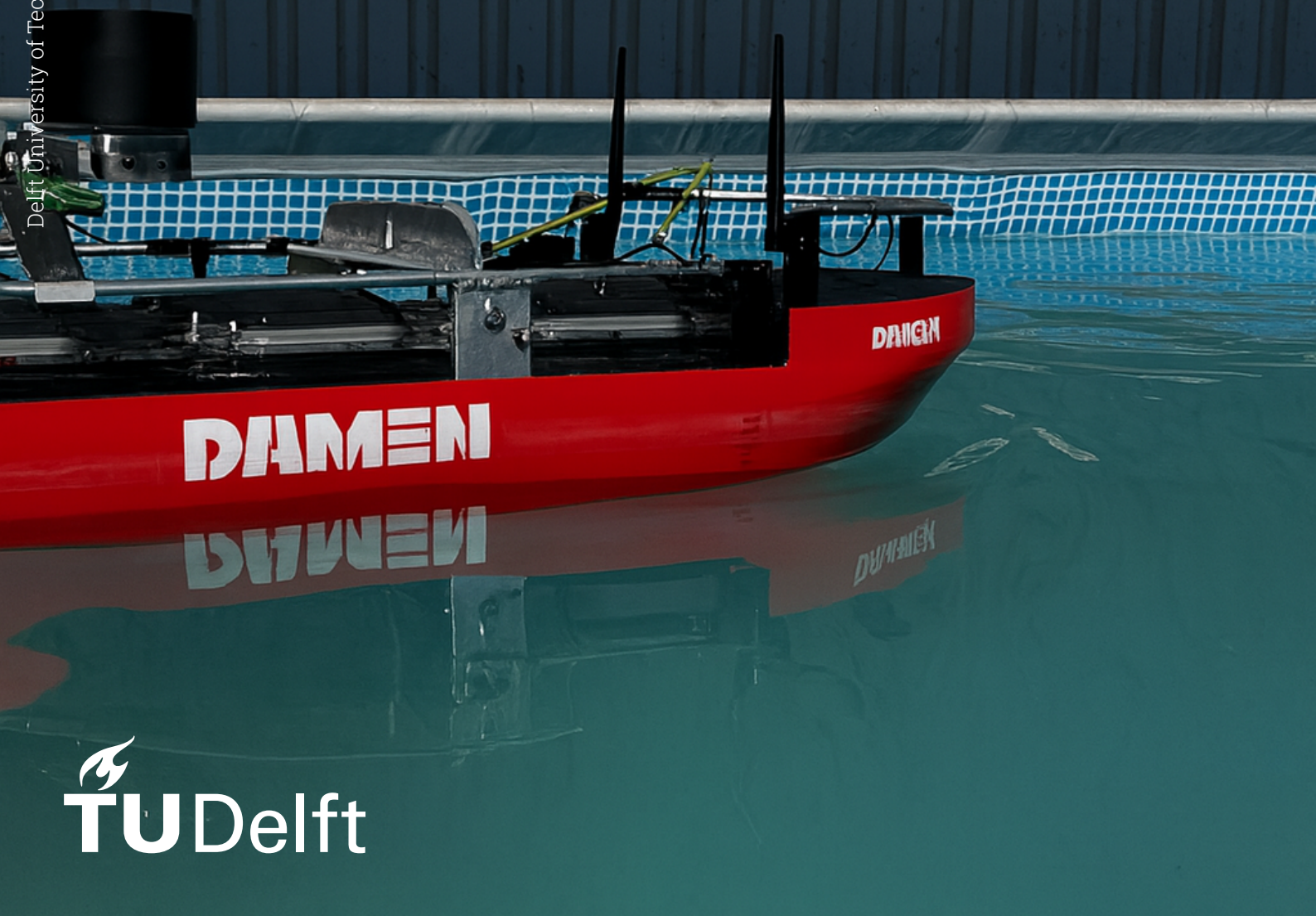
Dock-to-Dock Motion Control for Autonomous Vessels

MSc Thesis: Aerospace Engineering

Ugo Coveliers-Munzi



Delft University of Technology

TU Delft

Dock-to-Dock Motion Control for Autonomous Vessels

MSc Thesis: Aerospace Engineering

by

Ugo Coveliers-Munzi

Student number: 5007631

Thesis committee:

Ir. O. Stroosma	TUDelft, Chair
Dr. E. van Kampen	TUDelft, Supervisor
Dr. K. van der El	Damen, Supervisor
Dr. X. Wang	TUDelft, Examiner

Faculty: Faculty of Aerospace Engineering, Delft

TU Delft

Preface

This master thesis marks the culmination of a seven-month research project carried out at Damen, in conjunction with TU Delft, between January and September 2025. As I near the end of my university studies, there is not much to say about myself, so I would like to take this opportunity to express my gratitude to those who made this project possible and guided it to completion.

I would like to thank my supervisor at Damen, Dr. Kasper van der El, for persuading me to stay on after my internship and embark on what has been an intense yet rewarding journey. Having provided much of the foundational work on which this thesis builds, his expertise and enthusiasm were invaluable throughout the project.

My sincere thanks also go to my TU Delft supervisor, Dr. Erik-Jan van Kampen, whose dedication to both the subject of Reinforcement Learning and to his students was truly inspiring and greatly enriched this work.

Lastly, I would like to thank both TU Delft and Damen for providing the facilities, resources, and technical support that made this project possible.

Ugo Coveliers-Munzi
Delft, September 2025

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
ANN	Artificial Neural Network
ASV	Autonomous Surface Vessel
BC	Behavioural Cloning
BIS normalization	Velocity-free coefficient normalization used for low-speed hydrodynamics
CBF	Control Barrier Function
CG / CoG	Center of Gravity
CI	Confidence Interval
DAVE	Damen Autonomous Vessel
DDPG	Deep Deterministic Policy Gradient
DP	Dynamic Positioning
DOF	Degrees of Freedom
DRL	Deep Reinforcement Learning
EWMA	Exponentially Weighted Moving Average
FB	Feedback
FF	Feedforward
G&C	Guidance and Control
INS	Inertial Navigation System
IRL	Inverse Reinforcement Learning
ISA	International Standard Atmosphere
LOS	Line-of-Sight guidance
LP	Low-pass filter
MC	Monte Carlo
MDP	Markov Decision Process
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral (Control)
NED	North-East-Down reference frame
PID	Proportional-Integral-Derivative Controller
RL	Reinforcement Learning
SAC	Soft Actor-Critic (RL method)
SB3	Stable-Baselines3
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic Policy Gradient
USV	Unmanned Surface Vehicle
WP	Waypoint

Symbols

Symbol	Definition	Unit
A_{front}, A_{lat}	Frontal/lateral area above waterline	m^2
b_w	Wind sensor bias	m/s
$\mathbf{C}, \mathbf{C_{RB}}, \mathbf{C_A}$	Coriolis–centripetal matrix (rigid-body, added)	–
C_X, C_Y, C_N	Wind force coefficients (surge, sway, yaw)	–
$\mathbf{d}(\nu)$	Nonlinear hydrodynamic drag vector	N, N, N·m
Δd	Change distance error to goal	[m]
$\mathbf{D}, \mathbf{D}_L, \mathbf{D}_{NL}$	Damping matrix (linear, nonlinear)	–
D_x	Steady-state surge drag	N
e_X	X variable error	–
F_x, F_y	Surge and sway forces	N
$F_{x,\max}, F_{y,\max}$	Maximum feasible surge/sway force by propulsion	N
H_{target}	SAC target entropy	–
I_z	Yaw moment of inertia about z	kg·m^2
K	K-hold split count for RL success reward	–
K_p, K_i, K_d	PID controller gains	–
L	Vessel length	m
l_x, l_y	Thruster lever arms from CG	m
M_z	Yaw moment	N·m
$M_{z,\max}$	Maximum feasible yaw moment by propulsion	N·m
$\mathbf{M}$, $\mathbf{M}_{RB}$, $\mathbf{M}_A$	Mass matrix (rigid-body, added)	kg, kg·m^2
m	Vessel mass	kg
n	Propeller speed	rpm
$\dot{n}_{max}$	Max propeller speed slew rate	rpm/s
n_{FF}	Feedforward rpm for speed controller	rpm
$n_{\max}$	Maximum propeller speed	rpm
$N_{\text{thrusters}}$	Number of thrusters	–
$p_{x0} \dots p_{x8}$	Polynomial coefficients (surge drag)	–
$p_{y0} \dots p_{y5}$	Polynomial coefficients (sway drag)	–
$\mathbf{p}_x, \mathbf{p}_y$	Drag polynomial vectors	–
Q	State-action value function	–
R	Total RL reward	–
$R_b^n(\psi)$	Body-to-NED rotation matrix	–
T	Thruster output force magnitude	N
T_{nn}	Quadratic thrust coefficient	N/rpm^2
t_{dock}	Time to dock	s
u, v, r	Surge, sway, yaw rate body velocities	m/s
$v_{\text{transition}}$	Transition speed between controllers	m/s
V_w	Absolute Wind speed magnitude	m/s
V_{rw}	Relative wind speed magnitude	m/s
X_{N}	NED North position	m
Y_{E}	NED East position	m
x_g	Longitudinal CG location	m
α	Azimuth thruster angle	rad
$\alpha_{\max}$	Max azimuth angle	rad
$\dot{\alpha}_{\max}$	Max azimuth slew rate	rad/s
β_w	Wind angle/direction	rad
$\boldsymbol{\eta}_{\text{des}}$	Desired vessel pose	m, m, rad
$\boldsymbol{\eta} = [X_N, Y_E, \psi]^\top$	Vessel pose in NED frame	m, m, rad
γ	RL discount factor	–
$\nu = [u, v, r]^\top$	Body-fixed velocities (surge, sway, yaw rate)	m/s, m/s, rad/s
ω	Low-pass filter natural frequency	rad/s

Symbol	Definition	Unit
$\pi(a\|s)$	RL policy (probability of action given state)	–
ψ	Vessel heading	rad
ρ	Air density	kg/m^3
σ_w	Wind sensor noise standard deviation	m/s
σ_{gust}	Wind gust intensity	m/s
τ	SAC smoothing coefficient (target update)	–
$\boldsymbol{\tau} = [F_x, F_y, M_z]^\top$	Control force/moment vector	N, N, N·m
ζ	Low-pass filter damping ratio	–

1

Introduction

1.1. Motivation

In recent years, the maritime industry has experienced a significant shift toward autonomous shipping to enhance safety, improve efficiency and address a growing labour shortage [1], [2]. This shift has been supported by rapid advancements in computing power, artificial intelligence and sensor technology, enabling the development of perception, control and decision making systems required for autonomous vessel operations [3].

Research on motion control for marine vessels has significantly evolved since Minorsky's early work on automatic steering in 1922 [4], leading to widely adopted control systems such as Proportional-Integral-Derivative (PID) based heading autopilots, Dynamic Positioning (DP) systems for position keeping tasks and trajectory tracking controllers [5]. However, a lot of the research in this field focuses on two distinct tasks:

1. Position keeping tasks such as Dynamic Positioning, designed to keep a vessel at a fixed position and heading while being subjected to environmental disturbances.
2. Trajectory following tasks, relying on trajectory tracking controllers to navigate a vessel between static waypoints while adhering to a desired velocity profile.

While these methods have been widely studied and implemented in industry, fully autonomous dock-to-dock maneuvering remains less studied. Dock-to-dock maneuvering tasks are often more complex requiring automatic generation of the high level trajectory going from the starting to the final dock and a smooth transition between the high speed transit phase and the low speed docking phase as shown in Figure 1.1. The docking phase itself comes with many challenges such as "managing large sideslip angles, static and dynamic obstacles and navigation in complex port geometries" [6]. Well established solutions to the position keeping and trajectory following tasks could be modified, and combined to solve the dock-to-dock problem. However, two key questions remain; how and when to perform the transition from the transit to the docking phase as well as investigating what the optimal docking trajectory is in geometrically constrained environments subject to environmental disturbances.

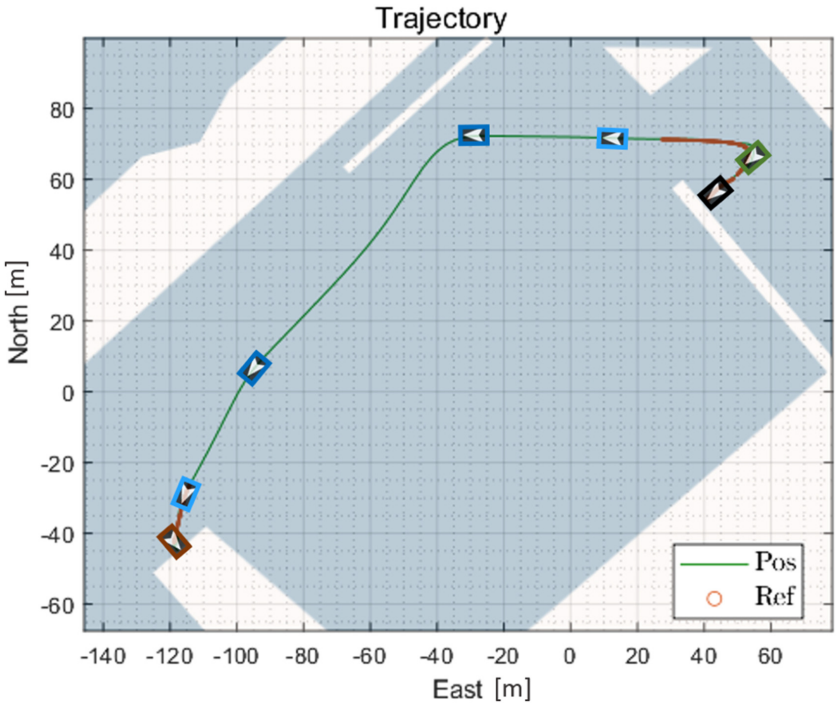


Figure 1.1: A dock-to-dock trajectory showing undocking (top right) and docking (bottom left) phase in brown and the transitions (light blue) to/from the sailing phase (dark blue) [7].

These questions drive the need for more advanced systems capable of higher level, integrated decision making. Advanced model-based motion control systems such as Model Predictive Control (MPC) could be leveraged due to their unique capability of performing both the trajectory generation and the control tasks in parallel. However, such methods come with their own challenges such as a complex design procedure, requiring precise dynamical models, and being computationally expensive to run in real time [8]. Additionally, such algorithms provide limited possibilities for exploration and optimisation of high-level decisions, such as determining when and how to perform the transition from high speed transit to docking phase. Many of the model-based challenges stem from requiring explicit mathematical formulations of the problem to be solved. Data driven strategies such as Reinforcement Learning (RL) could be leveraged to address these shortcomings, providing a way to explore, through interaction with an environment, different approaches that can be taken for the transition and docking phases without requiring any explicit knowledge of the problem. Additionally, such methods allow for the design of systems which can be robust to modeling errors and generalisable to many different scenarios. This could result in a more flexible system that removes the need to explicitly state all possible docking configurations that will be encountered during a vessel's operation.

A significant research gap remains in the development of motion control systems capable of a fully autonomous dock-to-dock mission by integrating trajectory generation, controller transitions and adaptability to environmental disturbances. This is addressed by developing a Guidance & Control (G&C) system for a dock-to-dock mission of an Autonomous Surface Vessel (ASV). Once a benchmark system based on traditional solutions is established, advanced methods, particularly RL, shall be employed to improve adaptability and performance in geometrically constrained environments subject to environmental disturbances providing a comparative analysis of different docking control systems.

Building on the motivation outlined in this section, this chapter continues with presenting the main research questions, how they will be tackled and ends with a project plan. Chapter 2 contains the stand-alone scientific article followed by the literature study in chapter 3. Chapter 4 proceeds with some supporting results providing deeper insights into the modeling, controller tuning and possible RL extensions. Finally, chapter 5 concludes this report followed by some recommended directions for future work in chapter 6.

1.2. Research Questions

Based on the research aim, a list of research questions are formulated to help break the problem down into smaller sub-problems as well as help guide the literature study.

1 What is the state-of-the-art in vessel motion control?

2 What requirements must be met for a dock-to-dock mission to be classified as successful?

- 2.1 What are the safety related requirements?
- 2.2 What are the performance related requirements?
- 2.3 Based on these requirements, what metrics should be used for the comparison of a benchmark G&C system with a system leveraging more advanced methodologies?

3 What are the key components needed for the design of a benchmark, PID-based G&C system for a dock-to-dock maneuver?

- 3.1 What mid level guidance algorithms and low level controllers should be used in both the docking and sailing phases?
- 3.2 How to transition between the two phases?
- 3.3 How to incorporate robustness to environmental disturbances?

4 How can advanced G&C methods be leveraged in the hierarchical control structure to improve upon the performance of the benchmark?

- 4.1 What is the optimal transition from the sailing phase to the docking phase?
- 4.2 What is the optimal trajectory during a docking maneuver?
- 4.3 What advanced G&C algorithms are most effective for high level decision making optimisation while being robust to environmental disturbances and generalisable to different mission scenarios?
- 4.4 How can safety and interpretability be ensured for the selected advanced algorithm?

5 Is a simplified simulation model for the Damen Autonomous Vessel adequate to allow for real-life transferability?

1.3. Methodologies

This section will give a brief overview of how each research question will be approached and tackled. Starting with Question 1, the state-of-the-art for vessel maneuvering and advanced algorithms such as model based and data driven approaches shall be investigated and presented in the literature study.

Question 2 defines the dock-to-dock mission through safety and performance requirements derived from literature, with distinct criteria for the transit and docking phases. It also establishes evaluation metrics to compare the benchmark and advanced G&C systems, grouped into three categories: tracking performance, energy efficiency, and safety (mission success).

Question 3 proceeds with the design of the benchmark G&C system by defining the guidance and control algorithms for the two mission phases. The driving idea behind these choices is to keep the design as simple as possible while still adhering to the mission requirements. Implementation will involve tuning the two controllers under different scenarios, adding a thrust allocation module to convert DP controller forces into actuator commands, and introducing a transition algorithm to smoothly switch between high-speed sailing and low-speed docking.

Question 4 seeks to improve upon the benchmark G&C system by addressing its main limitations, expected to be the discrete phase switch and static docking trajectory, which reduce adaptability to varying environments and geometries. These issues should be analysed using the metrics from Question 2, after which advanced algorithms are introduced; first with a simplified proof of concept, and then extended to tasks such as phase switching and disturbance rejection. Finally, the question touches upon the ’black box’ phenomenon for many advanced algorithms such as RL, where safety and interpretability can never be fully guaranteed. Methods allowing for more safe and interpretable RL should thus be investigated throughout the literature study.

Question 5 touches upon the simulation-to-real gap. The simulation model used for the design process should be accurate enough to allow for the same design to be run in real-life on the Damen Autonomous Vessel (DAVE). To ensure this, the simulation model used is for a similar vessel to DAVE, however, the transferability of the simulation model to the Damen Autonomous Vessel shall be quantitatively analysed with a real-life validation of the simulation results.

1.4. Planning

This thesis began with a ”Literature review & Research definition” phase where, over 6 weeks, relevant information on the current state of the art was collected. In addition, the scope of the thesis was refined by formulating and refinement of research questions. Once completed, the research part of the thesis began divided into two research phases split by a midterm review. Phase 1 began with generating the model required for performing simulations followed by an implementation of a benchmark, industry-standard, G&C system for an autonomous dock-to-dock mission satisfying the mission requirements. This was followed by an analysis of problematic areas in the benchmark system followed by a proof of concept that RL could indeed be used to solve such problems.

Following the midterm review, phase 2 was then used to perform a detailed implementation of advanced methodologies focused on the problematic areas. This was followed by a comparison of the benchmark vs advanced system through the use of metrics. Finally, a real-life validation was performed using the facilities provided at Damen. All related planning figures (Gantt Charts and Work Breakdown Structures) were presented in Appendix A.

2

Scientific Article

Dock-to-Dock Motion Control for Autonomous Vessels

Ugo Coveliers-Munzi

***Abstract*—This study evaluates control strategies for autonomous dock-to-dock sailing on inland waterways. A benchmark system using industry-standard PID controllers is compared to an all-in-one Reinforcement Learning (RL) controller and a third, hybrid system is proposed trading off the improved performance of the RL controller with the inherent stability guarantees of the benchmark system. Simulation results show all three controllers can successfully perform the mission. The RL controller docks significantly faster while rejecting higher lateral wind forces but struggles to generalise to unseen docking scenarios, while the hybrid system improves interpretability at the cost of performance. Furthermore, initial real-life testing of the benchmark system validates the simulation results.**

***Index Terms*—Autonomous Dock-to-Dock Maneuvering, Guidance and Control, Reinforcement Learning, Hybrid RL-PID Control Systems, Dynamic Positioning, Controller Regime Transition**

I. Introduction

In recent years, the maritime industry has experienced a significant shift toward autonomous shipping to enhance safety, improve efficiency and address a growing labour shortage [1], [2]. This shift has been supported by rapid advancements in computing power, artificial intelligence and sensor technology, enabling the development of perception, control and decision making systems required for autonomous vessel operations [3].

Research on motion control for marine vessels has significantly evolved since Minorsky's early work on automatic steering in 1922 [4], leading to widely adopted control systems such as PID-based heading autopilots, Dynamic Positioning (DP) systems for position keeping tasks and trajectory tracking controllers [5]. Most research in this field focuses on position keeping tasks under environmental disturbances or trajectory following tasks [6] often neglecting critical functionalities required for fully autonomous operations. Dock-to-dock maneuvering tasks are often more complex requiring high-level trajectory generation, transitioning between high-speed transit and low-speed docking phases, as well as precise low-speed maneuvering in constrained port geometries while subject to environmental disturbances [6]. Open questions remain regarding optimal switching between control phases and it's integration into high-level planning. Furthermore, research on generation of docking trajectories in complex, dynamically constrained environments remains limited.

This motivates the need for a system that is adaptable to different docking scenarios and capable of real-time decision making under uncertainty. Model-based approaches such as Model Predictive Control (MPC) and Model Predictive Path Integral (MPPI) have been investigated for this purpose, but suffer from high model dependency and, in some cases, high computational costs due to real-time optimisation [7], [8], [9]. Data driven methods, such as RL, mitigate some of these limitations by learning control policies directly from experience through interaction with an environment rather than relying on explicit problem formulations. This makes such methods particularly attractive for learning high-level decision making tasks, such as determining when to transition between mission phases or optimising trajectories for complex harbour geometries under environmental disturbances.

A clear research gap remains in the development of motion control systems capable of a full dock-to-dock mission by integrating trajectory generation, controller transitions and adaptability to environmental disturbances. This is addressed by developing a benchmark Guidance and Control (G&C) system for autonomous dock-to-dock maneuvering and investigating advanced methods, particularly RL, to improve adaptability and performance in geometrically constrained environments subject to environmental disturbances. The main contributions are:

- An industry-standard, benchmark G&C system for dock-to-dock operations including an initial real-life validation.
- An RL-based docking controller for geometrically constrained environments subject to environmental disturbances.
- A hybrid docking controller trading off performance for reliability and interpretability.
- A comparative evaluation of the three controllers.

The remainder of this paper is structured as follows: section II presents a background on motion control for marine vessels and Reinforcement Learning as well as some related work. Section III describes the simulation setup including generation of the dynamical model followed by the design process of the different controllers in section IV. Their simulation-based and real-life validation results are discussed in section V ending with a conclusion in section VI.

II. Background

A. The Dock-to-Dock Mission

A typical dock-to-dock scenario for a ferry operating in inland waterways is described in Figure 1. It comprises different phases as described below:

1) Undocking: the vessel leaves its starting dock and aligns itself with the trajectory.
2) Transit: The vessel begins sailing at constant speed.
3) Avoidance: The vessel avoids a static obstacle.

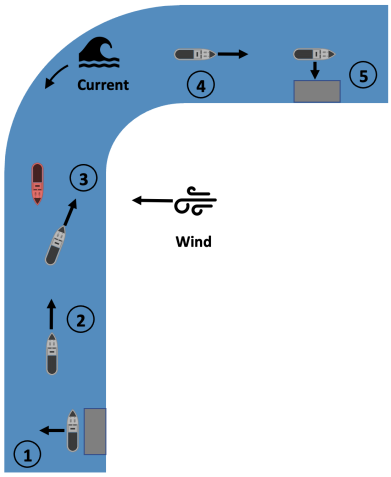


Fig. 1. A typical dock-to-dock scenario for a ferry operating in inland waters.

4) Approach: The vessel begins slows down to smoothly transition to the final approach phase.
5) Final Approach: The vessel begins docking where it aligns its position and heading with the end dock at low velocity.

This study implements only wind disturbances simplifying the design process while still showing the disturbance rejection capabilities for slowly varying disturbances (current is a similar quasi steady-state disturbance applied in a different manner). Waves are also omitted as they are considered minimal in inland waterways and are mostly due to ship wakes. It is assumed existing methods such as wave filtering [10] provide the necessary rejection capabilities.

A mission is defined as successful when the vessel stern and the bow (for lateral docking) make and maintain, by pushing 'into' the dock, contact with the dock while being aligned and having a low enough velocity. To simplify the successful condition, it is assumed in this study that docking is successful when all three of the following conditions are met simultaneously:

- Position error: $< 0.3\text{m}$
- Heading error: $< 10°$
- Velocity: $< 0.1\text{m/s}$

B. Marine Motion Control

Motion control is "the ability to accurately maneuver along a given path" [11] and, for marine applications can broadly be split in two tasks [5]:

- Position-keeping: Remain at a desired position and heading while rejecting environmental disturbances.
- Trajectory Following: Follow a desired geometric path and velocity profile.

Position keeping tasks make use of the DP controller as shown in Figure 2 [5]. The DP controller employs three independent PID controllers along the three axes (surge, sway and yaw) computing a desired force in each direction to remain at a desired position and heading while subject to environmental disturbances. A thrust allocation module then maps the desired force to actuator commands by distributing the required control effort across the vessel's actuators. Thrust allocation for vessel is a well researched constrained allocation problem ([12], [13], [14]) and is considered to be outside the scope of this study.

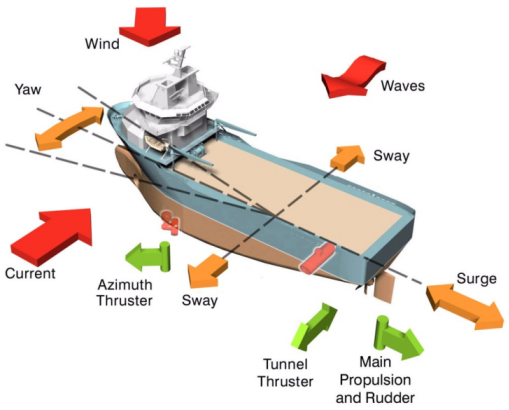


Fig. 2. The DP problem where the DP controller maintains the vessel within the desired degrees of freedom (orange arrows) while counteracting environmental disturbances (red arrows) through the use of its actuators (green arrows) [5].

Trajectory following refers to the ability of a vessel to follow a predefined geometric path and velocity profile. Essential for autonomous navigation, this task is achieved by a combination of mid-level guidance and low level control algorithms (G&C). The guidance module generates reference commands for the vessel to ensure it remains/converges back to the desired trajectory while the low level controller is responsible for ensuring the actuators output the desired commands. Line-of-Sight (LOS) is an example of such a guidance algorithm combining simplicity, predictive capabilities and convergence guarantees. Purely geometric, LOS computes a required heading for the vessel to converge to the trajectory at some point in the future called the LOS point as shown in Figure 3.

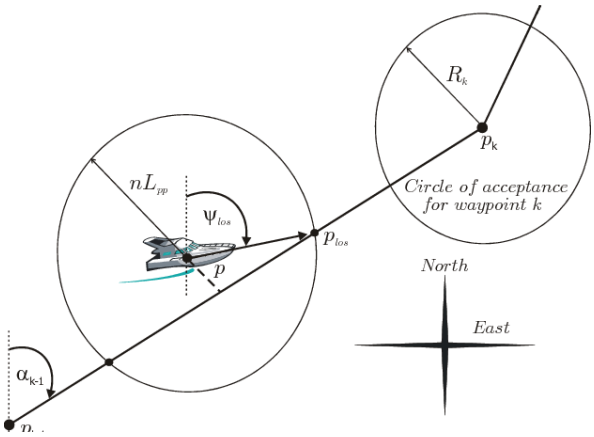


Fig. 3. The Line-of-Sight principle [15]

The controller required to execute the guidance commands can take many forms depending on the operating speed. At low

speeds, a DP controller can be used however, at higher speeds, it is common to make use of independent speed and heading PID controllers to avoid highly inefficient sway motion. Such a setup also allows for bypassing the thrust allocation module as the simpler task allows for directly relating speed/heading errors to thruster rpm and angle.

Robustness to environmental disturbances such as current, waves and wind is another essential ingredient for a motion control system. Although the integral term in the PID controllers can compensate for steady state errors caused by e.g. a constant wind force, they lack the ability to quickly adapt to dynamic conditions due to the slow nature of the integral term buildup. Hence, such disturbance forces are typically estimated through the use of nonlinear disturbance observers [16] and are applied in a feedforward manner. Additional measures, such as wave filters, have also been proposed to prevent high-frequency noise from destabilizing the control loop [10].

Assuming a pre-generated trajectory, several papers have combined the above motion control solutions into a G&C system for autonomous docking. [17] extends a PID-based DP controller to low speed tracking by employing a low pass filter guidance module, enabling accurate alignment with the final dock but neglects the transition from a previous sailing phase. Building on this, [18] addresses the full mission smoothly transitioning between docking and sailing controllers via integral-state matching (equal controller outputs at switch) although when to switch remains under-investigated. To avoid controller switching all together, [19] proposes a unified, model-based controller using a shaping function to limit inefficient sway motion at higher speeds but its feedforward and model-based nature tightly couple it to the simulation model used.

C. Model-Based Approaches

Model-based approaches explicitly incorporate vessel dynamics and environmental constraints into the control loop allowing for predictive planning to generate desired docking trajectories. Model Predictive Control (MPC) has been applied to the docking problem in [7], while hybrid MPC-PID methods have been shown to reduce MPC's model-based, feedforward nature by using it only to generate trajectories [20]. To address MPC's heavy computational expense due to performing real-time optimisation at each timestep, Model Predictive Path Integral (MPPI) replaces it with a gaussian sampling-based control selection [21] and has been applied to the docking problem in [8]. Many variations such as nonlinear MPC and iterative learning MPC exist [6] however, the fundamental model-based nature remains an issue for precise control in real-life operations. Additionally, such approaches generally require extensive weight tuning and cost function design for different scenarios.

D. Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning concerned with learning sequential decision-making policies through interaction with an environment. At each timestep, an RL agent observes the current state of the environment, selects an action, and receives a scalar reward based on the outcome. The goal of the agent is to learn a policy that maximizes the expected cumulative reward over time [22]. Unlike model-based approaches, RL avoids explicit problem formulations by directly interacting with an environment and learning through experience. This makes it well suited for complex tasks requiring the exploration of new solutions such as autonomous docking in constrained environments. By optimizing behavior over long horizons and under uncertainty, RL offers a promising framework for handling model uncertainties, adaptive learning, generalisability to unseen problems and exploration of unknown solutions [23].

RL methods can be divided into value-based and policy-based approaches. Value-based methods iteratively estimate a value function approximating the cumulative returns for state-action pairs, from which a policy is extracted using a greedy or ϵ-greedy algorithms. However, such methods are limited to discrete action spaces. Policy-based methods directly approximate the policy, enabling continuous and stochastic actions, though often suffer from high variance and training instability due to lack of 'guidance' by a value function during training [22].

Actor-Critic algorithms combine value- and policy-based methods allowing for continuous and stochastic action spaces while leveraging the higher stability of value based methods. This is achieved by introducing two distinct components: an actor which learns a parametrized policy mapping states to actions and a critic estimating the value (or action-value) function to provide feedback to the actor. Soft Actor-Critic (SAC) builds on other actor-critic methods such as Deep Deterministic Policy Gradients (DDPG) [24] and Twin Delayed Deep Deterministic Policy Gradient (TD3) [25] to provide an algorithm which is [26]:

- Robust to Q value overestimation bias due to introduction of twin Q networks.
- Robust learned policy due to incorporation of stochasticity in actor network.
- Automatically tuned entropy parameter α better balancing exploitation exploration.

RL has been applied to marine motion control with promising results. Deep RL (DRL)-based controllers show improved environmental disturbance rejection in trajectory following tasks [27] and allow for obstacle avoidance using real-time sensor data [28]. Although useful as a controller, RL's real advantage becomes apparent in the higher level planning module where the RL agent must make decisions based on many, dynamically varying factors. Although applied to different applications, both [29] and [30] show RL's ability to plan even in unseen scenarios.

As with all machine learning approaches, a major drawback of RL is its 'black box' nature [31]. [32] addresses safety and interpretability by employing a hybrid MPC-RL approach using the RL algorithm to adapt MPC weights and constraints. Lyapunov-based safe RL [33] tracks the stability of a system during training while [34] proposes Control Barrier Functions (CBFs) to ensure hard constraint handling through set invariance although these methods remain limited in handling transient performance (focused on steady-state stability guarantees).

III. Simulation Setup

This section describes the simulation setup where it's main components are shown in Figure 4. It begins with providing the theoretical background required for the maneuvering (vessel), thruster and wind models and ends with an overview of DAVE and how it's hydrodynamic and thruster coefficients were identified. The controller block in Figure 4 shall be discussed in section IV.

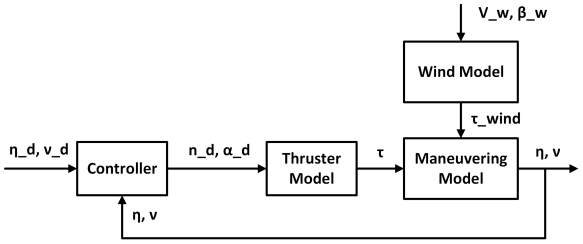


Fig. 4. A high level overview of the different simulation components.

A. Modeling

Reference Frames

The simulation model makes use of the Earth-tangent North-East-Down (NED) and the body-fixed frame as shown in Figure 5 where the NED frame assumes a flat, non-rotating Earth. Going from the body (b) to the NED (n) frame is based on the heading ψ:

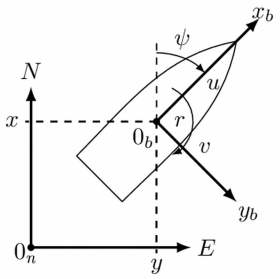


Fig. 5. North-East-Down (NED) vs. Body frame.

$$\dot{\boldsymbol{\eta}} = \mathbf{R}_b^n(\psi)\,\boldsymbol{\nu}_{\mathrm{b}}, \tag{1}$$

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

Where $\boldsymbol{\eta}$ is the NED state $[X_{north}, Y_{east}, \psi]$, commonly referred to as the 'pose' and $\boldsymbol{\nu}_{\mathrm{b}}$ is the velocity vector containing surge, sway and yaw rate velocities $[u, v, r]$.

Maneuvering Model

The maneuvering model is a 3 degree-of-freedom (3DOF) model used in navigation for surface vessels modeling forward (surge), lateral (sway) and rotational (yaw) motion through the use of the three body velocities as states (u, v, r). It's standard form is shown below [12]:

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{3}$$

Where the mass matrix, $\mathbf{M} = \mathbf{M_{RB}} + \mathbf{M_A}$, contains the rigid body dynamics and the added mass matrix describing the additional inertia due to movement of the surrounding water. The centripetal/coriolis matrix, $\mathbf{C} = \mathbf{C_{RB}} + \mathbf{C_A}$, captures the forces due to translational/rotational motions over the Earths surface and the damping matrix, $\mathbf{D} = \mathbf{D_L} + \mathbf{D_{NL}}$, contains the linear and nonlinear (high speed) damping terms. τ represents all the forces applied to the vessel including a control force and a disturbance force (wind and waves) while current is represented as a relative velocity in the model.

Thruster Model

The thruster dynamics model the thrust T to propeller speed n (in rpm) with a quadratic relation using the thrust coefficient T_{nn}:

$$T = T_{nn} n^2 \tag{4}$$

Where the thrust is further decomposed using the thruster azimuth angle α and the c.o.g. offsets l_x and l_y:

$$F_x = T\cos\alpha \tag{5}$$

$$F_y = T\sin\alpha \tag{6}$$

$$M_z = l_x F_y - l_y F_x \tag{7}$$

Additionally, actuator dynamics are modeled as simple linear rate limiters.

Wind Model

A wind model relating relative wind speed V_{rw} and angle β_w to force applied on the vessel is defined as follows [12]:

$$\boldsymbol{\tau}_{\text{wind}} = \frac{1}{2}\rho V_{\text{rw}}^2 \begin{bmatrix} C_X(\beta_w) A_{\text{front}} \\ C_Y(\beta_w) A_{\text{lat}} \\ C_N(\beta_w) A_{\text{lat}} L \end{bmatrix} \tag{8}$$

Where the wind coefficients $C_X(\beta_w), C_Y(\beta_w), C_N(\beta_w)$ are computed based on the Helmholtz-Kirchhoff plate theory (using the 'research vessel' parameters Table 8.3 in [12]), ρ is the density of air, A_{front} and A_{lat} are the frontal and lateral surface areas above the waterline and L is the length of the vessel.

Wind gusts are modeled with a Ornstein-Uhlenbeck Process where fluctuations in the x and y wind speed components are modeled as [35]:

$$V_{k+1} = e^{-\frac{\Delta t}{T_{\text{gust}}}} V_{x,k} + \sigma_{\text{gust}} \sqrt{1 - e^{-2\frac{\Delta t}{T_{\text{gust}}}}} \cdot \mathcal{N}(0,1) \tag{9}$$

The resulting V_x and V_y wind speeds are then converted to a wind speed magnitude and angle to be used in the wind model above.

Finally, to use the wind model in the controller design as a feedforward action, a wind sensor and observer must be implemented to account for measurement noise and avoid an ideal case where the feedforward control actions perfectly cancel out the wind experienced. As this is outside the scope of this research, the sensor was simply modeled by adding noise and a bias ($\sigma = 0.15, b = 0.05[m/s]$) based on typical maritime wind sensors [36]. The observer was a simple Exponentially Weighted Moving Average (EWMA) filter which filters out the high frequencies due to wind gusts and sensor noise. A block diagram of the wind model used in simulation is shown in Figure 6.

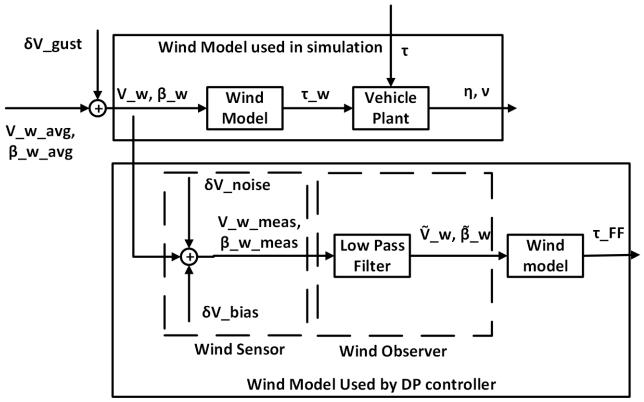


Fig. 6. Block diagram showing how the wind model is used in simulation and for the controllers.

B. The Damen Autonomous Vessel

The Damen Autonomous Vessel (DAVE) shall be used to perform final real-life validation in conjunction with a 10 by 5 meter testing pool. It is a 1:25 scale model vessel of a tug owned by Damen containing two azimuth thrusters at the stern as shown in Figure 7. DAVE is equipped will all the necessary hardware for logging and sending data commands to the vessel remotely. Additionally, it is equipped with an Inertial Navigation System (INS), developed at the Damen facilities, capable of estimating the full vessel state vector $[\eta, \nu, \omega]$ where η is the pose, ν are the linear velocities and ω are the angular velocities.

Fig. 7. An overview of the Damen Autonomous Vessel (DAVE).

DAVE's maneuvering and thruster models were identified in [37] where the thruster model was found to be accurate (and was hence used) but the maneuvering model omits key sway-yaw coupling dynamics. Hence, a validated maneuvering model developed for a similar-scale vessel, TitoNeri [38], is adopted and scaled to DAVE's dimensions using Froude similarity. To address the low speed issues of conventional velocity-based normalisation, the BIS normalisation method was applied (Table 7.2 in [12]) which is based on physical parameters such as length and mass. The final parameters for DAVE are listed in Appendix A, Table VII.

IV. Controller Design

This section concerns itself with the design of the controller block shown in Figure 4. It begins with the design of a benchmark controller for the dock-to-dock problem followed by the implementation of an all-in-one RL-based docking controller and ending in a hybrid system where the RL controller's increased performance is traded off with the inherent stability guarantees of the benchmark system.

A. Benchmark System

The benchmark system consists of a high speed PID-based sailing controller and a low speed PID-based docking controller as well as their respective guidance modules. Focusing on the G&C aspect of the mission, a dock-to-dock trajectory must be manually generated based on Figure 1 using Dubins method to impose curvature constraints ensuring feasibility [39]. The desired dock-to-dock trajectory is shown in Figure 16.

Sailing Controller

The sailing controller, used for the high-speed transit phase, consists of speed and heading PID controllers in combination with LOS guidance as shown in Figure 8.

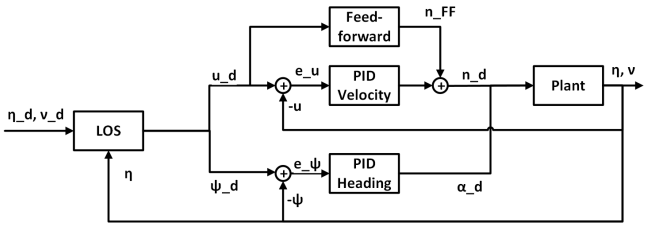


Fig. 8. Block diagram for complete sailing controller including guidance module.

Both controllers directly output thruster commands bypassing the need for a thrust allocation module. Additionally, the speed controller makes use of a feedforward term which accounts for the nonlinear steady-state damping at higher speeds (see Equation 12) avoiding the need for gain scheduling.

$$\alpha = K_p \cdot e_\psi(t) + K_i \cdot \int_0^t e_\psi(t)dt - K_d \cdot \dot{e_\psi}(t) \tag{10}$$

$$n = n_{FF} + K_p \cdot e_u(t) + K_i \cdot \int_0^t e_u(t)dt - K_d \cdot \dot{e_u}(t) \tag{11}$$

$$n_{FF} = \sqrt{\frac{D_x}{T_{nn} \cdot n_{thrusters}}} \tag{12}$$

Where n and α are the thruster rpm and angle, $[K_p, K_i, K_d]$ are the PID gains, $e(t)$ is the heading/surge speed error in the body frame, D_x is the steady state drag at the desired surge speed, u_d, and T_{nn} the thruster coefficient relating thrust to rpm.

To avoid the integral term building up to excessively large values when large setpoint changes occur (integral windup), a simple anti-windup scheme is employed where the integral term is only incremented if the actuators are not saturated [40].

Docking Controller

Shifting to the docking phase, a DP controller must be designed, and slightly modified, to allow for low speed pose tracking rather than position-keeping. To avoid over saturated DP controller outputs due to large position errors, a third-order low pass filter is employed (see Equation 13) to ensure

the desired pose smoothly moves between the current vessel and desired final pose preventing large, discrete jumps [17].

$$\frac{n_{d_i}}{r_i^n(s)} = \frac{\omega_{n_i}^3}{s^3 + (2\zeta+1)\omega_{n_i}s^2 + (2\zeta+1)\omega_{n_i}^2 s + \omega_{n_i}^3} \quad (13)$$

Where n_{d_i} is the output from the low pass filter (the dynamic waypoint), r_i is the desired final waypoint (static waypoint), ω the frequency of the low pass filter and ζ the damping ratio.

The DP controller often makes use of a wind feedforward term, as shown in Figure 9 and Equation 14, due to it's ease of measurement in real-time. This reduces the reliance on the slowly varying integral term for disturbance rejection.

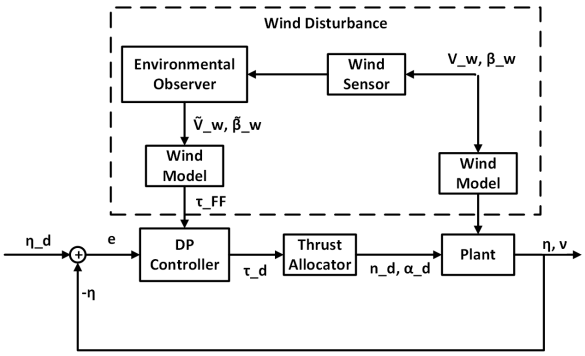


Fig. 9. Block Diagram for the Dynamic Positioning control loop.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{FF,wind}} + \mathbf{K}_p\, \boldsymbol{e}(t) + \mathbf{K}_i \int_0^t \boldsymbol{e}(\tau)\, d\tau + \mathbf{K}_d\, \dot{\boldsymbol{e}}(t) \quad (14)$$

Where $[\mathbf{K}_p, \mathbf{K}_i, \mathbf{K}_d]$ are the diagonal gain matrices (each has 3 gains for the three DOFs), $\boldsymbol{\tau}_{\text{FF,wind}}$ is the feedforward wind force and $\boldsymbol{e}(t) = \mathbf{R}_b^n\,(\boldsymbol{\eta}_{\text{des}}(t) - \boldsymbol{\eta}(t))$ is the pose error vector in the body frame,

The output desired force vector, $\boldsymbol{\tau_d} = [F_x, F_y, M_z]$, is subsequently translated into actuator rpm and angle commands in the thrust allocation module. To limit the scope of this study, a simple thrust allocation algorithm is used which performs a steady-state force decomposition to find the best rpm and angle for both thrusters to achieve the desired force (see section 12.3 in [12]).

Controller Tuning

The tuning process used for the PID controllers is based on the Ziegler-Nichols tuning method [41]:

1) Increase K_p until system reaches sustained oscillations (ultimate gain).
2) Reduce K_p by about half and introduce K_d for additional damping.
3) Finally, introduce K_i to eliminate the steady state error (if none present, still needed for disturbance rejection).

Throughout the tuning process, different metrics quantifying the closed loop response performance were monitored. These include: rise time, settling time (2%), overshoot and steady-state error. Additionally, the effect of operating speed on the different controllers is analysed and, if required, a feedforward term is included to avoid the need for gain scheduling. Finally, the lookahead distance was maximised to avoid an overly reactive controller while preventing excessive corner cutting and the low pass filter should be as fast as possible without introducing overshoot into the closed loop response. The final values are listed in appendix A.

Controller Transition

A mechanism to smoothly switch between the two controllers is now investigated. Three switching waypoints (WP), as shown in Figure 16, shall be placed along the trajectory with a specified along-track distance starting from the start/end dock (in terms of ship lengths from the dock). Their functionality is as follows:

- Undocking WP: Easiest point to place, must allow for enough clearance to dock when vessel rotates to align with the start of the transit phase.
- Approach WP: Ensure a smooth transition at the final approach wp by slowing down until reaching a velocity of 0.1m/s at the final approach wp.
- Final Approach WP: Vessel switches from the sailing controller back to the docking controller to perform the final docking maneuver. Must be placed far enough from final dock to give vessel time to slow down and align its heading with the dock.

Where as a general rule of thumb, the points should be tuned such that the time spent in the docking phase is minimised to ensure timely and efficient operation. The undocking waypoint is placed at 2 ship lengths from the start dock with it's only requirement being enough clearance to the dock when rotating. Using Figure 10 to place lower limits on the approach and final approach factors, values of 4 and 2 were chosen respectively.

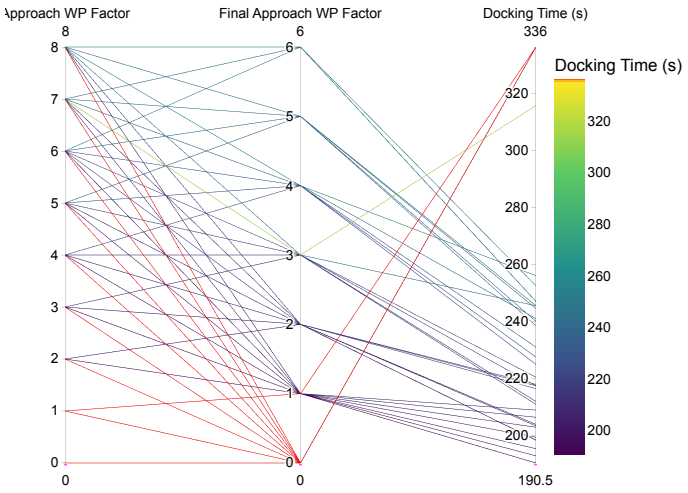


Fig. 10. Sensitivity analysis of approach and final approach factor placement w.r.t mission completion time (red=failure).

B. RL Docking Controller

An all-in-one RL controller, with direct access to the actuator commands, is now designed for the final docking problem providing the following improvements over the benchmark system:

- Built-in docking trajectory planning
- Built-in controller switch mechanism
- Generalisability to different geometric constraints & environmental conditions

Action & Observation Space

At each timestep, the agent receives the following observable states (all normalised to [-1, 1]):

- Errors to the goal pose in the body frame: $[x_e, y_e, \psi_e]$
- Body velocities: $[u, v, r]$
- Relative wind speed (body frame): $[v_{x_{rw}}, v_{y_{rw}}]$
- Lookahead points: 16 points at equidistant angles around vessel returning the distance to an object/safety bounds in that direction [42].
- Dock centroid (x, y): the lookahead points are in theory enough information for the agent but explicitly adding this speeds up training.
- Hold progress: variable indicating what step of the 'K-hold' process agent is currently in (see 'Reward Function' for more details)

Based on the observed state at each timestep, the agent must output 4 actions; propeller speed and thruster azimuth angle for two independent thrusters.

Reward Function

The reward function provides feedback to the agent on the quality of it's actions with respect to the ultimate task to be completed. To ensure the primary task remains dominant [43], the weights for individual components are carefully tuned (see Table I) and a potential-based distance reward is employed [44] providing the agent with directional information w.r.t the dock. Finally, to avoid destabilising the critic with large per step rewards at the success condition, a 'K-hold' technique is proposed where if the condition is reached, the state if frozen over the next K timestep and the agent receives $\frac{R_{success}}{K}$ at each step. To ensure full observability, an additional observation state on which hold step the agent is in is included.

$$R = R_{\text{progress}} + R_{\text{penalty}} + R_{\text{time}} + R_{\text{crash}} + R_{\text{success}} \quad (15)$$

$$R_{\text{progress}} = \begin{cases} w_{\text{p}} \cdot \Delta d, & \text{if } t > 0 \\ 0, & \text{if } t = 0 \end{cases} \quad \text{where} \quad \Delta d = d_{t-1} - d_t \quad (16)$$

$$R_{\text{penalty}} = -\big(w_\psi \cdot e_\psi + w_v \cdot e_v + w_d \cdot d_t\big) \quad (17)$$

$$R_{\text{crash}} = \begin{cases} -5.0, & \text{if outside safety bounds} \\ 0, & \text{otherwise} \end{cases} \quad (18)$$

$$R_{\text{time}} = -10^{-3} \quad (19)$$

$$R_{\text{success}} = \begin{cases} \frac{R_{\text{total succ}}}{K}, & \text{if in success hold phase} \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

Where $d_t = \frac{\|\boldsymbol{\eta}_t - \boldsymbol{\eta}_{\text{dock}}\|}{2 \max_{dist}}$ and $e_\psi = \frac{|\psi_{\text{dock}} - \psi_t|}{\pi}$ are the normalised distance and heading errors to the dock, $e_v = \frac{\|\boldsymbol{\nu}_t\|}{v_{\max}}$ is a normalised velocity penalty, K is the number of transitions over which to divide the success reward and w_{p}, w_ψ, w_{v}, w_{d} are the corresponding weighting factors.

TABLE I
REWARD FUNCTION STRUCTURE.

Reward	Description	Range	Weighting
R_{progress}	Potential-based distance reward encouraging agent to approach dock while providing directional information	[-0.1, 0.1]	$w_{\text{p}} = 20$
R_{penalty}	Reward penalising heading/position error to dock and penalising large velocities.	[-0.05, 0]	$w_\psi = 0.04$, $w_v = 0.01$, $w_d = 0.04$
R_{crash}	Truncation penalty applied when agent crashes	-5 or 0	-
R_{time}	Time penalty	-10^{-3}	-
R_{success}	Reward for reaching the goal condition	0 or 10 (split with k-hold)	K = 10

Environment Setup

To ensure a wide range of scenarios are covered, the environment scenario is generated randomly per episode where, if feasibility is not detected (no path connection), the process is repeated:

1) Randomly generate rectangular safety area within maximum distance bounds (20 meters).
2) Within this rectangle randomly initialise start and end pose at which to dock as well as wind disturbance.
3) Create dock object rectangle, place at $0.3m$ offset from goal pose. Merge safety and dock areas into single polygon with dock object cut out.

Training Approach

The SAC algorithm was implemented from scratch using PyTorch and was compared to the open source OpenAI StableBaselines3 (SB3) SAC package[1]. SB3 was used as, although final convergence was similar, it was much faster. Additionally, due to the complexity of the task, curriculum learning was employed where the task is divided into 3 subtasks to be learned sequentially where at each subsequent phase, the training is bootstrapped (load buffer with samples from previous policy) with the previous phase policy:

1) Phase 1: Learn the ideal policy in no wind and a fix dock/safety area size.
2) Phase 2: Add wind (up to 5m/s) randomisation in environment. Double network capacity (hence also reduce learning rates) as introducing wind increases problem complexity.
3) Phase 3: Add dock and safety area size randomisation (length and width).

The wind speed of 5m/s is based on the DP controller lateral wind rejection capability (3.5m/s) with an additional margin to investigate potential improvements. To avoid introducing too much static bias in the sample distribution, the bootstrapping was performed with 75,000 samples before

[1]https://stable-baselines3.readthedocs.io/en/master/

training. The hyperparameters, selected with the help of the sensitivity analysis performed below, are presented in appendix A, Table VIII.

RL Sensitivity Analysis

A sensitivity analysis on the SAC hyperparameters is only performed on the first training phase due to the time constraints[2]. Four main parameters are varied and their influence on the training reward curve is shown. Note that if instability is detected (exploding q values, large oscillations) the training run is terminated early to avoid unnecessary computational overhead.

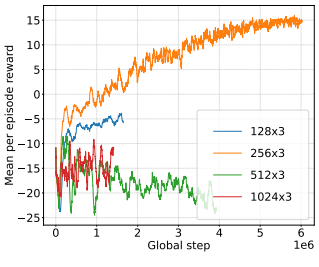


Fig. 11. Different network architectures for the RL controller phase 1 training.

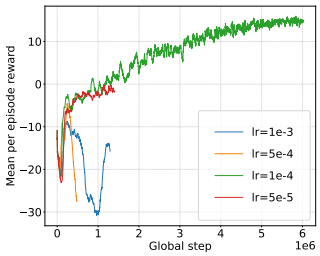


Fig. 12. Different learning rates for the RL controller phase 1 training.

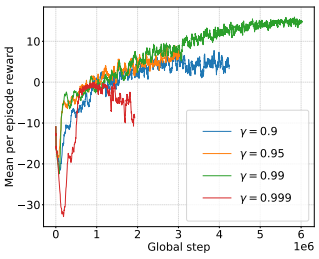


Fig. 13. Different discount factors γ for the RL controller phase 1 training.

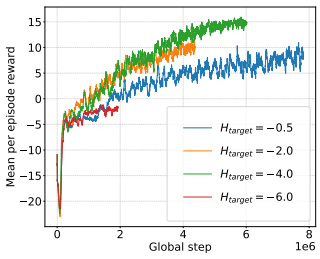


Fig. 14. Different target entropy for the RL controller phase 1 training.

Figure 11 shows that a network size of $[256, 256, 256]$ leads to the best reward where a smaller network results in clear underfitting while larger networks are unstable due to overfitting to training noise. Figure 12 shows that for the chosen network size, a learning rate of $1e-4$ is optimal where as a general rule of thumb, larger networks require smaller learning rates to mitigate their increased expressivity. The default discount factor $\gamma = 0.99$ is shown to work best in Figure 13 as it finds the balance between a large enough horizon without confusing the agent with too much delayed credit assignment. Finally, the target entropy was chosen to be $H_{target} = -4$ $(-action_{dim})$ as it provides the best exploitation (more negative) vs exploration (less negative) balance.

Although limited, this sensitivity analysis provides an initial indication that the values chosen consistently result in the highest performing training curve. This should be further extended to the second and third phase of the curriculum learning process aswell.

[2]Similarly, a sensitivity analysis on the hybrid controller presented later is omitted.

C. Hybrid RL system

To combat the black box nature of the all-in-one RL docking controller, a hybrid system is proposed trading-off the increased performance and functionality of the RL controller with the inherent stability guarantees of the benchmark G&C system. The RL controller is thus re-framed as a planning module which rolls out it's control policy to generate desired docking trajectories to be followed by the benchmark system (see Figure 15) where, if a roll-out violates safety constraints, the process is repeated. This is similar to the hybrid MPC-PID approach employed in [20] with the RL controller replacing the MPC controller.

An algorithm placing the switching points used by the benchmark system along the rolled-out trajectory must now be designed as shown in Figure 23. For this purpose, another SAC agent is trained by taking in the entire rolled out trajectory and placing the approach and final approach switch points shown below.

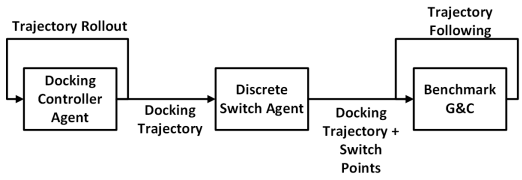


Fig. 15. A high level overview of the proposed hybrid RL approach.

Action & Observation Space

The agent is fed the entire rolled out trajectory (list of $[x_d, y_d, \psi_d, u_d]$ waypoints) where, due to the fixed nature of the observation vector size, it is interpolated to a fix number of points (128 gives enough resolution for max distance of 20m). The agent must output two scalar distances representing the along track distance from the end dock at which to place the two switching points.

Reward Function

The environment must now reward how well the benchmark system can execute the docking mission based on the switch placement resulting in a simple and sparse reward:

$$R = R_{time} + R_{sucess} \tag{21}$$

Where R_{time} (normalized to [-1, 0]) penalizes the time taken to reach the dock and $R_{success}$ simply returns 1 when docking is successful, 0 if not successful (but has not crashed) and -1 if the agent crashes.

The sparsity of the reward (single reward signal per episode) results in highly delayed credit assignment (which actions lead to good reward) limiting it's learning potential. The consequences and potential alternatives are discussed in subsection V-C.

Training Approach

The same SAC algorithm is used for this agent while its parameters have been slightly adapted compared to phase 1 of the RL controller. The network size is increased to [768, 768, 768] accounting for the large observation size (128x5) and the learning rates are decreased to 1e-5 accounting for the increased expressivity of the networks.

V. Results

The three systems shall now be validated and compared beginning with an individual analysis of the benchmark and RL docking controllers, and a comparison of the two systems on the benchmark docking scenario. This is followed by a comparison between the all-in-one RL controller with the hybrid system ending in a real-life validation of the benchmark system.

A. Benchmark System

Selecting an undocking waypoint factor of two, approach of four and final approach of two, the benchmark system can now be run on the fixed docking scenario where it's mission metrics are collected in Table II while the trajectory is plotted in Figure 16.

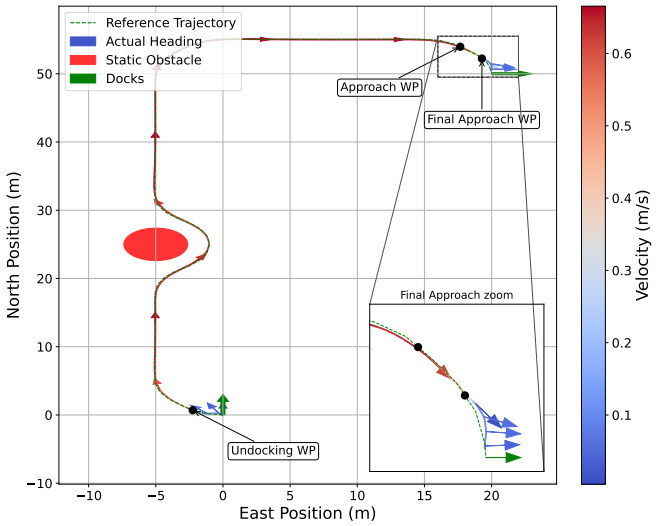


Fig. 16. Full dock-to-dock trajectory for the benchmark G&C system.

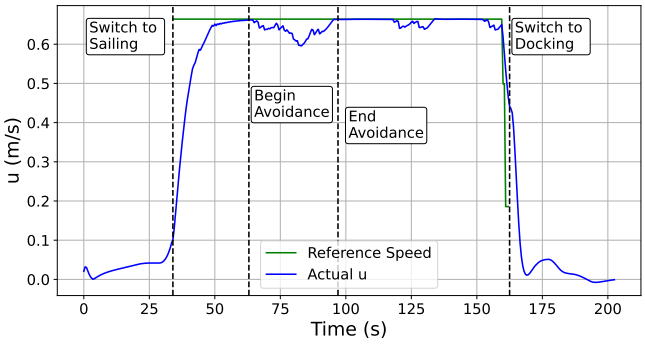


Fig. 17. Surge speed (u) profile related to Figure 16.

The successful docking attempt by the benchmark system (as seen by the final error metrics) provides some key take-aways:

- More conservative approach factors could be used to ensure $u = 0.1\text{m/s}$ at transition (currently 0.44m/s) however, Figure 16 shows an acceptably smooth transition.

TABLE II
Metrics for the ideal docking case.

Metric	Value
Total Time [s]	203.0
Switch to transit [s]	34.0
Switch to final approach [s]	162.5
RMSE Speed Error [m/s] (sailing phase)	0.079
RMSE Cross Track Error [m]	0.09
RMSE Heading Error [deg]	4.09
Final velocity at dock [m/s]	0.04
Final heading error at dock [deg]	2.39
Final position error at dock [m]	0.205

- Sailing controller performance is deemed adequate, with speed and heading RMSE errors mostly due to the tight turn in avoidance phase.
- Relative to distance covered, the docking phase is significantly slower compared to the transit phase.

This further emphasises the importance of the final docking phase and motivates the need to find better, more efficient solutions to the final approach and docking.

B. RL Docking Controller

The per episode training rewards for the different phases of the RL controller are shown in Figure 18. Both phase 1 and phase 2 reach a plateau at around +15 indicating the agent consistently picks up the +10 success reward where phase 2 is faster w.r.t global timestep due to bootstrapping. However, when introducing random dock and safety area sizes (phase 3), there are clear signs of a failure of the agent to learn the task.

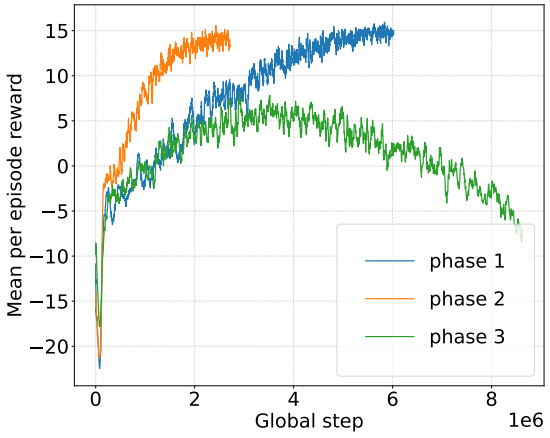


Fig. 18. The per episode reward training curves of the RL controller across the different curriculum phases.

The failure of the agent to learn the randomized geometry problem in phase 3 is further investigated with a MonteCarlo (MC) type validation where the three phases are ran on N randomly generated, unseen scenarios with its results shown in Table III and Figure 19. Equation 22 allows for computing the minimum number of samples required for a confidence interval (CI) of 95% ([45], Sec 4.4).

$$N = \frac{t^2\, p\,(1-p)}{d^2} \tag{22}$$

Where t is the normal quantile for desired confidence (1.96 for CI of 95%), p is the estimated success probability approximated apriori by running a small N=100 MC run ($p_{est} = 0.85$)[3] and d is the desired absolute margin of error ($\pm 4\%$). All MC simulations shall be produced using $\boldsymbol{N = 350}$.

TABLE III
BINARY RESULTS OF MONTECARLO ANALYSIS ON DIFFERENT RL TRAINING PHASES.

Metric	Phase1	Phase2	Phase3
Success %	95.7	**96.3**	47.1
Crash %	3.1	3.7	**32.9**

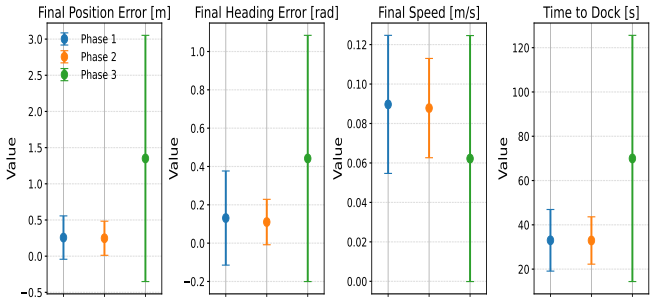


Fig. 19. Resulting metrics of MonteCarlo analysis on different RL training phases. Metrics are final errors at episode termination (success, crash or truncation).

Phase 1 and phase 2 (with wind) have near identical results showing that the agent has learned the wind rejection task. Figure 19 shows phase 2 performing slightly better likely due to the added domain randomization (wind) acting as regularisation allowing for larger networks compared to the no wind policy. This is consistent with Figure 18, where phase 1 reaches a slightly higher training reward plateau (15 vs 14), suggesting mild overfitting.

The drop in performance in phase 3 suggests the agent implicitly learned the constant dock and safety size in phase 1, rather than making use of the lookahead points to allow for generalisability to different geometries. This failure can be attributed to a weak learning gradient in the discrete R_{crash}. The inclusion of a continuous safety reward incentivising the agent to maximise distance to safety bounds [42] was implemented however, the optimal point between enough learning gradient and still allowing for the agent to approach the final dock safety bounds was not found.

Benchmark vs. RL Controller

Focusing on the final docking phase, the RL and benchmark controllers are now compared. The two controllers are first run under ideal conditions as shown in Figure 20 with the success/crash aswell as the time to dock collected in Table IV. The same scenario is then run under various wind speeds and angles in Figure 21 comparing the wind disturbance rejection of the two systems.

[3]The actual p_{est} was 0.95 however, a safety margin is added to ensure a worst case scenario (i.e. more samples required).

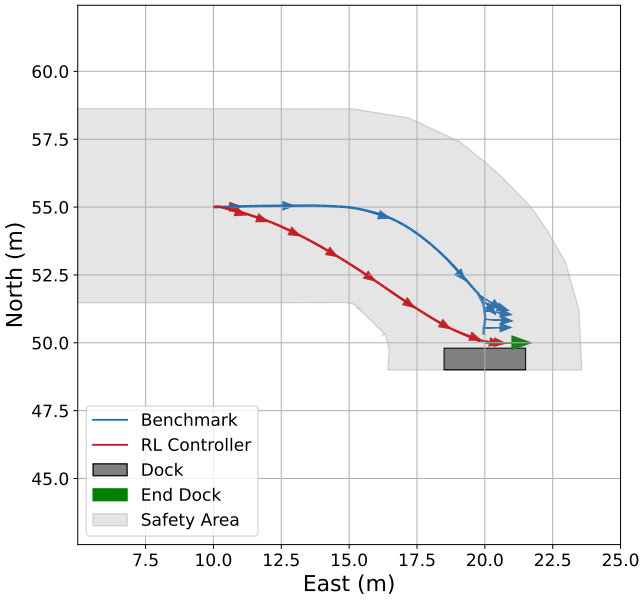


Fig. 20. RL vs benchmark controller on benchmark scenario.

TABLE IV
METRICS: RL VS BENCHMARK ON BENCHMARK SCENARIO.

Metric	Benchmark	RL controller
Success	Yes	Yes
Crash	No	No
t_{dock} [s]	60.5	31.2

Table IV shows the RL controller performs the same docking tasks in half the time of the benchmark. This can be attributed to the fact that the RL controller performs significant corner cutting (see Figure 20) and it minimises the time spent in the docking phase only aligning itself with the dock at the last possible moment avoiding inefficient sway motion.

Figure 21 evaluates the benchmark and RL controller's wind rejection capabilities where the benchmark asymmetry is due to the asymmetric trajectory; starboard turn before switch to docking can cause wind in certain directions to rotate vessel away from dock. Allowing for windspeeds up to $6\mathrm{m/s}$ in the worst case $\beta = 0°$ lateral wind direction while approximately halving the docking times shows its superior rejection capabilities. These gains in performance as well as a much more consistent wind profile (consistently 6 to $7\mathrm{m/s}$ in all directions) can mostly be attributed to the fact that it learns to avoid pure lateral wind force scenarios (higher surface area, lower thruster capabilities) by pointing the vessel bow 'into' the incoming wind.

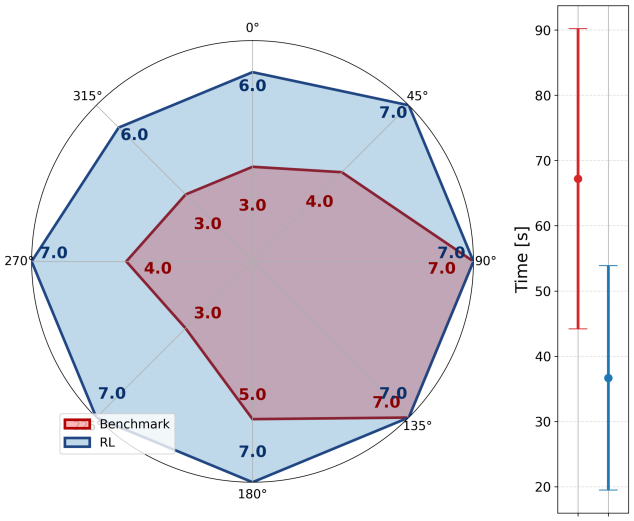


Fig. 21. Wind disturbance rejection capabilities of benchmark vs RL with maximum wind speeds [m/s] annotated in each outgoing direction. Mean and standard deviation of docking times are presented on the right where only successful runs for both controllers are considered.

C. RL Hybrid System

The all-in-one RL controller is now used to roll out a desired trajectory passed on to another SAC agent outputting two along-track distances (w.r.t end dock) denoting the placement of the approach and final approach waypoints as shown in Figure 23. The training curve is shown in Figure 22 where it is clear that although the agent learns a policy which does often end in mission success (although this seems better than it is due to a rolling average being used on the logged per episode reward), it fails at consistently reaching reward values close to +1.

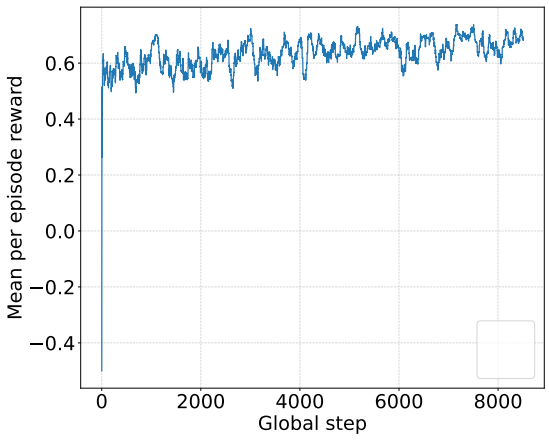


Fig. 22. The training curve for the discrete switch point placement RL agent where each step refers to a full episode rollout.

A similar MC analysis can be performed where an additional third system is compared placing the approach and final approach points at the constant benchmark values (4/2) resulting in the metrics presented in Table V and Figure 24.

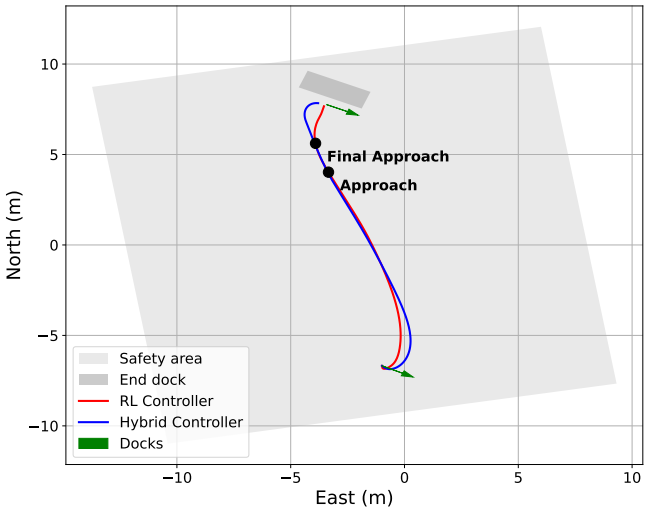


Fig. 23. Example of hybrid system where the RL controller output (red line) is used as the desired trajectory, with the switch points placed by a secondary RL algorithm, fed to the benchmark G&C system (blue line).

TABLE V
RESULTS OF MONTECARLO ANALYSIS ON RL CONTROLLER VS HYBRID SYSTEM VS HYBRID SYSTEM WITH FIXED SWITCH POINTS.

Metric	All-in-one RL	Hybrid RL	Hybrid fixed 4/2
Success %	**96.6**	25.0	73.8
Crash %	**1.7**	0.0	0.4

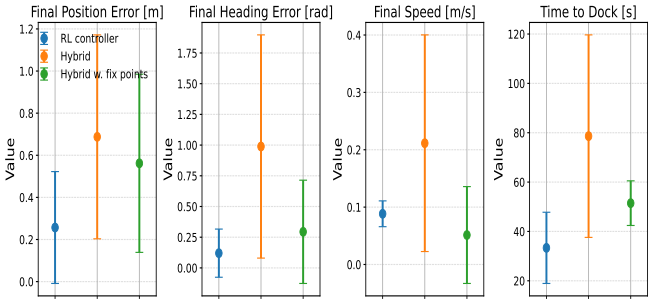


Fig. 24. Resulting metrics of MonteCarlo analysis on different RL training phases.

The clear drop in performance of the hybrid system can be attributed to the suboptimal policy learnt by the discrete switch agent. The low success rate combined with the low crash rate gives an indication that the agent prefers to place the final approach points far from the end dock to avoid crashing. This results in the vessel switching to the slow and inefficient (due to unconstrained sway motion) DP controller too soon causing it to never reach the dock due to time truncation at 300s. It is shown that placing the switch points at the fixed values found in the benchmark design (4/2) results in faster, and more successful docking.

Well suited to continuous control problems, SAC struggles with discrete, one-shot tasks more akin to a contextual bandit than a full MDP [22]. In such problems, the agent selects a single action per episode and receives a highly delayed and

sparse reward; a binary success flag and a scalar time penalty providing nearly no learning gradient needed for optimal credit assignment leading to slow or unstable policies. Although not implemented in this study a simpler, alternative algorithm could be based on the following rules:

1) Set final approach switch at point on trajectory where $v_{des} = v_{transition}$ to ensure smooth transition between controllers.
2) Set approach waypoint at a fix distance from the final approach, ensuring enough distance is given to slow down to $v_{transition}$.
3) If curvature is detected between approach and final approach points, move approach point further back to avoid slowing down during a turn (loss of maneuverability).

D. Real-Life Validation

A full real-life test including individual validation and comparison of the different systems as well as recording multiple runs to average out noise was planned. However, hardware issues limited the scope of this validation with only the benchmark system being tested on a manually generated docking trajectory. Figure 25 shows a screen capture of the real-life test video recording while the recorded data is further analysed below.

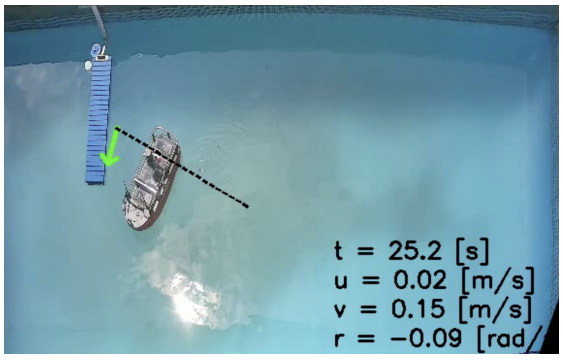


Fig. 25. A screen capture of the recorded real life test at the Damen facilities captured by top-down CCTV camera overlayed with digital data.

TABLE VI
METRICS: REAL VESSEL VS SIMULATION MODEL USING THE BENCHMARK SETUP.

Metric	Real	Simulation
Success	Yes	Yes
Crash	No	No
t_{dock} [s]	35.17	49.8

Comparing the position response of the vessel in Figure 26 and the corresponding docking times in Table VI, the simulation-to-real discrepancy can be considered quite small with the position response being very similar while the times being within 29.4% of each other. A more detailed analysis of the velocity responses in Figure 27 shows the model captures DAVE's surge dynamics very well but significantly deteriorates when it comes to the sway and yaw motions due to their tight cross-coupling effects becoming a potential problem in

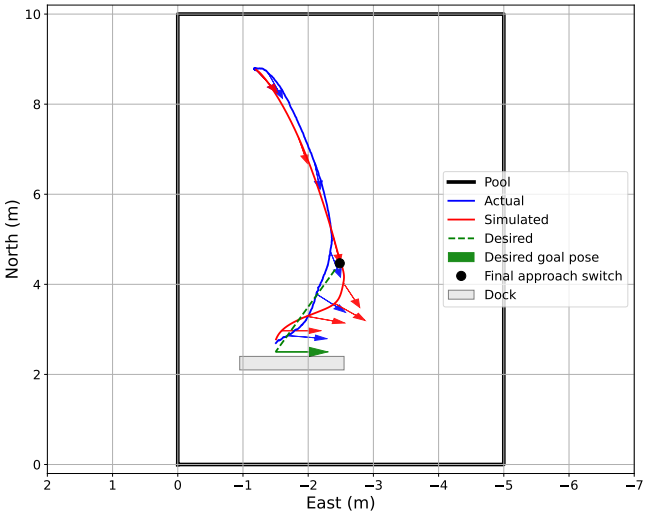


Fig. 26. Real vessel vs simulation run on the same docking scenario.

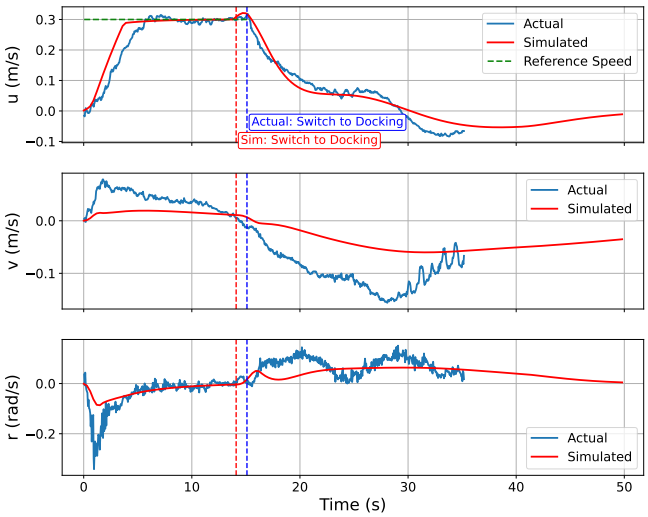


Fig. 27. Real vs simulated velocity profile related to Figure 26

low speed, 3DOF applications. Due to the closed-loop nature of this validation, explicit statements about the simulation model fidelity are avoided. However, close alignment does suggest a small enough simulation-to-real gap to allow for simulation based controller design.

VI. CONCLUSION

This study addressed the dock-to-dock motion control problem for autonomous vessels under environmental disturbances and geometric constraints. A benchmark guidance and control system was developed as a baseline and compared to reinforcement learning and hybrid approaches trading off the added performance for inherent stability guarantees. While the RL controller significantly improved performance and disturbance rejection capabilities, its current limitations in generalising safely to unseen docking scenarios and its black-box nature remain barriers to deployment in maritime settings. Hybrid approaches offer a promising compromise, with end-to-end

RL more justified in fast, dynamic domains such as aerial robotics, whereas in slower, safety-critical maritime settings where human oversight remains, interpretable and reliable conventional methods remain essential.

Acknowledgments

I would like to thank Damen for providing the necessary facilities and technical support during testing and gratefully acknowledge the guidance of the TU Delft supervisor, Dr. Erik-Jan van Kampen, and the Damen supervisor, Dr. Kasper van der El.

References

[1] A. Azarko and M. Molica Colella, "The rising tide of the autonomous ships market," *Open Access Government*, vol. 41, pp. 436–437, 2024. DOI: 10.56367/oag-041-10323.

[2] C. Bagoulla and P. Guillotreau, "Shortage and labor productivity on the global seafaring market," in *Seafarers : an international labour market in perspective*, P. Chaumette, Ed., Gomylex, 2016, pp. 15–27.

[3] S. Thombre, Z. Zhao, H. Ramm-Schmidt, *et al.*, "Sensors and ai techniques for situational awareness in autonomous ships: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 64–83, 2022. DOI: 10.1109/TITS.2020.3023957.

[4] N. Minorsky, "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, 1922. DOI: 10.1111/j.1559-3584.1922.tb04958.x.

[5] M. Breivik, "Topics in guided motion control of marine vehicles," PhD thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2010.

[6] S. J. N. Lexau, M. Breivik, and A. M. Lekkas, "Automated docking for marine surface vessels—a survey," *IEEE Access*, vol. 11, pp. 132 324–132 367, 2023. DOI: 10.1109/access.2023.3335912.

[7] S. Kockum, "Autonomous docking of an unmanned surface vehicle using model predictive control," Student paper, Lund University, Lund, Sweden, 2022.

[8] A. Vijayakumar, A. A, and A. Somayajula, "Model predictive path integral docking of fully actuated surface vessel," in *2025 IEEE Underwater Technology (UT)*, IEEE, 2025, pp. 1–6. DOI: 10.1109/ut61067.2025.10947451.

[9] M. L. Darby and M. Nikolaou, "Mpc: Current practice and challenges," *Control Engineering Practice*, vol. 20, no. 4, pp. 328–342, 2011. DOI: 10.1016/j.conengprac.2011.12.004.

[10] G. Juin-Gauthier, A. Babarit, B. Elie, O. Kermorgant, and V. Fremont, "Waves filtering in heading controllers: Impact on the power production of an energy ship," *IFAC-PapersOnLine*, vol. 58, pp. 458–463, 2024. DOI: 10.1016/j.ifacol.2024.10.096.

[11] M. Breivik, "Marine craft: 21st century motion control concepts," *Sea Technology*, vol. 47, no. 3, pp. 33–36, 2006.

[12] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: Wiley, 2011. DOI: 10.1002/9781119994138.

[13] L. Zhang, X. Peng, N. Wei, Z. Liu, C. Liu, and F. Wang, "A thrust allocation method for dp vessels equipped with rudders," *Ocean Engineering*, vol. 285, p. 115 342, 2023. DOI: 10.1016/j.oceaneng.2023.115342.

[14] C. de Wit, "Optimal thrust allocation methods for dynamic positioning of ships," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2009.

[15] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *IFAC Proceedings Volumes*, vol. 36, no. 21, pp. 211–216, 2003. DOI: 10.1016/s1474-6670(17)37809-6.

[16] D. Menges and A. Rasheed, "An environmental disturbance observer framework for autonomous surface vessels," *Ocean Engineering*, vol. 285, p. 115 412, 2023. DOI: 10.1016/j.oceaneng.2023.115412.

[17] S. Larsen, H. Helgesen, J. Walmsness, G. Kufoalor, and T. Johansen, "Automatic docking with extended dynamic positioning," *Journal of Marine Science and Technology*, vol. 29, pp. 770–788, 2024. DOI: 10.1007/s00773-024-01018-y.

[18] J. E. Walmsness, H. H. Helgesen, S. Larsen, G. K. M. Kufoalor, and T. A. Johansen, "Automatic dock-to-dock control system for surface vessels using bumpless transfer," *Ocean Engineering*, vol. 268, p. 113 425, 2023. DOI: 10.1016/j.oceaneng.2022.113425.

[19] M. Breivik and T. Fossen, "A unified concept for controlling a marine surface vessel through the entire speed envelope," in *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, 2005, pp. 1518–1523. DOI: 10.1109/.2005.1469807.

[20] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, "Trajectory planning and control for automatic docking of asvs with full-scale experiments," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 337–15 344, 2020. DOI: 10.1016/j.ifacol.2020.12.1451.

[21] G. Williams, A. Aldrich, and E. Theodorou, *Model predictive path integral control using covariance variable importance sampling*, 2015. arXiv: 1509.01149.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[23] L. Busoniu, T. de Bruin, D. Tolic, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018. DOI: 10.1016/j.arcontrol.2018.09.005.

[24] T. Lillicrap, J. Hunt, A. Pritzel, *et al.*, *Continuous control with deep reinforcement learning*. arXiv: 1509.02971.

[25] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1582–1591.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1861–1870.

[27] R. Deraj, s. Kumar, S. Alam, and A. Somayajula, "Deep reinforcement learning based controller for ship navigation," *Ocean Engineering*, vol. 273, p. 113 444, 2023. DOI: 10.1016/j.oceaneng.2023.113937.

[28] C. Zhou, Y. Wang, L. Wang, and H. He, "Obstacle avoidance strategy for an autonomous surface vessel based on modified deep deterministic policy gradient," *Ocean Engineering*, vol. 243, p. 110 166, 2022. DOI: 10.1016/j.oceaneng.2021.110166.

[29] Y. Guo, Z. Zhang, B. Zheng, and D. Yanhua, "An autonomous path planning model for unmanned ships based on deep reinforcement learning," *Sensors*, vol. 20, p. 426, 2020. DOI: 10.3390/s20020426.

[30] J. C. Sadlier, "Deep reinforcement learning for the automatic six-degree-of-freedom docking maneuver of space vehicles," M.S. thesis, Delft University of Technology, 2022.

[31] V. Hassija, V. Chamola, A. Mahapatra, and A. Singal, "Interpreting black-box models: A review on explainable artificial intelligence," *Cognitive Computation*, vol. 16, pp. 45–74, 2023. DOI: 10.1007/s12559-023-10179-8.

[32] W. Cai, A. B. Kordabad, H. N. Esfahani, A. M. Lekkas, and S. Gros, "Mpc-based reinforcement learning for a simplified freight mission of autonomous surface vehicles," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2990–2995. DOI: 10.1109/cdc45484.2021.9683750.

[33] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[34] M. Guerrier, H. Fouad, and G. Beltrame, *Learning control barrier functions and their application in reinforcement learning: A survey*, 2024. arXiv: 2404.16879.

[35] K. Pham, A. T. Nguyen, L. N. Nguyen, and T. Van Do Thi, "Application of the ornstein–uhlenbeck process to generate stochastic vertical wind profiles," *Journal of Aircraft*, vol. 62, no. 4, pp. 1067–1071, 2025. DOI: 10.2514/1.C038089.

[36] Airmar Technology Corporation, *Wx series weatherstation® multisensor (120wx, 220wx) specification sheet*, Rev. May 18, 2022, 2022.

[37] H. L. J. Thaams, "Guidance, navigation and control of autonomous vessels," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2019.

[38] A. Haseltalab, "Control for autonomous all-electric ships: Integrating maneuvering, energy management, and power generation control," PhD thesis, Delft University of Technology, Delft, The Netherlands, 2019.

[39] W. Cai, M. Zhang, Q. Yang, C. Wang, and J. Shi, "Long-range uwb positioning-based automatic docking trajectory design for unmanned surface vehicle," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–12, 2023. DOI: 10.1109/tim.2023.3271005.

[40] B. Srikanth and M. Sankar, "Implementation of anti-windup techniques for the improvement of pid performance," in 2021, pp. 585–593. DOI: 10.1007/978-981-15-8439-8_48.

[41] V. Patel, "Ziegler-nichols tuning method: Understanding the pid controller," *Resonance*, vol. 25, pp. 1385–1397, 2020. DOI: 10.1007/s12045-020-1058-z.

[42] K. Li, J. Chen, D. Yu, *et al.*, "Deep reinforcement learning-based obstacle avoidance for robot movement in warehouse environments," in *2024 IEEE 6th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, IEEE, 2024, pp. 342–348. DOI: 10.1109/ICCASIT62299.2024.10828100.

[43] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko, "Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications," *IEEE Access*, vol. 12, pp. 175 473–175 500, 2024. DOI: 10.1109/access.2024.3504735.

[44] H. Müller and D. Kudenko, *Improving the effectiveness of potential-based reward shaping in reinforcement learning*, 2025. arXiv: 2502.01307.

[45] W. G. Cochran, *Sampling Techniques*, 3rd ed. New York: John Wiley & Sons, 1977, ISBN: 978-0-471-16240-7.

Appendix A
Parameter Values

Simulation Model

The maneuvering model identified in [38] replaces the $\mathbf{C}_{\mathrm{A}}$, $\mathbf{D_L}$ and $\mathbf{D_{NL}}$ with a nonlinear drag polynmial for each degree of freedom collected in the vector $\boldsymbol{d}(\boldsymbol{\nu})$ leading to the following equation of motion:

$$\left(\mathbf{M}_{\mathrm{RB}}+\mathbf{M}_{\mathrm{A}}\right)\dot{\boldsymbol{\nu}}+\mathbf{C}_{\mathrm{RB}}(\boldsymbol{\nu})\,\boldsymbol{\nu}+\boldsymbol{d}(\boldsymbol{\nu})\;=\;\boldsymbol{\tau}_{\mathrm{control}}+\boldsymbol{\tau}_{\mathrm{disturbance}}. \tag{23}$$

Mass Matrix

$$\mathbf{M}_{\mathrm{RB}}=\begin{bmatrix} m & 0 & 0\\ 0 & m & m x_g\\ 0 & m x_g & I_z \end{bmatrix}, \qquad \mathbf{M}_{\mathrm{A}}=\begin{bmatrix} -X_{\dot{u}} & 0 & 0\\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}}\\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix}, \tag{24}$$

Rigid-Body Coriolis–Centripetal

$$\mathbf{C}_{\mathrm{RB}}(\boldsymbol{\nu})=\begin{bmatrix} 0 & 0 & -m\,(x_g r+v)\\ 0 & 0 & m\,u\\ m\,(x_g r+v) & -m\,u & 0 \end{bmatrix}. \tag{25}$$

Nonlinear Hydrodynamic Forces (Drag Polynomial)

$$\boldsymbol{d}(\boldsymbol{\nu}) = \begin{bmatrix} f_X(u,v) \\ f_Y(u,v) \\ f_R(r) \end{bmatrix}. \tag{26}$$

a) Surge polynomial:

$$f_X(u,v) = p_{x0}\,u \;+\; p_{x1}\,|v| \;+\; p_{x2}\,u^2 \;+\; p_{x3}\,u|v| \;+\; p_{x4}\,|v|^2 \\ +\; p_{x5}\,u^3 \;+\; p_{x6}\,u^2|v| \;+\; p_{x7}\,u|v|^2 \;+\; p_{x8}\,|v|^3. \tag{27}$$

b) Sway polynomial:

$$f_Y(u,v) = \operatorname{sgn}(v)\Big(p_{y0}\,|v| \;+\; p_{y1}\,u|v| \;+\; p_{y2}\,|v|^2 \;+\; p_{y3}\,u^2|v| \\ +\; p_{y4}\,u|v|^2 \;+\; p_{y5}\,|v|^3\Big). \tag{28}$$

c) Yaw polynomial built from sway:

$$f_R(r) \;=\; \frac{L}{3}\,f_Y\!\left(0,\; \frac{L}{3}\,r\right). \tag{29}$$

Benchmark G&C Design

Radius of acceptance: $1\,\mathrm{m}$

DP Controller Gains:

$$K_p^{\mathrm{DP}} = \mathrm{diag}\,(4.5,\,2,\,1.5) \quad (x,\,y,\,\psi)$$
$$K_i^{\mathrm{DP}} = \mathrm{diag}\,(0.05,\,0.01,\,0.055)$$
$$K_d^{\mathrm{DP}} = \mathrm{diag}\,(20,\,13.5,\,4.5)$$

Low-Pass Filter Gains:

$$\omega = \mathrm{diag}\,(0.2,\,0.2,\,0.15) \quad (x,\,y,\,\psi)$$
$$\zeta = \mathrm{diag}\,(1,\,1,\,1)$$

Sailing Controller Gains:

$$K_p^{\mathrm{Sail}} = \mathrm{diag}\,(8000,\,-4.0) \quad (\mathrm{v},\,\psi)$$
$$K_i^{\mathrm{Sail}} = \mathrm{diag}\,(1250,\,-0.2)$$
$$K_d^{\mathrm{Sail}} = \mathrm{diag}\,(100,\,-12)$$

Lookahead Distance: $2\,\mathrm{m}$

TABLE VII
DAVE MODEL PARAMETERS.

Parameter	Description	Value	Unit
Maneuvering model			
m	Vessel mass (rigid body)	39.23	[kg]
I_z	Yaw moment of inertia about z	4.76	[kg m^2]
x_g	Longitudinal CG position (body frame)	0.0585	[m]
L	Vessel length	1.19	[m]
$X_{\dot{u}}$	Added-mass derivative in surge	-2.79	[kg]
$Y_{\dot{v}}$	Added-mass derivative in sway	-114.35	[kg]
$Y_{\dot{r}}$	Cross added-mass derivative (sway–yaw)	0.0	[kg m]
$N_{\dot{r}}$	Added-mass derivative in yaw	-6.30	[kg m^2]
$\mathbf{p}_x = [p_{x0}, \ldots, p_{x8}]$	Surge drag polynomial coefficients for $f_X(u, \lvert v \rvert)$	$[3.56,\ -0.70,\ -1.86,\ 75.11,\ 10.98,\ 10.22,\ -143.71,\ 196.39,\ 7.89]$	[-]
$\mathbf{p}_y = [p_{y0}, \ldots, p_{y5}]$	Sway drag polynomial coefficients for $f_Y(u, \lvert v \rvert)$ (odd symmetry)	$[15.46,\ -40.31,\ -30.10,\ 195.13,\ 37.23,\ 234.39]$	[-]
u_{max}	Max. surge speed at max $n = 600rpm$	0.66	[m/s]
Thruster model			
$n_{thrusters}$	Number of thrusters	2	[-]
l_x	Longitudinal lever arm from CG to each thruster	-0.49	[m]
l_y	Lateral lever arm from CG to each thruster	± 0.1	[m]
T_{nn}	Quadratic thrust coefficient in $T = T_{nn}\, n^2$	6.29×10^{-6}	[N/(rpm)^2]
$n_{\max}$	Maximum propeller speed (per thruster)	600	[rpm]
$\dot{n}_{\max}$	Maximum propeller speed slew rate	450	[rpm/s]
$\alpha_{\max}$	Maximum azimuth angle (per thruster)	π	[rad]
$\dot{\alpha}_{\max}$	Maximum azimuth slew rate	$\pi/3$	[rad/s]
$F_{x_{max}}$	Maximum pure surge force	4.53	[N]
$F_{y_{max}}$	Maximum pure sway force (accounting for induced moment)	1.81	[N]
$M_{z_{max}}$	Maximum pure yaw moment	0.91	[Nm]
Wind model			
ρ	Air density	1.225	[kg/m^3]
A_{front}	Frontal area above waterline	0.055	[m^2]
A_{lat}	Lateral area above waterline	0.118	[m^2]

TABLE VIII
RL HYPERPARAMETERS BY CURRICULUM PHASE (SAC).

Parameter	Description	Value	Unit
Global SAC settings			
Max episode steps	Episode horizon	1000 (300s)	[-]
Algorithm	Base RL algorithm	SAC	[-]
Optimizer	Optimizer used for all nets	Adam	[-]
γ	Discount factor	0.99	[-]
τ	Target network smoothing coeff.	0.005	[-]
Replay buffer size	Max transitions stored	1e6	[steps]
Batch size	Minibatch size per gradient step	256	[samples]
Grad steps / env step	Training intensity	1	[-]
Learning starts	Steps before training begins	20,000	[steps]
H_{target}	Entropy target	$-action_{dim}$	[-]
Entropy tuning	Automatic temperature tuning	on	[-]
Exploration init	Initial std / noise strategy	-0.5	[-]
Clip grad norm	Gradient clipping	1.0	[-]
Obs/Reward norm	Running normalization	obs: off, rew: off	[-]
Seed	Random seed for reproducibility	0	[-]
Phase 1 (no wind, fixed dock/safety)			
Dock size range	Fixed domain (length/width)	L=3, W=1	[m]
Safety area range	Fixed domain (length/width)	L=20, W=20	[m]
Wind speed range	Fixed domain	0	[m/s]
Wind angle range	Fixed domain	0	[rad]
Init policy	Starting weights	Randomised	[-]
Total timesteps	Training steps in Phase 1	6e6	[steps]
Policy architecture	Actor (hidden layers)	[256, 256, 256]	[-]
Critic architecture	Q-networks (hidden layers/units)	[256, 256, 256]	[-]
Actor LR	Policy network learning rate	1e-4	[1/s]
Critic LR	Q-network learning rate	1e-4	[1/s]
Alpha LR	Temperature learning rate	1e-4	[1/s]
Phase 2 (wind randomization; bootstrapped from Phase 1)			
Dock size range	Fixed domain (length/width)	L=3, W=1	[m]
Safety area range	Fixed domain (length/width)	L=20, W=20	[m]
Wind speed range	Domain randomization	[0,5]	[m/s]
Wind angle range	Domain randomization	$[-\pi, \pi]$	[rad]
Init policy	Starting weights	Randomised	[-]
Warmup (buffer)	Bootstrap transitions from Phase 1 policy	75,000	[samples]
Total timesteps	Training steps in Phase 2	2.7e6	[steps]
Policy architecture	Actor (hidden layers)	[512, 512, 512]	[-]
Critic architecture	Q-networks (hidden layers/units)	[512, 512, 512]	[-]
Actor LR	Policy network learning rate	5e-5	[1/s]
Critic LR	Q-network learning rate	5e-5	[1/s]
Alpha LR	Temperature learning rate	5e-5	[1/s]
Phase 3 (wind + dock/safety randomization; bootstrapped from Phase 2)			
Dock size range	Domain randomization (length/width)	[1, 5]	[m]
Safety area range	Domain randomization (length/width)	[1, 20]	[m]
Wind speed range	Domain randomization	[0,5]	[m/s]
Wind angle range	Domain randomization	$[-\pi, \pi]$	[rad]
Init policy	Starting weights	From Phase 2	[-]
Warmup (buffer)	Bootstrap transitions from Phase 2 policy	75,000	[samples]
Total timesteps	Training steps in Phase 3	8.6e6	[steps]
Policy architecture	Actor (hidden layers)	[512, 512, 512]	[-]
Critic architecture	Q-networks (hidden layers/units)	[512, 512, 512]	[-]
Actor LR	Policy network learning rate	5e-5	[1/s]
Critic LR	Q-network learning rate	5e-5	[1/s]
Alpha LR	Temperature learning rate	5e-5	[1/s]

3

Literature Study

This chapter contains the literature study performed to gather the necessary background knowledge as well as to refine the thesis aim and research questions. It starts with an overview on the field of motion control for surface vessels, followed by a definition of the autonomous dock-to-dock maneuvering task accompanied by current research and methods used to solve it. This sets the foundation for the next section, where advanced G&C methods are investigated along with their potential applications for a dock-to-dock mission. An overview of Reinforcement Learning (RL) is then given starting from the fundamentals and ending in an overview of state-of-the-art RL algorithms. Finally, a summary is given where the key findings are tied back to the research questions.

3.1. Motion Control for Marine Surface Vessels

In its most basic form, motion control for marine vessels can be described as ”the ability to accurately maneuver along a given path” [9]. Motion control theory is fundamental to any autonomous system and, in marine applications, encompasses two main tasks: position-keeping and trajectory-following. Over the years various motion control strategies have been developed for the execution of such tasks ranging from conventional PID-based autopilots and position-keeping controllers to more advanced, model-based and data-driven approaches. However, most existing control methods focus on solving either the position-keeping or the trajectory-following tasks in isolation and methods for solving both tasks in a unified manner are still lacking [5]. This section begins by introducing the hierarchical control framework, followed by a review of different existing motion control techniques for executing the position-keeping and trajectory tracking tasks.

3.1.1. Hierarchical control

Motion control is often divided into different hierarchical levels to make the problem more manageable as shown in Figure 3.1. Such a modular framework allows for increased scalability and interpretability.

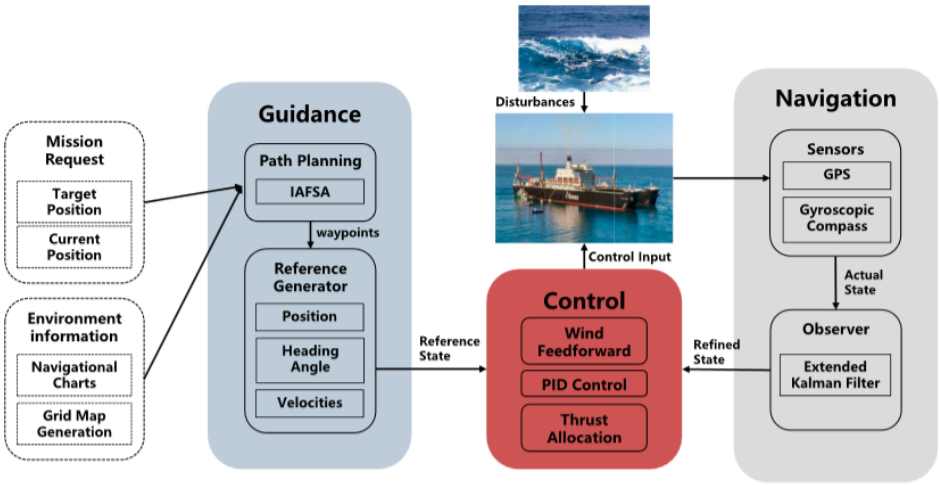


Figure 3.1: A generalised hierarchical control structure for an autonomous surface vessel taken from [10].

The definitions of each module (where planning is considered a separate module) can be found below [11]:

- Navigation: Responsible for estimating position and orientation of vessel continuously by integrating information from multiple sensors.
- Path Planning: The high-level module responsible for planning a trajectory based on an initial and final point. The trajectory consists of a path, a velocity profile and a heading profile while also containing safety bounds on these profiles.
- Guidance: The mid-level module taking in the entire trajectory from the path planner along with the state estimate from the navigation module and outputs feasible reference commands to best follow the trajectory.
- Control: The low-level module responsible for ensuring the reference commands from the guidance module are tracked as closely as possible through the use of actuators.

3.1.2. Position-keeping task

The concept of Dynamic Positioning (DP) first arose in the early 1960s motivated by the need for accurate placement of ships for offshore drilling purposes in locations where conventional barges with anchors cannot be used due to large water depths [5]. The goal of a DP controller is to position the vessel in the desired degrees of freedom (DOF) (position and heading) while counteracting the environmental forces through the vessels actuators as shown in Figure 3.2.

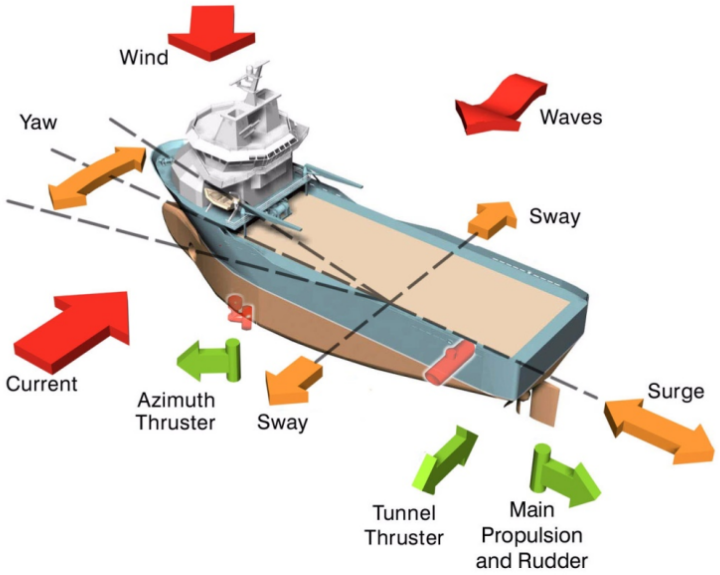


Figure 3.2: The classic Dynamic Positioning problem where the DP controller maintains the vessel within the desired degrees of freedom (orange arrows) while counteracting environmental disturbances (red arrows) through the use of its actuators (green arrows) [5].

The DP controller consists of 3 independent Proportional-Integral-Derivative (PID) controllers for each DOF outputting the required force (or moment) to maintain the vessel's position and orientation at the reference pose ($[x, y, \psi]$ waypoint). The PID controllers use the respective distance or heading errors to compute the desired forces and moment F_x, F_y, M_z. These are converted to low-level actuator commands through the use of a thrust allocation algorithm, a constrained optimisation problem where the goal is to solve for the thrust forces that best achieve the desired control forces and how to best distribute these over the different vessel actuators [11], [12], [13].

3.1.3. Trajectory-following tasks

Trajectory-following refers to the ability of a vessel to track a predefined geometric path while maintaining a desired velocity profile. Unlike position-keeping, trajectory-following enables autonomous navigation along a route while compensating for disturbances. This is achieved through a combination of a guidance and control (G&C) system.

At low speeds, a DP controller could in theory be applied however, this quickly becomes inefficient at higher speeds as it does not necessarily align the vessel heading with the course angle (direction of travel), leading to a rapid buildup of hydrodynamic forces. It is therefore common practice to rely on two independent controllers, ensuring trajectory alignment while also bypassing the thrust allocation module due to the simpler control structure:

- A velocity controller relating velocity error to thruster propeller speed n.
- A heading controller relating heading error to thruster azimuth α.

This setup requires the guidance module to supply both velocity and heading references in real time allowing for optimal trajectory tracking. In its simplest form, the guidance module selects the closest waypoint to the vessel and commands the corresponding velocity and heading. Although simple, such a guidance system introduces discrete jumps in the reference command leading to instability and actuator wear as well as inefficient path following with to predictive capabilities (corner cutting). A low pass filter is commonly applied to smooth out the references.

A more effective and widely used alternative is Line-of-Sight (LOS) guidance. Purely geometric, it computes the required heading ψ_d to converge with a point on the trajectory placed at a fix distance, the lookahead distance, from the current vessel pose. The velocity is then typically taken as the pre-assigned velocity (from the trajectory) on that path segment. This allows the vessel to anticipate

changes in the trajectory resulting in smoother and more efficient tracking [14].

$$\psi_d = \arctan\left(\frac{y_{los} - y}{x_{los} - x}\right) \tag{3.1}$$

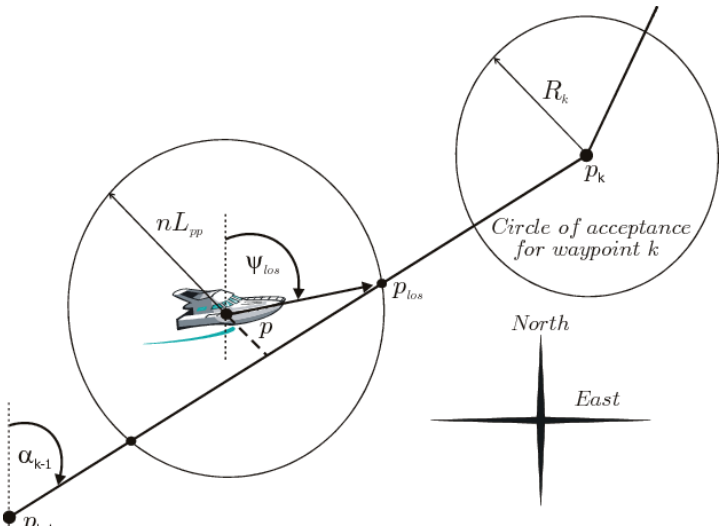


Figure 3.3: The Line-of-Sight principle [14].

Other, more involved, guidance approaches are presented in [15]: Lyapunov Based Virtual Target which generates a moving virtual target that travels along a trajectory and Jacobian Based Priority Task which formulates it as an optimisation problem where higher priority objectives (assigned manually) are more strongly enforced. However, Lyapunov based systems provide no guarantees on the transient response (focused on steady-state stability) while both approaches introduce significant complexity for minor gains in performance.

3.1.4. Environmental Disturbance Rejection

Robustness to environmental disturbances is a key requirement for both the position-keeping and trajectory-following tasks. In its most basic form, this robustness is achieved through the use of the integral term in the PID controllers, which, in theory, eliminates the steady-state error by applying a corrective action based on past errors. In practice however, relying solely on this integral term is not realistic especially in time sensitive maneuvers or highly dynamic environments such as the docking phase as it is slow to buildup (high integral gains lead to deterioration of ideal response). It is thus common practice to estimate the disturbance force vector and apply this estimate in a feedforward manner essentially 'canceling out' the disturbance forces experienced by the vessel. Two common ways of doing this are:

- Predicting the forces using mathematical models and real-time sensor measurements (e.g. wind vane).
- Estimating the forces through the use of an online observer.

[11] proposes a dynamical model for estimating the wind force experienced by the vessel based on a measured windspeed. However, generating mathematical models for current and wave disturbances becomes much more complex and in some cases, even unfeasible. Such is the case for current which is extremely difficult to accurately measure in real-time. To bypass these problems observers can be used estimating the disturbance forces based on observable vessel states. A simple observer would for example predict these forces using the deviation between the actual vessel position & orientation and the expected one. [16] proposes a nonlinear disturbance observer derived using a Lyapunov-based stability approach for this exact purpose. Additionally, [17] proposes the integration of an additional wave filter in the control loop to avoid the propagation of high frequency oscillations in the heading controller due to high frequency waves.

3.2. Autonomous Dock-to-Dock Maneuvering

The recent focus on automation in the maritime industry has driven extensive research into autonomous docking of surface vessels. In high-level terms, automated docking involves systems that "enable vessels to dock safely, independently and energy-efficiently at designated locations with a specific heading" [6] while incorporating controller transitions between phases. The overall dock-to-dock maneuvering problem is illustrated in Figure 3.4, and a possible state-machine transition mechanism between transit and docking phases is shown in Figure 3.5. Challenges specific to the final docking phase include [6]:

- Managing large sideslip angles.
- Static and dynamic obstacle avoidance.
- Navigation in complex port geometries.
- Transitioning from high speed sailing phase to the docking phase (unifying the speed regimes).

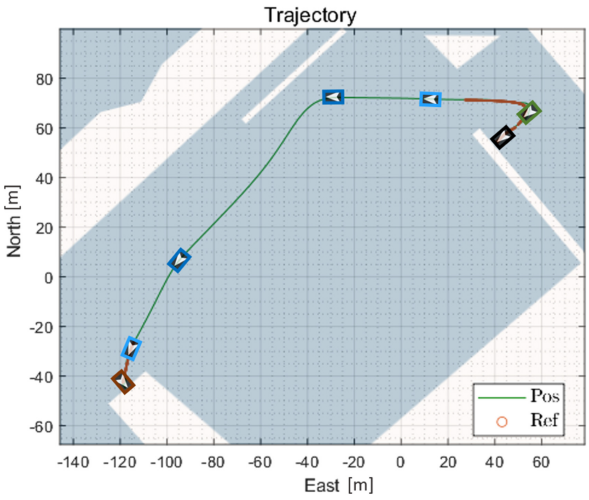


Figure 3.4: A dock-to-dock trajectory where the red coloured trajectory refers to undocking (top right) and docking (bottom left) while the green trajectory represents the high speed transit phase (sailing). Taken from [7].

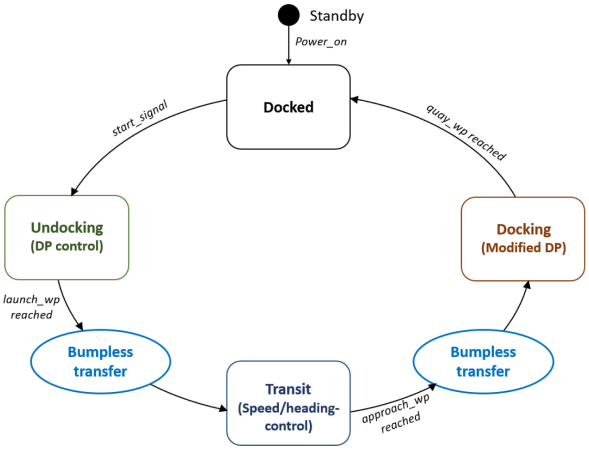


Figure 3.5: State machine illustrating transition between docking and sailing (transit) phase taken from [7].

[6] provides an extensive overview of research in the autonomous docking field spanning over 3 decades, assigning a Docking Characteristic Index to more than 200 papers, providing the reader with a semi-quantitative analysis of all this research. The papers investigated range from employing simple PID controllers to using AI-based methods such as Artificial Neural Networks (ANNs), Genetic algorithms and Reinforcement Learning (RL). Building on this, the docking problem can be structured according to the hierarchical control framework introduced in subsection 3.1.1, and is divided into the planning and control modules where the same trajectory-following guidance algorithms may be used.

3.2.1. Planning

Planning becomes of particular importance in the docking phase due to the strict requirements placed on both the positioning and the time related constraints of a vessel. Unlike the transit phase, straight line trajectories connecting high-level static waypoints are no longer adequate for the docking phase due to geometrically contrained harbour environments[6]. [18] proposes the use of Bezier curves, providing a smooth and flexible method of defining paths based on interpolation between control points, which are defined based on the vessel dynamics to avoid unfeasible trajectories. In contrast, [19] proposes the use of Dubins paths, providing the shortest path for a vehicle to translate and turn with maximum specified curvature.

Different guidance systems (e.g., LOS, Constant Bearing, Pure Pursuit) can then be applied to generate reference commands, with LOS already described in subsection 3.1.3 and others offering alternative convergence strategies [6].

3.2.2. Control

The control problem for autonomous surface vessels was already extensively described in section 3.1 hence, in this section, more emphasis will be placed on controllers which unite the high speed sailing phase with the low speed docking phase.

[7] proposes using two separate controllers; a modified DP controller for the fully actuated low speed regime and a velocity and heading controller for the underactuated high speed regime similar to the controllers presented in section 3.1 for the position-keeping and trajectory-following tasks. It derives a smooth, bumpless transfer between these two controllers through the use of an Integrator Resetting method (matching the output of the receiving controller with that of the previous controller) eliminating discontinuities in the control action which could lead to rapid accelerations or instability. This transition point between the two controllers is based on a predefined acceptance radius around the 'approach waypoint', a waypoint situated just outside the port.

Taking it one step further, [20] proposes a unified controller deriving a nonlinear, model-based velocity and heading controller through backstepping which seemlessly transitions between the two regimes through the use of a weighting function $\sigma(U)$, by limiting the sway motion based on the vessel's surge velocity. Additionally, it includes a feedforward term for the estimated bias vector due to environmental disturbances. In this case, the transition point is not a discrete point, but rather a continuous region before approaching the dock which is determined based on the vessel's velocity. Although elegant, its high model dependency rules it out from most real-life applications.

3.3. Advanced G&C methods

The widely adopted, industry standard G&C methods presented in section 3.1 have been successfully implemented in marine motion control for decades, providing effective solutions for surface vessels. However, as fully autonomous operation becomes increasingly viable, several limitations emerge:

- **No built-in trajectory planning**: Conventional systems rely on predefined straight-line paths, sufficient for transit but problematic in docking where trajectory design is closely tied to G&C performance. Without dynamic adjustment, maneuvers risk being inefficient or unsafe.
- **Limited adaptability**: Manual tuning is extensive and highly dependent on vessel dynamics and port conditions.
- **Limited robustness**: PID controllers struggle with strong or rapidly changing disturbances; while force observers help, they add complexity.
- **Lack of predictive capability**: Conventional methods react to current errors but cannot optimize future control actions.

To address these limitations, more advanced G&C strategies have been developed that integrate both model-based optimization and data-driven approaches. Such methods provide improved trajectory planning, predictive control, adaptability, and generalisability, making them more suitable for the complex and safety-critical nature of docking. They also offer possibilities for higher-level decision making, such as determining when to transition to the docking phase, an underexplored aspect in current research.

3.3.1. Model based approaches

Model-based approaches explicitly incorporate system dynamics and constraints, enabling predictive control and more optimal decision-making compared to widely used PID controllers, which rely on reactive error correction. By predicting future states and adjusting actions accordingly, these methods optimise control inputs over large time horizons, making them well suited for complex maneuvers such as docking, where precise trajectory generation and constraint handling are imperative. Although not inherently more robust than classical feedback methods, they can be more readily adapted for robustness [21].

One of the best known model based controllers is **Model Predictive Control** (MPC). MPC is an optimisation based control strategy that computes control actions based on predictions of future states over a finite time horizon by solving a constrained optimization problem at each timestep whose objective is to minimise a cost function while respecting constraints (actuator limits, obstacle avoidance, etc...)

as shown in Figure 3.6. A commonly used cost function for motion control includes tracking error and energy consumption as shown in Equation 3.2. By design, MPC integrates trajectory planning and control into a single framework.

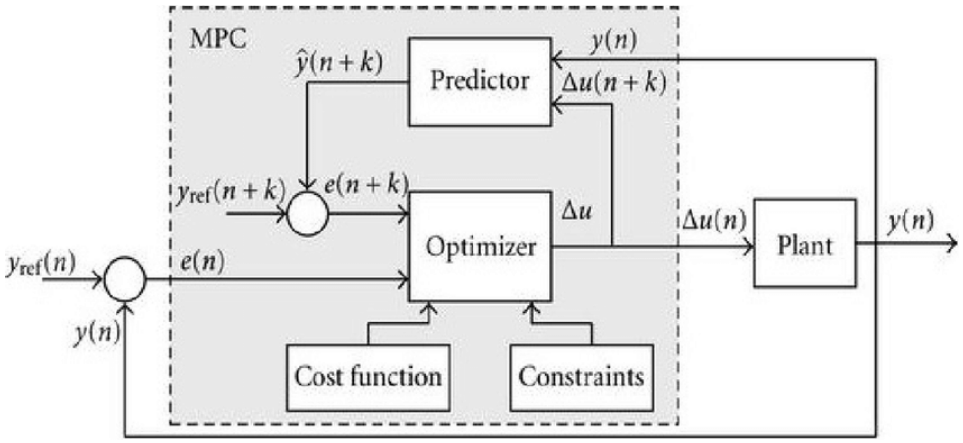


Figure 3.6: Model Predictive Control Schematic [22]

$$J = \sum_{k=0}^{N} x_k^T Q x_k + u_k^T R u_k \tag{3.2}$$

Where x_k is the state error at timestep k, u_k is the control input and Q and R are weighting matrices defining the relevant objectives.

While MPC provides optimal control actions within a constrained framework, it relies on linearised systems and convex cost functions reducing its effectiveness for highly non-linear problems [21] [8]. Mitigating these issues, **Model Predictive Path Integral Control** (MPPI) is a sampling based stochastic control method, which uses path integral formulation to evaluate many potential trajectories and select the most optimal one [23], [24]. Essentially, it replaces the optimisation problem in MPC by randomly sampling a distribution of possible control sequences at each timestep. This allows MPPI to handle complex nonlinear dynamics, high dimensional control tasks while removing the computationally expensive per-step optimisation [23].

Despite these advantages, model-based approaches introduce significant challenges:

- **High fidelity models**: Heavy reliance on model requires the expensive and time-consuming development of high fidelity dynamical models.
- **Computational costs**: Real-time implementation requires solving optimisation or sampling thousands of trajectories per timestep, which is computationally costly.

To mitigate these challenges, hybrid methods have been proposed. For instance, [25] applies MPC at a low update rate to generate feasible docking trajectories, while cost-effective PID controllers perform the high-rate trajectory tracking.

3.3.2. Data-driven Approaches

Data-driven approaches mitigate many of the issues encountered in model-based methods by learning control policies directly from experience rather than from explicit system formulations. Instead of predicting and optimising actions using a predefined model, they adaptively learn optimal actions through interaction with the environment. This makes them particularly attractive for high-level decision making tasks such as determining when to transition between mission phases or optimizing trajectories under unknown disturbances. Key advantages include [26]:

- **Handling model uncertainties**: data-driven approaches can adapt to unknown system dynamics making them more robust to modeling errors.

- **Adaptive learning**: Instead of relying on a fixed problem formulation, data-driven approaches can continuously learn and improve based on real-time data.
- **Generalisability to unseen problems**: If trained correctly, control policies generalise over a wide range of unseen problems.
- **Exploration of unknown solutions**: data-driven approaches (particularly Reinforcement Learning) can be used to explore novel, unknown solutions to a problem.

Applications to autonomous shipping are emerging. [27] demonstrates a DRL-based trajectory-following controller with improved robustness to disturbances compared to PID controllers, while [28] integrates DRL with real-time sensor data for dynamic obstacle avoidance. Data-driven approaches have also been leveraged for higher level planning and decision making tasks. [29] leverages DRL for optimised path planning showcasing its suitability to plan high-level tasks even for unseen missions. Similarly, [30] applies RL based docking strategies to a spacecraft docking problem highlighting the ability of RL to learn efficient, collision free docking maneuvers.

Despite their significant advantages, data-driven methods present several challenges [26]:

- **Stability**: Unlike traditional control methods derived based on stability considerations, data-driven approaches often times lack explicit stability guarantees.
- **Constraint Handling**: Hard constraint handling is not enforceable in many data-driven approaches.
- **Interpretability**: data-driven approaches often function as black boxes, making it difficult to understand their decision making, an essential component in ensuring safe control policies.
- **Sample Inefficiency**: data-driven approaches often require long and expensive training processes which quickly becomes infeasible for high dimensional, complex problems.

Research is addressing these challenges. For example, [31] introduces Lyapunov-based safe RL, which guarantees stability by restricting updates to safe actions (by tracking stability throughout training), though this emphasises steady-state behavior over transient performance. [32] proposes Control Barrier Functions (CBFs) for enforcing hard constraints through set invariance, or as a fail-safe filter modifying unsafe actions to ensure constraint compliance. A promising approach is hybrid RL-MPC as presented in [33] where RL is used to tune the MPC cost function weights and the constraints improving adaptability and generalisability while ensuring safe and interpretable control actions. Such an approach is particularly useful in applications such as autonomous docking where predictability and constraint handling is essential while still allowing for the exploration of novel solutions to the problem.

3.4. Reinforcement Learning

Reinforcement learning (RL) is a decision making framework where an agent learns a task by receiving rewards through interaction with an environment. The agent seeks to maximise the cumulative rewards over time by improving the action it takes through a process of trial and error [34]. RL can be formulated as a Markov Decision Process (MDP), a mathematical way of formalising sequential decision making where ”actions influence not just immediate rewards, but also subsequent states, and through those future rewards” [34]. As shown in Figure 3.7, the key components of a MDP are:

- State Space (S): Represents all possible environment configurations.
- Action Space (A): Contains all possible actions the agent can take.
- Transition function ($P(s'|s,a)$): The probability of transition to state s' given current state s and action a.
- Reward function ($R(s,a)$): The reward received when taking action a in state s.
- Policy ($\pi(a|s)$): The probability of taking action a in state s.

Having given a brief overview on the MDP problem formulation for RL, the next sections will focus on some RL fundamentals describing how agents optimise their behaviour in an MDP framework.

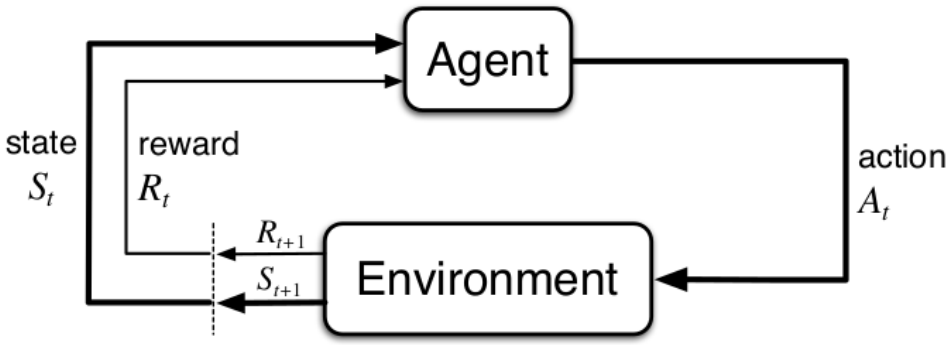


Figure 3.7: Agent-environment interaction in a MDP.

3.4.1. Value Functions & The Bellman Equation

Most RL algorithms involve estimating value functions; a function describing how good (in terms of expected future rewards) it is for the agent to be in the current state. The value function for MDPs can be defined as the expected return when starting in state s and following policy π thereafter [34]:

$$v_\pi(s) = E_\pi[G_t|S_t = s] \tag{3.3}$$

Where G_t is defined as the return (cumulative reward collected).

While $v_\pi(s)$ describes how good it is to be in state S followed by policy π, many RL algorithms estimate the action-value function $q_\pi(s, a)$, the expected return of taking action a in state s and following policy π afterwards defined as:

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] \tag{3.4}$$

Given that RL tasks involve sequential decision making, the value of a state depends both on the immediate reward and the value of the following state, leading to the recursive Bellman equation:

$$v_\pi(s) = \mathbb{E}_\pi\big[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s\big] \tag{3.5}$$

Where R_{t+1} is the immediate reward, γ is the discount factor determining the importance of future rewards ($0 \leq \gamma \leq 1$) and $v_\pi(S_{t+1})$ is the expected value of the next state.

3.4.2. The Learning Process

In high level terms, the goal of the learning process is to find an optimal policy π_* that maximises the value function $v_*(s)$ across all states, however the details of how this is done can differ significantly. This section will first touch upon how the agent collects the necessary data for learning, how they update their policies followed by different ways agents can improve their decision making and finally, how agents estimate and update values across states.

Online vs Offline

RL distinguishes between two ways of generating the necessary data from which the agent can optimise its decision making:

- **Online**: The agent collects new data/experiences by interacting with the environment in real-time.
- **Offline**: The agent does not interact with the environment in real-time, rather, it learns from pre-collected datasets.

Each method comes with its (dis)advantages, however, in safety critical tasks where exploring unsafe actions in real-time can be costly or even dangerous, offline is preferred. Additionally, online learning does not allow for the agent to restart the environment while learning (learning across multiple simulation tries e.g. try many docking attempts before settling on the optimal policy). For these reasons, only offline methods shall be considered from now on for autonomous dock-to-dock operations.

On-policy vs Off-policy

How an agent updates its policy during training can also be categorised into two main ideas:

- **On-Policy**: The agent improves the same policy being executed by the agent.
- **Off-Policy**: The agent improves a different policy from that being executed by the agent (e.g. learns from past experiences using a replay buffer).

Since on-policy methods update the policy being used, they tend to avoid large policy shifts leading to more stable and smooth learning. However, off-policy methods tend to be more sample efficient by reusing past experiences through the use of e.g. a replay buffer [34].

MonteCarlo vs Temporal Difference

How an agent estimates and updates values can also be split up into two distinct categories:

- **MonteCarlo (MC)**: estimates the value function using averaging sample returns where the value and policy estimations are changed only upon completion of a whole episode. MC methods sample and average returns (over whole episodes) for each state-action pair [34].
- **Temporal Difference (TD)**: estimates the value function at each step leveraging the idea of bootstrapping (update estimates based, in part, on other learned estimates from previous steps without needing to wait for a final outcome).

TD methods are linked to MC methods through the n-step TD concept (update estimates after n steps instead of every step). If n is increased to be equal to the number of steps in an entire episode, it essentially becomes a MC method. TD methods provide more flexibility and have generally been found to converge faster than MC methods.

Value vs Policy Iteration

Finally, the way a RL agent finds the optimal policy can also be categorised into two distinct methods:

- **Value iteration**: The agent first iteratively computes the optimal value function, making use of the Bellman equation, which is then used to extract the optimal policy (the one that maximises the optimal value function).
- **Policy iteration**: The agent directly improves the policy by evaluating the current policy and then refines its policy-based on this evaluation at each step until the policy converges.

Which method leads to faster convergence is still an ongoing discussion however, value iteration is generally simpler to implement as it removes the need to find an optimal policy (w.r.t to the current value function) at each step which is itself an extensive iterative process and is generally considered more stabled than policy-based methods [34].

3.4.3. An overview of existing RL solution methods

This section presents some solution methods used for learning. It begins with introducing a well known value-based method for discrete problems which is then extended to continuous problems. This is followed by an introduction to policy-based methods and finally, a hybrid method leveraging the advantages of both value-based and policy-based methods is presented.

Q-learning

Q learning builds upon the Temporal Difference (TD) method first introduced by [35]. It can be classified as a value-based, off-policy TD method for discrete state/action spaces. Q learning directly learns the optimal action-value function q_* independent of the policy being followed (off-policy) which allows it to determine the optimal action at each state purely based on maximizing future rewards. Additionally, this allows for a higher level of exploration during the learning process when compared to its on-policy counterparts such as SARSA as it allows using the greedy action (maximising future rewards) while still exploring with an exploratory policy (e.g. ϵ-greedy or random actions) when selecting actions. The update rule for Q learning is as follows [34]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{3.6}$$

Where γ is used to discount future rewards (importance of future rewards) and α is the learning rate.

Deep Q-learning

For all its benefits, Q-learning has the irredeemable drawback that it can only be used for discrete state/action spaces due to its need to store the learnt action-value function in a so called Q-table (each row represents a state and each column represents an action). Such tables become impossible to maintain as the state space becomes infinitely large. Deep Q-learning (DQN), first proposed in [36], is an extension of Q-learning which allows for continuous state spaces by approximating the Q function using Deep Neural Networks. It parametrizes Equation 3.6 using parameters θ, the weights of the neural network which are trained by minimising a sequence of loss functions $L_i(\theta_i)$ at each iteration i [36].

$$L_i(\theta_i) = \mathbb{E}[(y_i - Q(s, a; \theta_i))^2] \tag{3.7}$$

Where $y_i = \mathbb{E}[R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1})]$ is the target for iteration i [36].

In their simplest form, DQN's suffer from instability or even divergence when nonlinear function approximators are used to represent the action value function. This instability stems from several causes [37]:

- Consecutive observed experiences are highly correlated leading to inefficient learning.
- The action values (Q) used for learning and the target values (y_i) are highly correlated causing feedback loops and instability.
- Small updates to the action values (Q) can cause significant changes in the policy therefore changing the data distribution making learning non-stationary (hard to converge if data distribution changes over time).

[37] attempts to solve this by introducing two key ideas:

- Experience replay: removes correlation between successive experiences and smoothens out the data distribution changes by storing past experiences and then sampling them randomly.
- Target Network: A copy of the main Q network, which is only updated periodically, is used to compute target values y_i preventing a situation where the network learns from constantly changing values.

Policy Gradient Methods

So far the value-based methods presented learn a value function Q and select actions by maximising it. Selecting actions by maximising the Q function inherently limits the problem to discrete actions (the $\max_a$ argument assumes there are a finite set of actions to be compared). Policy gradient methods address this by taking a policy-based approach where a parameterized policy function $\pi(a|s, \theta)$ is directly optimized using gradient ascent [34] allowing for both discrete and continuous actions:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{3.8}$$

Where $\nabla J(\theta_t)$ is a "stochastic estimate whose expectation approximates the gradient of the performance measure with respect to its argument θ_t [34].

Actor Critic Methods

Policy gradient methods allow for continuous actions providing a solution to the main limitation of value-based methods. However, they suffer from high variance in their updates resulting in less stable and slow learning due to their inability to leverage value functions to guide the learning process. Actor-critic methods seek to combine the advantages of both value-based methods such as Deep Q-learning and policy-based methods (policy gradient methods) by using:

- An actor: a policy $\pi(a|s, \theta)$ which selects actions.
- A critic: a value function (or action value function $Q(s, a)$) that evaluates actions.

Instead of relying on high variance rollouts as in policy gradient methods, the critic helps 'guide' the actor by providing it with estimates of the value function resulting in more stable and sample efficient training. Such methods are widely used in control problems enabling both continuous states and actions while providing better learning capabilities when compared to policy gradient methods [34].

3.4.4. Some State-of-the-art RL algorithms

In this section an overview of some RL algorithms which can be used in the autonomous dock-to-dock problem shall be presented. The algorithms presented shall be restricted to model-free, offline methods for the following reasons:

- Model-free: Removes reliance on an accurate dynamical model, something which is time consuming and hard to obtain for complex environments subject to several external disturbances.
- Offline learning: Offline learning reduces the risk of taking unsafe actions during training, an important point for safety critical applications such as vessel autonomy.

Deep Deterministic Policy Gradients

[38] proposes adapting the ideas underlying the success of Deep Q-learning to the continuous action domain resulting in an actor-critic, model-free algorithm based on the deterministic policy gradient (see Figure 3.8). It solves the problems encountered in previous Determinstic Policy Gradient algorithms (instability for challenging problems, discrete actions) presented in [39] by incorporating the following improvements:

- Actor-Critic Framework allowing for more efficient learning for continuous action spaces.
- Experience replay preventing divergence in the learning process.
- Target Networks enhancing stability of the learning process.
- Off-policy allowing for enhanced sample efficiency by reusing past experiences

This makes the Deep Deterministic Policy Gradient (DDPG) algorithm well suited to high dimensional, continuous (both state and action space) control tasks. However, DDPG tends to suffer from overestimation bias and instability in value function learning.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Figure 3.8: The DDPG algorithm taken from [38].

Twin Delayed Deep Deterministic Policy Gradient

The Twin Delayed Deep Deterministic Policy Gradient (TD3), presented in [40], solves the overestimation bias and instability in value function learning encountered in the DDPG algorithm. It does this through the use of 3 key improvements:

- **Twin Q networks**: maintains two Q networks $Q_{\theta 1}$ and $Q_{\theta 2}$ and uses the minimum value for updating the target Q function to reduce the overestimation bias:

$$Q_{target} = R(s_t, a_t) + \gamma \min(Q_{\theta 1}(s_{t+1}, a_{t+1}), Q_{\theta 2}(s_{t+1}, a_{t+1})) \tag{3.9}$$

- **Delayed policy updates**: updates the actor (policy) less frequently than the critic ensuring the actor learns from more accurate Q-values reducing variance.
- **Target policy smoothing**: adds noise to the target actions used for updating the target network resulting in smoother target network updates and more robust policies.

These improvements result in TD3 being more robust and stable than DDPG for high-dimensional, continuous control tasks [40].

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

Figure 3.9: The Twin Delayed DDPG algorithm taken from [40].

Soft Actor Critic

The Soft Actor Critic (SAC) algorithm proposed in [41] further improves upon the TD3 algorithm by introducing entropy regularization to further encourage exploration and improve policy robustness. SAC learns a stochastic policy (unlike DDPG and TD3) balancing exploration and exploitation through an entropy objective:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))] \tag{3.10}$$

Where H is the entropy term and α the temperature parameter determining the relative importance of the entropy term (exploitation vs exploration balance) [41].

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Figure 3.10: The Soft Actor Critic algorithm taken from [41]

Imitation Learning

Imitation Learning is an approach where an agent learns by expert demonstrations provided to it. Many categories of imitation learning exist but the two main ones are:

- **Behavioural Cloning (BC)**: a supervised learning approach where an agent mimics data produced by an expert demonstrator [42].
- **Inverse Reinforcement Learning (IRL)**: instead of directly imitating actions, an agent attempts to learn the underlying reward function from expert demonstrations, then optimises its actions based on the learned reward function [43].

While both have their uses, BC is an interesting option for safety-critical maneuvering tasks for which there is an abundance of expert data available as is the case for dock-to-dock maneuvers. A generalisable framework for implementing BC algorithms is presented in [42]. BC can also be used as the first phase in a two phase learning process where one first learns an approximate policy-based on demonstrations after which this policy can be further optimised through exploration. Such an approach can significantly reduce the search space of the RL problem. Additionally, [44] proposes a hybrid SAC-BC approach where the SAC algorithm is used to enhance exploration capabilities, something which BC lacks. This allows for learning from demonstrations while still allowing for exploration enhancing generalisability to unseen tasks.

3.5. Summary

This literature study highlights the evolution of motion control strategies for surface vessels, from industry-standard PID controllers to more advanced model-based and learning-based approaches. While conventional methods have been extensively studied and applied for isolated tasks such as position-keeping or trajectory-following, they struggle in handling the full dock-to-dock mission, where smooth transitions and adaptability to complex environments are required. A clear research gap was identified in motion control systems that integrate trajectory generation, controller transitions, and adaptability to environmental disturbances.

Model-based approaches such as MPC address parts of this gap by combining trajectory optimisation with lower level control, but their dependence on accurate models and high computational cost limit real-life applicability. Data-driven methods, particularly RL, provide a promising alternative by reducing reliance on accurate models, improving generalisability, and enabling exploration of novel strategies. However, concerns remain around safety, interpretability, and limited real-life validation. This study lays the groundwork for the thesis, where RL shall be leveraged to address the core challenges of generalisable and robust dock-to-dock maneuvering.

The main research gaps found in this literature study are:

- Lack of motion control strategies addressing the full dock-to-dock envelope, including controller transitions.
- Limited research on incorporating environmental disturbances into docking trajectories.
- Insufficient focus on safety and interpretability of RL-based methods for autonomous docking.
- Limited real-life validation of results.

To conclude this chapter, the research questions shall be revisited to identify which question were partially or fully answered (note that question 5 is omitted as this will be answered during the validation phase).

1. What is the state-of-the-art in vessel maneuvering?

   This literature study highlights the evolution of motion control strategies for surface vessels, including PID, Dynamic Positioning, Model Predictive Control and Reinforcement Learning establishing both the industry standards and the current areas of research.

2. What requirements must be met for a dock-to-dock mission to be classified as successful?

   Many safety and performance related criteria were identified in existing research and collected throughout to be used for evaluation later on. A clear distinction between sailing and docking metrics was also made.

3. What are the key components needed for the design of a benchmark G&C system for a dock-to-dock maneuver?

   The main modules of traditional G&C systems were identified in subsection 3.1.1 providing a solid foundation on which to base the G&C design. A clear distinction between industry standard control strategies for high speed sailing and low speed, precise maneuvering is also made.

4. How can advanced G&C methods be leveraged in the hierarchical control structure to improve upon the performance of the benchmark?

   A detailed investigation of advanced model and learning based approaches was performed in section 3.3 and key benefits over traditional methods are listed helping to understand which methods should be used. It is also shown in related work that RL can suitably solve similar high level, integrated decision making problems.

4

Supporting Results

This chapter presents results and analyses that further support the main findings in the article. These results provide deeper insight into the modeling process, controller tuning, and disturbance rejection capabilities, and they discuss possible extensions of the RL controller.

4.1. Modeling

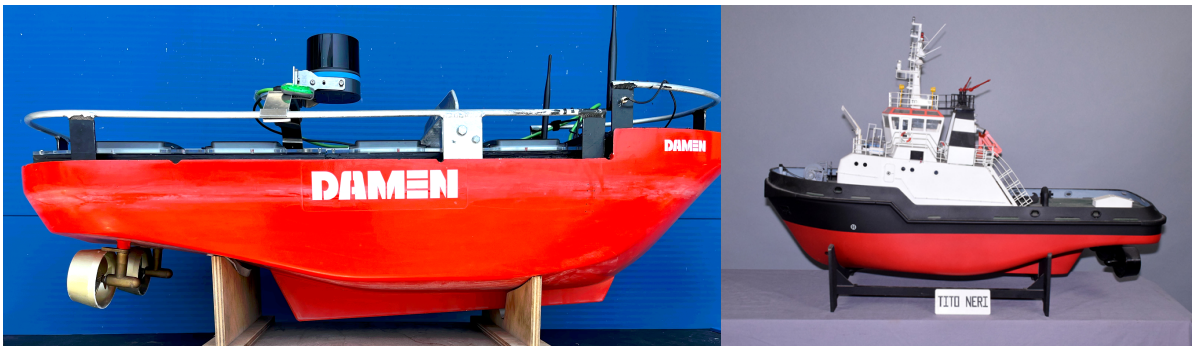
This section details the modeling of DAVE, focusing on the coefficient scaling procedure and extraction of some key characteristics.

4.1.1. Model Scaling

To obtain a model for DAVE, a validated maneuvering model developed for a similar scale vessel, TitoNeri [45], is adopted by scaling its coefficients. Figure 4.1 shows that, although dimensions between DAVE and TitoNeri differ significantly (see Table 4.1), the hull shape, and thus, hull dynamics are very similar further motivating scaling TitoNeri's model to DAVE's dimensions.

Table 4.1: A comparison of DAVE vs TitoNeri non-dimensionalisation parameters. TitoNeri parameters are taken from [45].

	Symbol	DAVE	TitoNeri	Unit
Length	L	1.19	0.97	m
Moment of Inertia	I_z	4.76	0.51	$\mathrm{kg} \cdot \mathrm{m}^2$
Mass	m	39.3	16.9	kg



(a) The Damen Autonomous Vessel (DAVE). **(b)** The TitoNeri scale model vessel [45].

Figure 4.1: A comparison of DAVE and TitoNeri model scale vessels.

Scaling TitoNeri's coefficients to DAVE's dimensions involves non-dimensionalising them and then re-scaling them to DAVE's dimensions. The BIS normalisation technique was selected for this purpose

(p.149, [11]) as it removes any division by velocity which is imperative for low speed docking applications. The non-dimensionalisation procedure is described below using a single coefficient ($X_{\dot{u}}$):

1. For each model coefficient, write out equation linking it to force:

$$F_x[N = kg \cdot \frac{m}{s^2}] = X_{\dot{u}}[?] \cdot \dot{u}[\frac{m}{s^2}] \tag{4.1}$$

2. Rearrange equation to solve for unknown unit:

$$\frac{F_x[N = kg\frac{m}{s^2}]}{\dot{u}[\frac{m}{s^2}]} = X_{\dot{u}}[kg] \tag{4.2}$$

3. Using Table 7.2 from [11], lookup associated BIS normlisation variable. E.g. for mass [kg]: $\mu \cdot \rho \cdot \nabla$ where μ is the body mass density ratio (=1 for floating ships) and ∇ is the hull contour displacement.
4. Simplify normalisation variable using equation $\mu = \frac{m}{\rho\nabla}$ from [11]:

$$\mu = 1 \rightarrow \rho\nabla = m \tag{4.3}$$

5. Non-dimensionalise coefficient:

$$X'_{\dot{u}} = \frac{X_{\dot{u}}}{m} \tag{4.4}$$

The above process can be repeated for all the coefficients present in the maneuvering model resulting in the normalisation variables in Table 4.2[1] which are subsequently used to rescale to DAVE's dimensions.

Table 4.2: Normalisation coefficients for TitoNeri Maneuvering Model.

Coefficient	Non-Dimensionalisation variable
$X_{\dot{u}}$	m
$Y_{\dot{v}}$	m
$Y_{\dot{r}}$	$m \cdot L$
$N_{\dot{v}}$	$m \cdot L$
$N_{\dot{r}}$	$m \cdot L^2$
Linear Coefficients	$\frac{m}{\sqrt{\frac{L}{g}}}$
Quadratic Coefficients	$\frac{m}{L}$
Cubic Coefficients	$\frac{m}{L\sqrt{Lg}}$

4.1.2. Model Characteristics

The vessel model, in conjunction with the thruster model presented in the article can now be used to make some preliminary performance assessments.

Maximum surge speed

The maximum forward (surge) speed u_{max} of DAVE is found by solving the X force balance (drag vs thrust) while setting the propeller speed to max rpm resulting in a max surge speed of 0.664 $\mathrm{m/s}$:

$$T_{\mathrm{ps}}(n_{\max}, \alpha = 0) + T_{\mathrm{sb}}(n_{\max}, \alpha = 0) - \left(p_0\, u + p_2\, u^2 + p_5\, u^3\right) = 0 \tag{4.5}$$

Maximum steady state yaw rate

Due to their coupling (high yaw rate mean lower surge speed) it is assumed the maximum steady state yaw rate r_{max} must ensure $u > 80\% u_{max}$ as found above. The thruster angle (resolution of $0.5°$) is swept across all possible combinations (with max propeller speed) and the resulting steady state velocities are recorded. The thruster angle leading to the largest steady-state yaw rate while remaining

[1] Note that coefficients with same units are only listed once

above 80% of the maximum surge speed was found to be $-15.556°$ resulting in a steady-state yaw rate of 0.162 $\mathrm{rad/s}$

Thruster force envelope
Because the thruster forces are not exerted at the c.o.g of the vessel, thruster outputs along one axis can induce unwanted motions in others (induced moment when outputting a lateral thrust force) thus, part of the available thrust must be allocated to counter these effects. A thruster force envelope is obtained by numerically sweeping thruster angles/propeller speeds and recording all achievable force combinations, allowing infeasible force commands to be clamped before passing them to the thrust allocation module. The thruster envelope for a max propeller speed of 600 rpm is shown in Figure 4.2.

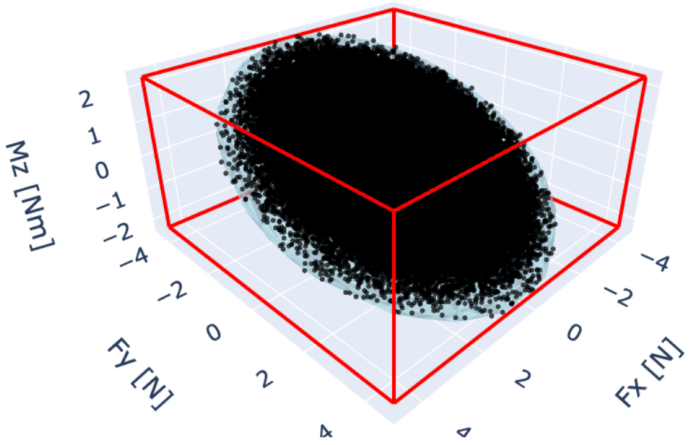


Figure 4.2: The thruster envelope for DAVE where red bounds denote the maximum theoretical force limits, the black points are the sampled wrench forces, and the blue volume is the feasible force envelope.

4.2. Benchmark Controller Design

This section provides a more detailed overview of the benchmark controller tuning process presenting the step response and associated metrics for each PID sub-controller. It begins with the Sailing controller and ends with the Dynamic Positioning controller.

4.2.1. Sailing Controller

The sailing controller is made up of two independent PID controllers; speed and heading. This section begins with an overview of the speed controller tuning, followed by the heading controller and finishing with the Line-of-Sight guidance system design.

Speed controller

The speed controller ensures the actuators output the correct rpm to achieve a certain speed as shown in Figure 4.5. Such a controller can be achieved in a variety of ways hence, the objective for the benchmark implementation is to achieve satisfactory response characteristics while keeping the design as simple as possible. This leads us to 3 proposed controllers:

- Pure feedforward (FF) based on the required propeller speed to overcome the drag at a certain velocity setpoint.
- Pure feedback (FB) through the use of a PID controller.
- A combination of a feedforward and feedback term.

The pure feedforward controller is ruled out due to its high model dependency, sluggish transient response, and inability to reject disturbances. Pure feedback (PID) solves these issues but, due to its linear formulation, requires gain scheduling at higher speeds where hydrodynamic forces become nonlinear. Combining both terms allows for a single set of gains across the entire speed regime while maintaining a similar response to the PID controller, as shown in Figure 4.3 and Table 4.3. At lower

speeds, the additional feedforward term introduces some overshoot (≈5%), which becomes negligible above $0.2\mathrm{m/s}$. To further improve robustness, a simple integrator anti-windup scheme is included: the integral term is only updated when the actuators are unsaturated, preventing the excessive error accumulation otherwise caused by large setpoint changes [46]. As shown in Figure 4.4, this ensures stable behaviour across the full speed range.

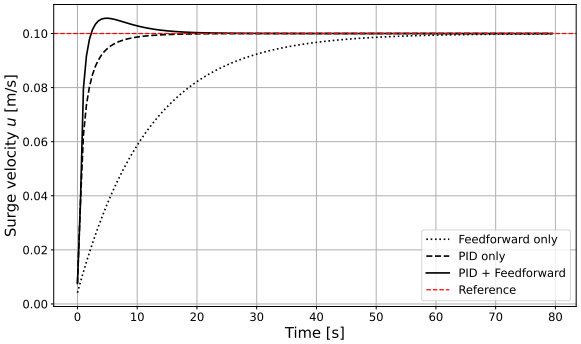


Figure 4.3: Setpoint regulation response under ideal conditions for different speed controllers.

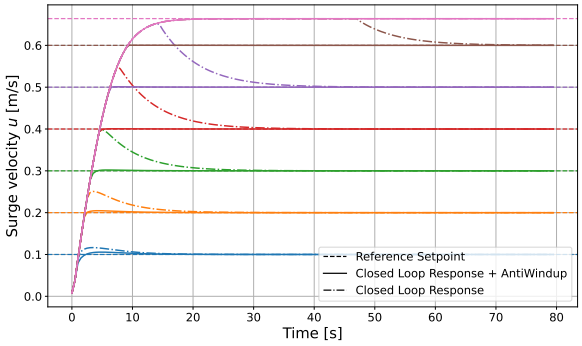


Figure 4.4: Speed Controller response to different speed setpoints emphasising the role of the integral anti-windup scheme.

Metric	FF	PID	PID + FF
Rise time [s]	26.0	3.1	1.1
Settling time 2% [s]	46.2	8.5	9.2
Overshoot [%]	0.0	0.0	5.4
Steady-State error [$\frac{m}{s}$]	0.0	0.0	0.0

Table 4.3: Performance metrics based on setpoint regulation in Figure 4.3 for different speed controllers.

The resulting control law is shown in Equation 4.6 and Figure 4.5 where n is the propeller speed in rpm, $[K_p, K_i, K_d]$ are the PID gains, $e(t)$ is the heading/surge speed error in the body frame, D_x is the steady state drag at the desired surge speed, u_d, and T_{nn} the thruster coefficient relating thrust to rpm.

$$n = n_{FF} + K_p \cdot e_u(t) + K_i \cdot \int_0^t e_u(t)dt - K_d \cdot \dot{e_u}(t) \tag{4.6}$$

$$n_{FF} = \sqrt{\frac{D_x}{T_{nn} \cdot n_{thrusters}}} \tag{4.7}$$

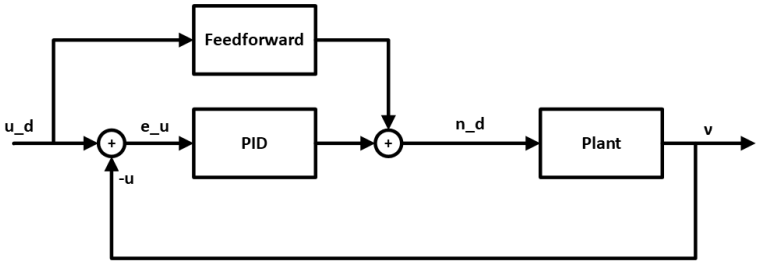


Figure 4.5: Block diagram for the velocity controller

Heading Controller

The heading controller ensures the actuators output the correct thruster deflection (azimuth) to achieve a certain change in heading angle (see Figure 4.8). Unlike the speed loop, it includes an integrator to convert yaw rate into heading, making the open-loop dynamics type 1 rather than type 0 [47]. Consequently, under ideal conditions the closed loop exhibits no steady-state error, and a PD design (Equation 4.8 with $K_i = 0$) yields the ideal response in Figure 4.6. Controller behaviour varies with propeller rpm since higher rpm increases available moment, but responses remain nearly identical above 200 rpm (Figure 4.7), where it is assumed the sailing controller will nearly always operate. Under environmental disturbances, however, steady-state offsets arise, requiring an integral term. To prevent integral windup, the controller updates the integrator only when actuators are unsaturated and the heading error is within $30°$, reflecting the maximum steady-state error observed under disturbance.

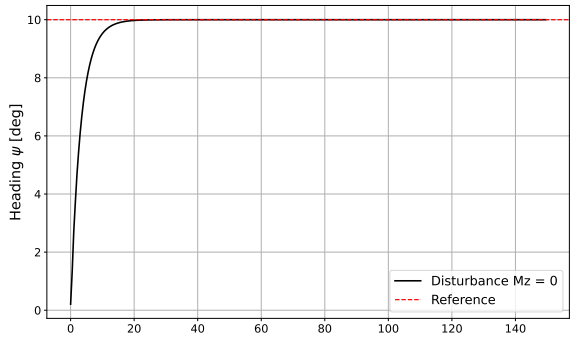


Figure 4.6: Step response of heading controller under ideal conditions.

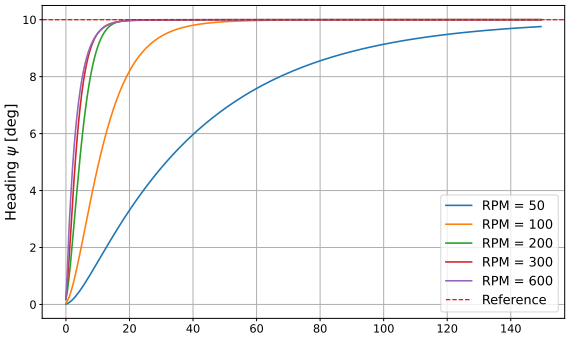


Figure 4.7: Heading step response under different rpm values.

The result is the control law shown in Equation 4.8 and Figure 4.8.

$$\alpha = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)dt + K_d \cdot \dot{e} \tag{4.8}$$

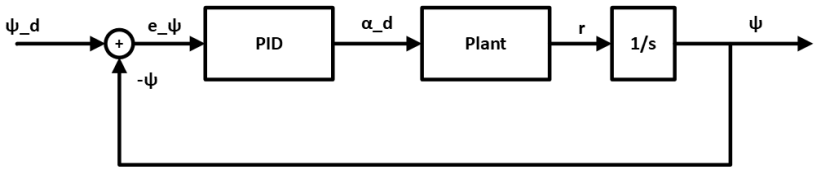


Figure 4.8: Block Diagram for the heading controller.

Line-of-Sight Guidance

The guidance module's purpose is to 'compute meaningful command signals to a vehicle control system such that the vehicle is able to achieve a given motion control objective' [5]. The Line-of-Sight guidance principle has a single tunable parameter: the lookahead distance influencing how far ahead on the trajectory convergence will occur. It should be maximised to avoid oscillatory/unstable behaviour while being small enough to prevent excessive corner cutting. Figure 4.9 shows that distances $\geq 3\text{m}$ result in excessive corner cutting while small distances such as 0.1m lead to oscillatory behaviour. A lookahead distance of $d_{LOS} = 2\text{m}$ is chosen. A final disturbance rejection analysis of the sailing controller with LOS guidance is performed for wind disturbance in Figure 4.10.

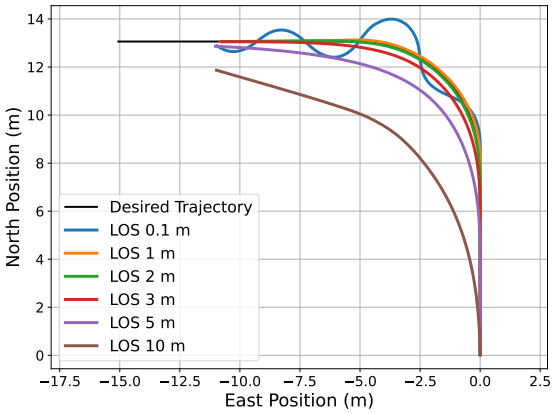


Figure 4.9: The effect of the lookahead distance on the sailing controller. Trajectory begins bottom right, ends top left.

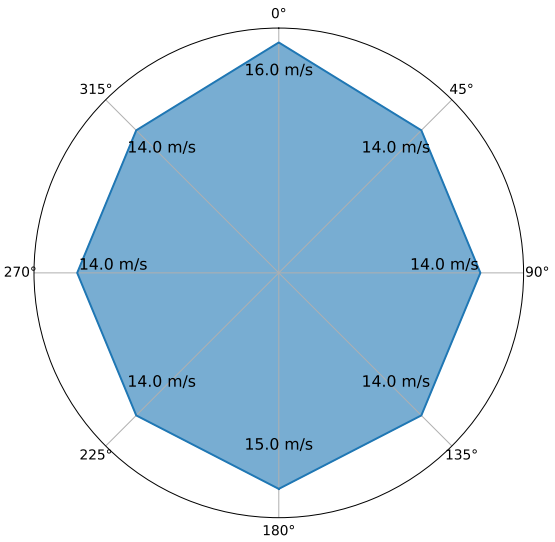


Figure 4.10: Disturbance rejection capabilities of the sailing controller for wind disturbance where a mission is successful if controller arrives within $1\mathrm{m}$ of desired pose.

4.2.2. Dynamic Positioning Controller

Originally designed for position keeping tasks under environmental disturbances, the Dynamic Positioning (DP) controller makes use of 3 PID controllers for controlling motion along the 3 degrees of freedom, taking in a position/heading error and outputting a desired force vector to maintain the desired pose (see Figure 4.11). This desired force vector is then passed on to a thrust allocation algorithm which finds the most optimal actuator commands to achieve the desired force vector. The PID based DP controller has the following control law where $\tau_{FF_{wind}}$ is the feedforward control action to counteract the wind force (using a wind model) and $e(t)$ is the position error vector $[e_x, e_y, e_\psi]^T$ in the vessel's body frame:

$$\tau = \tau_{FF_{wind}} + K_p \cdot e + K_i \cdot \int_0^t e(t)dt + K_d \cdot \dot{e} \tag{4.9}$$

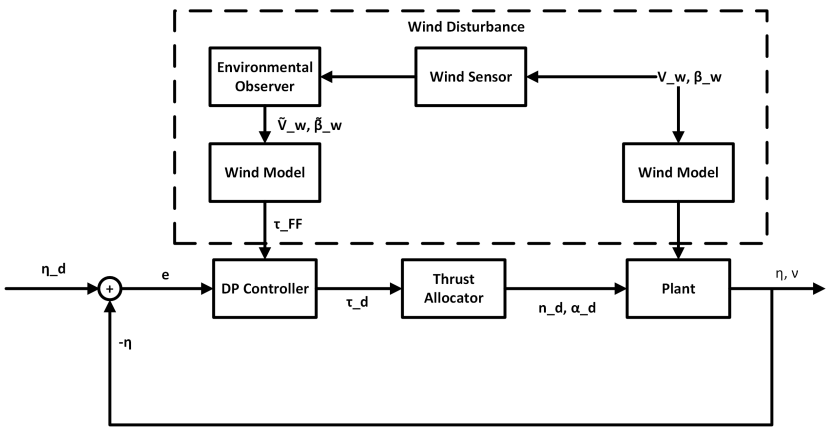


Figure 4.11: Block Diagram for the Dynamic Positioning control loop.

For tuning, the surge, heading, and sway controllers are adjusted sequentially. This order is important since pure sway motion cannot be produced without inducing a moment; tuning heading before sway allows counteraction of this effect. To showcase the improved position response due to the inclusion

of a feedforward wind term, the DP controller is ran three times while subject to the max allowable force in each direction ($F_x = -4.0, F_y = -0.8$ and $Mz = -0.4$) [2] in Figure 4.12. The feedforward term significantly improves the x- and y-position responses, while causing a slight deterioration in the heading response (15% overshoot). However, the substantial gains in surge and sway performance justify this minor reduction in heading accuracy while this can further be mitigated by reducing the integral gain of the heading controller trading off slower disturbance rejection for less overshoot. A final capability plot showcasing the wind rejection capabilities of the final DP controller is shown in Figure 4.13 where limited sway disturbance rejection is due to the thruster layout inducing unwanted moments (some thrust allocation must go to counter this) and from the higher hydrodynamic drag in the sway direction, both of which reduce effective thrust.

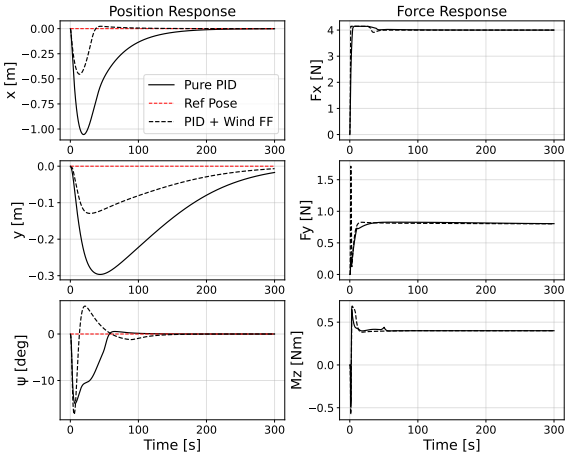


Figure 4.12: DP controller position and force response when subject to maximum force disturbance ($F_x = -4.0, F_y = -0.8$ and $Mz = -0.4$) in each DOF.

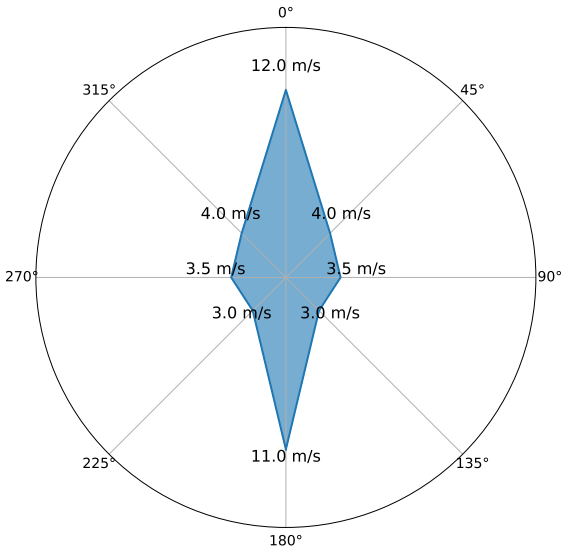


Figure 4.13: Capability plot of the DP controller w.r.t wind disturbance. A run is considered successful if it resists the disturbance force while returning to within 0.1 meters and 5 degrees (heading) of the desired position after $t_{max} = 500$s.

Low Speed Tracking

The main issue for low speed tracking is that the controller acts on position errors, so the distance to the reference point directly affects behaviour. If too large, it causes discrete jumps in the commanded forces and undesirable responses (see Figure 4.14). To counter this, it is common practice [11] to pass the reference waypoint through a low pass filter, ensuring the desired pose moves smoothly towards the target. The third-order low pass filter used is taken from [48] where n_{d_i} is the output from the low pass filter (the dynamic waypoint), r_i is the desired final pose (static waypoint), ω the frequency of the low pass filter and ζ the damping ratio:

[2] These values are based on the max allowable thruster force in each DOF including a safety margin.

$$\frac{n_{d_i}}{r_i^n(s)} = \frac{\omega_{n_i}^3}{s^3 + (2\zeta+1)\omega_{n_i}s^2 + (2\zeta+1)\omega_{n_i}^2 s + \omega_{n_i}^3} \tag{4.10}$$

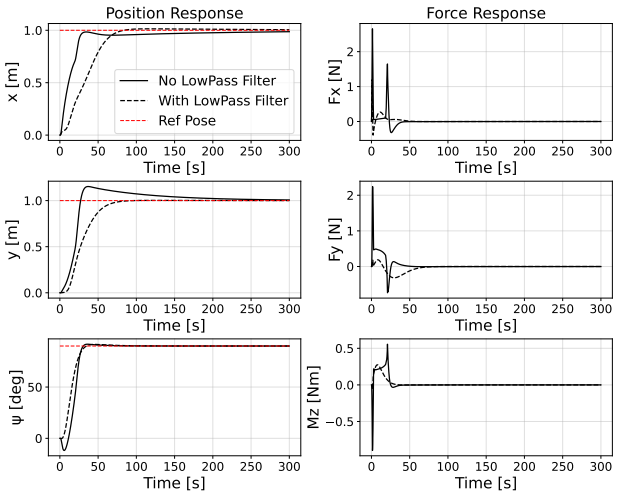


Figure 4.14: DP controller position and force response for low speed tracking emphasising the role of the low-pass filter.

Thrust Allocation

Although outside the scope of this study, a brief overview on the thrust allocation algorithm used is given below (this algorithm was provided by Damen). The thrust allocation algorithm serves the purpose of converting desired force commands to actuator commands. The underactuated nature of the system (2 thrusters for 3 dof's) complicates the thrust allocation as a single thruster must be able to output forces along different d.o.f.'s. The problem is nontrivial meaning that a single desired force can be executed by multiple different actuator commands effectively turning the problem into an optimisation problem where on must find the best thruster states to achieve a desired force. A high level overview of the thrust allocation problem is presented below.

Goal: Convert a desired body-frame force vector $[F_x, F_y, M_z]$ into 4 actuator states $[n_{ps}, \alpha_{ps}, n_{sb}, \alpha_{sb}]$.

1. Formulate 3 equilibrium constraints:

$$X_{\mathrm{ps}} + X_{\mathrm{sb}} = F_x \tag{4.11}$$

$$Y_{\mathrm{ps}} + Y_{\mathrm{sb}} = F_y \tag{4.12}$$

$$l_{x,\mathrm{ps}}\, Y_{\mathrm{ps}} - l_{y,\mathrm{ps}}\, X_{\mathrm{ps}} + l_{x,\mathrm{sb}}\, Y_{\mathrm{sb}} - l_{y,\mathrm{sb}}\, X_{\mathrm{sb}} = M_z \tag{4.13}$$

2. Generate additional constraint to have a unique (4 equations, 4 unkowns) solution (analytical or optimisation based conditions). Examples of possible choices include:

$$Y_{\mathrm{ps}} = Y_{\mathrm{sb}} \quad \text{OR} \quad X_{\mathrm{ps}} = X_{\mathrm{sb}} \qquad \text{(symmetric allocation)} \tag{4.14}$$

$$Y_{\mathrm{ps}} - Y_{\mathrm{sb}} = 2Y_{\mathrm{bias}} \qquad \text{(outward bias to reduce propeller wake interaction)} \tag{4.15}$$

$$\underset{X_{\mathrm{ps}}, Y_{\mathrm{ps}}, X_{\mathrm{sb}}, Y_{\mathrm{sb}}}{\text{minimize}} \; w_{\mathrm{ps}}\left(X_{\mathrm{ps}}^2 + Y_{\mathrm{ps}}^2\right) + w_{\mathrm{sb}}\left(X_{\mathrm{sb}}^2 + Y_{\mathrm{sb}}^2\right) \qquad \text{(energy)} \tag{4.16}$$

3. Solve for thruster forces $[X_{ps}, Y_{ps}, X_{sb}, Y_{sb}]$ analytically (analytical constraint) or through constrained minimisation (optimisation based constraint).

4. Map resulting thruster force vectors $[X, Y]$ into actuator commands $[n, \alpha]$ using thruster model.

$$T = \sqrt{X^2 + Y^2} \tag{4.17}$$

$$\alpha = atan2(Y, X) \tag{4.18}$$

$$n = \sqrt{\frac{T}{T_{nn}}} \tag{4.19}$$

Where T_{nn} is the thrust coefficient of the propeller.

5. Apply actuator limits, re-allocate if necessary.

The thrust allocation module in this study makes use of the outward bias condition to avoid significant drops in propeller efficiency due to wake interactions from the other propeller.

4.3. RL Controller: Extending to Multiple Objects

A quick note is made here to further motivate the use of lookahead points as observation states, first proposed in [49], even though they were shown to serve a limited purpose for the docking problem with a single object. When only the dock is present, the task can be solved by directly providing its position and size. However, this approach becomes unfeasible when a large, unknown number of objects must be avoided, since the observation vector must remain fixed. Lookahead points, by returning the distance to an object in a specific direction, elegantly address this limitation as they generalise to any number and size of obstacles. This is illustrated in Figure 4.15, where the agent, using only lookahead point information, learns to navigate through a complex geometry with randomly sized obstacles. Less attention was placed on precise alignment to the final pose (red arrow), explaining the suboptimal end alignment observed, as the focus of this study was specifically on obstacle avoidance capabilities.

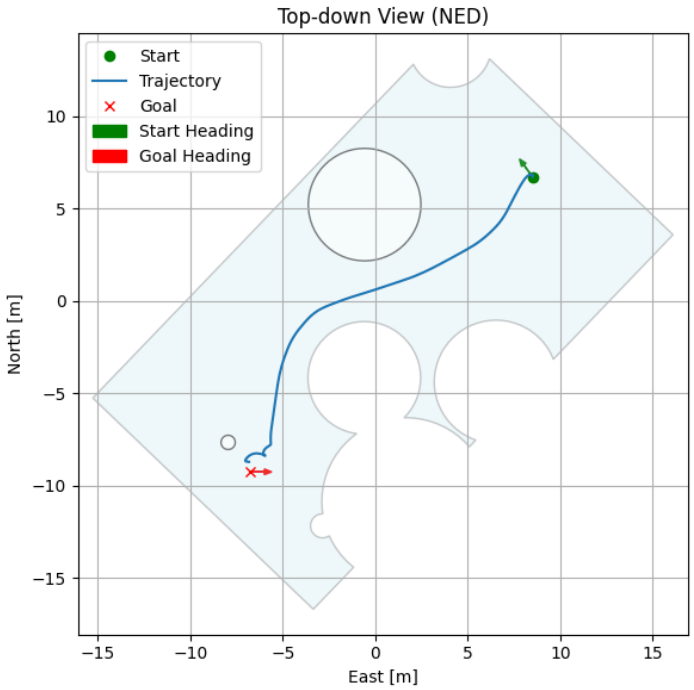


Figure 4.15: Learnt control policy focusing on the avoidance of a random number of objects with random size. The green arrow is the start, red is the end, each circle is an object. Note that this figure is taken from a **previously graded assignment in AE4350** and serves only as an illustrative example to support the claims made in this report.

5

Conclusion

The current shift toward autonomous systems in the maritime industry drives the need for vessels capable of operating fully autonomously over a wide range of scenarios. Although individual tasks such as position keeping have been solved, there remains little research on motion control systems capable of a full dock-to-dock mission by integrating trajectory generation, controller transitions and adaptability to environmental disturbances. This study aimed to address this by establishing a benchmark G&C system for an autonomous dock-to-dock mission, and subsequently and improving its adaptability and performance in geometrically constrained environments under environmental disturbances by employing reinforcement learning. With this objective in mind, five research questions were formulated to break the problem down into manageable modules.

1 What is the state-of-the-art in vessel motion control?

   The literature review in chapter 3 showed that PID controllers remain the industry standard due to their reliability and simplicity, while model-based approaches such as MPC and MPPI represent the current state-of-the-art in research for autonomous docking because of their ability to handle constraints and optimize trajectories. Given the limitations of model-based approaches, reinforcement learning methods are emerging as a promising alternative, offering improved adaptability and disturbance rejection while avoiding explicit problem formulations, though their black-box nature and limited safety guarantees still prevent widespread industry adoption.

2 What requirements must be met for a dock-to-dock mission to be classified as successful?

   A docking attempt was classified as successful if the vessel reaches and maintains contact with the dock while keeping its heading aligned and velocity within safe limits. To enable quantitative comparison across controllers, additional safety, performance, and energy metrics were defined and applied.

3 What are the key components needed for the design of a benchmark, PID-based G&C system for a dock-to-dock maneuver?

   The benchmark G&C system was designed using industry-standard methods: a PID-based speed and heading controller with Line-of-Sight guidance for the sailing phase, a modified PID-based dynamic positioning controller for docking and a discrete controller transition algorithm. Simulation results show that the benchmark system can complete the full dock-to-dock mission, but its docking phase is comparatively slow and the lack of built-in trajectory planning limits adaptability to varying geometrical constraints and environmental disturbances.

4 How can advanced G&C methods be leveraged in the hierarchical control structure to improve upon the performance of the benchmark?

Analysis of the benchmark system revealed inefficiencies in discrete controller transitions and the absence of trajectory planning. Reinforcement learning was introduced to address these limitations, offering a flexible framework for learning high-level decisions such as controller transitions. The RL controller reduced docking times (up to twice as fast) and improved wind rejection ($6\,\mathrm{m/s}$ vs. $3.5\,\mathrm{m/s}$), but its black-box nature raised concerns for safety-critical applications. Balancing performance and interpretability, a hybrid system was proposed where the RL controller acts as a planning module while execution remains with the PID-based benchmark. While the RL and hybrid controllers did not fully exploit their capabilities, namely suffering from poor generalisability to varying geometries, their evaluation highlighted key limitations of the benchmark controller and pointed towards promising alternative directions.

5 Is a simplified simulation model for the Damen Autonomous Vessel adequate to allow for real-life transferability?

The benchmark system was validated in a real-life setting using the Damen Autonomous Vessel to assess the simulation-to-real gap. A comparison of simulated and real benchmark controller performance revealed similar closed-loop behaviour, with docking times within $15\mathrm{s}$ of each other ($49\mathrm{s}$ vs. $35\mathrm{s}$). Due to the closed-loop nature of the validation, explicit statements about full model fidelity are omitted. Nevertheless, the results demonstrate that the simulation model is sufficiently accurate to support controller design with no modifications required for transfer.

Taken together, this study demonstrates while reinforcement learning significantly improved performance and disturbance rejection capabilities, its current limitations in generalising safely to unseen docking scenarios and its black-box nature remain barriers to deployment in maritime settings. Hybrid approaches offer a promising compromise, with end-to-end RL more justified in fast, dynamic domains such as aerial robotics, whereas in slower, safety-critical maritime settings where human oversight remains, interpretable and reliable conventional methods remain essential.

6

Future Work

While this study provided new insights into autonomous dock-to-dock control, several avenues remain for further research. The recommendations below stem directly from the limitations and findings discussed in the results and conclusion.

Methodological improvements:

- Improved vessel modeling: Extend the current point-mass representation by modeling the vessel as a bounding box, providing more realistic docking success conditions and closer alignment with physical behaviour.
- Geometry generalisation: Enhance RL controller performance on random dock sizes by incorporating and carefully weighting a continuous safety reward, encouraging better use of lookahead points as proposed in [49].
- Multi-object scenarios: If lookahead point utilisation is improved through better reward shaping, multi-object collision avoidance could be introduced by including multiple obstacles during training.
- Hybrid system improvement: Since RL struggled with one-shot tasks such as placing switching points, a simpler, rule-based algorithm should be investigated to improve the robustness and reliability of the hybrid system.

Experimental validation:

- Comprehensive real-life validation: The results presented in this thesis are primarily simulation-based, with only an initial real-life validation of the benchmark controller due to hardware constraints. Future work should therefore test both the RL and hybrid controllers on the Damen Autonomous Vessel to confirm simulation findings and assess transferability under real-world conditions such as sensor noise, varying environmental disturbances, and dock geometries.

Addressing these directions would strengthen the reliability and generalisability of advanced G&C methods for autonomous docking, while more comprehensive real-life validation would provide stronger support for the simulation-based results.

References

[1] A. Azarko and M. Molica Colella, "The rising tide of the autonomous ships market," *Open Access Government*, vol. 41, pp. 436–437, 2024. DOI: 10.56367/oag-041-10323.

[2] C. Bagoulla and P. Guillotreau, "Shortage and labor productivity on the global seafaring market," in *Seafarers : an international labour market in perspective*, P. Chaumette, Ed., Gomylex, 2016, pp. 15–27.

[3] S. Thombre, Z. Zhao, H. Ramm-Schmidt, *et al.*, "Sensors and ai techniques for situational awareness in autonomous ships: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 64–83, 2022. DOI: 10.1109/TITS.2020.3023957.

[4] N. Minorsky, "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, 1922. DOI: 10.1111/j.1559-3584.1922.tb04958.x.

[5] M. Breivik, "Topics in guided motion control of marine vehicles," PhD thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, 2010.

[6] S. J. N. Lexau, M. Breivik, and A. M. Lekkas, "Automated docking for marine surface vessels—a survey," *IEEE Access*, vol. 11, pp. 132 324–132 367, 2023. DOI: 10.1109/access.2023.3335912.

[7] J. E. Walmsness, H. H. Helgesen, S. Larsen, G. K. M. Kufoalor, and T. A. Johansen, "Automatic dock-to-dock control system for surface vessels using bumpless transfer," *Ocean Engineering*, vol. 268, p. 113 425, 2023. DOI: 10.1016/j.oceaneng.2022.113425.

[8] M. L. Darby and M. Nikolaou, "Mpc: Current practice and challenges," *Control Engineering Practice*, vol. 20, no. 4, pp. 328–342, 2011. DOI: 10.1016/j.conengprac.2011.12.004.

[9] M. Breivik, "Marine craft: 21st century motion control concepts," *Sea Technology*, vol. 47, no. 3, pp. 33–36, 2006.

[10] L. Zhao, F. Wang, and Y. Bai, "Route planning for autonomous vessels based on improved artificial fish swarm algorithm," *Ships and Offshore Structures*, vol. 18, pp. 1–10, Jun. 2022. DOI: 10.1080/17445302.2022.2081423.

[11] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: Wiley, 2011. DOI: 10.1002/9781119994138.

[12] L. Zhang, X. Peng, N. Wei, Z. Liu, C. Liu, and F. Wang, "A thrust allocation method for dp vessels equipped with rudders," *Ocean Engineering*, vol. 285, p. 115 342, 2023. DOI: 10.1016/j.oceaneng.2023.115342.

[13] C. de Wit, "Optimal thrust allocation methods for dynamic positioning of ships," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2009.

[14] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *IFAC Proceedings Volumes*, vol. 36, no. 21, pp. 211–216, 2003. DOI: 10.1016/s1474-6670(17)37809-6. [Online]. Available: http://dx.doi.org/10.1016/s1474-6670(17)37809-6.

[15] E. Saggini, E. Zereik, M. Bibuli, G. Bruzzone, M. Caccia, and E. Riccomagno, "Performance indices for evaluation and comparison of unmanned marine vehicles' guidance systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 12 182–12 187, 2014. DOI: 10.3182/20140824-6-ZA-1003.01055.

[16] D. Menges and A. Rasheed, "An environmental disturbance observer framework for autonomous surface vessels," *Ocean Engineering*, vol. 285, p. 115 412, 2023. DOI: 10.1016/j.oceaneng.2023.115412.

[17] G. Juin-Gauthier, A. Babarit, B. Elie, O. Kermorgant, and V. Fremont, "Waves filtering in heading controllers: Impact on the power production of an energy ship," *IFAC-PapersOnLine*, vol. 58, pp. 458–463, 2024. DOI: 10.1016/j.ifacol.2024.10.096.

[18] B. de Kruif, "Autonomous docking of a feeder vessel," *Journal of Marine Engineering & Technology*, Nov. 2023. DOI: 10.1080/20464177.2023.2281742.

[19] W. Cai, M. Zhang, Q. Yang, C. Wang, and J. Shi, "Long-range uwb positioning-based automatic docking trajectory design for unmanned surface vehicle," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–12, 2023. DOI: 10.1109/tim.2023.3271005.

[20] M. Breivik and T. Fossen, "A unified concept for controlling a marine surface vessel through the entire speed envelope," in *Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005.*, 2005, pp. 1518–1523. DOI: 10.1109/.2005.1469807.

[21] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. DOI: 10.1016/0005-1098(89)90002-2.

[22] W. Farag, "Model-predictive-control complex-path tracking for self-driving cars," *International Journal of Modelling, Identification and Control*, vol. 34, p. 265, Jan. 2020. DOI: 10.1504/ijmic.2020.10033906.

[23] G. Williams, A. Aldrich, and E. Theodorou, *Model predictive path integral control using covariance variable importance sampling*, 2015. arXiv: 1509.01149. [Online]. Available: https://arxiv.org/abs/1509.01149.

[24] A. Vijayakumar, A. A, and A. Somayajula, "Model predictive path integral docking of fully actuated surface vessel," in *2025 IEEE Underwater Technology (UT)*, IEEE, 2025, pp. 1–6. DOI: 10.1109/ut61067.2025.10947451.

[25] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, "Trajectory planning and control for automatic docking of asvs with full-scale experiments," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15 337–15 344, 2020. DOI: 10.1016/j.ifacol.2020.12.1451.

[26] L. Busoniu, T. de Bruin, D. Tolic, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018. DOI: 10.1016/j.arcontrol.2018.09.005.

[27] R. Deraj, s. Kumar, S. Alam, and A. Somayajula, "Deep reinforcement learning based controller for ship navigation," *Ocean Engineering*, vol. 273, p. 113 444, 2023. DOI: 10.1016/j.oceaneng.2023.113937.

[28] C. Zhou, Y. Wang, L. Wang, and H. He, "Obstacle avoidance strategy for an autonomous surface vessel based on modified deep deterministic policy gradient," *Ocean Engineering*, vol. 243, p. 110 166, 2022. DOI: 10.1016/j.oceaneng.2021.110166.

[29] Y. Guo, Z. Zhang, B. Zheng, and D. Yanhua, "An autonomous path planning model for unmanned ships based on deep reinforcement learning," *Sensors*, vol. 20, p. 426, 2020. DOI: 10.3390/s20020426.

[30] J. C. Sadlier, "Deep reinforcement learning for the automatic six-degree-of-freedom docking maneuver of space vehicles," M.S. thesis, Delft University of Technology, 2022.

[31] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[32] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431. DOI: 10.23919/ECC.2019.8796030.

[33] W. Cai, A. B. Kordabad, H. N. Esfahani, A. M. Lekkas, and S. Gros, "Mpc-based reinforcement learning for a simplified freight mission of autonomous surface vehicles," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2990–2995. DOI: 10.1109/cdc45484.2021.9683750.

[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

[35] C. J. C. H. Watkins, "Learning from delayed rewards," PhD thesis, University of Cambridge, 1989.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, *Playing atari with deep reinforcement learning*, 2013. arXiv: 1312.5602. [Online]. Available: https://arxiv.org/abs/1312.5602.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, Feb. 2015. DOI: 10.1038/nature14236.

[38] T. Lillicrap, J. Hunt, A. Pritzel, *et al.*, *Continuous control with deep reinforcement learning*. arXiv: 1509.02971. [Online]. Available: https://arxiv.org/abs/1509.02971.

[39] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, vol. 32, PMLR, 2014, pp. 387–395.

[40] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1582–1591.

[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1861–1870.

[42] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*. Oxford University PressOxford, 2000, pp. 103–129. DOI: 10.1093/oso/9780198538677.003.0006.

[43] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103 500, 2021. DOI: 10.1016/j.artint.2021.103500.

[44] A. Geist *et al.*, "Multi-task offline reinforcement learning," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2019.

[45] A. Haseltalab, "Control for autonomous all-electric ships: Integrating maneuvering, energy management, and power generation control," PhD thesis, Delft University of Technology, Delft, The Netherlands, 2019.

[46] B. Srikanth and M. Sankar, "Implementation of anti-windup techniques for the improvement of pid performance," in 2021, pp. 585–593. DOI: 10.1007/978-981-15-8439-8_48.

[47] W. Yan, Y. Yang, and X. Guo, "Analyze the transient response of a control systems," *Advanced Materials Research*, vol. 706-708, pp. 639–643, Jun. 2013. DOI: 10.4028/www.scientific.net/AMR.706-708.639.

[48] S. Larsen, H. Helgesen, J. Walmsness, G. Kufoalor, and T. Johansen, "Automatic docking with extended dynamic positioning," *Journal of Marine Science and Technology*, vol. 29, pp. 770–788, 2024. DOI: 10.1007/s00773-024-01018-y.

[49] K. Li, J. Chen, D. Yu, *et al.*, "Deep reinforcement learning-based obstacle avoidance for robot movement in warehouse environments," in *2024 IEEE 6th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, IEEE, 2024, pp. 342–348. DOI: 10.1109/ICCASIT62299.2024.10828100.

A

Planning

This appendix contains all the planning tools used throughout. It begins with a high level timeline of the thesis, followed by an in depth Gantt (where green line represents midterm or greenlight date) chart and Work Breakdown Structure (where red colored block indicate the task is optional) for both research phases.

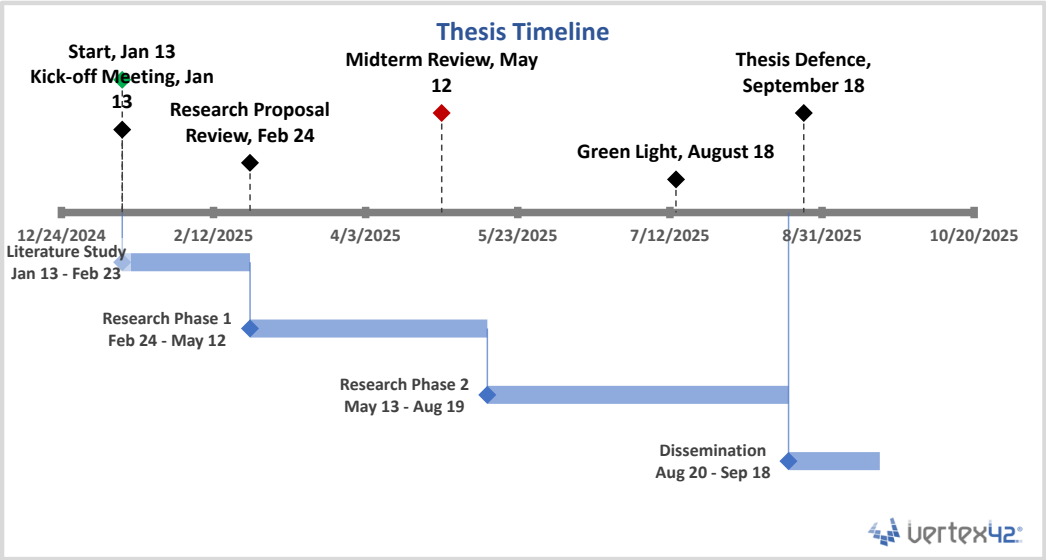
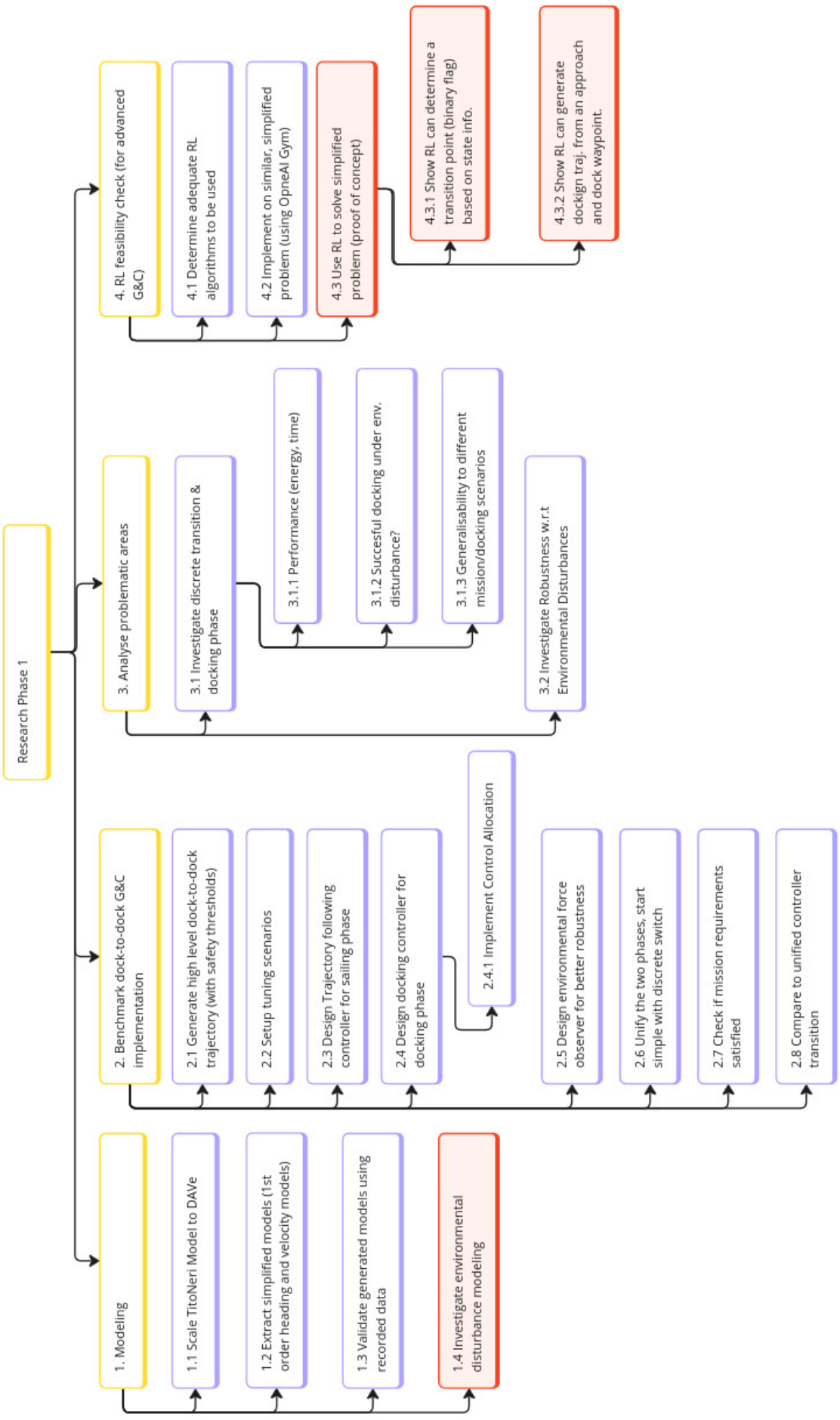


Figure A.1: High level Thesis timeline.

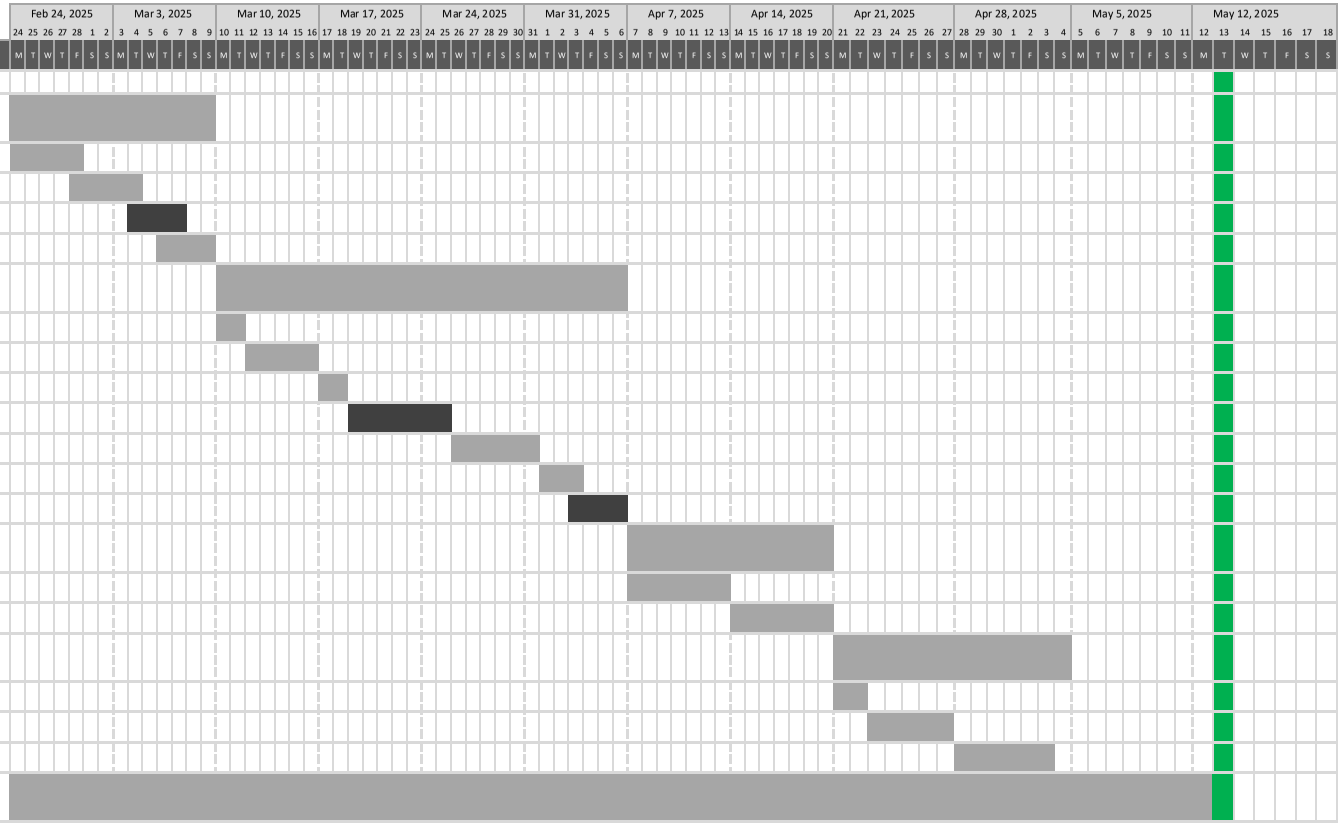

Research Phase 1
1. Modeling
1.1 Scale TitoNeri Model to DAVe
1.2 Extract simplified models (1st order heading and velocity models)
1.3 Validate generated models using recorded data
1.4 Investigate environmental disturbance modeling
2. Benchmark dock-to-dock G&C implementation
2.1 Generate high level dock-to-dock trajectory (with safety thresholds)
2.2 Setup tuning scenarios
2.3 Design Trajectory following controller for sailing phase
2.4 Design docking controller for docking phase
2.4.1 Implement Control Allocation
2.5 Design environmental force observer for better robustness
2.6 Unify the two phases, start simple with discrete switch
2.7 Check if mission requirements satisfied
2.8 Compare to unified controller transition
3. Analyse problematic areas
3.1 Investigate discrete transition & docking phase
3.1.1 Performance (energy, time)
3.1.2 Succesful docking under env. disturbance?
3.1.3 Generalisability to different mission/docking scenarios
3.2 Investigate Robustness w.r.t Environmental Disturbances
4. RL feasibility check (for advanced G&C)
4.1 Determine adequate RL algorithms to be used
4.2 Implement on similar, simplified problem (using OpneAI Gym)
4.3 Use RL to solve simplified problem (proof of concept)
4.3.1 Show RL can determine a transition point (binary flag) based on state info.
4.3.2 Show RL can generate dockign traj. from an approach and dock waypoint.

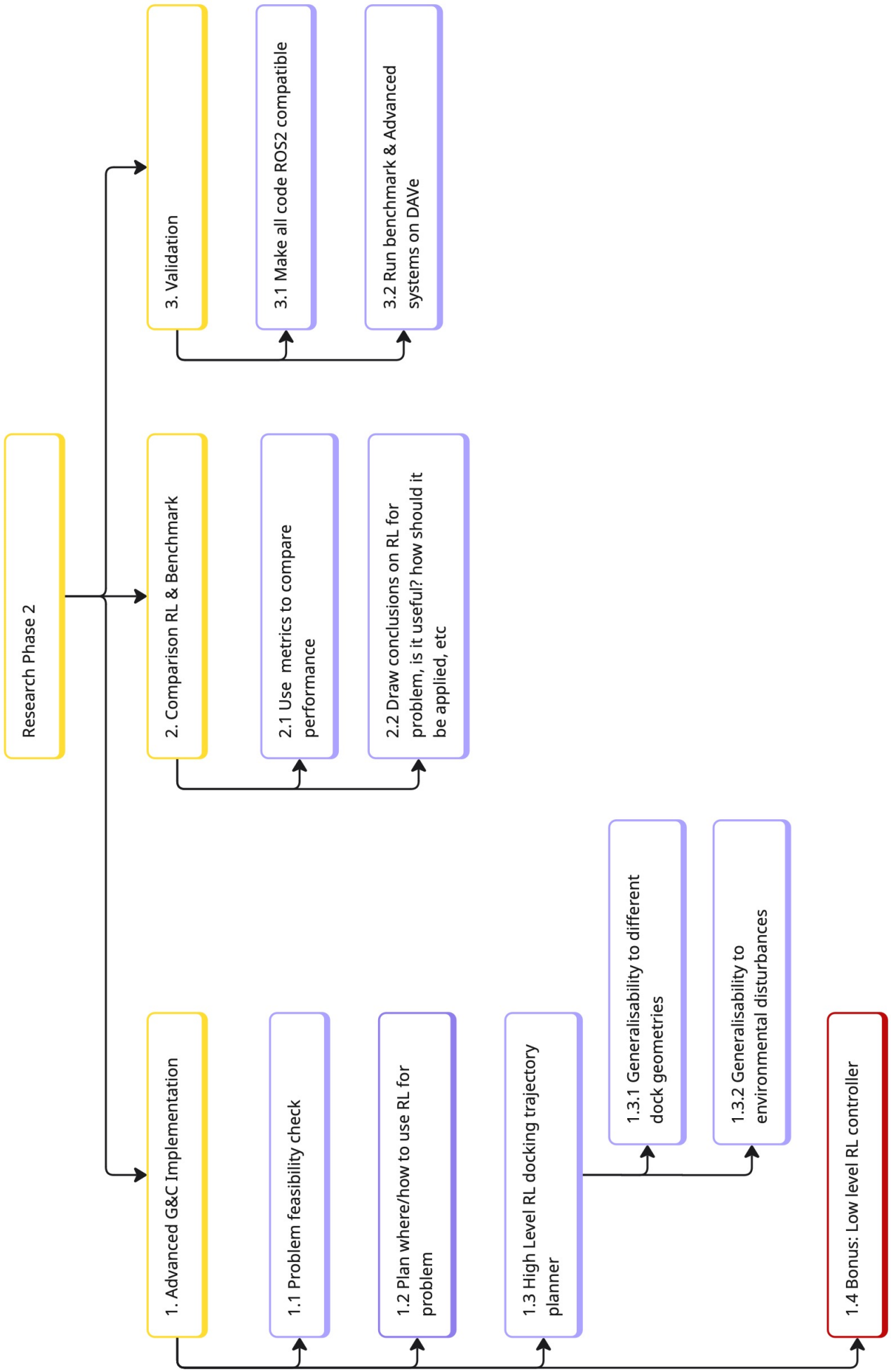
Thesis Research Phase 1

Ugo Coveliers-Munzi

Project Start: Mon, 2/24/2025
Today: Wed, 3/5/2025
Display Week: 1

TASK	PROGRESS	START	END
1. Modeling	100%	2/24/25	3/9/25
1.1 Scaling TitoNeri model to DAVe	100%	2/24/25	2/28/25
1.2 Extract Simplified Models	100%	2/28/25	3/4/25
1.3 Validate Models with recorded real-	0%	3/4/25	3/7/25
1.4 Disturbance Modeling	100%	3/6/25	3/9/25
2. Benchmark G&C	100%	3/10/25	4/6/25
2.1 Generate dock-dock trajectory	100%	3/10/25	3/11/25
2.2 Sailing Phase Trajectory Following C	100%	3/12/25	3/16/25
2.3 Docking Phase Controller	100%	3/17/25	3/18/25
2.4 Environmental Force Observer	0%	3/19/25	3/25/25
2.5 Transition point: Unifying two phas	100%	3/26/25	3/31/25
2.6 Check mission reqs. using metrics	100%	4/1/25	4/3/25
2.7 Compare to Unified Controller	0%	4/3/25	4/6/25
3. Analysis of Problematic Areas	100%	4/7/25	4/20/25
3.1 Investigate Transition & Docking ph	100%	4/7/25	4/13/25
3.2 Investigate robustness w.r.t environ	100%	4/14/25	4/20/25
4. RL feasibility check (start of advanced G&C)	100%	4/21/25	5/4/25
4.1 Determine adequate RL algorithms	100%	4/21/25	4/22/25
4.2 Initial RL implementation on similar	100%	4/23/25	4/27/25
4.3 Apply RL to learn transition point (p	100%	4/28/25	5/3/25
5. Reporting	100%	2/24/25	5/12/25




Research Phase 2
1. Advanced G&C Implementation
1.1 Problem feasibility check
1.2 Plan where/how to use RL for problem
1.3 High Level RL docking trajectory planner
1.3.1 Generalisability to different dock geometries
1.3.2 Generalisability to environmental disturbances
1.4 Bonus: Low level RL controller
2. Comparison RL & Benchmark
2.1 Use metrics to compare performance
2.2 Draw conclusions on RL for problem, is it useful? how should it be applied, etc
3. Validation
3.1 Make all code ROS2 compatible
3.2 Run benchmark & Advanced systems on DAVe

Thesis Research Phase 2

Ugo Coveliers-Munzi

Project Start: Thu, 5/15/2025
Today: Thu, 7/31/2025
Display Week: 1

TASK	PROGRESS	START	END
1. RL Implementation	100%	5/12/25	7/6/25
1.1 Feasibility Check	100%	5/12/25	5/18/25
1.2 Plan where/how to use RL in problem	100%	5/12/25	5/25/25
1.3 High Level RL docking trajectory planne	100%	5/26/25	6/22/25
1.3.1 Focus first on generlisability to differe	100%	5/26/25	6/8/25
1.3.2 Add generlisabvility to env. disturban	100%	6/9/25	6/22/25
1.3.3 Hybrid system to combat black box na	100%	6/23/25	7/6/25
2. Comparison RL & Benchmark	100%	7/7/25	7/13/25
2.1 Compare using metrics	100%	3/10/25	3/11/25
2.2 Conclude if RL useful, how and when to	100%	3/12/25	3/16/25
3. Real-Life Validation	50%	7/14/25	8/3/25
3.1 Make all code ROS2 compatible	100%	7/14/25	7/20/25
3.2 Run benchamrk vs advanced on DAVE ii	50%	7/21/25	8/3/25
5. Reporting	100%	5/12/25	8/16/25