dot{\hat{\boldsymbol{\eta}}} = \boldsymbol{R}(\psi)\hat{\boldsymbol{\nu}} + \boldsymbol{L}_1\bar{\boldsymbol{\eta}}, \tag{5.1.1a}$$

$$\boldsymbol{M}\dot{\hat{\boldsymbol{\nu}}} = -\boldsymbol{D}\hat{\boldsymbol{\nu}} + \boldsymbol{R}(\psi)^\top\hat{\boldsymbol{b}} + \boldsymbol{R}(\psi)^\top\boldsymbol{L}_2\bar{\boldsymbol{\eta}} + \boldsymbol{\tau}, \tag{5.1.1b}$$

$$\dot{\hat{\boldsymbol{b}}} = \boldsymbol{L}_3\bar{\boldsymbol{\eta}}, \tag{5.1.1c}$$

and the corresponding error dynamics are given by

$$\dot{\bar{\boldsymbol{\eta}}} = \boldsymbol{R}(\psi)\bar{\boldsymbol{\nu}} - \boldsymbol{L}_1\bar{\boldsymbol{\eta}}, \tag{5.1.2a}$$

$$\boldsymbol{M}\dot{\bar{\boldsymbol{\nu}}} = -\boldsymbol{D}\bar{\boldsymbol{\nu}} + \boldsymbol{R}(\psi)^\top\bar{\boldsymbol{b}} - \boldsymbol{R}(\psi)^\top\boldsymbol{L}_2\bar{\boldsymbol{\eta}}, \tag{5.1.2b}$$

$$\dot{\bar{\boldsymbol{b}}} = -\boldsymbol{L}_3\bar{\boldsymbol{\eta}}. \tag{5.1.2c}$$

Here, the estimation errors are defined by $\bar{\boldsymbol{\eta}} := \boldsymbol{\eta} - \hat{\boldsymbol{\eta}}$, $\bar{\boldsymbol{\nu}} := \boldsymbol{\nu} - \hat{\boldsymbol{\nu}}$, and $\bar{\boldsymbol{b}} := \boldsymbol{b} - \hat{\boldsymbol{b}}$. The injection gains are given as matrices $\boldsymbol{L}_1$, $\boldsymbol{L}_2$, and $\boldsymbol{L}_3$, and are determined through a tuning process.

Stability of (5.1.2) can be evaluated by Lyapunov analysis. The equilibrium $[\bar{\boldsymbol{\eta}}, \bar{\boldsymbol{\nu}}, \bar{\boldsymbol{b}}]^\top = \boldsymbol{0}$ of (5.1.2) will be uniformly globally asymptotically stable provided that the following conditions on the injection gains and damping matrix are satisfied (Værnø and Skjetne, 2017):

- $\boldsymbol{D} + \boldsymbol{D}^\top \geq \boldsymbol{0}$
- $\boldsymbol{L}_1$, $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ are symmetric and positive definite,
- $\boldsymbol{L}_1$ and $\boldsymbol{L}_1$ are commutative, i.e. $\boldsymbol{L}_1\boldsymbol{L}_3 = \boldsymbol{L}_3\boldsymbol{L}_1$, and
- the symmetric matrices $(\boldsymbol{L}_1\boldsymbol{L}_2 + \boldsymbol{L}_2\boldsymbol{L}_1 - 2\boldsymbol{L}_3)$ and $(\boldsymbol{L}_3^{-1}\boldsymbol{L}_1 - \boldsymbol{L}_2^{-1})$ are positive definite.

The conditions on the injection gains will be met if they are diagonal with strictly positive entries. Then, each entry represents a weight on the corresponding DOF – either surge, sway, or yaw. Moreover, the condition on the damping matrix is generally true for a vessel in calm waters, since energy will be transferred to the water as waves are generated by vessel motions. For proofs of the stability properties of the observer (5.1.1) the reader is referred to Værnø and Skjetne (2017).

The bias terms $\boldsymbol{b}$ are included to account for unmodeled dynamics and slowly varying environmental forces, as mentioned in Chapter 3. Hence, $\hat{\boldsymbol{b}}$ will be nonphysical as it covers several components (Fossen, 2011, p. 306). In the special case of no bias, the equilibrium $(\bar{\boldsymbol{\eta}}, \bar{\boldsymbol{\nu}})^\top = \boldsymbol{0}$ can be proven uniformly globally exponentially stable. Exponential stability can also be accomplished if the estimate of the bias dynamics (5.1.1c) includes low-pass filtering, that is $\hat{\boldsymbol{b}} = -\boldsymbol{T}^{-1}\hat{\boldsymbol{b}} + \boldsymbol{L}_3$, $\boldsymbol{T} > \boldsymbol{0}$, and $\boldsymbol{D} + \boldsymbol{D}^\top$ is strictly positive (Fossen, 2011, p. 318). Although exponential stability is preferred, since the rate of convergence will be quantified, it is beyond the scope of this thesis to explore these alternatives.

5.2 Path generator

Given an ordered sequence of WPs from the path planner, $V_P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$, where $\boldsymbol{p}_i = (x_n, y_n)^\top$ represents the NE-coordinates of each waypoint. Then, path-generation is the problem of determining a continuous and feasible path that passes through each waypoint in V_P, as defined in Section 2.2.2. The problem can be relaxed by introducing a circle of acceptance around each waypoint. This relaxation implies that the path does not have to pass directly through each waypoint, but inside a region in its vicinity. The format of the generated path has to be compatible with the path-following controller in use, which in this work is the maneuvering controller presented in Section 5.3.

In the literature, there are several approaches to post-processing the planned WPs in order to obtain a feasible path for path-following. Post-smoothing, also known as the shortcut algorithm, is one such method, in which nodes are successively erased in between two vertices that are visible to one another (Nash and Koenig, 2013). This generally removes superfluous changes in heading along the planned path, but will not change the path topology in how obstacles are circumnavigated. A method for smoothing a path produced by a hybrid A* algorithm is proposed in Dolgov et al. (2008). It is a two-stage optimization procedure: The first step involves a nonlinear optimization program on the WPs, and improves both length and smoothness of the path. In the second step, non-parametric interpolation based on the conjugate gradient method (see e.g. Nocedal and Wright, 2006, ch. 5) is performed. Here, effort has been put into exploring a method based on Bézier curves presented in Knædal (2019). In the following, the design and implementation of the Bézier-based path generator is presented, but first a brief introduction to the Bézier curve is given.

5.2.1 The Bézier curve

In the early 1960s, the automotive industry put effort into deriving mathematical descriptions of free-form shapes. The aim was to accurately specify the geometry of car bodies, and thereby eliminate inaccuracies and ambiguities when interpreting paper drawings and sculpted clay models. As a result, Pierre Étienne Bézier, an engineer at Renault manufacturer, patented and popularized the Bézier curve. At Citröen, Paul de Casteljau finished similar work. However, his findings were kept internal to the company, and therefore Bézier's name is associated with the curve. The possibility of representing complex shapes with smooth curves proved to be an indispensable tool in computer-aided design (Farouki, 2012). More recently, the potential of using Bézier curves in path-planning and path-generation has been explored. Among others, Choi, Curry, and Elkaim (2008) presented two path-planning algorithms based on Bézier curves for an autonomous vehicle with waypoints and corridor constraints, and Jolly, Kumar, and Vijayakumar (2009) proposed a path-planning technique based on Bézier curves in a multi-agent system. A maritime application is found in Hassani and Lande (2018), where Bézier curves are exploited as basis for generating paths for an ASV similar to the method in Knædal (2019).

A Bézier curve is a planar parametrization, $\boldsymbol{B}(\theta) = [x(\theta), y(\theta)]^\top \in \mathbb{R}^2$. It is defined by a set of control points $\boldsymbol{P}_i,\ i \in \{0, \ldots, n\}$, where n represents the degree of the curve. The first and last control points are endpoints of the curve, whereas intermediate ones do not necessarily lie on the curve. A general formulation of the Bézier curve is given by (Hassani and Lande, 2018; Knædal, 2019)

$$\boldsymbol{B}(\theta) = \sum_{i=0}^{n} b_i^n(\theta)\boldsymbol{P}_i, \quad \theta \in [0,1]. \tag{5.2.1}$$

Although neither Bézier nor Casteljau explicitly formulated the curves in terms of Bernstein polynomials,

$$b_i^n(\theta) = \binom{n}{i}\theta^i(1-\theta)^{n-i}, \quad i = 0,1,\dots,n, \tag{5.2.2}$$

these polynomials are now considered the core of the Bézier curve (Farouki, 2012) and serve as blending functions. A matrix formulation of the Bézier curve, given by (Joy, 2000)

$$\boldsymbol{B}(\theta) = \begin{bmatrix}1 & \theta & \theta^2 & \dots & \theta^n\end{bmatrix} \begin{bmatrix} b_{0,0} & 0 & 0 & \dots & 0 \\ b_{1,0} & b_{1,1} & 0 & \dots & 0 \\ b_{2,0} & b_{2,1} & b_{2,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ b_{n,0} & b_{n,1} & b_{n,2} & \dots & b_{n,n} \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{P}_n \end{bmatrix} \tag{5.2.3a}$$

$$b_{i,j} = (-1)^{i-j}\binom{n}{i}\binom{i}{j}, \quad i,j \in \{0,\dots,n\}, \tag{5.2.3b}$$

is convenient for computer implementation. Only the first vector in (5.2.3a) is dependent on the path parameter θ, which simplifies the process of finding derivatives. Note that control points are constant, and hence the derivative of a Bézier curve is determined by computing the derivative of the Bernstein polynomials (5.2.2). The derivative of an arbitrary Bézier curve of degree n will be a Bézier curve of degree $n-1$, and hence defined by n control points.

5.2.2 Design and implementation

Methods proposed in Knædal (2019) and code available on GitHub[1] formed the basis for the design and implementation of the Bézier-based path generator. Two methods, referred to as the *pragmatic* and the *optimization* approach, were explored. The methods differ in how intermediate control points are defined. Both methods are briefly presented in Sections 5.2.3 and 5.2.4. For further details on the two approaches see Knædal (2019).

Both methods are derived under the assumptions that (Knædal, 2019)

- there exists a straight corridor of breadth $\zeta_{k+1} > 0$, defined such that it is free of static obstacles and encloses the last visited and the next WP, and that
- initially, only the current position of the ASV and the next WP is known. Thereafter, all previous WPs and the next WP will be known.

The collision-free corridor is ideally defined by the path planner, based on information about the environment. It can be computed as the shortest distance to any static obstacle along a straight line connecting two WPs, illustrated in Figure 5.2.1. Note that in the quasi-static environment (see Sections 4.2.1 and 4.2.2) in which a path is planned and generated, a dynamic obstacle and the associated avoidance region will be treated as if they were static.

[1]`path_generator` codebase: `https://github.com/magnuok/path_generator`.

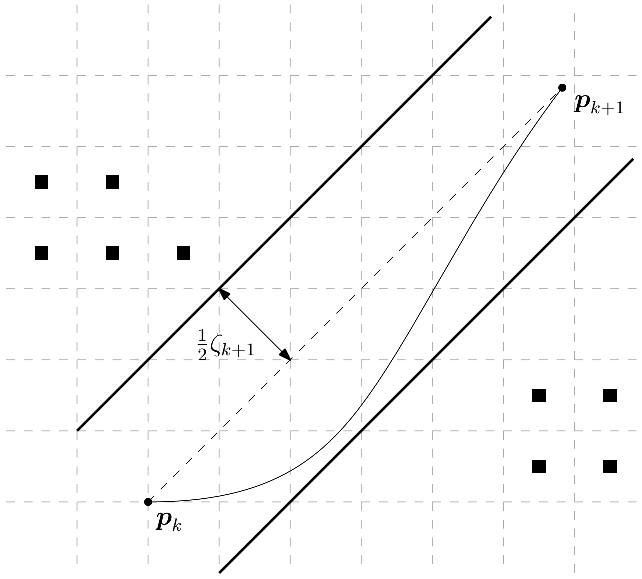


Figure 5.2.1: A straight corridor enclosing two WPs, k and $k+1$, with breadth given by the shortest distance from the straight line between $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$ to an obstructed cell.

Two WPs will be provided by the path planner at any time instant, being the previous and the WP ahead. The path generator computes a Bézier curve (5.2.1) for the pair of WPs, such as the curved path segment in Figure 5.2.1. There will be one WP at each end of the curve (i.e. $\boldsymbol{P}_0 = \boldsymbol{p}_k$ and $\boldsymbol{P}_n = \boldsymbol{p}_{k+1}$), and each path segment will be linked to the previous one in a WP to produce the entire path. Therefore, the parametric continuity at the endpoints of the Bézier curves determine the degree of continuity along the entire path. As argued in Knædal (2019), the septic Bézier curve is of the lowest degree to freely set the change in curvature at endpoints. Continuity in curvature is required when defining the desired heading angle along the path by the path-tangential angle

$$\psi_d(s) = \arctan 2(y_d^s(s), x_d^s(s)), \tag{5.2.4}$$

with the 1st and 2nd order derivatives with respect to the path parameter s given by

$$\psi_d^s = \frac{x_d^s y_d^{2s} - x_d^{2s} y_d^s}{|\boldsymbol{p}_d^s|^2}, \tag{5.2.5a}$$

$$\psi_d^{2s} = \frac{x_d^s y_d^{3s} - x_d^{3s} y_d^s}{|\boldsymbol{p}_d^s|^2} - 2\frac{\left(x_d^s y_d^{2s} - x_d^{2s} y_d^s\right)\left(x_d^s x_d^{2s} + y_d^s y_d^{2s}\right)}{|\boldsymbol{p}_d^s|^4}. \tag{5.2.5b}$$

It is seen that path derivatives with respect to s up to order three, denoted by a super script $3s$, are needed to compute the derivatives of the desired heading (5.2.5) up to order two. Every curve of order higher than 7 would also provide the required parametric continuity, but additional control points increase the computational complexity. Therefore, the septic Bézier curve with a total of 8 control points, of which two are the current and the next WPs, is used.

Since only one WP ahead is assumed to be known at a time, the desired heading of the vessel when approaching this WP is set to

$$\psi_{d,k} = \arctan 2(x_{k+1} - x_k, y_{k+1} - y_k). \tag{5.2.6}$$

Here, subscripts k and $k+1$ indicate the WP visited last and the WP ahead. The desired heading at the next WP (5.2.6) equals to the orientation of the straight line between the two WPs. It might neither be the most practical nor a feasible departure angle if the next line segment forms a sharp angle to the current line segment. However, unless the vessel has to perform an avoidance maneuver, the change in angle between line segments will be kept at a minimum by the path planner (see Section 4.2).

Control points specify the shape of the Bézier curve. The placements of the three first intermediate control points, $\boldsymbol{P}_{i,k+1},\ i=1,2,3$, are constrained by the requirement of $\mathcal{C}^3$-continuity along the path. These constraints can be expressed in terms of a system of linear equations (Knædal, 2019)

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_{1,k+1} \\ \boldsymbol{P}_{2,k+1} \\ \boldsymbol{P}_{3,k+1} \end{bmatrix} = \begin{bmatrix} 2\boldsymbol{P}_{n,k} - \boldsymbol{P}_{n-1,k} \\ -2\boldsymbol{P}_{n-1,k} + \boldsymbol{P}_{n-2,k} \\ 2\boldsymbol{P}_{n,k} - 3\boldsymbol{P}_{n-1,k} - 3\boldsymbol{P}_{n-2,k} - \boldsymbol{P}_{n-3,k} \end{bmatrix} \tag{5.2.7}$$

by evaluating the derivatives of the Bézier curve (5.2.1). The matrix in (5.2.7) is full rank, and hence the placements of $\boldsymbol{P}_{i,k+1},\ i=1,2,3$ are uniquely determined by $\boldsymbol{P}_{i,k},\ i=4,\dots,7$ for $n=8$. Given the desired heading at the last visited WP (5.2.6), the former three points will lie on a straight line inclined at this angle. Since the Bézier curve is completely contained inside the *convex hull*[2] of its control points, the curve will be within the collision-free corridor if all control points are bounded by the corridor. Consequently, the following corridor constraints are imposed on the 4th through 7th control points (Knædal, 2019):

$$|\boldsymbol{P}_{7,k} - \boldsymbol{P}_{6,k}| \leq \frac{1}{2}\zeta_{k+1}, \tag{5.2.8a}$$

$$|3\boldsymbol{P}_{7,k} - 4\boldsymbol{P}_{6,k} + \boldsymbol{P}_{5,k}| \leq \frac{1}{2}\zeta_{k+1}, \tag{5.2.8b}$$

$$|7\boldsymbol{P}_{7,k} - 12\boldsymbol{P}_{6,k} + 6\boldsymbol{P}_{5,k} - \boldsymbol{P}_{4,k}| \leq \frac{1}{2}\zeta_{k+1}. \tag{5.2.8c}$$

However, by assumption, the corridor breadth of the next curve segment ζ_{k+1} will not be known in advance since only one WP ahead is known at a time. Therefore, a constant corridor of minimum breath $\zeta = \zeta_{min}$ will be used for the present. As mentioned, the last control point $\boldsymbol{P}_{7,k}$ coincides with the WP in position $\boldsymbol{p}_k$. Thus, only control points 4 through 6 are not fixed and need to be placed by an appropriate strategy.

5.2.3 Pragmatic approach

The placements of the control points $\boldsymbol{P}_{i,k},\ i=4,5,6$ are ultimately determined under the proposition that a reasonable path for a low-speed ASV minimizes traveled distance and respects nonholonomic constraints (see Section 1.3) at all times. The strategy is to place $\boldsymbol{P}_{4,k}$ a distance δ away from the next WP in position $\boldsymbol{p}_{k+1}$, and $\boldsymbol{P}_{i,k},\ i=5,6$ close to this WP according to

$$\boldsymbol{P}_{4,k} = \boldsymbol{p}_{k+1} - \delta\left[\cos(\psi_{d,k+1}),\quad \sin(\psi_{d,k+1})\right]^\top \tag{5.2.9a}$$

$$\boldsymbol{P}_{5,k} = \boldsymbol{p}_{k+1} - \frac{\delta}{\mu}\left[\cos(\psi_{d,k+1}),\quad \sin(\psi_{d,k+1})\right]^\top \tag{5.2.9b}$$

$$\boldsymbol{P}_{6,k} = \boldsymbol{p}_{k+1} - \frac{\delta}{2\mu}\left[\cos(\psi_{d,k+1}),\quad \sin(\psi_{d,k+1})\right]^\top \tag{5.2.9c}$$

[2]The smallest convex polygon enclosing all control points (for a definition of convexity, see Figure 4.1.4).

By proper tuning of a scaling factor μ, this will keep both the curve length and the curvature low. The distance δ determined from (Knædal, 2019)

$$\delta = \min\left\{\frac{1}{2}\left|\boldsymbol{p}_{k+1} - \boldsymbol{p}_k\right|,\ \delta_c(\kappa_{max}, U_{ref}(t))\right\}, \tag{5.2.10}$$

in which the former parameter is to avoid maneuvers leading the ASV further away from the next WP and δ_c assures that the upper bound on curvature κ_{max} at the prevailing reference speed $U_{ref}(t)$ is not violated. The pragmatic approach to placing the control points is not optimal, but it is computationally efficient and yields good results if tuned properly (Knædal, 2019).

5.2.4 Optimization approach

For each path segment, an optimization program (4.1.10) with the objective to minimize path length, or energy consumption, and subject to boundary and curvature constraints is solved. The unfixed control points $\boldsymbol{P}_{i,k},\ i = 4, 5, 6$ constitute the decision variables. An optimal placement of these three control points for a path segment k is given by

$$\min_{\boldsymbol{P}_{i,k},\ i=4,5,6} \quad \int_0^1 \sqrt{x^\theta(\theta)^2 + y^\theta(\theta)^2 d\theta} \tag{5.2.11a}$$

$$\text{subject to} \quad \boldsymbol{P}_{i,k} \leq \boldsymbol{P}_{i+1,k},\ i = 0, \dots, 6, \tag{5.2.11b}$$

$$\kappa_k(\theta) \leq \kappa_{max}, \tag{5.2.11c}$$

$$\kappa_{k+1}(\theta) \leq \kappa_{max}, \tag{5.2.11d}$$

$$\text{(5.2.7) and (5.2.8).}$$

Since $\boldsymbol{P}_{i,k},\ i = 4, 5, 6$ define the shape of the curve, the objective function will indeed be a function of the decision variables. The constraints (5.2.11b) ensure that the decision variables are in correct order, e.g. $\boldsymbol{P}_{4,k}$ does not appear before $\boldsymbol{P}_{6,k}$, and that $\boldsymbol{P}_{i,k},\ i = 4, 5, 6$ are in between the last visited and the next WP. Curvature constraints are imposed through (5.2.11c) and (5.2.11d). The former considers feasibility of the current path segment with respect to an upper bound on path curvature, κ_{max}. Feasibility of the next path segment in a worst-case scenario, defined as a $\pm 90°$ turn, is evaluated by the latter curvature constraint. It is computed based on knowledge of control points 0 through 4 from (5.2.7) and by generating four artificial points to create the largest curvature one can expect.

Being a nonlinear problem, (5.2.11) is intrinsically hard to solve. However, having only a number of three decision variables, the search space (4.1.11) is relatively small. Therefore, (5.2.11) is solved by a brute-force technique: A discrete set of N_{ss} feasible solutions is constructed, of which the candidate yielding the smallest cost (5.2.11a) is selected. Advantages of this strategy include robustness, in that the minimum will be identified if one exists, and, further, it will not get stuck in local minima. The selected candidate will converge to the real optimum with an increasing size of the set of feasible solutions, but at the cost of increased run-time.

5.3 Maneuvering controller

Path-following is achieved through an appropriate maneuvering controller. The maneuvering problem comprises the task of following the path provided by the path generator from Section 5.2.2 at a desired speed, as presented in Section 2.2.3. Here, the maneuvering controller is designed according to a backstepping approach, and the speed assignment is given by a unit-tangent gradient update law. Alternatives to the backstepping controller include a sliding-mode controller, as presented in e.g. Fossen (2011, p. 519) and Skjetne (2005), or the enhanced version in Skjetne (2020) based on cascade-backstepping. The former is a robust nonlinear design technique that handles model uncertainty, and the latter has fewer tuning parameters and exponential stability properties. However, these control methods were not explored further as the guidance system, comprising the path planner and path generator, was the focus of attention here. In this section, the controller design is presented. A generalized inverse control allocation scheme was derived, but since no physical experiments could be conducted, details on the allocation scheme is left to Appendix C.

5.3.1 Backstepping control design

The control law is established by a backstepping design technique. Backstepping refers to a recursive construction of the feedback control law $\boldsymbol{\tau}$ by determining stabilizing controls for cascaded subsystems through Lyapunov analysis (Fossen, 2011, p. 457). Two steps are needed to obtain $\boldsymbol{\tau}$ for the maneuvering control model (3.2.1). These steps are presented in the following, as well as the update law on the path parameter s.

Step 1

The 1st step state variable is defined as the deviation in position and heading from the desired path in $\{b\}$, that is $\boldsymbol{z}_1 := \boldsymbol{R}(\psi)^\top (\boldsymbol{\eta} - \boldsymbol{\eta}_d(s))$. A control Lyapunov function (CLF) candidate for the $\boldsymbol{z}_1$-subsystem is

$$V_1 = \frac{1}{2}\boldsymbol{z}_1^\top \boldsymbol{z}_1, \tag{5.3.1a}$$

$$\dot{V}_1 = \boldsymbol{z}_1^\top \left(\boldsymbol{\nu} - \boldsymbol{R}(\psi)^\top \boldsymbol{\eta}_d^s(s)\dot{s}\right). \tag{5.3.1b}$$

The virtual control is defined as $\boldsymbol{\nu} := \boldsymbol{\alpha}_1 + \boldsymbol{z}_2$, where $\boldsymbol{z}_2$ is the 2nd step state variable and the stabilizing function is a feedback controller

$$\boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t) = -\boldsymbol{K}_1\boldsymbol{z}_1 + \boldsymbol{R}(\psi)^\top \boldsymbol{\eta}_d^s(s) v_d(s,t), \quad \boldsymbol{K}_1 = \boldsymbol{K}_1^\top > \boldsymbol{0} \tag{5.3.2}$$

with a unit-tangent gradient update law on the path parameter s. The update law is given by

$$\dot{s} = v_d(s,t) - \frac{\mu}{\left|\boldsymbol{\eta}_d^s(s)\right|} V_1^s(\boldsymbol{\eta}, s) = v_d(s,t) + \mu \frac{\boldsymbol{\eta}_d^s(s)^\top}{\left|\boldsymbol{\eta}_d^s(s)\right|} \left(\boldsymbol{\eta} - \boldsymbol{\eta}_d(s)\right), \quad \mu \geq 0, \tag{5.3.3}$$

and avoids a varying gain from $V_1^s(\boldsymbol{\eta}, s)$ along the path by normalizing the tangent vector $\boldsymbol{\eta}_d^s(s)$. Moreover, the second term in (5.3.3), that is the speed assignment error

$$\omega = \dot{s} - v_d(s,t) = -\frac{\mu}{\left|\boldsymbol{\eta}_d^s(s)\right|} V_1^s, \tag{5.3.4}$$

provides some elasticity in solving the dynamic task (2.2.8) as opposed to a pure tracking update law (i.e. $\omega = \mu = 0$). If the scaling factor μ is large, the path parameter s will be driven rapidly to minimize $s \to V_1(\boldsymbol{\eta}, s)$. By defining a tuning function

$$\rho := -\boldsymbol{z_1}^\top \boldsymbol{R}(\psi)^\top \eta_d^s(s) = V_1^s, \tag{5.3.5}$$

a bound on the time derivative of the 1st step CLF (5.3.1b) is given by

$$\dot{V}_1 \leq -\lambda_{min}(\boldsymbol{K}_1)\,|z_1|^2 + \rho\omega + \boldsymbol{z}_1^\top \boldsymbol{z}_2, \tag{5.3.6}$$

where $\lambda_{min}(\boldsymbol{K}_1)$ denotes the minimum eigenvalue of the control gain $\boldsymbol{K}_1$ and $\rho\omega \leq 0$ for the update law (5.3.3). The 2nd step state variable $\boldsymbol{z}_2$ is disregarded in this step, and thus the stabilizing control (5.3.2) with (5.3.3) renders $\boldsymbol{z}_1 = \boldsymbol{0}$ uniformly globally exponentially stable and solves the dynamic task (2.2.8) in the limit as $\boldsymbol{z}_1 \to 0$ (Skjetne, 2005; Skjetne, 2019).

Step 2

The 2nd step state variable is defined in step 1 as the deviation in vessel velocities $\boldsymbol{\nu}$ and the virtual control $\boldsymbol{\alpha}_1$, that is $\boldsymbol{z}_2 := \boldsymbol{\nu} - \boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t)$. A CLF candidate for the 2nd step is

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{z}_2^\top \boldsymbol{M} \boldsymbol{z}_2, \tag{5.3.7a}$$

$$\begin{aligned}\dot{V}_2 = &-\boldsymbol{z}_1^\top K_1 \boldsymbol{z}_1 + \rho\omega - \frac{1}{2}\boldsymbol{z}_2^\top \left(\boldsymbol{D} + \boldsymbol{D}^\top\right) \boldsymbol{z}_2 \\ &+ \boldsymbol{z}_2^\top \left(\boldsymbol{z}_1 - \boldsymbol{D}\boldsymbol{\alpha}_1 + \boldsymbol{R}(\psi)^\top \boldsymbol{b} + \boldsymbol{\tau} - \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1\right).\end{aligned} \tag{5.3.7b}$$

To obtain the time derivative of the 2nd step CLF (5.3.7b), the maneuvering control model (3.2.1) has been applied. A backstepping control law that renders (5.3.7b) negative definite, and hence the equilibrium $[\boldsymbol{z}_1, \boldsymbol{z}_2]^\top = \boldsymbol{0}$ uniformly globally exponentially stable, is given by

$$\boldsymbol{\tau} = -\boldsymbol{z}_1 - \boldsymbol{K}_2\boldsymbol{z}_2 + \boldsymbol{D}\boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t) - \boldsymbol{R}(\psi)^\top \hat{\boldsymbol{b}} + \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1(\boldsymbol{\eta}, s, t), \quad \boldsymbol{K}_2 = \boldsymbol{K}_2^\top > \boldsymbol{0}, \tag{5.3.8}$$

where $\hat{\boldsymbol{b}}$ denote bias estimates provided by the observer (see Section 5.1). The gain matrices $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$, as well as the scaling factor μ, are determined by tuning. Figure 5.3.1 illustrates how the $\boldsymbol{z}_1$- and $\boldsymbol{z}_2$-subsystem are interconnected. For the complete set of equations describing the maneuvering controller, see Appendix B.

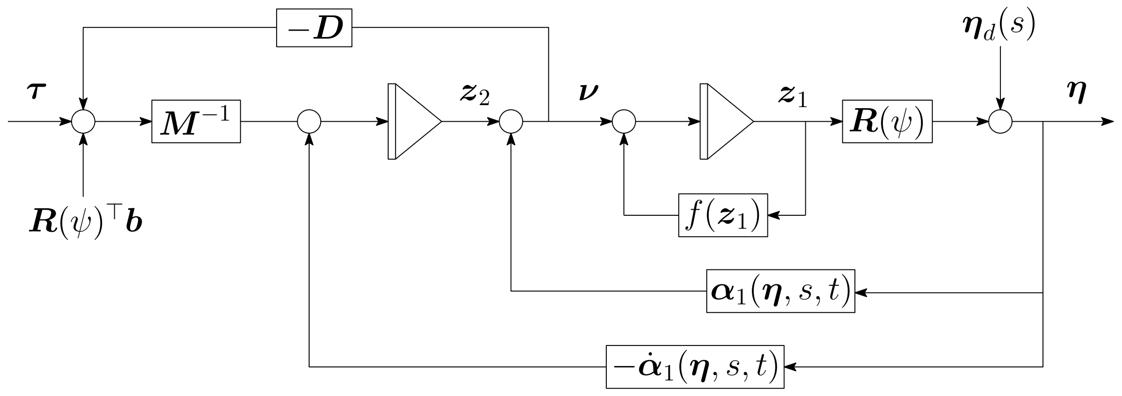


Figure 5.3.1: The backstepping control system, in which stabilization of the $\boldsymbol{z}_1$-subsystem is achieved through a virtual control $\boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t)$. Note that $f(\boldsymbol{z}_1) = -\boldsymbol{S}(r)\boldsymbol{z}_1 + \boldsymbol{z}_1^s \dot{s}$.

5.4 LOS path-following control

To quickly test and assess the outputs of the path-planning algorithms, a simple LOS path-following law is used. According to the lookahead-based steering principle in Fossen (2011, p. 258–262), and in absence of current, the desired heading can be set to

$$\psi_d(e) = \psi_p + \psi_r(e), \qquad \text{with} \tag{5.4.1a}$$
$$\psi_p \quad = \arctan 2(y_{k+1} - y_k, x_{k+1} - x_k), \tag{5.4.1b}$$
$$\psi_r(e) = \arctan\left(\frac{-e}{\Delta}\right) = \arctan(-k_p e). \tag{5.4.1c}$$

Herein, the first term (5.4.1b) is the path-tangential angle, whereas the second term (5.4.1c) is interpreted as a saturating control with a proportional gain $k_p > 0$. The latter directs the vessel's velocity vector towards a point on the path, determined by a lookahead distance Δ. The aim is to drive the cross-track error $e(t)$ towards zero, which is the distance from the vessel normal to the straight line between the last visited and the next WPs. It is determined together with the along-track distance $s(t)$ to the next WP from

$$[s(t), e(t)]^\top = \boldsymbol{R}_p(\psi_p)^\top (\boldsymbol{p}(t) - \boldsymbol{p}_k). \tag{5.4.2}$$

Lookahead-based steering comes with the advantage of being valid for any $e(t)$. A circle of acceptance with radius r_{k+1} around each WP defines when the WP is considered reached, and is given by Fossen (2011, p. 265)

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 \leq r_{k+1}^2. \tag{5.4.3}$$

According to Fossen (2011), a suitable value for r_{k+1} is twice the vessel length. Figure 5.4.1 illustrates the lookahead-based principle and its parameters.

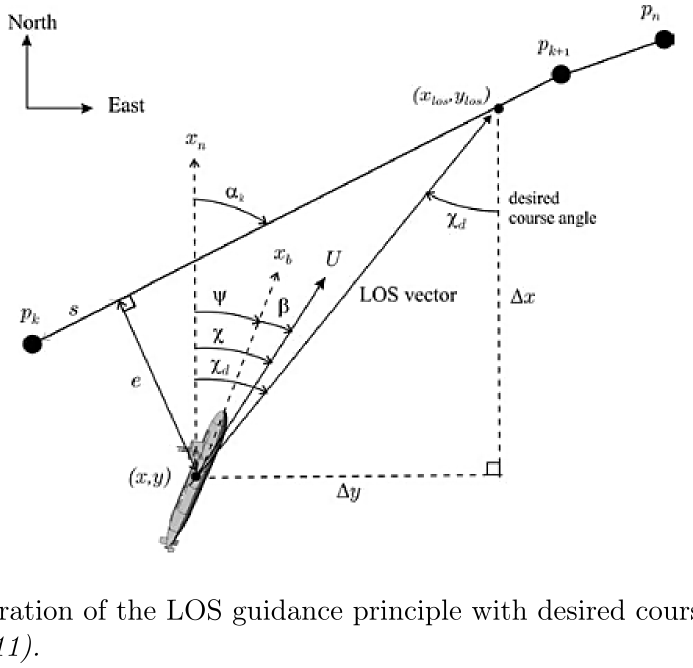


Figure 5.4.1: Illustration of the LOS guidance principle with desired course angle χ_d. *Courtesy of Fossen (2011).*

Chapter 6

Simulation studies

The performance of the guidance system, and the path-planning subsystem in particular, is demonstrated through simulations. First, the path-planning algorithms are combined with the simple LOS path-following law presented in Section 5.4 to examine the search trees, evaluate the KPIs from Section 4.2.6, and demonstrate how different encounter situations are handled. Thereafter, the path planner is evaluated as a part of a GNC system (see Figure 2.2.1); comprising the nonlinear passive observer presented in Section 5.1, the Bézier-based path generator from Section 5.2, and the maneuvering controller derived in Section 5.3. Both the pragmatic and optimal path-generation methods are tested and evaluated in relationship with the path planner. The hybrid A* path search with two different expansion strategies, as presented in Section 4.2.2, was used for the path planner, whereas the MIP given in Section 4.2.3 was discarded as it produced impracticable WPs. A brief discussion on the MIP is included in this chapter. Before looking into different simulation scenarios and discussing results therefrom, the ASV model, to which the GNC system is applied, and the software environment is presented.

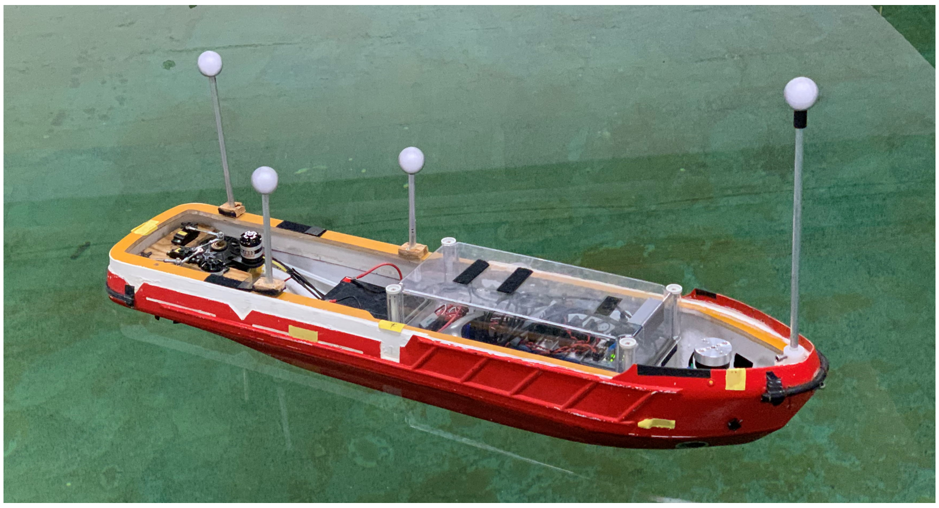


Figure 6.0.1: The CSEI model vessel.

6.1 Software and hardware

The path-planning algorithms are implemented in the MATLAB R2018b programming environment. MATLAB comes with a high-level programming language that expresses matrix and array mathematics directly, and is therefore particularly suitable for implementing the matrix-vector models in Chapter 3. Moreover, MATLAB supports model-based design through Simulink, which is a block diagram environment for simulating dynamic nonlinear systems. Here, Simulink is used for the implementation of the modular GNC system including the ASV model. Furthermore, MATLAB is interfaced with Java classes and objects. Hence, by using the Java-based AMPL application programming interface, the MIP in Section 4.2.3 can be invoked directly from MATLAB. Last, but not least, MATLAB provides tools for visualization of results and has commands for graphics in both two and three dimensions. For these reasons, MATLAB was considered suitable for the simulation and evaluation of the path-planning algorithms and the GNC system as a whole. Evaluation of the path-planning algorithms included calculations of computation times (see Section 4.2.6), and therefore specifications of the machine running the simulations are provided in Table 6.1.1.

Table 6.1.1: Computer specifications.

Processor	Intel® Core i5™-2520M
Clock speed	2.50 GHz
Memory	4.00 GB RAM
Operating system	Windows 7 Professional x64

6.2 CyberShip Enterprise I

A simulation model of CyberShip Enterprise I (CSEI), depicted in Figure 6.0.1, serves as the platform for testing the GNC system. CSEI is a 1 : 50 scale model of a tug boat, fitted with two Voith-Schneider propellers astern and a bow thruster. The main dimensions of CSEI are given in Table 6.2.1a, and Table 6.2.1b and Figure 6.2.1 specify its thruster configuration. System identification of CSEI through towing tests conducted by former master's students has resulted in a 3DOF maneuvering model (3.1.5). The coefficients and model matrices are attached in Appendix A. For more information regarding the model boat, the reader is referred to NTNU (2020) and references therein.

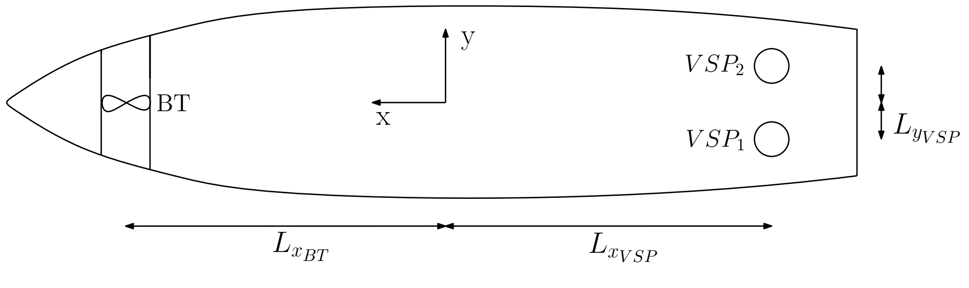


Figure 6.2.1: Arrangement of actuators on CSE1. *Adapted from Valle (2015).*

6.2.1 Froude scaling of velocity

Since CSEI is a 1:50 scale model, the length of the real tug boat is $50L = 55.25[m]$. In order to find an appropriate velocity for CSEI along the desired path, the *Froude number* is evaluated. In naval architecture, this is a dimensionless quantity used to determine the resistance of a partially submerged body moving through water. Vessels with equal Froude number produce a similar wake, even if their geometry and size are different. The Froude number is defined by (Fossen, 2020, p. 2)

$$Fr = \frac{U}{\sqrt{gL}}, \tag{6.2.1}$$

in which L is the length of the vessel at the water line level, g is the gravitational acceleration, and U is the speed of the vessel. Thus, in order for two geometrically similar vessels to produce the same wave pattern, the smaller model has to run at slower speeds than the larger ship, according to

$$U_{model} = \sqrt{\frac{L_{model}}{L_{ship}}} U_{ship}. \tag{6.2.2}$$

A speed of $U_{model} = 0.2$[m/s] was used in the simulations, which gives $U_{ship} \approx 4$[kn]. The choice of speed was limited by the validity of the maneuvering model (3.2.1). Although 4 knots is a rather slow speed for the transit phase, it may indeed be appropriate in congested and complex environments.

Table 6.2.1: CSEI data.

(a) Main dimensions

Parameter	Unit	Symbol	Value
Length over all	[m]	L	1.105
Breadth	[m]	B	0.248
Weight displacement	[kg]	Δ	14.11

(b) Position of actuators

Actuator	x [m]	y [m]
VSP_1	-0.4574	-0.055
VSP_2	-0.4574	0.055
BT	0.3875	0

6.3 The MIP path-planning scheme

The MIP as presented in Section 4.2.3 did not produce practicable WPs. As seen in Figure 6.3.1, obstacles are not circumvented and sequence of WPs do not lead the vessel to the target. That is, the MIP fails even in this rather simple environment of limited size and with static obstacles only. The aim was to formulate an MILP, and therefore the kinematic model (4.2.14) was employed. However, as mentioned in Section 4.2.3, the matrix of neural activities introduces nonlinearities to the objective function. Furthermore, the solver seems to struggle in finding a solution due to the minimal differences in neural activity associated with nodes in the free space. Thus, altering the BINN or using another guidance model with a landscape of less steepness could possibly help. Another issue is probably the size of the search space: The MIP (4.2.18) is not restricted to discrete nodes and theoretically evaluates any position in the continuous search space.

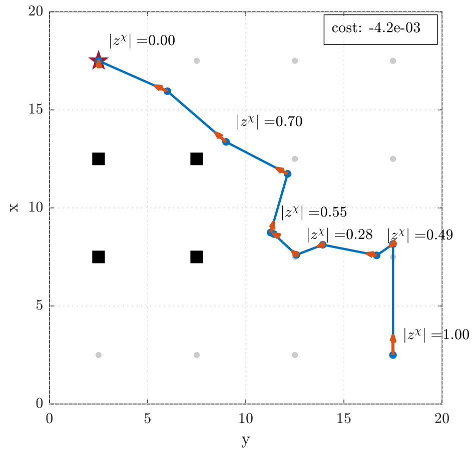


Figure 6.3.1: An example of how the MIP (4.2.18) fails to find a feasible path in a relatively simple environment, even when relaxing the constraints on $\boldsymbol{z}^\psi$.

Even though the MIP did not work as intended, the receding horizon approach was adopted in the hybrid A* implementation in order for the system to be reactive: Only the first WP in the planned sequence of WPs to the target, which is optimal in the current quasi-static environment, is forwarded to the path generator and the planning process is repeated as this WP is reached. Note that the hybrid A* search plans a path all the way to the target, whereas the MIP in an MPC scheme would plan only a certain number of WPs ahead restricted by the time horizon. Thus, the planned path will not only be suboptimal, but potentially misguide the vessel if neural activities do not monotonically decrease towards the target. A solution could be to add a terminal cost minimizing the distance to the target, but such a cost ignores the presence of obstacles. For these reasons, the hybrid A* path search was deemed more promising, and the work on the MIP was aborted.

6.4 Simulations with LOS path-following control

The LOS path-following control law presented in Section 5.4 is utilized in the first step of testing and evaluating the path-planning algorithms. First the performance of the two hybrid A* path search strategies is assessed based on the selected KPIs in Section 4.2.6. Thereafter follow simulations of different encounter situations, demonstrating the influence of the avoidance region on how the path progresses and how this, in turn, enforces the OS to act in compliance with COLREGs. A test of the robustness of the path planner is also carried out by introducing random changes in the heading of the TS. Values for the simulation parameters were retained for all simulations and are given in Table 6.4.1. Note that the radius of the detection region is a guesstimate based on the OS length, and the radius of acceptance is stricter than the value suggested by Fossen (2011) (ref. Section 5.4) which results in sharper turns. Legends of the NE-plots from simulations are given in Figure 6.4.1, and specify the colors, markers and line styles used to indicated the different elements.

Table 6.4.1: Parameter values for simulations.

Parameter	Unit	Symbol	Value
Time step	[s]	h	0.01
Workspace dimensions	[m]	$L_x,\ L_y$	42.0
Cell dimensions	[m]	$dX,\ dY$	3.0
Vessel initial position	[m]	$\boldsymbol{p}_0$	$(1.5,\ 40.5)$
Vessel initial heading	[rad]	ψ_0	0
Vessel speed	[m/s]	U	0.2
Radius of acceptance	[m]	r	0.5
Radius of $\mathcal{R}_d$	[m]	r_d	20

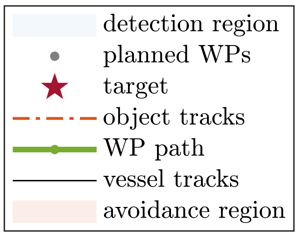
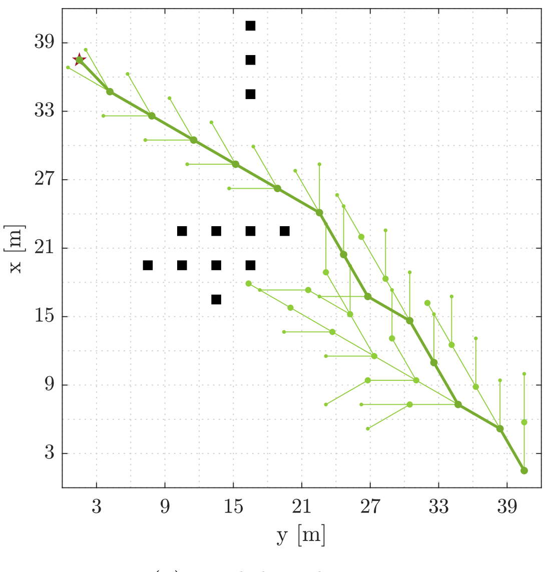


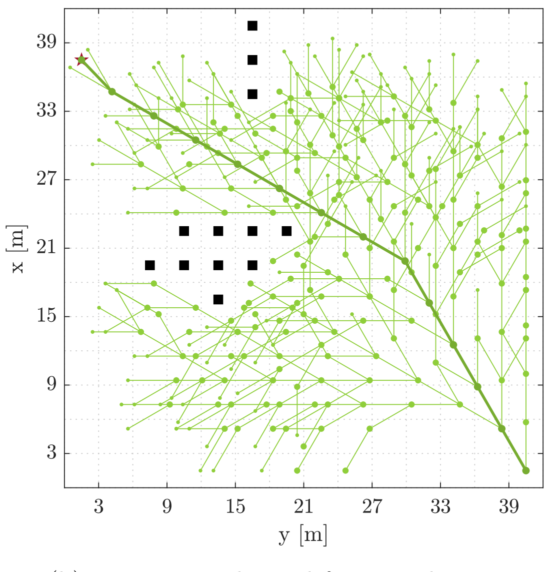
Figure 6.4.1: Legends of the NE-plots.

6.4.1 Evaluation of the hybrid A* path search strategies

The two hybrid A* path search strategies using either straight-line or kinematics-based motion primitives, as described in Section 4.2.2, are evaluated in an environment with both static obstacles and a moving object. The 1st step search trees for both strategies with the two different heuristics – the Euclidean distance to the target, or the modification based on neural activity and with penalty on changes in search direction – are depicted in Figure 6.4.2 and Figure 6.4.3. With straight-line motion primitives, the resulting piecewise linear path will be continuous, in that the endpoint of a motion primitive is the starting point of a subsequent one, but the path tangential angle in a joint between two motion primitives will not be continuous. Conversely, and by definition, the kinematics-based motion primitives form a path that has continuous first derivatives. Neither path will have continuity in curvature. However, as part of the GNC system illustrated in Figure 2.2.1, the path planner is only required to produce a set of feasible WPs while the smooth parametric path definition is the responsibility of a path generator.



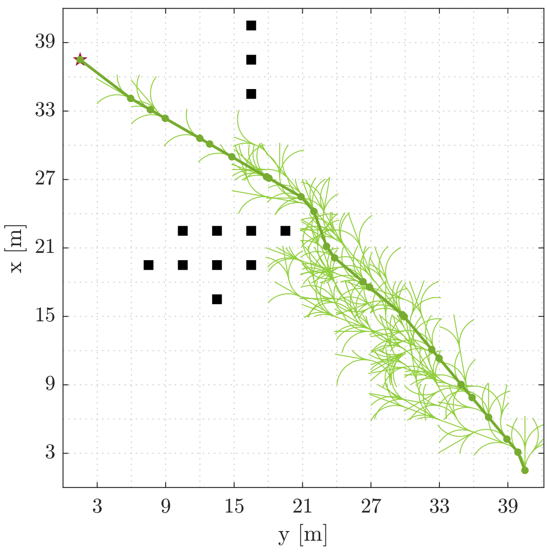
(a) Euclidean heuristics



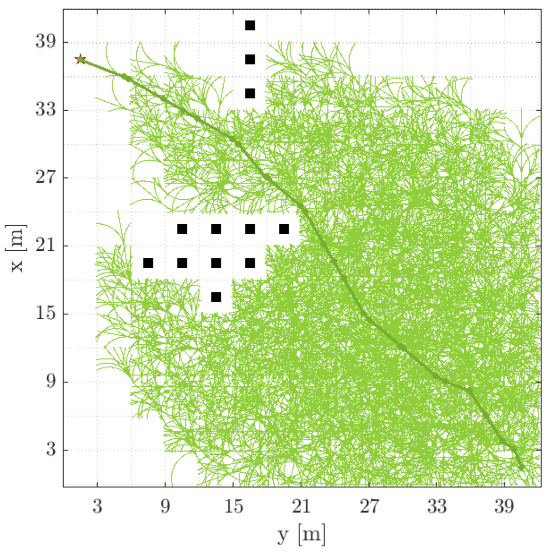
(b) Heuristics adjusted for neural activity

Figure 6.4.2: First step straight-line hybrid A* search trees with the resulting paths in bold. The relative angle between motion primitives is ±30°.

It is evident from Figure 6.4.2 that the Euclidean heuristics is far more efficient than the heuristics adjusted for neural activity. This is seen in the number of expansions, but it is also reflected in the computation times in Table 6.4.2. The heuristic function adjusted for neural activity is slightly slower than the Euclidean distance. This substantiates the claim in Section 4.2.2 that the latter heuristics is an underestimate of the former, and hence admissible. Yet, one does not want the A* algorithm to spend too much time on expanding nodes that seem promising according to the heuristics but eventually turn out to be misleading. When evaluating the resulting paths, it is however seen that the heuristics adjusted for neural activity result in a path with less turns due to the added cost on changes in search direction. Both of the resulting paths are composed of 13 line segments, and thus of the same length since each line segment is equisized. Hence the choice of heuristics will be a compromise between efficiency and path quality. An alternative to the two heuristics applied here could be to use the maximum $h = \max_i\{h_i\},\ i = 1, 2, \ldots$ of several admissible heuristics $h_1, h_2, \ldots$, that is the underestimate closest to the true path cost.



(a) Euclidean heuristics



(b) Heuristics adjusted for neural activity

Figure 6.4.3: First step kinematics-based hybrid A* search trees with the final piecewise linear WP path in bold. The turn rate defining the arcs is set to $0.7r_{max}$ with $r_{max} = \frac{3}{2}L$.

When comparing the straight-line and kinematics-based hybrid A* search trees in Figure 6.4.2 and Figure 6.4.3, it is evident that the latter generally expands more nodes. This can probably be imputed to the variable length of the kinematics-based motion primitives, which sometimes yields quite short line segments and thus a larger set of nodes to evaluate and possibly expand further. Another issue is the restriction on the search direction at a node, which is ignored by the Euclidean distance heuristics. This does also apply to the straight-line search, but since the kinematics-based motion primitives may bend and yield an instant turn away from the target, such expansions are more prone to the underestimated distance from the endpoint to the target. Furthermore, kinematics-based motion primitives are created by increasing the central angle in an iterative manner until a neighboring cell is encountered, which is necessarily more demanding than a simple straight-line expansion calculated in one turn. The computation time of the kinematics-based search tree using Euclidean heuristics is about twice the time needed to compute the straight-line expansion trees. At the same time the iterative construction of kinematics-based motion primitives guarantees that the resulting paths are feasible. The process of increasing the central angle ends immediately when encountering a neighboring cell. On the contrary, the strategy to prevent the straight-line motion primitives from crossing obstructed cells, inspired by the Bresenham algorithm (ref. Section 4.1.2), is seen to fail in Figure 6.4.2b. To summarize, the straight-line path search is superior to the kinematics-based one when it comes to computation time, but falls short when feasibility is concerned.

Table 6.4.2: KPIs of the two hybrid A* path search methods with heuristics given by the Euclidean distance and the version adjusted for neural activity.

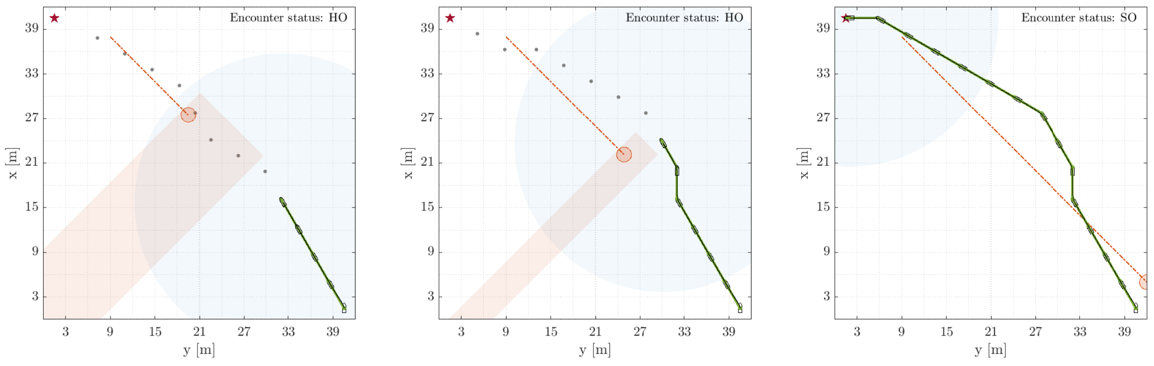
KPI	Unit	Straight-line		Kinematics-based	
		Euclidean	neural	Euclidean	neural
Path length	[m]	81.50	81.50	61.62	60.07
Computation time	[s]	$2.59 \cdot 10^{-5}$	$3.55 \cdot 10^{-5}$	$6.34 \cdot 10^{-5}$	$3.29 \cdot 10^{-4}$
Accumulated turns	[rad]	5.58	5.58	5.42	5.35

6.4.2 Different encounter situations

The potential of the path planner presented in Section 4.2 to handle different encounter situations is demonstrated with the straight-line hybrid A* path search algorithm. Similar results are achieved with the kinematics-based search strategy and are attached in Appendix D. Five different scenarios, one for each encounter type as defined in Section 2.1, were simulated with an object moving in a straight line. The situations are correctly classified by the path planner, and COLREGs compliant maneuvers are initiated. Note that the presented scenarios are rather simplistic, with a single TS moving in a straight line, but they exemplify how the path planner handles encounters of different types. Further testing must be performed to validate the algorithms, in which environmental impacts from waves, wind, and ocean currents should be included. Since there is no quantitative definition of the safe distance at which the OS should pass another vessel (ref. Section 4.1.4), it is adjusted to the identified situation and defined as a factor times the OS length. The initial conditions on the object as well as the chosen safe distance are stated together with plots of the respective encounter scenarios. Videos of the encounters are attached in the digital submission of the thesis. Note that the path sometimes changes even when the object is out of sight and one would expect the WPs to remain in place. The reason is that the path search takes the current vessel state as the starting point, which does not necessarily coincide with the WP considered reached.

Head on

An HO encounter scenario is simulated with the object in initial position $\boldsymbol{p}_{0,obj} = (38,\ 9)$, heading $\psi_{obj} = \frac{3}{4}\pi$ and speed $0.9U$. Figure 6.4.4a shows how the avoidance region impels the OS to take a starboard turn and stay clear of the object, as seen in Figure 6.4.4b. The OS is forced to keep a safe distance of at least 2 times its ship length to the object. If the object would comply with the rules itself and turn to its starboard side, the safe distance between the two vessels would obviously be larger. As the OS detects and steers clear of the object with the COLREGs-compliant maneuver, the path is completely replanned. Therefore, the vessel is guided directly towards the target as the object is passed, rather than driving it back on the initially planned path – which would have lead to an unnecessary turn and probably a longer travel distance. The final path is displayed in Figure 6.4.4c.

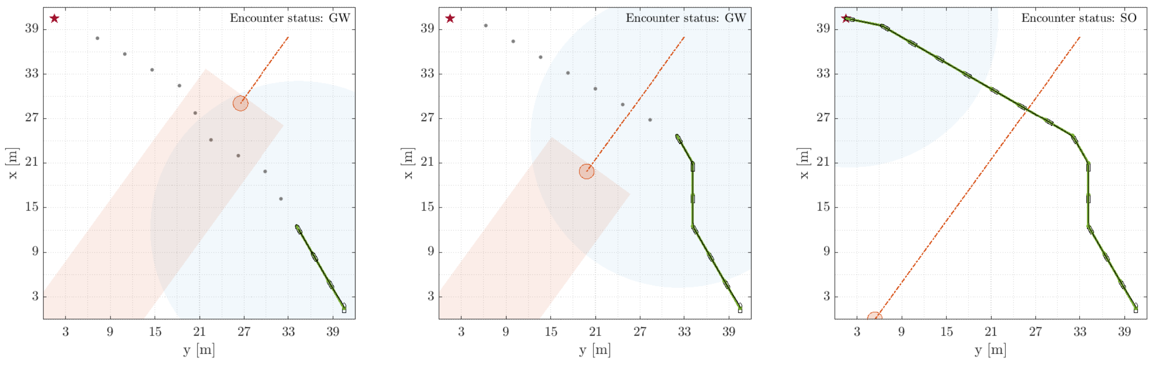


(a) Object detected (b) Maneuver to starboard (c) Target reached

Figure 6.4.4: An HO situation.

Give way

A crossing situation in which the OS is obliged by COLREGs to GW is simulated with the object in initial position $\boldsymbol{p}_{0,obj} = (38,\ 33)$, heading $\psi_{obj} = -\frac{4}{5}\pi$ and speed $0.9U$. An avoidance region that forces the OS to pass behind the object is created instantly as the object is detected. The OS has to stay clear of the object by a margin of 5 times the OS length, as seen in Figure 6.4.5a and Figure 6.4.5c. Consequently, the resulting behavior of the OS is in line with rule 15 in COLREGs as stated in Section 2.1. Figure 6.4.5c displays the resulting path after the target eventually is reached.

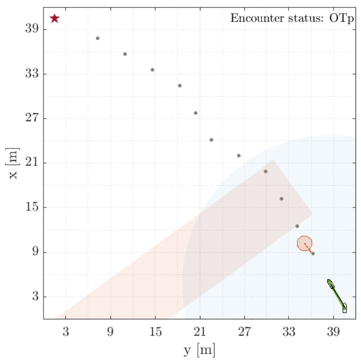


(a) Object detected (b) Passing behind the TS (c) Target reached

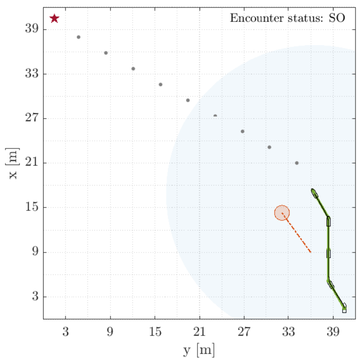
Figure 6.4.5: A GW situation.

Overtaking

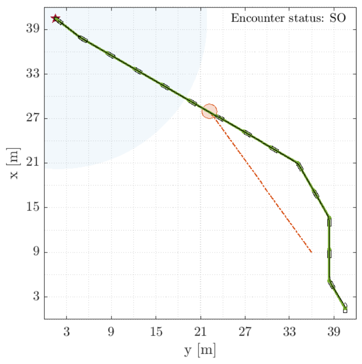
Recall that there are two distinct options when the OS is overtaking another vessel: Either passing with the TS on starboard or on port side. The appropriate actions are given by the relative angle between the two vessels. Therefore, two OT scenarios are simulated. The initial conditions on the moving object are set to $\boldsymbol{\eta}_{0,obj} = (9,\ 36,\ -\frac{1}{5}\pi)^\top$ and $\boldsymbol{\eta}_{0,obj} = (8,\ 36,\ -\frac{1}{8}\pi)$ to incite port and starboard turns, respectively. Furthermore, its speed is reduced to $0.4U$ so that the OS is able to actually overtake the TS. The avoidance regions seen in Figure 6.4.6a and Figure 6.4.7a spur the OS to take actions compliant with COLREGs. They are shifted slightly ahead of the object to take its progression into account. A downside with such a forward shift of the avoidance region is that the OS is allowed to come close to the object before initiating an evasive maneuver, which contradicts rule 8 in COLREGs (ref. Section 2.1). Furthermore, the OS is observed to pass the object by a relatively small margin in the OTp situation depicted in Figure 6.4.6c as it is heavily attracted by the target. Nevertheless, the OS stays clear of the object in both OT scenarios, and a desired safe distance should be achievable by adjusting the parameter values defining the avoidance region.



(a) Object detected

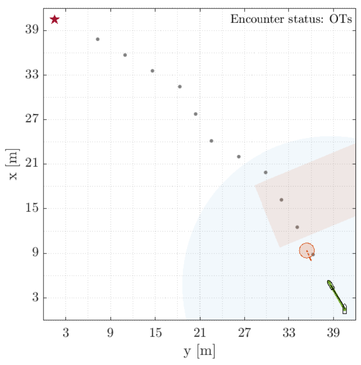


(b) Maneuver to starboard

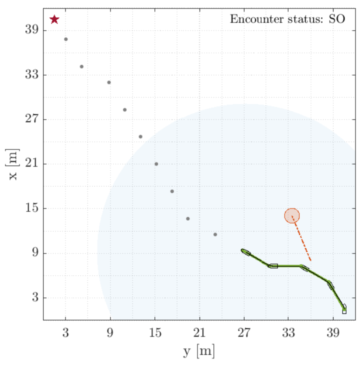


(c) Target reached

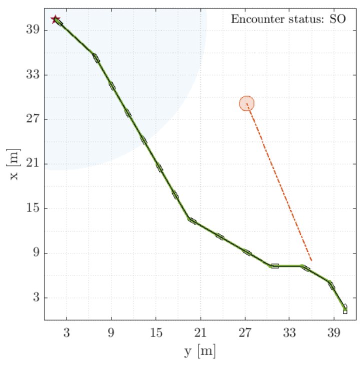
Figure 6.4.6: An OTp situation.



(a) Object detected



(b) Maneuver to port

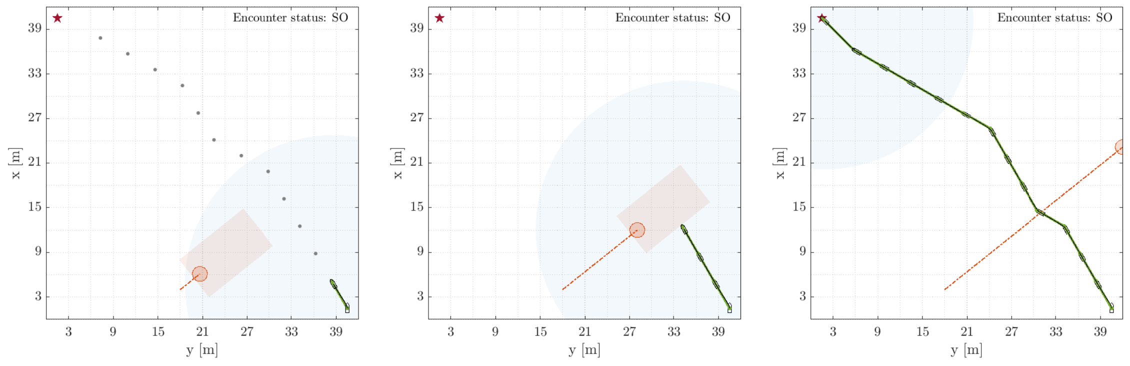


(c) Target reached

Figure 6.4.7: An OTs situation.

Stand on

An SO encounter scenario is simulated with the object in initial position $\boldsymbol{p}_{0,obj} = (4,\ 18)$, with heading $\psi_{obj} = \frac{2}{7}\pi$, and speed $0.9U$. Note that in any situation where the OS detects a moving object but is not obliged to take action, the encounter type is here classified as SO. Therefore, the status 'SO' is seen in some snapshots of the above presented encounter situations after the OS has initiated the appropriate action. However, there are crossing situations, defined by rule 15 in COLREGs, where the OS is requested to keep its course and speed in the first instance and the TS is obliged to GW. Figure 6.4.8a demonstrates such a scenario. Since the moving object does not take action, the OS enters a DP mode and waits for the object to pass in Figure 6.4.8b. The strategy to enter a DP mode might be questionable considering the large inertia of vessels and the corresponding stopping distance, and other MRCs should be evaluated. Assuming that the OS is able to stop in time, collision with the object is avoided and the OS arrives safely at the target in Figure 6.4.8c.

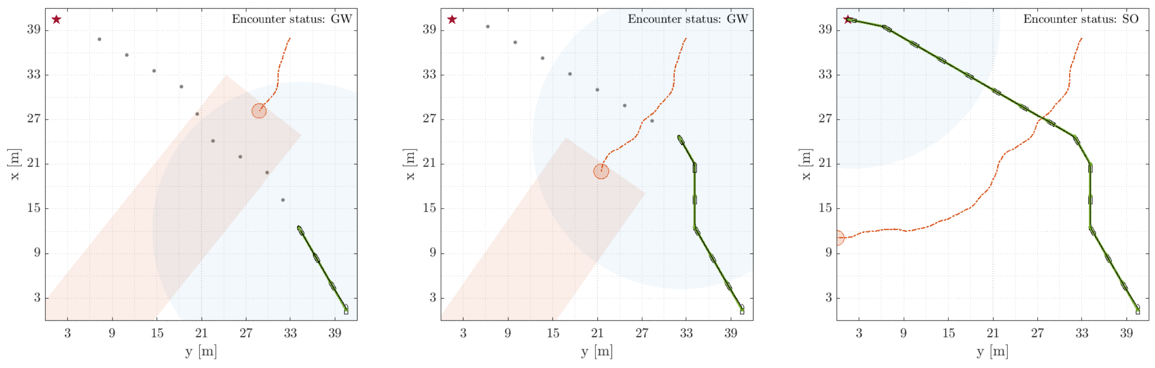


(a) Object detected (b) DP mode (c) Target reached

Figure 6.4.8: An SO situation. The OS enters DP mode as the TS takes no action.

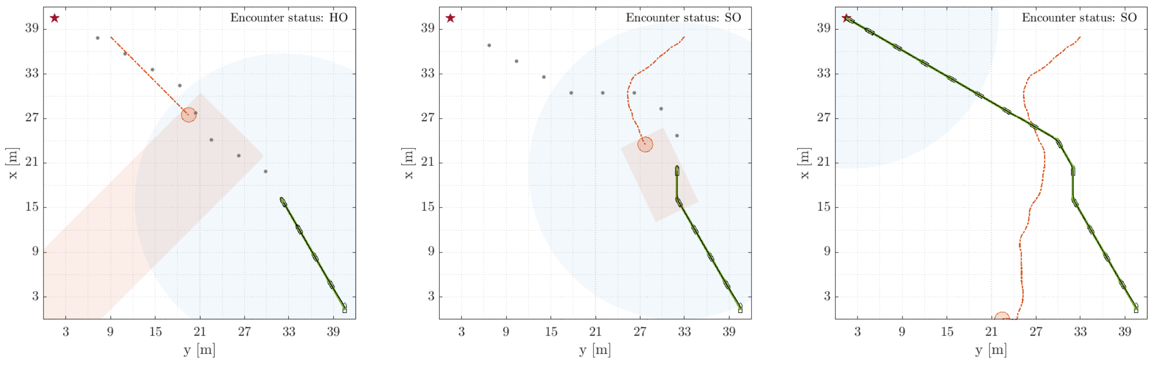
6.4.3 Random motions of the TS

A test of the robustness of the path planner and the adaptive avoidance region method to obtain COLREGs-compliant behavior is performed by simulating random object motions. The heading of the object is allowed to vary within ±1.0 degrees at each time step, and the actual heading change is determined by MATLAB's `rand` function for random number generation. A physical interpretation of such random motions can be that signals from the detection sensors are uncertain and contaminated by noise. Figure 6.4.9 presents snapshots of a GW scenario from a simulation with random object motions. The OS safely circumvents the object and follows the same path as in Figure 6.4.5. Another simulation, which demonstrates a pitfall of the avoidance region method, follows in Figure 6.4.10. In the latter simulation, the OS is trapped inside the avoidance region, as seen in Figure 6.4.10b. Consequently, as no path to the target can be established, the OS enters DP mode leading to a potentially hazardous situation. Fortunately, the object steers away from the OS so that the OS is released from the avoidance region and can safely navigate towards the target again. Nevertheless, collision avoidance should not be a matter of luck, and therefore the risk that the OS gets trapped inside the avoidance region is a topic for future work.



(a) Object detected (b) Passing behind (c) Target reached

Figure 6.4.9: A GW encounter with a randomly moving object.



(a) Object detected (b) OS is trapped (c) Target reached

Figure 6.4.10: A HO encounter with a randomly moving object where the OS gets trapped in the avoidance region, leading to a potentially hazardous situation.

6.5 Simulations with the GNC system

The path planner is integrated as a part of a GNC system, including the path generator, the motion controller and the observer from Chapter 5. A dynamic environment with a moving object in addition to static obstacles was simulated to demonstrate the GNC system. The environment, together with the generated Bézier-based paths and vessel tracks, is depicted in Figure 6.5.3. In the following, outputs from the observer and the controller are assessed, and the path generator with both the pragmatic and optimization approach are evaluated in relationship with the path planner. For the path search, straight-line expansions with neural heuristics was primarily used, but the path generator was also tested together with the kinematics-based hybrid A*.

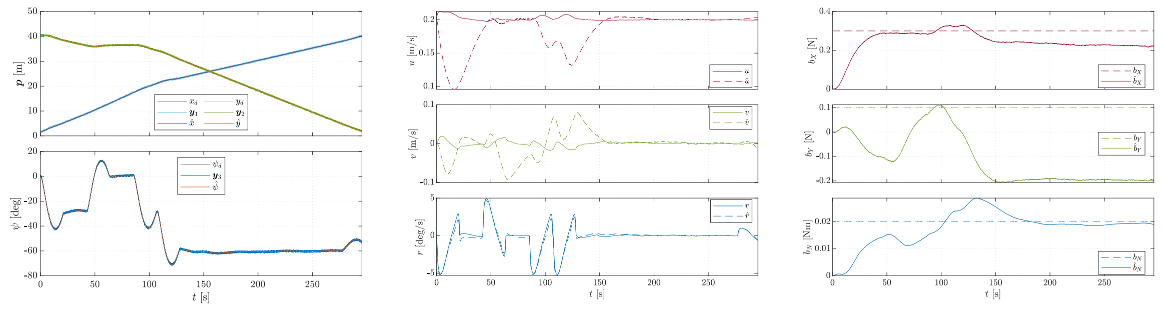
6.5.1 Observer and motion controller

The GNC system was simulated with a constant, nonzero bias, and with measurement signals corrupted by white noise. The bias was set to $\boldsymbol{b} = [0.3, 0.1, 0.02]^\top$ with units of forces and moment, that is [N] and [Nm]. The values were selected by considering the thrust generating limits of the vessel. Zero-mean white Gaussian noise was added to the measurement signals through Simulink's `Band-Limited White Noise` block. The noise power was calculated to obtain a standard deviation σ on the GNSS position signals of 0.2[m], and of 0.5[deg] on the heading signal, according to

$$\sigma^2 t_c, \qquad \text{with} \quad t_c = \frac{1}{100} \cdot \frac{2\pi}{\omega_{max}}. \tag{6.5.1}$$

With a correlation time $t_c = 0.005$[s], corresponding to a system bandwidth of $\omega_{max} = 0.08$[rad/s], the noise power was calculated to be $[16, 16, 0.04] \cdot 10^{-5}$, with units [m^2/Hz] and [rad^2/Hz]. The resulting measurement signals and the respective observer estimates from a simulation case with both static obstacles and a moving object are displayed in Figure 6.5.1. Observer gains of $\boldsymbol{L}_1 = \boldsymbol{I}$, $\boldsymbol{L}_2 = \text{diag}([0.1, 0.1, 0.02])$, and $\boldsymbol{L}_3 = \text{diag}([0.1, 0.1, 0.05])$ were applied.

As seen in Figure 6.5.1a, estimates of position and heading are quite accurate, and the observer clearly solves the task (2.2.2) with respect to $\boldsymbol{\eta}(t)$. These are the estimates relevant for the path-planner, and hence WPs were correctly placed. In contrast, the velocity and bias estimates in surge and sway are rather poor. On closer inspection, the surge and sway velocity estimates are seen to mirror the true signals. The reason is probably that the coordinate axis are inverted in the NE-frame, whereas the path-generator was originally defined with a horizontal x-axis and a vertical y-axis. Thus, the desired setpoints, and more specifically the along-path derivatives $\boldsymbol{\eta}^s$, $\boldsymbol{\eta}^{2s}$, and $\boldsymbol{\eta}^{3s}$, should be inverted. However, this requires for the Bernstein polynomials (5.2.2) to be inverted, which turned out to be a nontrivial task. In fact, with the desired along-path speed profile (2.2.6a) set to zero, the estimated velocity in surge converges to the true value, which substantiates the suspicion that $\boldsymbol{\eta}^s$ and $\boldsymbol{\eta}^{2s}$ might be erroneous.

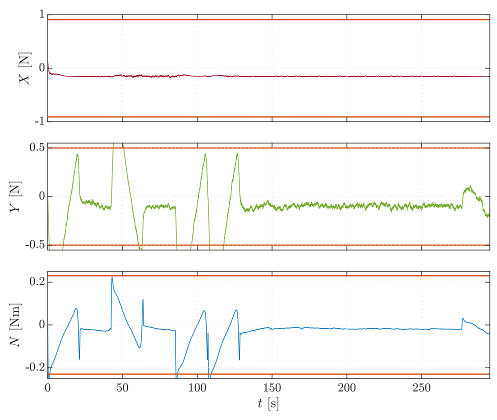


(a) Desired, estimated and measured $\boldsymbol{\eta}(t)$ (b) True and estimated $\boldsymbol{\nu}(t)$ (c) True and estimated $\boldsymbol{b}$

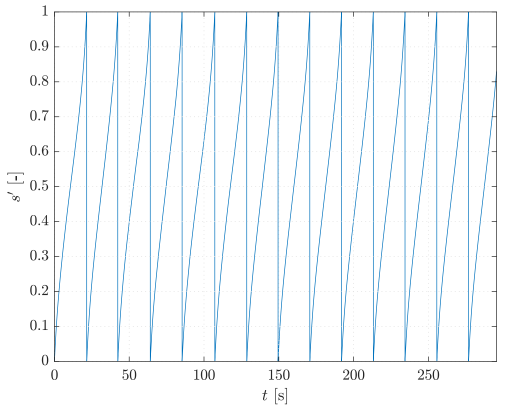
Figure 6.5.1: Observer estimates of vessel position, heading and velocities, as well as the bias.

From Figure 6.5.2a, it is seen that the commanded $\boldsymbol{\tau}$ exceeds the thrust generating capabilities of CSEI in sway and slightly in yaw. Limits are indicated by the upper and lower orange lines. The saturating controls are probably related to the poor velocity and bias estimates in surge and sway, and are expected to diminish with accurate estimates. In surge, the control input is seen to be almost constant, which is consistent with the vessel forward motion at constant speed. In addition to violate the thruster limits, the control commands oscillate, which is undesired as it may result in wear on the actuators. This can probably be reduced by proper tuning of the control gains $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$. The control gains used here were $\boldsymbol{K}_1 = 0.01\boldsymbol{I}$ and $\boldsymbol{K}_2 = 20\boldsymbol{I}$.

Time series of the path parameter s' are plotted in Figure 6.5.2b. In accordance with the hybrid parametric path definition in (2.2.4), s' goes from 0 to 1 for each of the 14 line segments. In other words, the vessel is immediately directed along a new line segment as a WP is reached. This implies that replanning of the current path is only considered upon reaching a WP. However, the criterion for when to replan the path should rather be related to changes in the operational environment. In the simulation studies presented here, this means that replanning should be initiated the instant new objects and obstacles are detected or move.



(a) Commanded $\boldsymbol{\tau}$



(b) Path parameter

Figure 6.5.2: Commanded control forces and moment, and the path parameter s'.

6.5.2 The pragmatic versus the optimal path generator

The pragmatic and optimal path generators based on Bézier curves, presented in Section 5.2.2, were tested together with the straight-line hybrid A* path planner. The resulting paths and the environment in which the path generators were tested are displayed in Figure 6.5.3. Parameter values associated to the two path-generation approaches are listed in Table 6.5.1. Some parameter values were adjusted compared to the values in Knædal (2019) to better fit with the discretization of the external environment and the path planner. For instance, the upper bound on the distance between two WPs, Δ_{max}, was set to two times the discretization size, dX. Note that the corridor breadth, ζ, is fixed, although ζ as function of the distance to the closest obstacle, as proposed in Section 5.2, would probably be more appropriate and promote consistency between the path planner and the path generator. However, due to time constraints this possibility was not explored further.

Table 6.5.1: Parameters of the pragmatic and optimal path generators.

(a) Pragmatic

Parameter	Value
κ_{max}	3
ζ	$\frac{1}{2}dX$
μ	6
δ_{min}	2

(b) Optimal

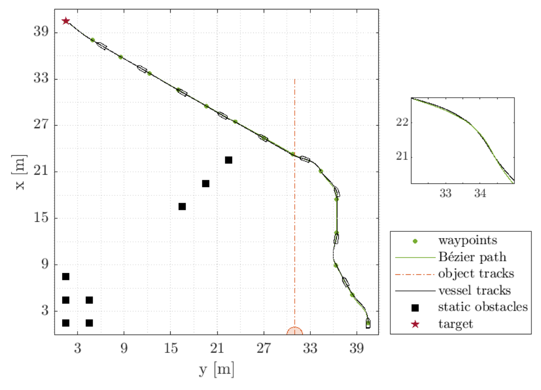
Parameter	Value
κ_{max}	3
ζ	$\frac{1}{2}dX$
N_{ss}	40
Δ_{max}	$2dX$

The visual appearance of the two paths in Figure 6.5.3 is quite similar. Initially, both paths are attracted straight towards the target. However, as the OS detects the moving object on its starboard side, the course is altered to starboard and the OS passes behind the object in compliance with Rule 15 in COLREGs (see Section 2.1). Although the two generated paths seem quite alike and both are closely followed by the OS, there are some differences when evaluating the selected KPIs. In addition to the KPIs used for the path planner (see Section 4.2.6), two quantitative measures of how well the OS is able to follow the generated path are evaluated:

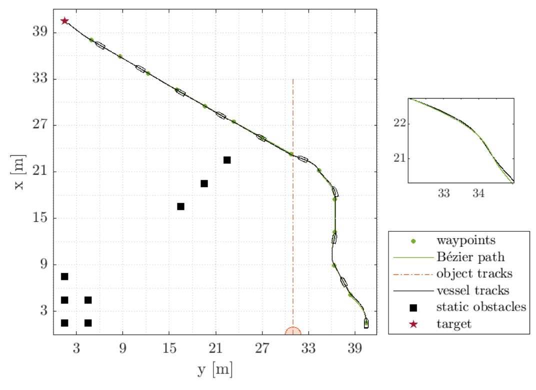
$$\frac{1}{N_{dat}} \sum_{k}^{N_{dat}} \left| \boldsymbol{p}_{d,k} - \boldsymbol{p}_k \right|, \quad \text{and} \tag{6.5.2a}$$

$$\max_{k \in N_{dat}} \left| \boldsymbol{p}_{d,k} - \boldsymbol{p}_k \right|. \tag{6.5.2b}$$

The former is the average deviation from the path and the latter specifies the maximum deviation between the OS position and the path, with N_{dat} being the number of data points. Given the parameter values in Table 6.5.1 and the environment illustrated in Figure 6.5.3, the optimal approach yields a path that the OS is less capable of following compared to the pragmatic Bézier-based path. However, the difference in maximal deviation (6.5.2b) of 1mm is negligible considering the OS length L and discretization of the environment dX, dY. Furthermore, the average deviation (6.5.2a) from the path is identical. Hence, when it comes to generating a path that the OS is able to follow, both the pragmatic and the optimal approach can be regarded equally feasible.



(a) Pragmatic



(b) Optimal

Figure 6.5.3: Bézier-based paths given WPs from the straight-line hybrid A* path planner in a dynamic landscape with a moving object and static obstacles. The smaller axes display a zoomed cut of the paths and the OS tracks.

Even if the deviations between the OS tracks and the paths are negligible, the slight difference between the two path-generation approaches can probably be imputed to the curvature along the path. Table 6.5.2 reveals that the pragmatic Bézier-based path has a maximum curvature below the assumed limit of $\frac{1}{R_{min}}$ with $R_{min} = \frac{3}{2}L$, whereas the maximum curvature along the optimal path is 20 percent above and corresponds to a steady-state turn radius of approximately $R = \frac{5}{4}L$. In fact the parameter value for maximum curvature, κ_{max}, had to be relaxed to 3[m^{-1}] to obtain a path at all, and even 2[m^{-1}] was too restrictive. Nevertheless, since the OS is able to follow the optimal path closely, a minimum turning radius of $R_{min} = \frac{3}{2}L$ is apparently a conservative choice.

Path length is a measure adopted from the evaluation of the path planners, but it is also relevant for the path generator as it is responsible for producing the actual path to be followed. Similar to the deviations between the OS tracks and the path, one can hardly distinguish between the path-generation approaches based on path length relative to L and dX. The difference in path length is 10cm, but contrary to the deviations between the OS tracks and the generated paths, it is in favor of the optimal approach. The difference can probably also be imputed to the curvature along the paths: Since the curvature along the optimal Bézier-based path is higher, it will take sharper turns and thereby yield a shorter path.

Table 6.5.2: KPIs of the pragmatic and optimal Bézier path generators.

KPI	Unit	Pragmatic	Optimal
Path length	[m]	59.4	59.3
Computation time	[s]	$7.9 \cdot 10^{-5}$	$2.7 \cdot 10^{-4}$
Max curvature	[m^{-1}]	0.45	0.72
Average deviation	[m]	0.065	0.065
Max deviation	[m]	0.21	0.22

Lastly, the generation of a desired parametric path definition is timed. As mentioned in Section 1.3, the path planner is subject to onboard computer limitations. The associated time constraints do necessarily also apply to the path generator. Therefore, execution times of the path-generation algorithms are important to assess. The path generator is called at each time step in order to constantly extend the path as the OS moves forward. Therefore the computation time is given as the average time for the path generator to produce the current desired pose $\boldsymbol{\eta}_d$ and its derivatives. The first call to the path generator is excluded from the average value since start-up times are generally longer than the times of subsequent calls to a program. Furthermore, the time given in Table 6.5.2 is the mean value of 5 completed simulations, as the elapsed times differ slightly for each simulation. Both path-generation approaches seem efficient, considering that the simulation step is fixed to 1e−3s. However, the pragmatic approach is on average 3.4 times faster than the optimal approach. Considering the computational efficiency of a well-tuned pragmatic Bézier-based path generator along with its relatively simple design compared to the optimal approach, it seems promising in solving the path-generation problem given in Section 2.2.2.

There are several thinkable improvements of the Bézier-based path generators. An enhancement already mentioned is to utilize information from the path planner and use a variable corridor breadth ζ that depends on the distance between the straight-line segment connecting two succeeding WPs and the nearest obstructed cell, as illustrated in Figure 5.2.1. Furthermore, if several WPs ahead would be provided by the path planner, one could reconsider if the path necessarily has to pass through all WPs and hence relax constraints on the path generator. Even if all WPs must be included in the path, the knowledge of two or three WPs ahead would potentially result in a more convenient heading at the next WP than the prevailing desired heading that is aligned with the straight line between the previous and the next WP. The latter choice is indeed practical when only one WP ahead is known, since it yields zero curvature at that WP and hence makes a turn towards starboard and port side equally feasible. However, if the second next WP is known, curved path segments, as observed in Figure 6.5.3, could potentially be smoothed. In short, more information possessed by the path planner should be utilized by the path generator.

6.5.3 Interaction between the path planner and the path generator

It seems like the Bézier-based path generator works the best for paths defined by equispaced WPs. If two WPs are close to each others, as might happen with the kinematics-based hybrid A*, the generated path can become difficult for the vessel to follow. An example is illustrated in Figure 6.5.4, where two WPs in the zoomed plot to the right are about a vessel length L apart from each other and issue a winding path given the parameter values in Table 6.5.1a. Decreasing the minimum distance δ_{min} between control points 4 and 7 seems to help. In other words, path-generating parameters adjusted to the length of the current motion primitive length would probably benefit the kinematics-based hybrid A*. Nevertheless, closely placed WPs leave little room for the path generator to trace out a path. With these considerations in mind, the Bézier-based path-generation methods are probably best suited for WPs placed by the straight-line hybrid A* path search.

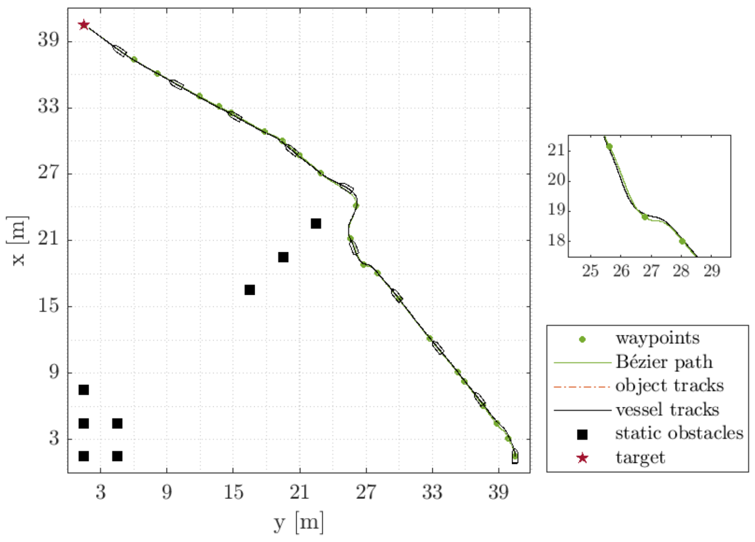


Figure 6.5.4: Path generated with the pragmatic approach given WPs from the kinematics-based hybrid A* path planner. Tight and uneven spacing between WPs complicates the task of generating a feasible path.

Chapter 7

Conclusions and further work

7.1 Conclusions

A path planner for an autonomous surface vehicle – composed of a guidance model, an anti-collision strategy, and a path search algorithm – was developed. The bio-inspired neural network landscape in Yang and Meng (2001) served as the guidance model, and effectively provided a hybrid-state A* path search algorithm with information about the locations of the target, obstructed cells, and free space. Two methods for expanding the hybrid A* search tree were proposed, of which expansions by straight-line segments of constant length together with a heuristic cost based on the Euclidean distance to the target finished the path search in the shortest time.

Simulations of different encounter situations demonstrated that the anti-collision scheme yielded paths compliant with COLREGs. An adaptive avoidance region of rectangular shape, that the vessel under control ought to steer clear of, was defined according to the relative distance and angle between the two encountering vessels. A pitfall of the method is that the ownship might get trapped inside the region, which may lead to potentially hazardous situations. Therefore a redefinition of the avoidance region is required, allowing the vessel to escape or at least to enter a state of minimum risk.

The path planner was integrated into a guidance, navigation, and control system with a path generator based on Bézier curves, a backstepping maneuvering controller and a nonlinear passive observer. Simulations indicated that the WPs provided by the path planner were suitable for path-generation, and the hybrid A* path search with straight-line expansions together with a pragmatic approach of generating a Bézier path yielded particularly promising results. Further testing is needed to validate the path planner, in which environmental impacts from waves, winds and ocean currents should be included and with several ships present.

7.2 Recommendations for further work

Suggestions for a way forward include:

- Conduct more realistic simulations by considering effects of ocean currents, wind and waves, as well as traffic separation schemes.
- Investigate the performance of the path planner in congested waters with multi-ship encounters, and how communication between vessels can enhance the replanning of a path including situations where there initially is no risk of collision.
- The planner should immediately replan the path when changes in the environment are detected, rather than waiting until the next WP is reached.
- Experimental testing on an ASV scale model.
- Reconsider the discretization of the environment and, for instance, test if Voronoi fields more elegantly can produce paths that adhere to a safety distance to obstacles.
- The proposed path planner is intended for an ASV in transit mode, and, in order to enable a complete voyage, it should be merged together with planners for in-harbor maneuvering and docking, as addressed by fellow master students Jakob S. Jensen and Elias Gauslaa, respectively.
- Assess if other parameters than the turn rate and minimum turning circle of the vessel should be considered when path feasibility is concerned.
- Explore alternative shapes of the avoidance region, and how to omit that the vessel gets trapped inside the region. Define feasible safe states, in addition to a DP mode, that the vessel can enter when the complexity exceeds the capability of the autonomous system, referred to as MRCs in DNV GL (2018).
- Forward two or several of the planned waypoints to the Bézier-based path generator, and thereby utilize more of the information possessed by the path planner. A suggestion for a path generation strategy when two waypoints ahead are known is illustrated in Figure 7.2.1. Here, straight lines connecting the current and the two waypoints ahead define the boundaries of the control polygon.

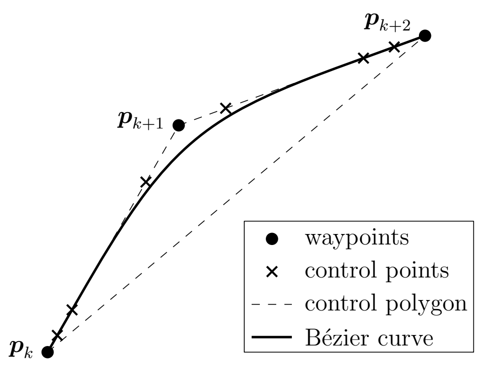


Figure 7.2.1: Suggestion for a Bézier-based path-generation strategy if two WPs ahead are given by the path planner: Let the straight lines connecting the WPs be the control polygon.

Bibliography

Aggarwal, S. and N. Kumar (2020). Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications* 149, pp. 270–299. doi: 10.1016/j.comcom.2019.10.014.

Barile, M. and E. W. Weisstein (n.d.). *Arc.* From MathWorld – A Wolfram Web Resource. Available at: https://mathworld.wolfram.com/Arc.html (accessed: May 20, 2020).

Beard, R. W. and T. W. McLain (2012). *Small Unmanned Aircraft. Theory and Practice.* Princeton, NJ, USA: Princeton University Press.

Bousson, K. (2008). Model predictive control approach to global air collision avoidance. *Aircraft Engineering and Aerospace Technology* 80, pp. 605–612. doi: 10.1108/00022660810911545.

Campbell, S. et al. (2020). "Path Planning Techniques for Mobile Robots A Review." *6th International Conference on Mechatronics and Robotics Engineering (ICMRE).* Barcelona, Spain, pp. 12–16. doi: 10.1109/ICMRE49073.2020.9065187.

Chauvin, C. et al. (2013). Human and organisational factors in maritime accidents: Analysis of collisions at sea using the HFACS. *Accident Analysis & Prevention* 59, pp. 26–37. doi: doi.org/10.1016/j.aap.2013.05.006.

Chen, L., H. Hopman, and R. R. Negenborn (2018). Distributed model predictive control for vessel train formations of cooperative multi-vessel systems. *Transportation Research Part C: Emerging Technologies* 92, pp. 101–118. doi: 10.1016/j.trc.2018.04.013.

Chiang, H. L. and L. Tapia (2018). COLREG-RRT: An RRT-Based COLREGS-Compliant Motion Planner for Surface Vehicle Navigation. *IEEE Robotics and Automation Letters* 3(3), pp. 2024–2031. doi: 10.1109/LRA.2018.2801881.

Choi, J., R. Curry, and G. Elkaim (2008). "Path Planning Based on Bézier Curve for Autonomous Ground Vehicles." *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, pp. 158–166. doi: 10.1109/WCECS.2008.27.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik* 1, pp. 269–271. doi: 10.1007/BF01386390.

DNV GL (2015). *Årsaksanalyse av grunnstøtinger og kollisjoner i norske farvann.* (Sjøsikkerhetsanalysen 2014 2014-1332 C). Available at: `www.kystverket.no/Nyheter/2015/November/Peker-ut-videre-kurs-for-arbeid-med-sjosikkerhet/` (accessed: Apr. 14, 2020).

DNV GL (2018). *DNVGL-CG-0264 Class guideline: Autonomous and remotely operated ships.* Available at: `www.dnvgl.com/maritime/autonomous-remotely-operated-ships/class-guideline.html` (accessed: May 7, 2020).

Dolgov, D. et al. (2008). Practical Search Techniques in Path Planning for Autonomous Driving. *AAAI Workshop - Technical Report.*

Dragland, K. (n.d.). *Autonomous pilot ferry for concept testing and to study behaviour of the other boat traffic.* Image. NTNU. Available at: `https://www.ntnu.edu/autoferry` (accessed: Apr. 28, 2020).

Dubins, L. E. (1957). On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics* 79(3), pp. 497–516. doi: `10.2307/2372560`.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer* 22(6), pp. 46–57. doi: `10.1109/2.30720`.

Eriksen, B.-O. H., G. Bitar, et al. (2020). Hybrid Collision Avoidance for ASVs Compliant with COLREGs Rules 8 and 13-17. *Frontiers in Robotics and AI* 7(11). doi: `10.3389/frobt.2020.00011`.

Eriksen, B.-O. and M. Breivik (2017). "MPC-based Mid-level Collision Avoidance for ASVs using Nonlinear Programming." doi: `10.1109/CCTA.2017.8062554`.

Farouki, R. T. (2012). The Bernstein polynomial basis: A centennial retrospective. *Computer Aided Geometric Design* 29(6), pp. 379–419. doi: `10.1016/j.cagd.2012.03.001`.

Ferguson, D. and M. Likhachev (2008). "Efficiently Using Cost Maps For Planning Complex Maneuvers." *Proceedings of International Conference on Robotics and Automation Workshop on Planning with Cost Maps.*

Ferrer, G. and A. Sanfeliu (2014). "Proactive kinodynamic planning using the Extended Social Force Model and human motion prediction in urban environments." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Chicago, IL, pp. 1730–1735. doi: `10.1109/IROS.2014.6942788`.

Flanagan, C. (n.d.). *The Bresenham Line-Drawing Algorithm.* Available at: `https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html` (accessed: May 30, 2020).

Ford (n.d.). *One Step Ahead of Pedestrians.* Available at: `www.corporate.ford.com/articles/products/petextrian.html` (accessed: Apr. 16, 2020).

Foss, B. and T. A. N. Heirung (2016). *Merging Optimization and Control.* Norwegian University of Science and Technology (NTNU).

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control.* 1st ed. Chichester, UK: John Wiley & Sons.

Fossen, T. I. (2020). *Handbook of Marine Craft Hydrodynamics and Motion Control.* 2nd ed. Chichester, UK: John Wiley & Sons.

Fox, D., W. Burgard, and S. Thrun (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1), pp. 23–33. doi: `10.1109/100.580977`.

Hagen, I. B. et al. (2018). "MPC-based Collision Avoidance Strategy for Existing Marine Vessel Guidance Systems." *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7618–7623. doi: `10.1109/ICRA.2018.8463182`.

Hart, P. E., N. J. Nilsson, and B. Raphael (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2), pp. 100–107. doi: `10.1109/TSSC.1968.300136`.

Hassani, V. and S. V. Lande (2018). Path Planning for Marine Vehicles using Bézier Curves. *IFAC-PapersOnLine* 51(29), pp. 305–310. doi: `10.1016/j.ifacol.2018.09.520`.

Helsedirektoratet (2020). *The Norwegian Directorate of Health has issued a decision to close schools and other educational institutions.* Available at: `https://www.helsedirektoratet.no/nyheter/the-norwegian-directorate-of-health-has-issued-a-decision-to-close-schools-and-other-educational-institutions` (accessed: May 31, 2020).

Hodgkin, A. L. and A. F. Huxley (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology* 117(4), pp. 500–544. doi: `10.1113/jphysiol.1952.sp004764`.

Hu, Y. and S. X. Yang (2004). "A knowledge based genetic algorithm for path planning of a mobile robot." *IEEE International Conference on Robotics and Automation. Proceedings. ICRA '04.* Vol. 5. New Orleans, LA, USA, pp. 4350–4355. doi: `10.1109/ROBOT.2004.1302402`.

Huang, Y. et al. (2020). Ship collision avoidance methods: State-of-the-art. *Safety Science* 121, pp. 451–473. doi: `https://doi.org/10.1016/j.ssci.2019.09.018`.

International Maritime Organization (IMO) (1972). *COLREG: Convention on the International Regulations for Preventing Collisions at Sea.* eng. London, UK.

International Maritime Organization (IMO) (2019). *Interim guidelines for MASS trials.* MSC.1 /Circ.1604. London.

International Maritime Organization (IMO) (n.d.). *Autonomous shipping.* Available at: `www.imo.org/en/MediaCentre/HotTopics/Pages/Autonomous-shipping.aspx` (accessed: May 6, 2020).

Jaillet, L., J. Cortes, and T. Simeon (2008). “Transition-based RRT for path planning in continuous cost spaces.” *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2145–2150. doi: 10.1109/IROS.2008.4650993.

Johansen, T. A., T. Perez, and A. Cristofaro (2016). Ship Collision Avoidance and COLREGS Compliance Using Simulation-Based Control Behavior Selection With Predictive Hazard Assessment. *IEEE Transactions on Intelligent Transportation Systems* 17(12), pp. 3407–3422. doi: 10.1109/TITS.2016.2551780.

Jokioinen, E. (2016). “Introduction.” In: *Remote and Autonomous Ships – The next steps.* (white paper). Rolls-Royce plc. Available at: https://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/aawa-whitepaper-210616.pdf (accessed: Jan. 23, 2020).

Jolly, K., R. S. Kumar, and R. Vijayakumar (2009). A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems* 57(1), pp. 23–33. doi: https://doi.org/10.1016/j.robot.2008.03.009.

Joy, K. I. (2000). *A Matrix Formulation of the Cubic Bézier Curve.* On-Line Geometric Modeling Notes. Visualization and Graphics Research Group at the University of California, Davis.

Kamal, W., D.-W. Gu, and I. Postlethwaite (2005). MILP and its application in flight path planning. *IFAC Proceedings Volumes* 38(1). 16th IFAC World Congress, pp. 55–60. doi: 10.3182/20050703-6-CZ-1902.02061.

Khatib, O. (1985). “Real-time Obstacle Avoidance for Manipulators and Mobile Robots.” *Proceedings of the IEEE International Conference on Robotics & Automation.* St. Louis, MO, USA, pp. 500–505. doi: 10.1109/ROBOT.1985.1087247.

Knædal, M. (2019). “Stepwise Path-Generation using Bézier Curves.” Specialization project report. Norwegian University of Science and Technology (NTNU).

Kongsberg (2020). *Automatic ferry enters regular service following world-first crossing with passengers onboard.* Press release. Available at: www.kongsberg.com/maritime/about-us/news-and-media/news-archive/2020/first-adaptive-transit-on-bastofosen-vi (accessed: Apr. 15, 2020).

Kongsberg (n.d.). *Autonomous Ship Project, Key Facts about YARA Birkeland.* Available at: www.kongsberg.com/maritime/support/themes/autonomous-ship-project-key-facts-about-yara-birkeland (accessed: Nov. 24, 2019).

Kühner, A. (2017). Unmanned ships on the horizon. *Maritime Impact* 2017(1), pp. 20–23. Available at: www.issuu.com/dnvgl/docs/dnv_gl_maritime_impact_01-2017 (accessed: May 7, 2020).

Li, Z. et al. (2016). Trajectory-Tracking Control of Mobile Robot Systems Incorporating Neural-Dynamic Optimized Model Predictive Approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46(6), pp. 740–749. doi: 10.1109/TSMC.2015.2465352.

Lu, D. V., D. Hershberger, and W. D. Smart (2014). "Layered costmaps for context-sensitive navigation." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 709–715. doi: 10.1109/IROS.2014.6942636.

Mercedes-Benz (n.d.). *By far the best: Mercedes-Benz assistance systems.* Available at: www.mercedes-benz.com/en/innovation/by-far-the-best-mercedes-benz-assistance-systems1 (accessed: Apr. 16, 2020).

Nash, A. and S. Koenig (2013). Any-Angle Path Planning. *AI Magazine* 34(4), pp. 85–107. doi: 10.1609/aimag.v34i4.2512.

Nikola Tesla Museum (n.d.). *Interactive Chronology. 1898 Demonstrates operation of "teleautomaton" boat.* Available at: https://nikolateslamuseum.org/en/chronology/ (accessed: Apr. 1, 2020).

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization.* 2nd ed. Springer Series in Operations Research and Financial Engineering. New York, NY, USA: Springer.

Nordstoga, A. K. (2019). "AutoVoyage: Autonomous path-planning, path-generation, and path-following for autonomous ships in transit." Master's Thesis. Norwegian University of Science and Technology (NTNU).

Norwegian Ministry of Trade, Industry and Fisheries (NFD) (2019). *Blue Opportunities – The Norwegian Government's updated ocean strategy.* Available at: www.regjeringen.no/en/dokumenter/the-norwegian-governments-updated-ocean-strategy/id2653026 (accessed: Apr. 19, 2020).

Norwegian University of Science and Technology (NTNU) (2020). *CyberShip Enterprise I, User Manual.* Department of Marine Technology, NTNU. Revised by Gauslaa, E., Jensen, J. S., and Fleischer, C.

Norwegian University of Science and Technology (NTNU) (n.d.). *NTNU AMOS – Centre for autonomous marine operations and systems. Associated Projects.* Available at: https://www.ntnu.edu/amos/research (accessed: Apr. 1, 2020).

Oh, S.-R. and J. Sun (2010). Path following of underactuated marine surface vessels using line-of-sight based model predictive control. *Ocean Engineering* 37(2), pp. 289–295. doi: https://doi.org/10.1016/j.oceaneng.2009.10.004.

Panda, M. et al. (2020). A Comprehensive Review of Path Planning Algorithms for Autonomous Underwater Vehicles. *International Journal of Automation and Computing.* doi: 10.1007/s11633-019-1204-9.

Petereit, J. et al. (2012). “Application of Hybrid A* to an autonomous mobile robot for path planning in unstructured outdoor environments.” *ROBOTIK 2012: 7th German Conference on Robotics*, pp. 1–6.

Poikonen, J. et al. (2016). “Technology.” In: *Remote and Autonomous Ships – The next steps.* (white paper). Rolls-Royce plc. Available at: `https://www.rolls-royce.com/~/media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/aawa-whitepaper-210616.pdf` (accessed: Jan. 23, 2020).

Porathe, T. (2017). Is COLREG enough? Interaction Between Manned and Unmanned Ships. *International Journal on Marine Navigation and Safety of Sea Transportation.* 12th International Conference on Marine Navigation and Safety of Sea Transportation, pp. 191–194. doi: `10.1201/9781315099132-33`.

Premakumar, P. (2016). *A* (A Star) search for path planning tutorial.* Version 1.2.0.1. Available at: `https://se.mathworks.com/matlabcentral/fileexchange/26248-a-a-star-search-for-path-planning-tutorial` (accessed: Nov. 1, 2019).

Reif, J. and M. Sharir (1985). “Motion planning in the presence of moving obstacles.” *26th Annual Symposium on Foundations of Computer Science.* Portland, OR, USA, pp. 144–154. doi: `10.1109/SFCS.1985.36`.

Richards, A., E. Feron, et al. (2002). Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming. *Journal of Guidance Control and Dynamics* 25. doi: `10.2514/2.4943`.

Richards, N. D., M. Sharma, and D. G. Ward (2004). “A Hybrid A*/Automaton Approach to On-Line Path Planning with Obstacle Avoidance.” *AIAA 1st Intelligent Systems Technical Conference.* doi: `10.2514/6.2004-6229`.

Savitz, S. et al. (2013). “Introduction.” In: *U.S. Navy Employment Options for Unmanned Surface Vehicles (USVs).* RAND Corporation, pp. 1–6.

Schouwenaars, T. et al. (2001). “Mixed integer programming for multi-vehicle path planning.” *2001 European Control Conference (ECC)*, pp. 2603–2608. doi: `10.23919/ECC.2001.7076321`.

Scibilia, F., U. Jørgensen, and R. Skjetne (2012). “AUV Guidance System for Subsurface Ice Intelligence.” *Proceedings of the ASME 2012 31st International Conference on Ocean, Offshore and Arctic Engineering.* Rio de Janeiro, Brasil. doi: `10.1115/OMAE2012-83811`.

Šeda, M. (2007). “Roadmap methods vs. cell decomposition in robot motion planning.” *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation (ISPRA’07).* Stevens Point, WI, USA, pp. 127–132.

Skjetne, R. (2005). “The Maneuvering Problem.” PhD thesis. Norwegian University of Science and Technology (NTNU).

Skjetne, R. (2019). *TMR4243 Marine Control Systems II – Lecture 7: Maneuvering Control Design*. Department of Marine technology, NTNU.

Skjetne, R. (2020). *Maneuvering-based guidance design for dynamic positioning*. Technical note. Version Revision B (Draft). Norwegian University of Science and Technology (NTNU).

SNAME (1950). *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid*. Technical and Research Bulletin 1–5. The Society of Naval Architects and Marine Engineers, pp. 1–15.

Song, L. et al. (2019). Collision avoidance planning for unmanned surface vehicle based on eccentric expansion. *International Journal of Advanced Robotic Systems* 16(3). doi: 10.1177/1729881419851945.

Sørensen, A. J. (2018). *Marine Cybernetics – Towards Autonomous Marine Operations and Systems*. Lecture Notes. Department of Marine Technology, NTNU.

Tam, C. and R. Bucknall (2010). Collision risk assessment for ships. *Journal of Marine Science and Technology (JMST)* 15, pp. 257–270. doi: 10.1007/s00773-010-0089-7.

Tesla, N. (1898). "Method of and Apparatus for Controlling Mechanism of Moving Vessels or Vehicles." United States patent 613809.

The Tesla Team (2016). *All Tesla Cars Being Produced Now Have Full Self-Driving Hardware*. Blog post. Available at: www.tesla.com/blog/all-tesla-cars-being-produced-now-have-full-self-driving-hardware (accessed: Apr. 16, 2020).

Thyri, E. H. et al. (2020). "Reactive collision avoidance for ASVs based on control barrier functions." *4th IEEE Conference on Control Technology and Applications (CCTA 2020)*. Forthcoming. Montreal, Canada.

Trudeau, R. (1993). *Introduction to Graph Theory*. Dover Books on Mathematics. Mineola: Dover Publications.

Værnø, S. A. and R. Skjetne (2017). *Observer for simplified DP model: Design and proof*. Norwegian University of Science and Technology (NTNU), Trondheim.

Valle, E. (2015). "Marine Telepresence System." Master's thesis. Norwegian University of Science and Technology (NTNU).

Wärtsilä (2018). *World's first Autodocking installation successfully tested by Wärtsilä*. Press release. Available at: www.wartsila.com/media/news/26-04-2018-world-s-first-autodocking-installation-successfully-tested-by-wartsila-2169290 (accessed: Apr. 15, 2020).

Waymo (n.d.). *Our Journey*. Available at: www.waymo.com/journey (accessed: Apr. 16, 2020).

Weisstein, E. W. (n.d.[a]). *Metric.* From MathWorld – A Wolfram Web Resource. Available at: `https://mathworld.wolfram.com/Metric.html` (accessed: May 20, 2020).

Weisstein, E. W. (n.d.[b]). *NP-Problem.* From MathWorld – A Wolfram Web Resource. Available at: `https://mathworld.wolfram.com/NP-Problem.html` (accessed: Apr. 28, 2020).

Weisstein, E. W. (n.d.[c]). *Vector Norm.* From MathWorld – A Wolfram Web Resource. Available at: `https://mathworld.wolfram.com/VectorNorm.html` (accessed: May 20, 2020).

Wiig, M. S., K. Y. Pettersen, and A. V. Savkin (2017). "A reactive collision avoidance algorithm for nonholonomic vehicles." *2017 IEEE Conference on Control Technology and Applications (CCTA).* Mauna Lani, HI, pp. 1776–1783. doi: `10.1109/CCTA.2017.8062714`.

Williams, J. F. (2016). *Sea Hunter gets underway on the Williammette River following a christening ceremony in Portland, Ore.* Image. Official U.S. Navy Page on Flickr. Available at: `https://flickr.com/photos/56594044@N06/25702146834` (accessed: Apr. 28, 2020).

Wright, R. G. (2020). *Unmanned and Autonomous Ships. An Overview of MASS.* 1st ed. Abingdon, UK: Routledge.

Yang, S. X. and M. Meng (2001). Neural Network Approaches to Dynamic Collision-Free Trajectory Generation. *IEEE Transactions on Systems, Man, and Cybernetics,* 31(3), pp. 302–318. doi: `10.1109/3477.931512`.

Yara International ASA (2019). *Yara Birkeland.* Image. Available at: `https://www.yara.com/news-and-media/media-library/image-library/` (accessed: Apr. 28, 2020).

Zaccone, R., M. Martelli, and M. Figari (2019). "A COLREG-Compliant Ship Collision Avoidance Algorithm." *18th European Control Conference (ECC).* Naples, Italy, pp. 2530–2535. doi: `10.23919/ECC.2019.8796207`.

Appendix

Together with this report follows a folder with figures, videos, and the MATLAB/Simulink code files.

A Vessel specific matrices and coefficients for CSE1

Vessel specific matrices for the maneuvering model (3.1.5) of CSEI are given in NTNU (2020). These include the inertia and added mass matrix

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} = \boldsymbol{M}^\top > 0, \tag{A.1}$$

the Coriolis-centripetal matrix

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}) \tag{A.2}$$
$$= \begin{bmatrix} 0 & 0 & (-mx_g + Y_{\dot{r}})r + (-m + Y_{\dot{v}})v \\ 0 & 0 & (m - X_{\dot{u}})u \\ (mx_g - Y_{\dot{r}})r + (m - Y_{\dot{v}})v & (-m + X_{\dot{u}})u & 0 \end{bmatrix}.$$

and the hydrodynamic damping matrix

$$\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D} + \boldsymbol{D}_n(\boldsymbol{\nu}) = \begin{bmatrix} d_{11}(u) & 0 & 0 \\ 0 & d_{22}(v,r) & d_{23}(v,r) \\ 0 & d_{32}(v,r) & d_{33}(v,r) \end{bmatrix}. \tag{A.3}$$

The two former system matrices are composed of rigid-body and added mass components, denoted with a subscript RB and A respectively, and the latter is the sum of a linear $\boldsymbol{D}$ and a nonlinear $\boldsymbol{D}_n(\boldsymbol{\nu})$ damping matrix. The hydrodynamic damping components are given by

$$d_{11}(u) = -X_u - X_{|u|u}|u| - X_{uuu}u^2, \tag{A.4a}$$
$$d_{22}(v,r) = -Y_v - Y_{|v|v}|v| - Y_{|r|v}|r| - Y_{vvv}v^2, \tag{A.4b}$$
$$d_{23}(v,r) = -Y_r - Y_{|v|r}|v| - Y_{|r|r}|r| - Y_{rrr}r^2, \tag{A.4c}$$
$$d_{32}(v,r) = -N_v - N_{|v|v}|v| - N_{|r|v}|r| - N_{vvv}v^2, \tag{A.4d}$$
$$d_{33}(v,r) = -N_r - N_{|v|r}|v| - N_{|r|r}|r| - N_{rrr}r^2. \tag{A.4e}$$

The values of the parameters entering the model matrices are given in Table A.1 and Table A.2. Furthermore, the 3DOF rotation matrix transferring the generalized coordinates from $\{b\}$ into $\{n\}$ is given by

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.5}$$

where ψ is the heading of the vessel. Properties of the rotation matrix (A.5) include

$$\boldsymbol{R}(\psi)^\top \boldsymbol{R}(\psi) = \boldsymbol{I}, \tag{A.6a}$$

$$\dot{\boldsymbol{R}}(\psi) = \boldsymbol{R}(\psi)\boldsymbol{S}(r), \tag{A.6b}$$

$$\boldsymbol{S}(r) = -\boldsymbol{S}(r)^\top \tag{A.6c}$$

where $\boldsymbol{I}$ is the identity matrix, $r = \dot{\psi}$ is the yaw rate, and $\boldsymbol{S}(r)$ is a skew-symmetric matrix given by

$$\boldsymbol{S}(r) = \begin{bmatrix} 0 & -r & 0 \\ r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.7}$$

Table A.1: CSEI rigid body and added mass parameters. *Adopted from NTNU (2020).*

Rigid body		Added mass	
Parameter	Value	Parameter	Value
m	14.11	$X_{\dot{u}}$	-2.0
I_z	1.760	$Y_{\dot{v}}$	-10.0
x_g	0.0375	$Y_{\dot{r}}$,$N_{\dot{v}}$	-0.0
y_g	0.0	$N_{\dot{r}}$	-1.0

Table A.2: CSEI hydrodynamic damping parameters. *Adopted from NTNU (2020).*

Surge		Sway		Yaw	
Parameter	Value	Parameter	Value	Parameter	Value
X_u	-0.6555	Y_v	-1.33	N_v	0.0
$X_{\|u\|u}$	0.3545	$Y_{\|v\|v}$	-2.776	$N_{\|v\|v}$	-2.088
X_{uuu}	-3.787	Y_{vvv}	-64.91	N_{vvv}	0.0
X_v	0.0	Y_r	-7.25	N_r	-1.9
$X_{\|v\|v}$	-2.443	$Y_{\|r\|r}$	-3.45	$N_{\|r\|r}$	-0.75
X_{vvv}	0.0	Y_{rrr}	0.0	N_{rrr}	0.0
-	-	$Y_{\|r\|v}$	-0.805	$N_{\|r\|v}$	0.130
-	-	$Y_{\|v\|r}$	-0.845	$N_{\|v\|r}$	0.080

B Maneuvering controller equations

In the following, the compact set of equations describing the backstepping maneuvering controller presented in Section 5.3 is listed.

Backstepping state variables

The 1st step state variable in the backstepping design is given by

$$\boldsymbol{z}_1 = \boldsymbol{R}(\psi)^\top \left(\boldsymbol{\eta} - \boldsymbol{\eta}_d(s)\right), \tag{B.1a}$$

$$\dot{\boldsymbol{z}}_1 = -\boldsymbol{S}(r)\boldsymbol{z}_1 + \boldsymbol{\nu} + \boldsymbol{z}_1^s \dot{s}, \tag{B.1b}$$

$$\boldsymbol{z}_1^s = -\boldsymbol{R}(\psi)^\top \boldsymbol{\eta}_d^s(s), \tag{B.1c}$$

and the 2nd step state variable is defined such that

$$\boldsymbol{z}_2 = \boldsymbol{\nu} - \boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t), \tag{B.2a}$$

$$\dot{\boldsymbol{z}}_2 = \boldsymbol{M}^{-1}\left(-\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{R}(\psi)^\top \boldsymbol{b} + \boldsymbol{\tau}\right) - \dot{\boldsymbol{\alpha}}_1(\boldsymbol{\eta}, s, t). \tag{B.2b}$$

Stabilizing virtual control

A stabilizing function for the $\boldsymbol{z}_1$-subsystem is

$$\boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t) = -\boldsymbol{K}_1\boldsymbol{z}_1 + \boldsymbol{R}(\psi)^\top \boldsymbol{\eta}_d^s(s) v_d(s,t), \quad \boldsymbol{K}_1 = \boldsymbol{K}_1^\top > \boldsymbol{0} \tag{B.3a}$$

$$\dot{\boldsymbol{\alpha}}_1(\boldsymbol{\eta}, s, t) = -\boldsymbol{K}_1\dot{\boldsymbol{z}}_1 + \boldsymbol{S}(r)\boldsymbol{z}_1^s v_d(s,t) + \boldsymbol{R}(\psi)^\top \boldsymbol{\eta}_d^{2s}(s) v_d(s,t)\dot{s} - \boldsymbol{z}_1^s \dot{v}_d(s,t). \tag{B.3b}$$

Unit-tangent gradient update law

Following from a unit-tanget update law on the path parameter s, the speed assignment error and a tuning function are respectively given by

$$\omega = \dot{s} - v_d(s,t) = -\frac{\mu}{\left|\boldsymbol{\eta}_d^s(s)\right|} V_1^s, \tag{B.4a}$$

$$\rho = V_1^s. \tag{B.4b}$$

Control Lyapunov functions

The 1st step CLF is defined such that

$$V_1 = \frac{1}{2}\boldsymbol{z}_1^\top \boldsymbol{z}_1, \tag{B.5a}$$

$$\begin{aligned} \dot{V}_1 &= -\boldsymbol{z}_1^\top \boldsymbol{K}_1 \boldsymbol{z}_1 + \rho\omega + \boldsymbol{z}_1^\top \boldsymbol{z}_2 \\ &\leq -\lambda_{min}(\boldsymbol{K}_1)\left|\boldsymbol{z}_1\right|^2 + \boldsymbol{z}_1^\top \boldsymbol{z}_2, \end{aligned} \tag{B.5b}$$

$$V_1^s = -\boldsymbol{\eta}_d^s(s)^\top \left(\boldsymbol{\eta} - \boldsymbol{\eta}_d(s)\right), \tag{B.5c}$$

and the 2nd step CLF is given by

$$V_2 = V_1 + \frac{1}{2}\boldsymbol{z}_2^\top \boldsymbol{M}\boldsymbol{z}_2, \tag{B.6a}$$

$$\begin{aligned} \dot{V}_2 &= -\boldsymbol{z}_1^\top \boldsymbol{K}_1 \boldsymbol{z}_1 + \rho\omega - \frac{1}{2}\boldsymbol{z}_2^\top \left(\boldsymbol{D} + \boldsymbol{D}^\top\right)\boldsymbol{z}_2 - \boldsymbol{z}_2^\top \boldsymbol{K}_2 \boldsymbol{z}_2 \\ &\leq -\lambda_{min}(\boldsymbol{K}_1)\left|\boldsymbol{z}_1\right|^2 - \lambda_{min}(\boldsymbol{K}_2)\left|\boldsymbol{z}_2\right|^2. \end{aligned} \tag{B.6b}$$

Feedback control law

The resulting feedback control law, rendering $[\boldsymbol{z}_1, \boldsymbol{z}_2]^\top = \mathbf{0}$ uniformly globally exponentially stable and hence solves the maneuvering problem given by (2.2.7) and (2.2.8), is

$$\boldsymbol{\tau} = -\boldsymbol{z}_1 - \boldsymbol{K}_2\boldsymbol{z}_2 + \boldsymbol{D}\boldsymbol{\alpha}_1(\boldsymbol{\eta}, s, t) - \boldsymbol{R}(\psi)^\top\hat{\boldsymbol{b}} + \boldsymbol{M}\dot{\boldsymbol{\alpha}}_1(\boldsymbol{\eta}, s, t), \quad \boldsymbol{K}_2 = \boldsymbol{K}_2^\top > \mathbf{0}. \tag{B.7}$$

C Control allocation

The control law (5.3.8) solves the maneuvering problem (2.2.7) and (2.2.8). However, in order to realize the commanded thrust forces and moment $\boldsymbol{\tau}$, these must be transformed into setpoints $\boldsymbol{u}$ for each actuator according to (2.2.9). The control model (3.2.1) is derived under the assumption of a linear relationship between $\boldsymbol{\tau}$ and $\boldsymbol{u}$ given by (3.2.3). Hence, an explicit solution to the control allocation problem (2.2.9) is obtained from the generalized inverse allocation scheme (Fossen, 2011, pp. 403-405):

$$\boldsymbol{u} = \boldsymbol{K}^{-1}\boldsymbol{T}_W^\dagger\boldsymbol{\tau}, \tag{C.1a}$$

$$\boldsymbol{T}_W^\dagger = \boldsymbol{W}^{-1}\boldsymbol{T}^\top\left(\boldsymbol{T}\boldsymbol{W}^{-1}\boldsymbol{T}\right)^{-1}. \tag{C.1b}$$

The generalized inverse $\boldsymbol{T}_W^\dagger$ exists for an arbitrary matrix $\boldsymbol{T}$, including nonsquare matrices. If control forces are equally weighted $\boldsymbol{W} = \boldsymbol{I}$, (C.1b) reduces to the Moore-Penrose pseudo-inverse,

$$\boldsymbol{T}^\dagger = \boldsymbol{T}^\top\left(\boldsymbol{T}\boldsymbol{T}\right)^{-1}. \tag{C.2}$$

Although the solution (C.1) is valid for all thruster angles $\boldsymbol{\alpha}_0$, it is not optimal if $\boldsymbol{\alpha}_0$ varies in time. In the case of rotatable thrusters, a solution (C.1) is obtained by using the extended thrust configuration and coefficient matrices, $\boldsymbol{T}_e$ and $\boldsymbol{K}_e$. The contribution from a rotatable thruster i acting in the horizontal plane can be decomposed into two forces,

$$X_i = K_i u_i \cos\alpha_i = K_i u_{i,x}, \tag{C.3a}$$

$$Y_i = K_i u_i \sin\alpha_i = K_i u_{i,y}, \tag{C.3b}$$

yielding two elements, $u_{i,x}$ and $u_{i,y}$, in the extended control vector $\boldsymbol{u}_e$. Setpoints for rotatable thrusters can then be derived from the elements in $\boldsymbol{u}_e$ through the mapping given by

$$u_i = \sqrt{u_{i,x}^2 + u_{i,y}^2}, \tag{C.4a}$$

$$\alpha_i = \arctan 2(u_{i,y}, u_{i,x}). \tag{C.4b}$$

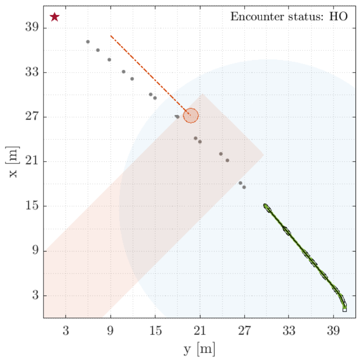
Accordingly, given the data in Table 6.2.1b, the combined extended thrust configuration and coefficient matrix for CSEI can be derived as

$$\boldsymbol{B}_e = \boldsymbol{T}_e\boldsymbol{K}_e = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ y_{VSP_1} & x_{VSP_1} & y_{VSP_2} & x_{VSP_2} & x_{BT} \end{bmatrix} \tag{C.5}$$

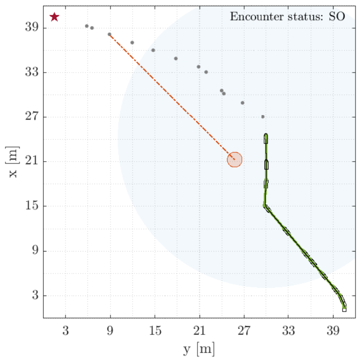
with extended input vector $\boldsymbol{u}_e = [u_{VSP_1,x}, u_{VSP_1,y}, u_{VSP_2,x}, u_{VSP_2,y}, u_{BT}]^\top$. The allocation problem (2.2.9) is unconstrained, implying that bounds on $\boldsymbol{f}$, $\boldsymbol{\alpha}$, and $\boldsymbol{u}$ are disregarded. Thus, a shortcoming of the solution (C.1) is that it may contain inappropriate setpoints for the actuators. A rotatable thruster may, for instance, end up in the wake flow of another. A constrained control allocation program for rotatable thrusters is given in Fossen (2011, pp. 408-411). However, being a nonconvex nonlinear optimization problem, the search for a solution entails a significant computational effort. Therefore, despite of ignoring actuator constraints, the simpler generalized inverse method (C.1) is considered adequate for the purpose of demonstrating the path-planning module.

D Simulations of encounter situations with kinematics-based hybrid A* path search

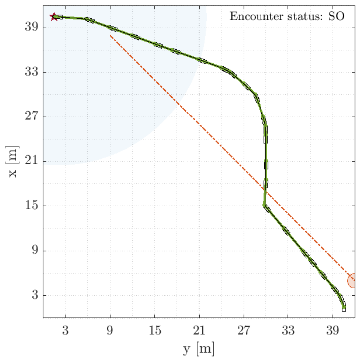
The path planners capability of handling several encounter situations in a COLREGs-compliant manner was demonstrated with the straight-line hybrid A* path search in Section 6.4.2. Similar results were achieved with the kinematics-based strategy. Plots of the five different encounter scenarios, under the exact same conditions as in Section 6.4.2, are presented in the following. The kinematics-based hybrid A* generally yields smoother paths than the straight-line alternative without a path generator. This is due to the variable length of the kinematics-based motion primitives (see Figure 4.2.3), which is typically small when the path turns, whereas the straight-line motion primitives have a constant length. Nevertheless, the overall behavior of the OS is quite similar with both the straight-line and kinematics-based strategy.



(a) Object detected

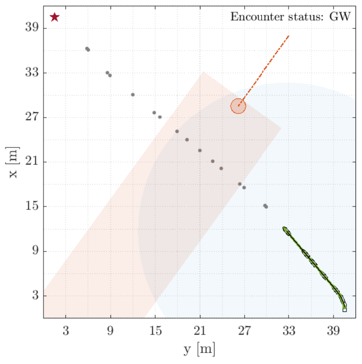


(b) Maneuver to starboard

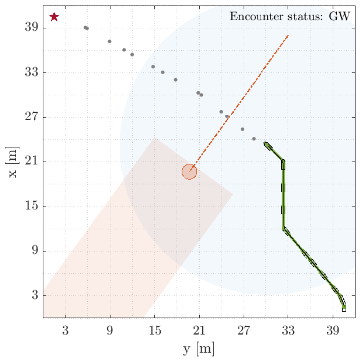


(c) Target reached

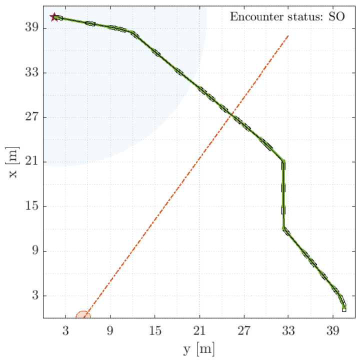
Figure D.1: An HO situation with kinematics-based hybrid A* path search.



(a) Object detected

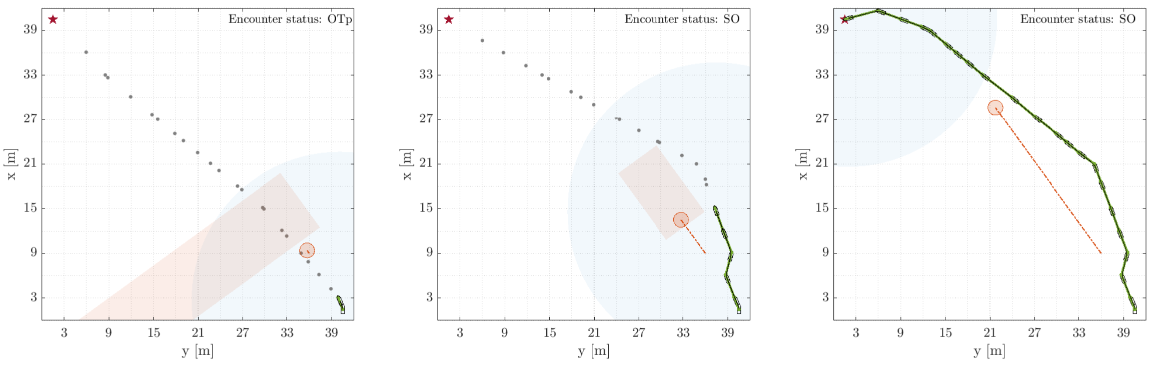


(b) Passing behind the TS



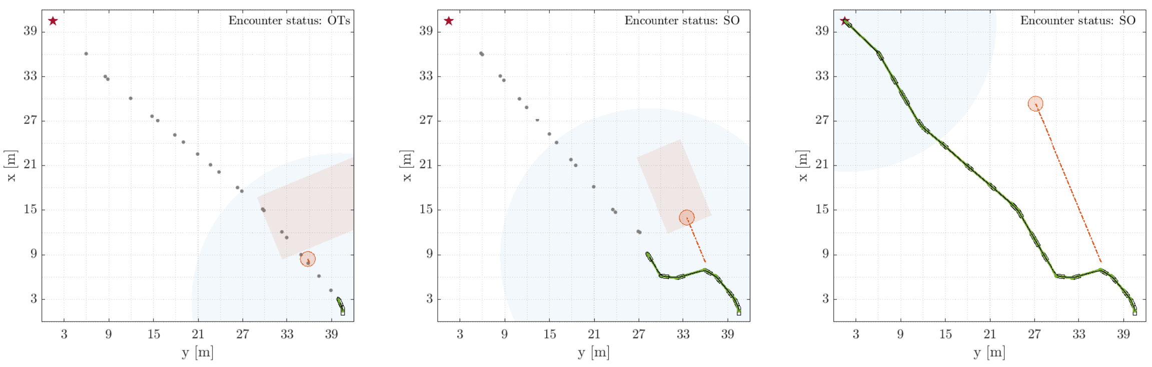
(c) Target reached

Figure D.2: A GW situation with kinematics-based hybrid A* path search.



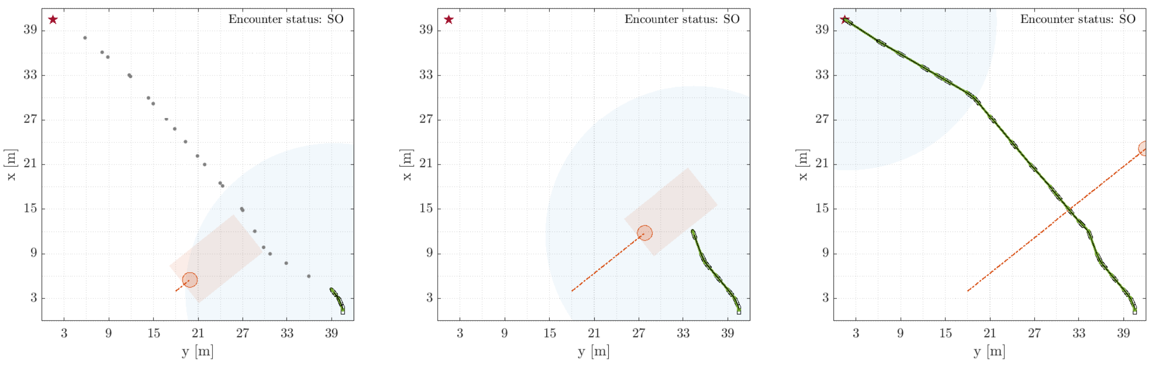
(a) Object detected (b) Maneuver to starboard (c) Target reached

Figure D.3: An OTp situation with kinematics-based hybrid A* path search.



(a) Object detected (b) Maneuver to port (c) Target reached

Figure D.4: An OTs situation with kinematics-based hybrid A* path search.



(a) Object detected (b) DP mode (c) Target reached

Figure D.5: An SO situation with kinematics-based hybrid A* path search. The OS enters DP mode as the TS takes no action, and thereafter passes in back of the TS.