TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Bink, K. J. A., Düz, B., & Weymouth, G. D. (2026). Autonomous sailing with sim-to-real reinforcement learning. *Engineering Applications of Artificial Intelligence*, *182*, Article 115804. https://doi.org/10.1016/j.engappai.2026.115804

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

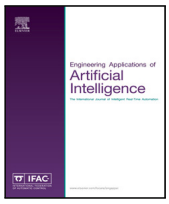
This work is downloaded from Delft University of Technology.

Engineering Applications of Artificial Intelligence 182 (2026) 115804

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research paper

Autonomous sailing with sim-to-real reinforcement learning

Kiki J.A. Bink [a,b,*], Bülent Düz [b], Gabriel D. Weymouth [a]

[a] Faculty of Mechanical Engineering, TU Delft, 2628 CD, Delft, The Netherlands
[b] Maritime Research Institute Netherlands (MARIN), Haagsteeg 2, 6708 PM, Wageningen, The Netherlands

ARTICLE INFO

Keywords:
Autonomous sailing
Sim-to-real reinforcement learning
Simulation
Experiment
Domain randomization
Reality gap
Roll tacking

ABSTRACT

Autonomous sailing offers a sustainable alternative for reducing greenhouse gas emissions in maritime transport, aligning with global environmental targets. This study explores the application of reinforcement learning (RL) to autonomous sailing, addressing challenges in handling dynamic and unpredictable environmental conditions. Leveraging a sim-to-real transfer methodology, RL agents were trained in a simulation environment with the domain randomization technique to enhance adaptability and robustness, and tested in real-world scenarios using a robotic sailboat in the Offshore Basin at MARIN. The study quantified the reality gap between simulation and real-world environments, identifying key discrepancies in actuator latency and simulation modeling accuracy. In real-world basin experiments, the best-performing RL agent successfully completed the course in 12 out of 12 runs. Unlike conventional controllers, the trained agents demonstrated enhanced sailing capabilities like roll tacking and recovery from wind-stalled conditions. This work advances the understanding of autonomous sailing control and highlights pathways to bridge the reality gap, contributing to the broader adoption of RL in dynamic real-world applications.

1. Introduction

Wind-assisted ships have (re)gained considerable attention as part of an effort to vastly reduce the maritime industry's greenhouse gas emissions, with a target of a 50% reduction by 2050 set by the International Maritime Organization (IMO, 2023). However, the control of wind-assisted ships is challenging due to their low control authority and the advance planning that is required. This challenge is further compounded by the constantly changing and unpredictable maritime environment it has to operate in, requiring precise maneuvering and the optimal utilization of sails. Safely and efficiently controlling these ships requires specialized skills, but due to its novelty, there are not many people with expertise in this area (Chou et al., 2021).

To overcome these challenges, there is a need to explore innovative control systems that autonomously operate wind-assisted ships in a way that harnesses the energy of the wind efficiently. Smart control systems, which leverage advanced technologies such as artificial intelligence, have shown promise in various control applications, including the control of power-propelled ships (Chen et al., 2020, 2019). These systems are able to integrate real-time data, optimize sailing strategies, and adapt to changing environmental conditions. In this work, "sailing" denotes wind-propelled surface vessels equipped with sails, while "autonomous" refers to the control of the sail and rudder without human intervention.

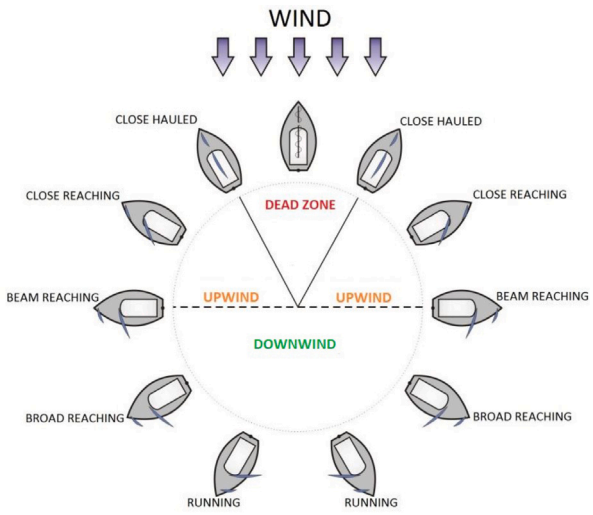
Unlike motorized vessels, sailboats are unable to sail directly against the wind. This is depicted as the "Dead Zone" in Fig. 1(a). Sailing downwind is fairly straightforward, as this is something that automatically happens. However, to sail upwind, the sailboat has to depend on the complex interaction between aerodynamic and hydrodynamic forces. Forward power is generated through lift, produced by maintaining an optimal angle of attack on the sail, similar to an airplane wing (Kimball, 2009). The forces on the sail are generated at the price of heeling and/or leeway. The centerboard (or "keel" depending on the type of sailboat) generates lift in the other direction to counter this leeway and heeling moment by acting as a hydrofoil. The resulting forces on the centerboard and sail are shown in Fig. 2.

The counteracting forces result in a forward motion of the sailboat. This makes the prediction of the speed of a sailing vessel very complex when going upwind. To reach an upwind target, a sailboat follows a zigzag trajectory called *beating*, alternating between close-hauled courses on opposite sides of the wind direction, Fig. 1(a). This involves *tacking*, a maneuver where the boat turns its bow through the wind to switch directions. Efficient tacking minimizes distance and maintains speed toward the target. Such trajectories are shown in Fig. 1(b).

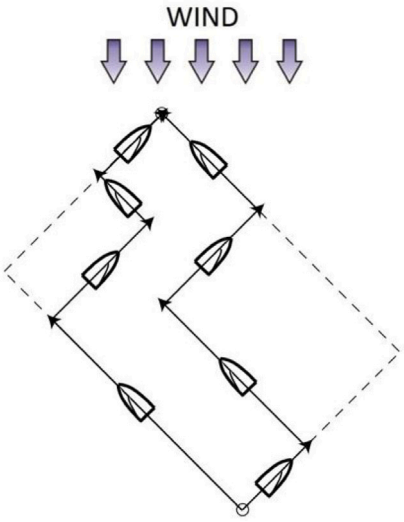
In previous research on autonomous sailing control (ASC), global path planning algorithms, such as A* (Erckens et al., 2010) and RRT* (Saoud et al., 2015), optimize routes while balancing efficiency and adaptability in changing conditions. Local path planning approaches

* Corresponding author at: Faculty of Mechanical Engineering, TU Delft, 2628 CD, Delft, The Netherlands.
E-mail addresses: kiki.bink@vanoord.com (K.J.A. Bink), b.duz@marin.nl, duzbulen@itu.edu.tr (B. Düz), G.D.Weymouth@tudelft.nl (G.D. Weymouth).

https://doi.org/10.1016/j.engappai.2026.115804
Received 26 March 2025; Received in revised form 17 July 2026; Accepted 24 July 2026
Available online 10 August 2026
0952-1976/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).



(a) Points of sail as the different directions a sailboat can travel relative to the wind.



(b) Upwind trajectories illustrating the beating maneuver through alternating tacks. These would take equal time in steady wind.

Fig. 1. Navigating wind angles with a sailboat.

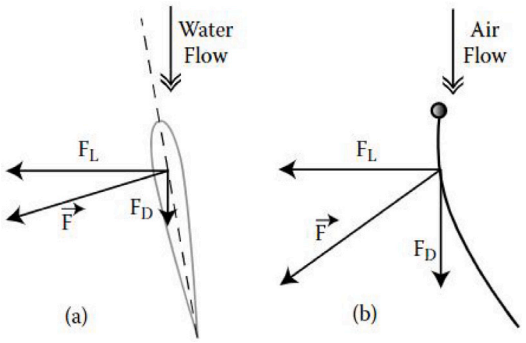


Fig. 2. Forces on the (a) centerboard and (b) sail.

like Line-of-Sight (Deng et al., 2020) and Potential Field (Ulysse et al., 2019) methods focus on near-field navigation to manage obstacles and real-time adjustments. Path following, which manages rudder and sail adjustments, often uses PID, adaptive fuzzy control (Stelzer et al., 2007), or hybrid control techniques to respond to environmental changes. Notable work includes (Lemaire et al., 2018) which presents an adaptive system for selecting optimal tacking strategies for small sailing robots. The system uses a PID controller for low-level rudder control, and sail control is managed through a predefined lookup table that maps apparent wind direction to sail sheet length.

Conventional control algorithms rely on predefined models and rule-based approaches to manage systems, while reinforcement learning (RL) leverages trial-and-error interactions with the environment to optimize decision-making through learned policies. In RL, a learner called the agent performs an action in the environment at each time step and the environment returns a state along with a scalar reward. The agent learns to accomplish a task by gradually improving its policy and maximizing long-term rewards. For path planning and control in ASC, Suda and Nikovski (2022) presents an application of RL to optimize decision-making for sailing a boat upwind under uncertain wind conditions. The study demonstrates that learned policies can outperform conventional control approaches, providing improvements in sailing efficiency. da Silva Junior et al. (2020) proposes a high-level path planning algorithm for autonomous sailboats using RL, optimizing sailing routes by accounting for wind directions, obstacles, and safety while generating feasible trajectories in various scenarios, including upwind navigation. In addition to sailing-specific applications, RL has also been investigated for low-level control of surface vessels, primarily for motor-driven platforms. Chen et al. (2020, 2019) applied RL to the control of heading and speed of power-propelled ships operating under environmental loads, showing that learned policies can improve robustness compared to conventional controllers. RL has also been explored for dynamic positioning, where learned policies maintain position under environmental loads, demonstrating its suitability for continuous marine control problems (Øvereng et al., 2021). Furthermore, RL has been proposed as a method for collision-aware maneuvering for motor-driven vessels, where agents are trained to simultaneously control heading and speed in dynamically changing scenarios, showing their ability to adapt to evolving vessel interactions (Chun et al., 2024).

One of the main challenges in RL is safely and autonomously collecting extensive data, which is best handled through simulation. Simulation allows for rapid, parallelized data collection, reducing training costs and human supervision requirements (Ibarz et al., 2021). However, transfer of RL agents trained in simulation to real-world robotic systems faces the "reality gap", or inaccuracies in simulations due to modeling errors and assumptions (Zhao et al., 2020). This gap is widened by real-world robotic-system issues like sensor noise, actuator latency, and variability in environmental dynamics, which might not be fully represented in simulations. Although RL research in ASC has primarily focused on simulated models (da Silva Junior et al., 2020; Suda and Nikovski, 2022), bridging the reality gap is crucial to achieving reliable real-world performance.

To address the challenges of transferring learned policies from simulation to reality, domain randomization (DR) has become a widely used and effective technique, particularly in dynamic and continuous control environments. DR introduces variability into simulation parameters to improve the generalization of learned policies to real-world conditions, prioritizing robustness over simulation fidelity. Two common approaches to DR are dynamics randomization and initial condition randomization. For example, in quadruped locomotion, Tan et al. (2018) randomized parameters like friction coefficients and applied perturbation forces, while Wada et al. (2022) focused on variations in aerodynamic coefficients for aerial vehicles. Similarly, Loquercio

et al. (2020), Kaufmann et al. (2023), and Wang et al. (2022) randomized initial conditions, such as lighting, starting positions, and state perturbations. Kaufmann et al. (2023) extended their approach by incorporating noise models, while Wang et al. (2022) demonstrated how DR could be combined with progressive task complexity to further enhance policy robustness, both achieving real-world task success comparable to or even surpassing professional human performance. In this study, DR was limited to variability in the initial state of the vessel and the environmental wind setting, while the underlying system and actuator dynamics were kept fixed.

While these sim-to-real studies demonstrate the potential of DR to enable effective policy transfer, none have explicitly measured or quantified the reality gap that was bridged. This gap, representing the discrepancy between simulated environments and real-world conditions, is a critical yet often overlooked factor. Quantifying it could provide valuable insights into the effectiveness of sim-to-real transfer methods, help refine simulation setups by identifying areas where improvements are needed, and establish benchmarks for evaluating transfer success. Without a clear measure of the reality gap, it is challenging to establish benchmarks for transfer effectiveness or to compare the performance of different methods systematically. By addressing this overlooked aspect, future research could strengthen the foundation of sim-to-real transfer and expand its applicability across a broader range of control tasks. In addition, existing work on RL for surface vessel control focuses predominantly on motor-driven vessels and simulation-based evaluation. While RL has been applied to wind-propelled sailing in prior studies (Suda and Nikovski, 2022; da Silva Junior et al., 2020), these studies remain simulation-based, and real-world validation of RL-driven sailing control has not yet been demonstrated.

This study explores the viability of RL for controlling underactuated systems, such as wind-assisted ships, in dynamic and unpredictable real-world conditions. The research focuses on developing an RL-based control system for a small sailboat in a simulation environment and subsequently transferring it to a real-world environment, specifically the Offshore Basin (OB) at MARIN (The Maritime Research Institute Netherlands). The primary objective is to assess the effectiveness of RL policies trained in simulation when applied to real-life scenarios. By analyzing their performance and investigating the reality gap, this work aims to enhance understanding of RL's potential and limitations in autonomous sailing control. While prior work on RL for autonomous sailing has primarily focused on simulation-based performance, this study provides systematic evaluations of learned sailing policies in a controlled real-world basin environment and relates their performance to differences between simulation and reality. In doing so, the study contributes to broader research on sim-to-real transfer in dynamic environments.

The paper is organized as follows: Section 2 describes the environments and the RL agent with the state and action spaces as well as the reward function. Furthermore, the section highlights the training details for the RL agents and the evaluation process yielding the RL agents to be transferred to the real world. Finally, the baseline controller is explained, which is used for comparison with the RL agents. Section 3 presents the results in the simulation environment. First, the reality gap is quantified and analyzed, and subsequently results are presented to demonstrate that RL agents are able to learn to sail upwind in the simulation environment. In Section 4, sim-to-real transfer results are given, by first explaining the sim-to-real training setup and then the deployment of the agent in the basin. This is followed by the results demonstrating the fundamental and advanced sailing capabilities that the agents displayed in the basin. Lastly, Section 5 finalizes the paper with conclusions and ideas for future research.

2. Method

The reinforcement learning agent is trained in a simulation environment. The policies saved during training are then subjected to an evaluation process, and the selected ones are transferred to the basin. An established successful robotic sailing controller is used as a baseline to compare against the policies. This section presents detailed information on these steps.

Table 1
Main particulars of the Optimist assembly.

Main particular	Value	Unit
Lpp	2.041	m
Beam	1.112	m
Ta	0.141	m
Tf	0.075	m
Hull mass	108.96	kg
Ballast mass	42.59	kg
Boom mass	1.65	kg

2.1. Environments

The following sections provide detailed information on the two environments used in this study: the simulation environment where the training of the RL agent was done and the basin environment where the trained agents were transferred without any further tuning or training.

2.1.1. Basin environment and the optimist

The experiments to test the trained agents were conducted in the OB at MARIN. The OB has the dimension of 45 m × 36 m × 10.2 m and is equipped with wave, wind and current generation mechanisms as well as a carriage. Wave and current generators were not used during the experiments, and the wind was generated using electrical fans installed on a platform of 24 m width. The carriage can follow the movements of the Optimist in both directions on the horizontal plane. During the experiments several precautions were taken to prevent any accident. The Optimist was allowed to move in an inner domain safely away from the sides of the basin. The perimeter of the inner domain was demarcated by a rope barrier. Furthermore, two members of the basin crew were in the water at a safe distance from the Optimist to recover it in case of a risky situation. Fig. 3 shows a view of the OB with the wind fans, Optimist, carriage and basin crew. Fig. 4 illustrates the Global Coordinate System (GCS) in the basin with the dimensions of the inner domain and the basin. The xy-plane of the GCS is aligned with the Still Water Line (SWL). The z-axis points upwards.

The electrical fans generated the wind at two levels; 40% or 60%. Prior to the experiments, steady wind velocity measurements were conducted at three elevations (0.5 m, 1.0 m and 1.5 m) at the two wind levels since the Optimist was not equipped with a sensor to measure the wind. The wind probe array consisted of 64 (x, y) positions inside the inner domain. These measurements were converted into a spatial non-temporal wind field. This data was used during both the tests in the basin and training in the simulation.

The robotized Optimist is shown in Fig. 5 and the main particulars are listed in Table 1. It features three actuators, for moving the rudder, setting the sail, and moving the ballast. The ballast can be moved only in the transverse direction and is intended to imitate a person on board. The carriage tracks and follows the Optimist, and features the Optotrak Certus HD which is *a Dynamic Measuring Machine for 3D and 6-DOF motion tracking and analysis* (NDI, 2024). This system is able to track the positions and motions of infrared LEDs (markers) on the Optimist in real-time at a high speed of approximately 495 frames per second. Using at least 5 of the 8 LEDs, the position and motions of the Optimist can be captured at high accuracy. The carriage is also equipped with a wireless transmitter/receiver that can send and receive data in real-time.

The choice of a small sailboat like the Optimist is advantageous since it can be tested and controlled within the confines of a basin that already has sensor systems and camera setup, reducing the complexity

Fig. 3. A view of the Offshore Basin with the carriage following the Optimist and wall of wind fans.

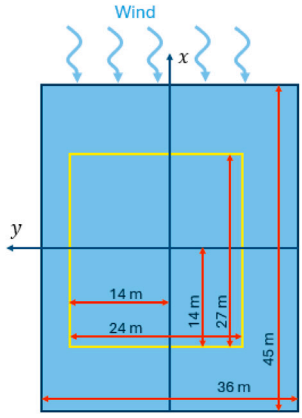


Fig. 4. Top view of the basin with the global coordinate system, dimensions and wind direction. The xy-plane of the coordinate system is aligned with the still water line. The z-axis points upwards. The inner domain is indicated by the yellow rectangle.

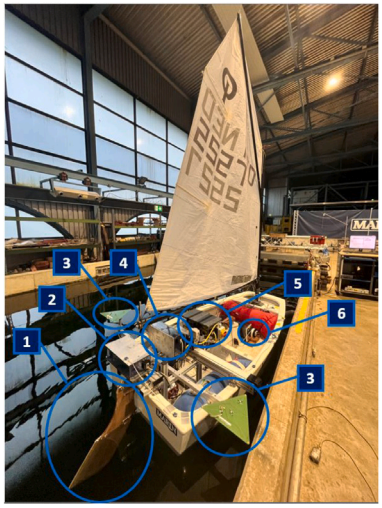


Fig. 5. Robotized Optimist for autonomous sailing control, with: 1. Rudder 2. Servo motor to control the rudder angle 3. Triangles with a LED on each corner 4. Servo motor to move the ballast 5. Ballast/battery 6. Winch for sail control.

and risks associated with open water trials. Moreover, there is existing knowledge and experience in ASC of small sailboats, which can serve as a foundation for developing control strategies and as a benchmark for evaluating the RL agents.

2.1.2. Simulation environment

The numerical model of the sailing Optimist is implemented using MARIN's time-domain seakeeping and maneuvering software known as xSimulation (Quadvlieg and Rapuc, 2019). Fig. 6 shows the definitions of the ship motions and the right-handed local coordinate systems (LCSs). The setup of the numerical model of the Optimist includes three main elements: the environment, the Optimist assembly, and the numerical integrator. The environment is a digital copy of the OB (see Fig. 11). The measured wind speed in the OB is imported into the simulations in order to calculate the apparent wind speed and direction using the heading and location of the Optimist. The numerical integrator executes the integration in time using the implicit Euler method. The Optimist assembly is comprised of three bodies with associated LCSs as shown in Fig. 6(b), each with its own mass and moment of inertia. These bodies are hull, ballast and boom\sail, and linked together by means of constraints. The origin of the boom is at the intersection between the boom profile and the centerline of the mast. The origin of the ballast is chosen at the center of the aluminum profile, which is the support of the ballast actuator on the hull edges.

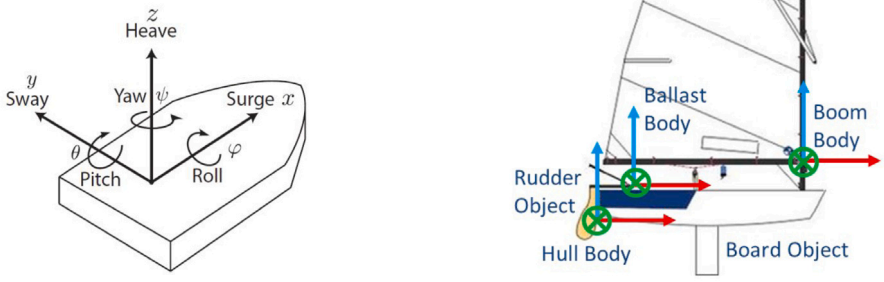
A key feature of the simulator is its ability to estimate the hydrostatic and hydrodynamic forces on the Optimist. To calculate the aerodynamic forces, the lift and drag coefficients of the Optimist's sail were measured in a previous study (Andersson et al., 2018) in steady flow conditions. The driving and side forces acting on the sail were then calculated. The leeway angle was assumed to be negligibly small. The application position, or centre of effort, of the sail forces was assumed to be in the center of the sail area of the Optimist. The angle of attack was taken as the difference between the apparent wind direction and the measured boom angle. These adjusted steady-state coefficients are then applied to the unsteady motion of the 6-DOF. Calculated aerodynamic forces and their application positions were then used in combination with the hydrostatic and hydrodynamic forces to solve the equations of motion in 6-DOF.

2.2. Reinforcement learning agent

Reinforcement Learning is a framework for sequential-decision making, where an agent interacts with an environment, learning through trial and error to maximize long-term rewards. Formally, RL problems are often modeled as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(s_{t+1}|s_t, a_t)$ is the unknown transition dynamics with $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in \mathcal{A}$, $r(s_t, a_t)$ is the reward function, and $\gamma \in (0, 1]$ is the discount factor. In the MDP framework, the agent observes the current state s_t at each time step and selects an action a_t according to its policy $\pi(a_t|s_t)$. The environment then transitions to a new state s_{t+1} according to its transition dynamics p and provides a reward r. This process repeats, generating a trajectory of state–action-reward sequences. The agent's objective is to learn an optimal policy that maximizes the expected cumulative discounted reward,

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{(s_t,a_t)\sim\pi} \left[\gamma^t r(s_t, a_t)\right]. \tag{1}$$

The Markov property assumes that the future state depends only on the current state and action, not on the history of previous states, which makes the problem tractable for computational solution. RL algorithms solve MDPs by iteratively improving the policy through



(a) Ship motions in 6-DOF

(b) Optimist assembly with three bodies and associated right-handed LCSs

Fig. 6. Definitions of the ship motions and local coordinate systems.

experience, either by learning a value function that estimates expected future rewards or by directly optimizing the policy parameters. Here, the discount factor γ plays a crucial role in balancing immediate versus future rewards. A value of γ close to one places more emphasis on long-term performance, encouraging the agent to plan ahead, whereas smaller values prioritize short-term gains. In our sailing application, a high discount factor ($\gamma = 0.998$ as listed in Table 5) was chosen to ensure that the agent considers the long-term consequences of its actions along the path to the upwind target, rather than focusing solely on immediate gains. This is essential because successful upwind sailing requires a sequence of well-coordinated maneuvers (tacking) where short-term speed loss during a tack must be accepted to achieve the ultimate goal of reaching the target.

Haarnoja et al. (2018a) proposed the Soft Actor–Critic (SAC) agent, which extends maximum entropy RL by optimizing a policy to maximize both expected reward and entropy, promoting exploration. The objective of the SAC agent is

$$J(\pi) = \sum_{t=0}^{\infty} \mathbb{E}_{(s_t,a_t)\sim\pi} \left[\gamma^t (r(s_t,a_t) + \alpha \mathcal{H}_t)\right]. \tag{2}$$

Here, $\mathcal{H}_t$ represents the entropy term indicating the stochasticity of the policy, and the temperature parameter $\alpha > 0$ controls the trade-off between reward maximization and exploration. The entropy term encourages exploration by preventing premature policy convergence to suboptimal deterministic solutions. A higher α leads to more stochastic policies, favoring exploration, whereas a lower α results in more exploitative behavior. In this work, α was not selected manually. Instead, the automatic entropy tuning mechanism proposed by Haarnoja et al. (2018b) was used, where α is learned online during training. This allows the agent to adaptively adjust its exploration–exploitation trade-off based on the learning progress and task complexity, removing the need for manual tuning.

In the maritime domain, the SAC agent has found applications, particularly in enhancing the capabilities of autonomous marine vehicles and improving navigation safety. Greep et al. (2025) investigated both model-free and model-based approaches with the SAC agent in heading-keeping of ships in waves. Xu et al. (2022) extended upon the SAC agent and applied it to the collision avoidance problem of an autonomous underwater vehicle. Zheng et al. (2022) presented a linear active disturbance rejection control strategy based on the SAC agent and applied to the path tracking control of unmanned surface vessels under wind and wave disturbances.

The SAC implementation from the Stable Baselines3 library (Raffin et al., 2021) was used in this project. Detailed information regarding training parameters and the architecture of the agent is presented in Section 2.5.

Table 2
State and action spaces.

Action space ($\mathcal{A}$)	State space (S)	Unit
Move rudder	Rudder angle (δ_r)	rad
Let sheet out/in	Sheet length (l)	m
Move Ballast	Ballast position (x_b)	m
	Boom angle (δ_s)	rad
	Distance to target (x_t, y_t)	m
	Velocity (u, v)	m/s
	Yaw angle (ψ)	rad
	Yaw rate (r)	rad/s
	Roll angle (ϕ)	rad
	Wind heading (α_w)	rad
	Wind speed (v_w)	m/s

2.3. State and action spaces

In our problem, both the state and action spaces are continuous, as listed in Table 2. All the components in the state space were normalized to the range of [−1, 1]. The actions that the policy outputs were also in the range of [−1, 1], which were then scaled to the corresponding range of each individual actuator and clamped to within the maximum allowed actuator rates.

Since the Optimist was not equipped with any sensors to measure the wind speed or direction, these were calculated in the simulation model using pre-measured wind data as described in Section 2.1.1.

2.4. Reward function and terminations

In all our experiments there are multiple objectives that the agent needs to achieve simultaneously. Each objective is represented by a distinct reward component that measures a specific aspect of the simulation, often tied to a physical quantity. These components are combined into a single scalar reward value. Details about the reward components are provided in Table 3. Each reward component is designed to promote a specific aspect of the desired sailing behavior. The distance reward encourages steady progress toward the upwind target by reducing the distance to the target line. The control cost discourages abrupt or excessive control actions, promoting smoother rudder, ballast, and sail adjustments. The gybing penalty discourages rapid boom reversals, which are mechanically stressful and increase the risk of capsizing, thereby promoting safer maneuvering. When the control policy triggers a termination, a substantial reward is given, see Table 4. The termination rewards are designed to enforce safety and task completion. Large negative rewards are assigned when the agent leaves the operating area, exceeds roll angle limits, or runs out of

Table 3
Reward components.

Reward component	Description	Calculation
Distance	Encourage the agent to reduce the distance between its actual position (x_{ship}) and the target line ($x_{target} = 10\,m$)	$2 \times (\frac{1}{\lvert x_{target} - x_{ship} \rvert + 1} - (x_{target} - x_{ship}))$
Gybing	Punish the agent if the boom swiftly changes sides. The sudden and large change of boom angle can put excessive stress on the sails and other equipment while also causing a risk of capsizing.	−100
Control cost	A negative reward to penalize the agent for taking actions that are too large	$-5 \times$ (Rudder Rate + Ballast Rate + Sheet Rate)

Table 4
Terminations triggering the end of an episode with a corresponding reward.

Termination	Termination Criteria	Reward
Out of Time	Agent runs out of time, (t > 360 s)	−1000
Out of Boundaries	Agent goes outside the boundaries of the inner domain shown in Fig. 4	−3000
Capsize Risk	Agent runs the risk of capsizing the boat, (Roll > 36°)	−3000
Reaching Target	Agent reaches the target line, ($x_{ship} \geq x_{target} = 10\,m$)	+2000

time, discouraging unsafe or non-productive behavior. A large positive reward is given when the target line is reached, reinforcing successful task completion.

Several studies emphasize the importance of reward design in RL and the trade-off between sparse and dense rewards (Hwangbo et al., 2019; Henderson et al., 2018). Sparse rewards, which only provide feedback at episode termination, can lead to slow or unstable learning in complex environments requiring extensive exploration (Larsen et al., 2021). In this sailing task, the continuous action space and low probability of reaching the target through random exploration make purely sparse rewards impractical, consistent with findings by Suda and Nikovski (2022). Intermediate rewards were therefore introduced to promote progress toward the target while discouraging unsafe behavior through termination penalties and task-specific constraints such as limiting excessive gybing. Guided by these considerations, the relative weighting of the reward components was tuned through a limited grid search, where higher distance rewards encouraged aggressive sailing behavior and stronger control penalties produced overly conservative policies. The selected weights represent a compromise between reliable task completion and smooth control behavior.

2.5. Training details

The SAC agent's architecture comprises two neural networks: the policy network for the actor and the Q-network for the critic. In this case, both networks were designed as a multi-layer perceptron with two hidden layers, each consisting of 64 units with the tanh activation function. The weights in both networks were initialized with Glorot initialization (Glorot and Bengio, 2010). The actor receives the state as its input and outputs a Gaussian distribution for each action. The critic predicts the Q-value for a state–action pair that it receives. Table 5 lists the common SAC agent parameters used in this study. The values were obtained after a basic grid search.

To take the stochasticity in the environment and the policy into account, RL agents were trained in four separate trials. The training duration was 400,000 time steps. The agents were trained on a cluster node featuring dual Intel(R) Xeon(R) Gold 6126 CPU @ 2.60 GHz 12 core CPUs, dual NVIDIA Tesla P100 GPUs (16 GB RAM and 3584 CUDA cores each), 768 GB RAM memory, and a 500 GB SSD local disk.

After a training was completed, the agents were put through an evaluation process explained in detail in Section 2.6.

Table 5
Hyperparameters for the RL agents.

Parameter	Value
Optimizer	Adam
Learning rate	0.0003
Discount factor	0.998
Target value smoothing constant	0.005
Number of hidden layers	2
Hidden layer size	64
Activation function	Tanh
Replay buffer size	400,000
Batch size	256
Initial exploration steps	20,000
Policy training frequency	1
Policy training iterations	1

Table 6
Values of the initial conditions considered in the 36 evaluation runs.

Initial condition	Values
Wind speed setting	40%, 60%
x-position	−8.0 m
y-position	−6.0 m, 0.0 m, 8.0 m
Heading angle	−45.8°, −11.4°, 34.3°
Forward speed	0.2 m s^{-1}, 0.6 m s^{-1}

2.6. Evaluation of the agents in the simulation

The learned policies of the agents trained in the simulation as described in Section 2.5 are put through an evaluation process again in the simulation. This process consists of firstly running a large number of evaluation runs for the policies, secondly assessing the results using five metrics, and finally taking a weighted average of these metrics to obtain an overall performance score. Depending on the overall performance score, successful policies are selected to be transferred to the basin.

The evaluation runs are started with various initial conditions covering wind speed, starting position in the domain, heading angle and forward speed, see Table 6 for more details. The Cartesian product of all the initial condition values results in 36 evaluation runs per policy.

The results of the evaluation runs are examined using five metrics listed in Table 7. Each metric is defined to assess a specific aspect of a policy's performance. The metrics are calculated as an average over the 36 evaluation runs.

Finally, a weighted average of these metrics is calculated to obtain an overall performance score for each evaluated policy, facilitating a

Table 7
Metrics used in the evaluation process.

Metric	Description
Success rate	Percentage of episodes where the agent reaches the target line
Time to complete	Average time it takes for the agent to reach the target line
Energy ratio	Average ratio of the energy the agent puts into the rudder and ballast to the energy the agent extracts from the wind via the sail
Control cost	Average rudder, ballast and sheet rates
High-risk behavior	Gybing and rolling are considered in this metric. The total number of gybes is averaged over the total duration of the evaluation runs. For rolling, no penalty was given for roll angles below $0.2 \cdot \theta_{\max}$, and above this value penalty grows exponentially with the magnitude of the roll angle. $\theta_{\max}$ was calculated as $\theta_{\max} = \arctan(GM/BM)$.

quick way to compare them. For full description of the metrics and their weighted average, the reader is referred to Bink (2024). The same evaluation process is applied to the trained agents and the baseline controller which is described next.

2.7. Baseline controller

RL agents are compared to the control system developed by Southampton University's Sailing Robot Team and described in Lemaire et al. (2018). Hereafter, this controller will be referred to as the Sailing Robot Controller of Southampton (SRCS). The SRCS was initially developed for a one meter long autonomous sailing boat with the goal to win the World Sailing Robot Competition. The code is open-source and based on the Robot Operating System.

Several modifications were done in the SRCS to adapt it to the problem at hand. The parts of the control system necessary to reach an upwind target were extracted. The *high-level control* is comprised of two key components: *tack decision* and *goal heading*. In our problem, the *tack decision* has been modified to be based on "tacking zones", which are placed 6.0 m from each side of the basin, giving the Optimist space to execute its tacking maneuver. A tack command is triggered when the Optimist is sailing with its sail on the portside and enters the left tacking zone, and vice versa. Among the four different tack maneuvers implemented in the SRCS, the basic tack maneuver was selected for this setup, as it is best suited to the conditions present in the basin. The *goal heading* calculation in the SRCS was based on the relative angle to the designated target. When the target is upwind, the goal heading is set to 45° in the SRCS.

The *low-level control* consists of the sail and rudder control and did not require any adaptations besides making it compatible with our simulation environment. The sail setting is controlled with a look-up table based on the apparent wind direction, which is computed from the known wind field and the vessel heading. We took the values for the sail setting from tests that the sailing robot team conducted with a Laser dinghy, which is similar in size to the Optimist.

A PID controller was used for the rudder, which takes the error in heading as input and outputs the rudder setting at every 0.1 s. The PID controller was tuned in simulations and its performance was validated using the data available in Lemaire et al. (2018). The overall behavior when sailing upwind was very similar with a robust and stable performance.

The SRCS did not include any control mechanism for moving the ballast. Therefore, in all our runs with the SRCS, the ballast was kept fixed at the center line of the boat.

3. Results in the simulation environment

3.1. Quantifying the reality gap

Before testing the learned policies in the real basin environment, it is important to quantify the modeling error in the simulation with respect to the reality in the basin. For this purpose, five runs were performed in the basin using open-loop controls with the same predetermined rudder and sheet actions. The basin runs were then replicated in the simulation, matching the measured initial conditions and using the same control actions. The difference in the boat trajectories predicted in the simulation environment and those measured in the Offshore Basin is attributed to the so-called *reality gap* (Tobin et al., 2017) in RL terminology. We then assumed that the reality gap is mainly due to two sources: the latency in the basin and the modeling error in the simulation.

Table 8
Average RMSE and NRMSE with ± one standard deviation in the rudder and sheet length signals.

	RMSE	NRMSE
Rudder Angle	0.74 ± 0.08 deg	0.89% ± 0.10%
Sheet Length	7.54 ± 0.60 mm	14.98% ± 0.75%

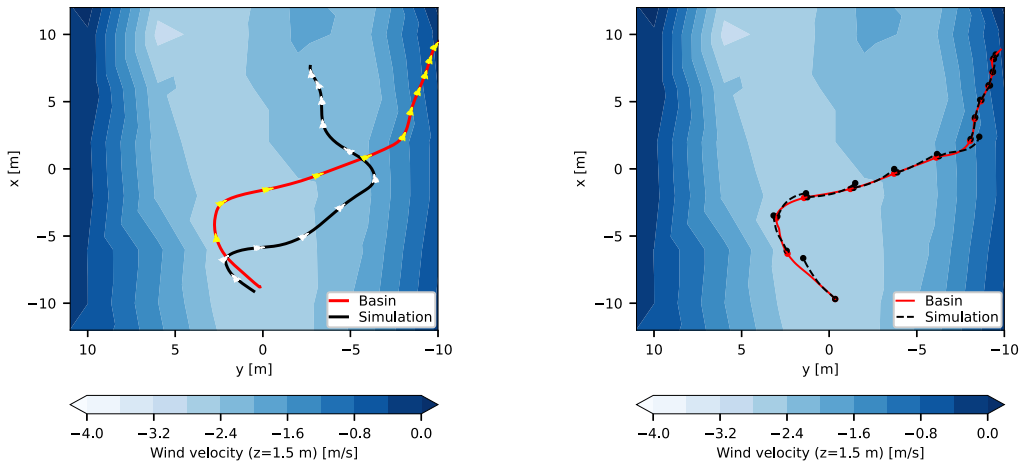
The signals for the actuators from the simulation and the basin were overlaid and synchronized. The latency in the basin for the rudder angle was obtained as 0.2 s, and for the sheet length as 0.3 s. The remaining error in the actuator signals is then calculated using the Root Mean Squared Error (RMSE) and the Normalized Root Mean Squared Error (NRMSE) as:

$$\begin{aligned} \text{RMSE} &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(s_t^{\text{basin}} - s_t^{\text{simulated}})^2} \\ \text{NRMSE} &= \frac{\text{RMSE}}{s_{max}^{\text{basin}} - s_{min}^{\text{basin}}} \times 100 \end{aligned} \tag{3}$$

where t indicates the time index, T the number of time steps over the considered duration and s the signal. The average RMSE and NRMSE values over the 5 runs are given in Table 8. The error in the rudder angle is below 1% and in the sheet length about 15%. Therefore, the rudder angle signal is sufficiently accurately reproduced in the simulation. The sheet length signal still has some deviation in the simulation compared to the measurement, which might be the source for a part of the reality gap.

Fig. 7(a) shows the measured trajectory from one of the five runs and its reproduced counterpart in the simulation. It is clear that the trajectories differ significantly with errors accumulating over time. To investigate the reality gap in more detail, each trajectory in the basin runs is split in time increments of 4 s, which is approximately the average time it takes for the Optimist to traverse a distance of one body length. The initial conditions at each 4 s interval were obtained from the basin data and used as input for the simulation for all 5 runs. The resulting time-segmented trajectory reproduced in the simulation for one of the five runs is shown in Fig. 7(b). For each 4-second increment, the average position error is calculated as the Euclidean distance $\overline{d}$ using the basin position of the Optimist $(x_t^{\text{basin}}, y_t^{\text{basin}})$ and the simulated one $(x_t^{\text{simulated}}, y_t^{\text{simulated}})$ at each time index t,

$$\overline{d} = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\left(x_t^{\text{basin}} - x_t^{\text{simulated}}\right)^2 + \left(y_t^{\text{basin}} - y_t^{\text{simulated}}\right)^2} \tag{4}$$



(a) Measured trajectory in the basin and its reproduced counterpart in the simulation

(b) Basin run compared to the time-segmented run performed in the simulation

Fig. 7. Open-loop runs as an attempt to quantify the reality gap.

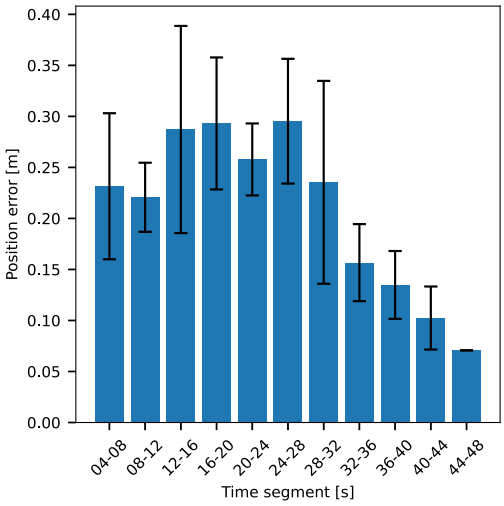


Fig. 8. Mean position error for each 4-second time increment. The error bars indicate ± one standard deviation. The mean and standard deviation for each 4-second increment are calculated over the five runs. Note that the number of runs in the last two time segments is smaller compared to the rest since the durations of the runs were not consistent.

Table 9
Mean RMSE in a 4-second time increment averaged over five runs (with ± one standard deviation) for the position, speed in x and y directions, roll, pitch and yaw motions.

	RMSE
Position	0.2 ± 0.06 m
Speed in x-dir.	0.03 ± 0.02 m/s
Speed in y-dir.	0.03 ± 0.01 m/s
Roll	0.75 ± 0.31 deg
Pitch	0.35 ± 0.08 deg
Yaw	8.64 ± 4.87 deg

Table 10
Summary of SAC training results. The success rate column depicts the percentage of times the agent reaches the target during training.

Run No	Total runtime	Episodes	Success rate
1	22 h 46 m	5083	97.0%
2	23 h 7 m	5317	97.6%
3	22 h 53 m	5166	98.2%
4	20 h 17 m	4008	95.8%

with T indicating the number of time steps in a 4-second increment. Mean errors of the other ship motions and velocities between the simulation and the basin runs were obtained by calculating the RMSE using Eq. (3) for each 4-second increment.

The mean position error is shown in Fig. 8, where the mean and standard deviations are calculated over the five runs. Since the mean position error for the first 4-second increment is relatively high due to the start-up procedure in the basin, it was excluded from the modeling error analysis for all the runs. Fig. 8 illustrates the mean position error across the remaining time increments. The error is highest between 8 and 32 s corresponding to when the Optimist navigates in areas of high wind speeds and executes tacks.

The average RMSE values are listed in Table 9. These are calculated in each 4-second time increment averaged over all the increments in five runs. The average RMSE of yaw is notably high with a high standard deviation as well, suggesting a large variability over the time segments. Furthermore, the simulation typically overestimates the amplitude of the pitch motion. Overall, the analysis reveals discrepancies between simulation and reality, particularly in highly dynamic conditions. These findings highlight areas where the model can be refined to capture real-world behavior.

3.2. Verification of RL training in the simulation environment

To confirm that the RL agents are able to learn a reasonable sailing policy with the setup described in the previous sections, four parallel training runs were conducted with the same initial conditions. The results are listed in Table 10. It is clear that due to the randomness in the agent's exploration process and the policy, the performance of the agent varies even when trained using the same setup. Looking at the success rate during the trainings, all four agents perform well.

Fig. 9 shows the learning curve of the agent in one of the runs from Table 10. In the beginning of the training, the agent is heavily exploring, hitting large penalties that drop the reward significantly. As the learning progresses, the agent improves its policy and demonstrates asymptotic stability within the final 4000 episodes. Fig. 10 shows the paths of the boat in different episodes from this training process. The agent gradually learns a beating maneuver to reach the upwind target as it enhances its policy during training. Furthermore, it also learns to stay within the area of the domain where the wind velocity is high.

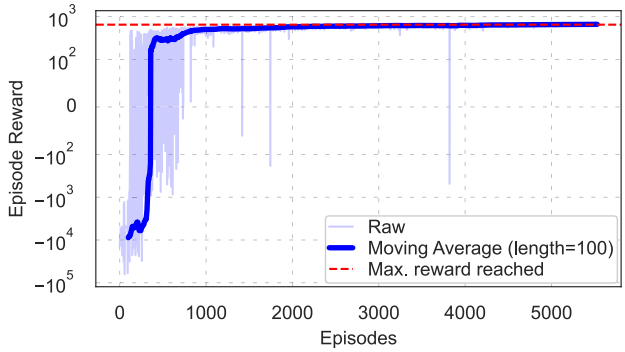


Fig. 9. Cumulative episode rewards from a training listed in Table 10. The solid thick line represents a 100-episode moving average, while the light thin line indicates raw reward. To visualize the transition from initial exploration penalties (≈ -63000) to the stable convergence phase (≈ 640) the y-axis is displayed on a symmetric logarithmic scale.

Table 11
Initial conditions for the SAC agent with and without domain randomization.

Initial condition	SAC	SAC with domain randomization
Wind speed setting	60%	{40%, 60%}
x-position	−8.37 m	[−10.0 m, −6.0 m]
y-position	−6.13 m	[−8.0 m, 12.0 m]
Heading angle	30.0°	[−45.0°, 45.0°]
Forward speed	0.3 m s^{-1}	[0.0 m s^{-1}, 0.6 m s^{-1}]

4. Results of sim-to-real transfer

When the RL agents trained in the simulation were transferred to the basin, the sensor data from the basin was sent to the RL agent and the actions from the RL agent were sent to the basin via the simulation environment, see Fig. 11. In this case, however, the simulation environment was simply used as a means for communication between the RL agent and the basin, and not as a constrained dynamics solver for ship motions. The wind speed and heading were obtained in the simulation environment using the measured position and heading of the Optimist and pre-measured wind data. These were also sent to the RL agent in addition to the sensory data from the basin. Furthermore, when sending the actions from the RL agents to the basin, the simulation model checks them using the real rates of the actuators in the basin.

4.1. Sim-to-real training setup

Two types of agents were considered for the sim-to-real study:

1. *SAC trained with Domain Randomization*: The domain randomization is achieved by randomly varying the initial conditions of the environment during training, including the wind speed setting, initial position, heading angle, and forward speed of the vessel. The range of initial conditions is listed in Table 11. Considering the limited testing time in the basin, two policies with the highest overall performance score were selected for transferring to the basin. They will be referred to as *SAC_RI_3* and *SAC_RI_4* hereafter. Both SAC_RI_3 and SAC_RI_4 are chosen as they performed comparably in the evaluation process described in Section 2.6.
2. *SAC trained without Domain Randomization*: This agent was trained with the same setup as in the verification runs in Section 3.2. After evaluating the policies according to the process in Section 2.6, one was selected for testing in the basin. Hereafter, this agent will simply be referred to as *SAC*. The initial condition for this agent is listed in Table 11.

4.2. Deployment of the agent in the basin

Before starting any tests, the measurement systems were calibrated. The rudder, ballast and sheet were zeroed. The position measurement system was calibrated as well making sure that the carriage followed the Optimist so that at least 5 LEDs were kept in the frame and the global position could be recorded. Once all the systems were verified to work, preliminary sanity checks were performed using the open-loop controls. The fans were set at wind levels of either 40% or 60% and a buffer period of about 2 min was allowed for the wind to stabilize before proceeding with the tests.

Safety was of the highest importance during the testing procedure. A basin crew of at least 3 people were always present to oversee the systems in the basin, the hardware on the Optimist and the carriage. Two members of the crew were in the water in the basin at all times during the tests. They could intervene at any time to make sure that the Optimist did not capsize, hit the carriage or perform any other type of dangerous behavior. They also brought the Optimist back to the desired starting position after each test. In some cases the runs had to be stopped early, for example, when the carriage was unable to follow the Optimist. This occurred when the Optimist sailed too fast, gybed, or rolled excessively.

It was not possible to evaluate each agent in the basin as was done in the simulation environment described in Section 2.6. This was due to the high cost of the basin time. Instead, 3 different starting positions with both wind levels were used for each agent with the aim to repeat runs at least 2 times with the same initial conditions to investigate the consistency and reliability of the results. The starting x-position was set as $x \approx -10\,\text{m}$, while the starting y-position was chosen from $y \approx \{-7.5\,\text{m}, 0\,\text{m}, 7.5\,\text{m}\}$. The starting heading was 30° when starting at the first two y-positions, and −30° at the last one. The initial forward speed was provided by the basin crew gently pushing the Optimist at the start of a run.

Besides assessing the RL agents, the basin time was used to run the open-loop control tests as described in Section 3.1 and to assess the SRCS explained in Section 2.7 serving as a baseline.

4.3. Demonstration of fundamental sailing capability

The average success rate, energy ratio and completion time of the successful runs are summarized in Fig. 12. The lower the energy ratio, the better, since it represents the ratio of rudder and ballast energy used compared to the wind energy harnessed. The performances of the three agents and the baseline controller are compared in the task of reaching the upwind target in both the simulation and basin (real-world) environments.

The SRCS performed very well in both the simulation and basin. In each environment it was able to complete the tasks all but one time, resulting in a success rate of 97.2% in the simulation environment over 36 runs and 93.8% across 16 runs in the basin environment. The average completion time is shorter in the simulation compared to the basin for the SRCS as well as the RL agents. The only basin run the SRCS failed to reach the target line is shown in Fig. 13, where the Optimist was not able to generate enough speed to sail upwind. Even in the failed run, the SRCS showed robust behavior in the sense that it still sailed within the boundaries and did not engage in a risky behavior. These results underscore the robustness and reliability of the classic control methods in general and the SRCS in particular.

The SAC agent (trained without domain randomization) achieved a success rate of 100% in the simulations but with varying completion times. These are due to the different initial conditions it was evaluated on, which were not encountered during training. It was still able to complete the task every time. In the basin however, it encountered more of a challenge, with its success rate falling to 56.2% over 16 trials. The agent was able to do quite well with the 40% wind speed setting in the basin, with most runs succeeding from the different

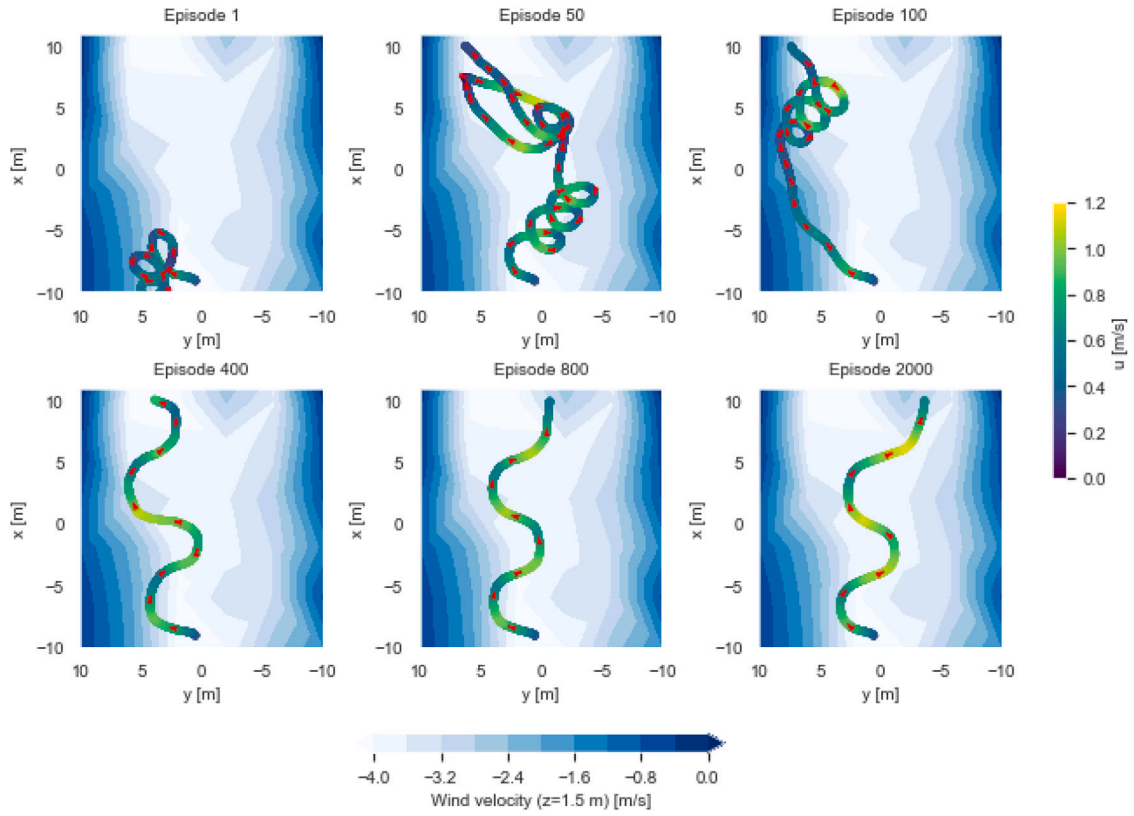


Fig. 10. Paths of the boat in different episodes as the policy improves during training.

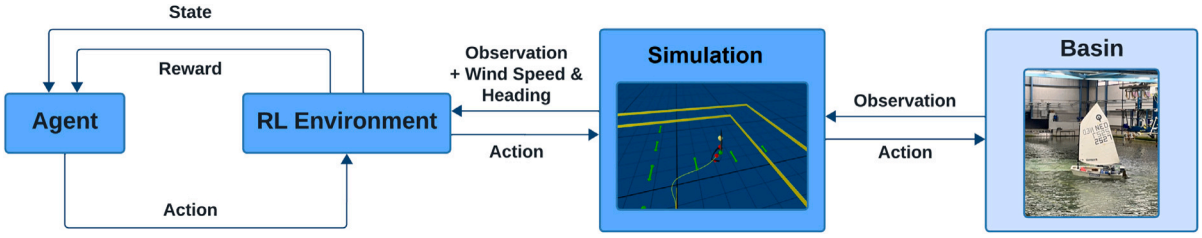


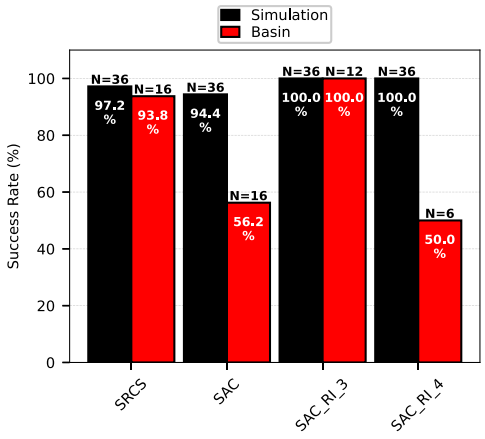
Fig. 11. Sim-to-real deployment architecture. During basin experiments, the simulation environment was used solely as a communication interface between the RL agent and the physical system.

starting positions. However, it exhibited "cheating" behaviors in the basin; once it gained a bit of speed it would turn into the wind and use quick movements of the rudder (*sculling*) and/or the ballast (*rocking*) to propel itself as evidenced by its inferior energy ratio compared to the other controllers, see Fig. 12(b). This could explain why most of the failures in the basin occurred with the 60% wind speed setting as the wind was too strong to succeed by trying to go directly into it. The only times it was able to complete the task with the 60% wind speed setting were from the similar starting position as it was trained on, with 2 out of the 3 runs being successful as shown in Fig. B.18. As such, the SAC agent's policy was not generalizable to conditions unseen during training.

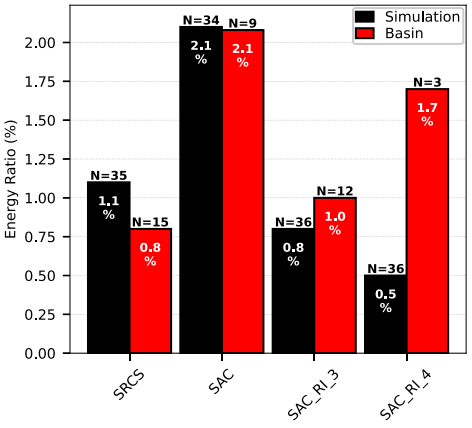
A particularly intriguing aspect of the study is the comparison between SAC_RI_3 and SAC_RI_4, both of which were subjected to the same domain randomization training. Despite identical training setups and comparable performances in the simulation, their performances in the basin were quite different. SAC_RI_3 maintained an impressive success rate of 100% over 12 runs, mirroring its simulation performance. However, SAC_RI_4, which had demonstrated a 100% success rate in simulation, saw its basin success rate halved to 50% across 6 runs. A Fisher's Exact Test yielded a p-value of 0.025, indicating a statistically significant difference in performance between the two agents. This discrepancy between SAC_RI_3 and SAC_RI_4 highlights the inherent variability in learned policies by RL agents, even when identical training setups are employed. This variability arises from the stochastic nature of RL training and the sailing task's sensitivity to small control differences, which can lead agents to converge to different learned behaviors that are further amplified by real-world uncertainties and disturbances. As a result, even extensive simulation evaluation may not fully predict an RL agent's real-world performance, especially when the agent is not exposed to such phenomena during training.

SAC_RI_4 succeeded in half the runs in the basin, with all of the successful runs being at the wind speed setting of 40%. In the failed runs at the 60% wind speed setting (such a run is illustrated in Fig. B.19), it was able to sail quite well for a while, staying within the boundaries and demonstrating that it was trying to go upwind by beating. However, it was unable to make enough progress going upwind due to drifting after a tack maneuver, thus running out of time.

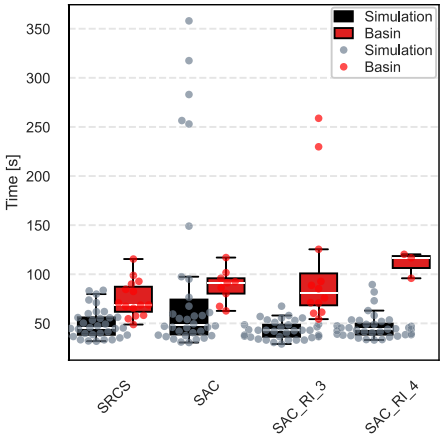
To compare the success rates of SAC_RI_3 and the SRCS, Fisher's Exact Test was applied, yielding a high p-value of 1.0. This outcome indicates that there is no statistically significant difference in the success rates between the two controllers at the conventional significance



(a) Success rate



(b) The lower the energy ratio, the more effective the controllers are in utilizing the wind energy.



(c) Completion time

Fig. 12. Performance of the SRCS and RL agents in the simulation and basin.

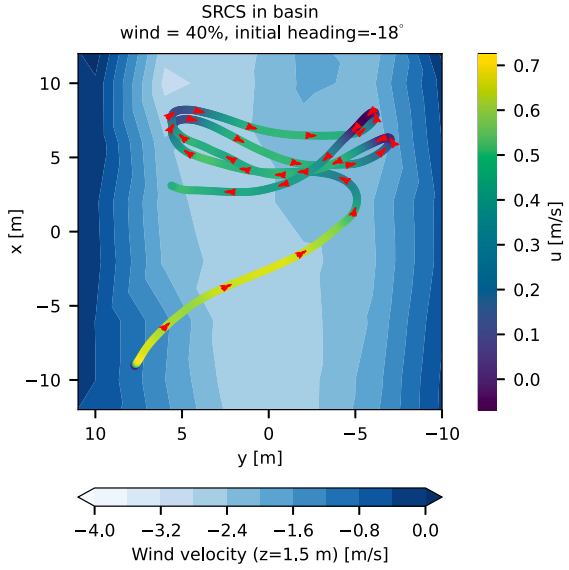


Fig. 13. The SRCS's single failed run in the basin, showing that the speed loss after the tacks is detrimental.

levels. Furthermore, except for two outliers, the completion times of the SAC_RI_3 agent are generally comparable to those of the SRCS. This is underscored using the Mann–Whitney U test resulting in a p-value of 0.252, indicating that the distributions of the data are similar.

In Fig. 12(b), the energy ratios of all the successful runs of the agents are summarized. The energy ratios for the SRCS and SAC_RI_3 are quite low suggesting that they mainly use the wind energy to propel themselves in reaching the target. Furthermore, their energy ratios are comparable when transferring from sim-to-real. The SAC agent has the most inferior energy ratio, which is in line with the observations described above. The SAC_RI_4 agent has a favorable energy ratio in simulations, but its behavior worsens significantly when transferred to the basin.

4.4. Demonstration of advanced sailing capability

During the runs in the basin with the SAC_RI_3 agent, two behaviors caught attention particularly. These are explained below in detail.

Roll tacking: Using ballast effectively during tacking

Roll tacking is a technique used in sailing, especially in competitive dinghy or small keelboat racing, to optimize a boat's performance during a tack. It involves intentionally heeling and rolling the boat during the maneuver to maintain or even gain speed while changing direction.

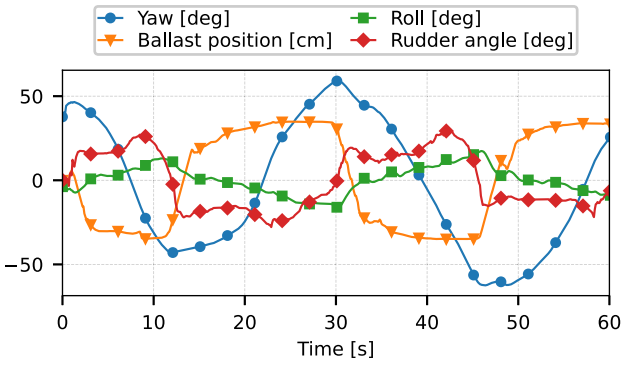


Fig. 14. Time traces of yaw, roll, ballast position and rudder angle with the SAC_RI_3 agent from a run in the basin where roll tacking strategy was observed. The path of the boat from this run is shown in Fig. 15.

The SAC_RI_3 agent employed roll tacking quite often in the basin. Fig. 14 shows time traces of four motions, and Fig. 15 illustrates the path of the boat from such a run. At each tack, the agent moves the ballast until the allowed limit to leeward (the side away from the wind). This helps the boat turn more easily and reduces drag on the rudder. After the roll, the agent flattens the boat by quickly moving the ballast back toward the centerline. The agent uses this strategy in a systematic way for each tack as can be seen in Fig. 14. Snapshots from this run during a roll tacking are displayed in Appendix A. Note that since the SRCS does not control the ballast position, a comparison is not possible.

Recovering from a stuck-in-the-wind situation

Fig. 12(c) illustrates that two runs with the SAC_RI_3 agent took significantly more time to complete compared to the rest of the runs. One of the two runs was actually a repeat run to reproduce the behavior observed in the first one.

Fig. 16 shows the path of the Optimist in the basin during the two runs. They were both started with similar initial conditions ($y = 7.5$ m, 40% wind and initial headings of $-41°$ and $-59°$) and they both ended up taking around 250 s to complete. During these runs, the agent got stuck in the wind at similar positions in the basin ($y = 0$ m, $x = 8$ m) with no forward speed. However, from this situation it sailed downwind back towards the start line, increased its speed and made a second attempt to carry out a series of tacks. It was able to resume sailing and successfully reached the target line. The SRCS, SAC and SAC_RI_4 also got stuck in various situations shown in Figs. 13, B.18 and B.19, respectively, but they were not able to recover.

5. Conclusions

This study investigates sim-to-real reinforcement learning for autonomous sailing, focusing on transferring control policies from simulation to real-world environments. By training RL agents in simulation and subsequently testing them in a controlled basin environment, we addressed critical challenges associated with the reality gap, robustness, and adaptability of autonomous sailing systems.

Although identical setups were used in simulated training and in the basin, the RL agents exhibited significant performance differences in the two environments. For example, SAC_RI_4, while achieving a 100% success rate in simulations, displayed only a 50% success rate in the basin. This suggests that the performance of the RL agents in the simulation might not give an accurate indication regarding their performance in the real world. More advanced evaluation techniques in the simulation environment might mitigate this issue, possibly providing a more realistic assessment for the suitability of the RL agents prior to real-world deployment.

Quantifying the reality gap revealed discrepancies between simulation and real-world conditions, primarily due to actuator latency and modeling inaccuracies. While the rudder angle signals aligned closely between simulation and reality, the sheet length signals exhibited greater deviations, contributing to the observed reality gap. Moreover, the reality gap was found to be more significant in areas with higher wind speeds and during maneuvers.

Despite the challenges posed by the reality gap, our results show that the RL agents demonstrated the capability to learn complex sailing strategies. The introduction of domain randomization during training in simulation significantly enhanced the robustness of the agents, enabling them to adapt to diverse conditions in the basin environment. When transferring RL-trained policies to the basin environment, the results highlighted that the SAC_RI_3 agent trained with domain randomization achieved higher success rates and demonstrated greater adaptability to real-world conditions compared to the SAC agent trained without randomization. SAC_RI_3 consistently performed well in the basin, achieving a success rate of 100% in 12 runs. The baseline SRCS controller displayed robust performance in both simulation and the basin, underscoring the reliability of traditional control methods. Notably, the SAC_RI_3 agent matched the success rate of SRCS while demonstrating advanced behaviors, such as roll tacking and recovering from being stuck in the wind. This ability to recover from difficult scenarios highlights the advantage of RL in handling dynamic and unpredictable environments, where traditional control methods may falter. While the basin experiments demonstrate the feasibility of sim-to-real RL for autonomous sailing, the limited number of basin trials and the variability observed between independently trained agents indicate that more comprehensive real-world testing is needed to fully characterize the robustness and reproducibility of the learned policies.

Although domain randomization improved robustness, it was limited to variations in the initial conditions and wind settings, and did not account for the identified discrepancies in actuator latency and modeling inaccuracies. Actuator latency leads to delayed corrective actions, while inaccuracies in sheet modeling directly affect forces on the sail, thereby affecting upwind progress. Since the agents were not explicitly exposed to these sources of the reality gap during training, their performances might differ when exposed to such uncertainties in the basin. This may also have contributed to the performance difference observed between the SAC_RI_3 and SAC_RI_4 agents.

As the experiments in this study were conducted in a controlled basin environment, a promising direction for future research on sim-to-real transfer in autonomous sailing would be to evaluate the agents in open-water conditions. Such evaluations would provide valuable insight into the generalization capability of the learned policies and serve as a baseline for further improvements. In open-water conditions, additional challenges are introduced such as more complex and time-varying wind fields, wave-induced disturbances, and currents. These phenomena directly affect vessel dynamics, sensor measurements, and control effectiveness while also introducing partial observability challenges. For example, wave-induced roll may interfere with the learned roll-tacking strategy, while currents may affect the agent's perception of progress toward the target. To account for these effects, training and evaluation could be extended using mechanisms such as dynamics randomization, online adaptation, or augmented state estimation. Furthermore, given that actuator latency and modeling inaccuracies were identified as significant contributors to the reality gap, extending domain randomization to such effects could further improve the robustness of sim-to-real transfer. Developing systematic methods to measure and minimize the reality gap would also be a fruitful avenue for future research. This would provide deeper insights into simulation fidelity and real-world transfer. Finally, hybrid control systems combining RL with traditional control methods may be promising, leveraging the strengths of RL in handling dynamic and unpredictable scenarios while maintaining the reliability of traditional control for standard conditions.

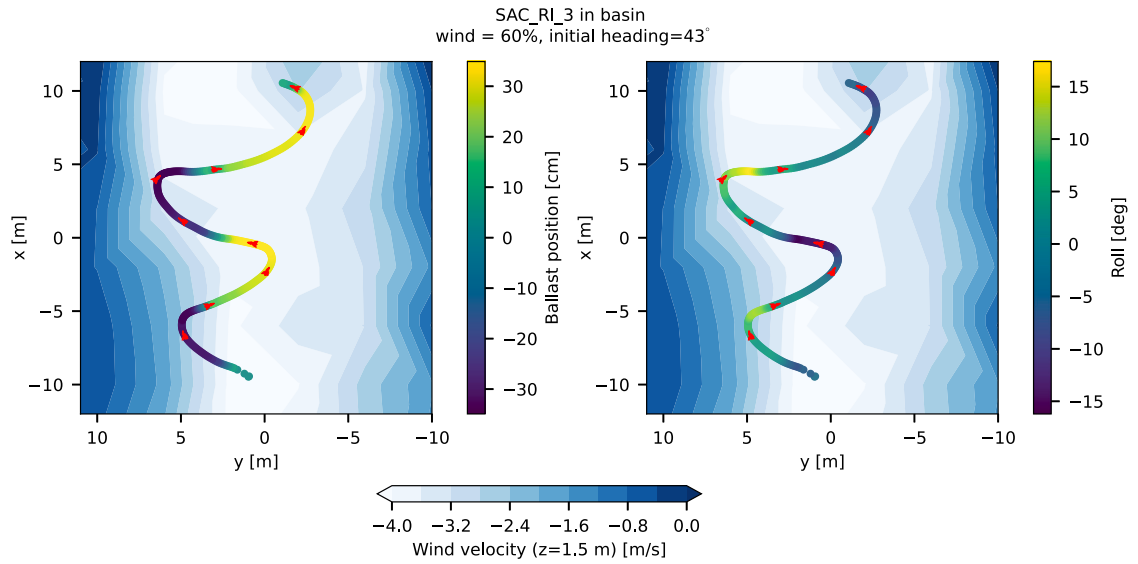


Fig. 15. Roll tacking behavior from the agent. The agent moves the ballast to the maximum limits during tacking (left plot), which causes large roll angles (right plot).

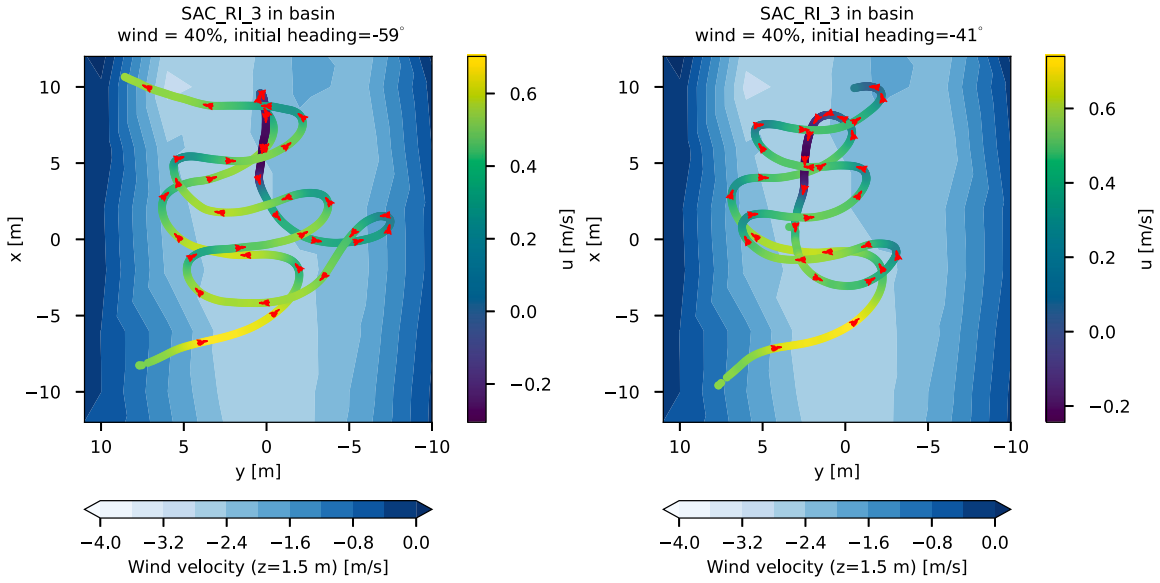


Fig. 16. Two basin runs with the SAC_RI_3 agent which got stuck in the wind but recovered itself and reached the target line.

CRediT authorship contribution statement

Kiki J.A. Bink: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Bülent Düz:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Software, Resources, Methodology, Investigation, Data curation, Conceptualization. **Gabriel D. Weymouth:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to express their sincere gratitude to Rudy Negenborn, Eelco Frickel, Bart Mak, Fanny Rebiffe, Olivier Schnitzeler, Douwe Rijpkema, Jacco van Soest, Victor Ferrari, Julio Polo and Hannes Bogaert for the insightful discussions during the project and their support in preparing the manuscript.

Appendix A. Supplementary material for roll tacking behavior

The figures in Appendix A and Appendix B are provided to enhance the reader's experience by presenting detailed visual information separately, ensuring the main text remains concise and focused.

Fig. A.17 shows six snapshots from the basin run with the SAC_RI_3 agent analyzed in Fig. 14 and Fig. 15.

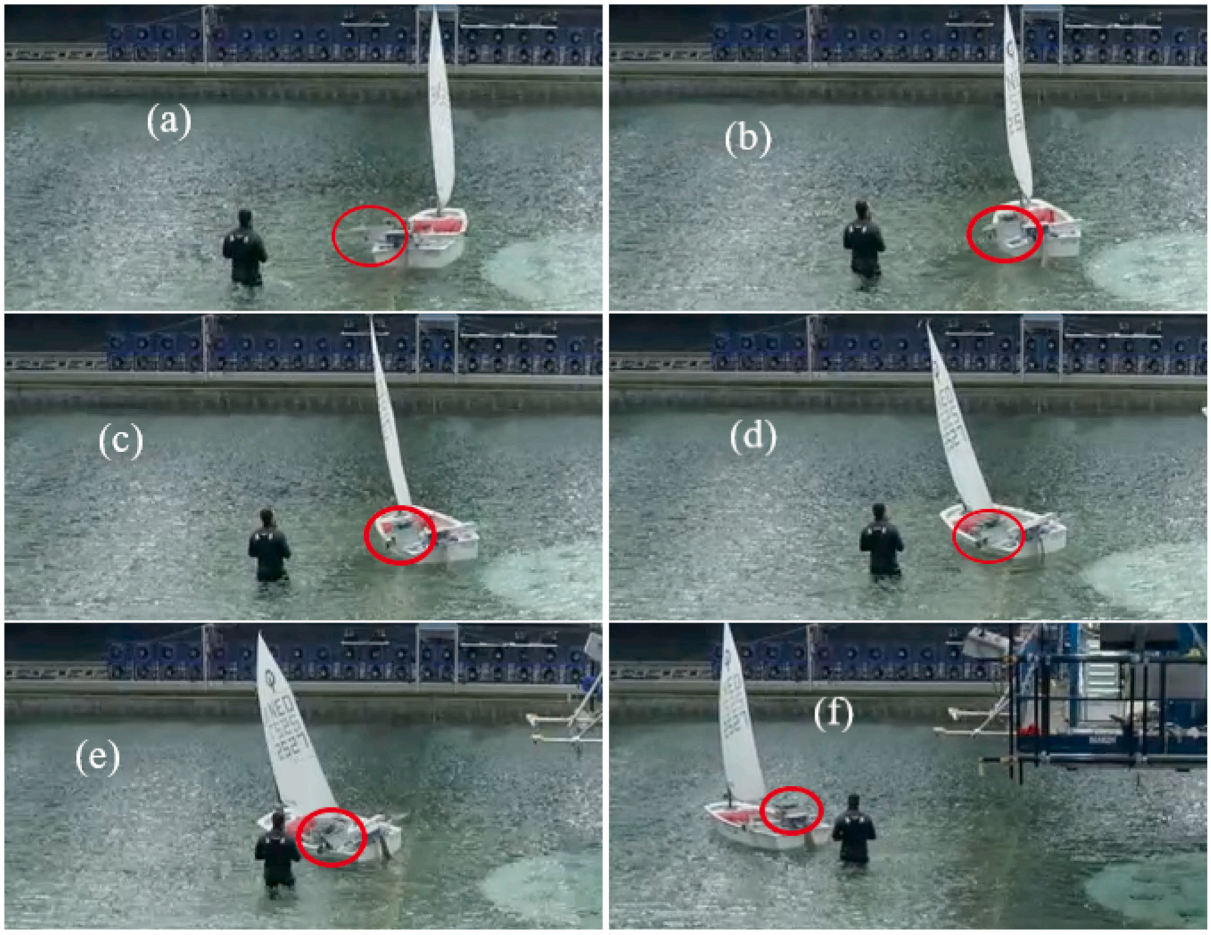


Fig. A.17. Roll tacking with the SAC_RI_3 agent in the basin. Snapshots are ordered alphabetically. The red circle shows the location of the moving ballast. The agent keeps the ballast at the maximum limit on the leeward side (the side away from the wind) during tacking, see the snapshots from (a) to (e). After tacking, the agent flattens the boat by quickly moving the ballast back toward the centerline as shown in the snapshot (f).

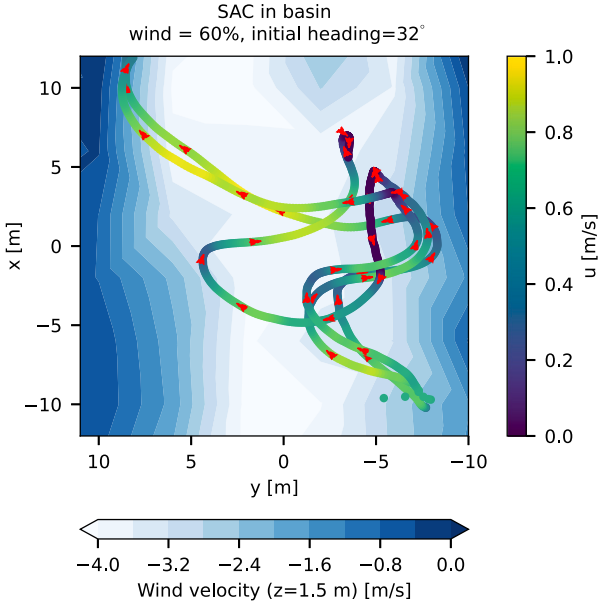


Fig. B.18. Three basin runs with the SAC agent at 60% wind speed setting. The agent reaches the target line in two runs while it fails in the third as it gets stuck in the wind at the position of ($x \approx 7\,\mathrm{m},\ y \approx -4\,\mathrm{m}$).

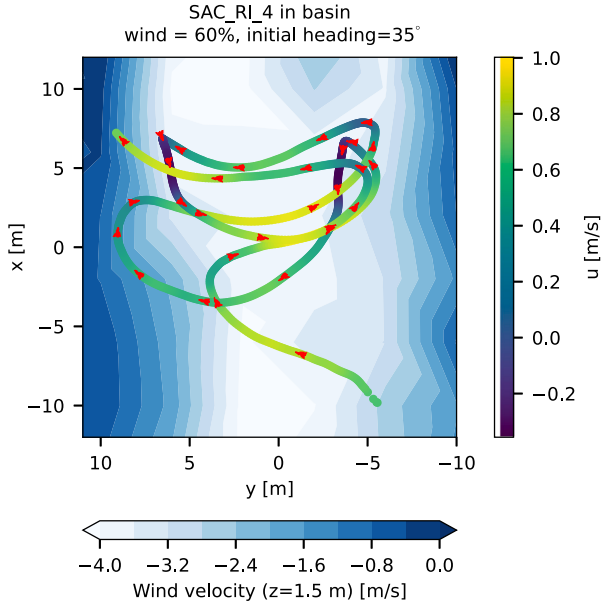


Fig. B.19. A failed run with the SAC_RI_4 agent at the 60% wind speed setting.

Appendix B. Supplementary results from the SAC and SAC_RI_4 agents

Trajectories of the Optimist from the basin runs with the SAC and SAC_RI_4 agents are provided in Figs. B.18 and B.19. We especially focus on the runs where the agents fail to investigate the weaknesses in their policies.

Data availability

The authors do not have permission to share data.

References

Andersson, A., Barreng, A., Bohnsack, E., Larsson, L., Lundin, L., Olander, G., Sahlberg, R., Werner, E., Finnsgård, C., Persson, A., Brown, M., McVeagh, J., 2018. Design of a foiling optimist. J. Sail. Technol. 3 (01), 1–24. http://dx.doi.org/10.5957/jst.2018.06.

Bink, K.J.A., 2024. Autonomous Sailing with Sim-to-Real Reinforcement Learning (Master's thesis). Faculty of Mechanical Engineering, Delft University of Technology (TU Delft), The Netherlands, URL: https://resolver.tudelft.nl/uuid:1284613a-d9e7-4076-b120-8349acdca4ca.

Chen, C., Chen, X.Q., Ma, F., Zeng, X.J., Wang, J., 2019. A knowledge-free path planning approach for smart ships based on reinforcement learning. Ocean Eng. 189, 106299. http://dx.doi.org/10.1016/j.oceaneng.2019.106299.

Chen, C., Ma, F., Liu, J., Negenborn, R.R., Liu, Y., Yan, X., 2020. Controlling a cargo ship without human experience using deep Q-network. J. Intell. Fuzzy Syst. 39 (5), 7363–7379. http://dx.doi.org/10.3233/JIFS-200754.

Chou, T., Kosmas, V., Acciaro, M., Renken, K., 2021. A comeback of wind power in shipping: An economic and operational review on the wind-assisted ship propulsion technology. Sustainability 13 (4), 1880. http://dx.doi.org/10.3390/SU13041880.

Chun, D.H., Roh, M.I., Lee, H.W., Yu, D., 2024. Method for collision avoidance based on deep reinforcement learning with path-speed control for an autonomous ship. Int. J. Nav. Archit. Ocean. Eng. 16, 100579. http://dx.doi.org/10.1016/j.ijnaoe.2023.100579.

da Silva Junior, A., Dos Santos, D., de Negreiros, A., Silva, J., Gonçalves, L., 2020. High-level path planning for an autonomous sailboat robot using Q-learning. Sensors (Switzerland) 20 (6), http://dx.doi.org/10.3390/s20061550.

Deng, Y., Zhang, X., Zhang, G., 2020. Line-of-sight-based guidance and adaptive neural path-following control for sailboats. IEEE J. Ocean. Eng. 45 (4), 1177–1189. http://dx.doi.org/10.1109/JOE.2019.2923502.

Erckens, H., Büsser, G.A., Pradalier, C., Siegwart, R.Y., 2010. Avalon: Navigation strategy and trajectory following controller for an autonomous sailing vessel. IEEE Robot. Autom. Mag. 17 (1), 45–54. http://dx.doi.org/10.1109/MRA.2010.935792.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. ML Research Press, PMLR, Cambridge, MA, USA, pp. 249–256, URL: https://proceedings.mlr.press/v9/glorot10a.html.

Greep, J., Bayezit, A.B., Mak, B., Rijpkema, D., Kınacı, Ö.K., Düz, B., 2025. Ship course-keeping in waves using sample-efficient reinforcement learning. Eng. Appl. Artif. Intell. 141, 109848. http://dx.doi.org/10.1016/j.engappai.2024.109848.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018a. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning. ML Research Press, PMLR, Cambridge, MA, USA, pp. 1861–1870, URL: https://proceedings.mlr.press/v80/haarnoja18b.html.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S., 2018b. Soft Actor-Critic algorithms and applications. arXiv:1812.05905.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D., 2018. Deep reinforcement learning that matters. In: 32nd AAAI Conference on Artificial Intelligence. AAAI 2018, pp. 3207–3214. http://dx.doi.org/10.1609/AAAI.V32I1.11694.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M., 2019. Learning agile and dynamic motor skills for legged robots. Sci. Robot. 4 (26), http://dx.doi.org/10.1126/scirobotics.aau5872.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., Levine, S., 2021. How to train your robot with deep reinforcement learning: lessons we have learned. Int. J. Robot. Res. 40 (4–5), 698–721. http://dx.doi.org/10.1177/0278364920987859.

IMO, 2023. IMO adopts revised strategy to reduce greenhouse gas emissions from international shipping. https://www.imo.org/en/MediaCentre/PressBriefings/Pages/Revised-GHG-reduction-strategy-for-global-shipping-adopted-.aspx. (Online; accessed 11 January 2025).

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D., 2023. Champion-level drone racing using deep reinforcement learning. Nature 620 (7976), 982–987. http://dx.doi.org/10.1038/s41586-023-06419-4.

Kimball, J., 2009. Physics of Sailing. CRC Press, http://dx.doi.org/10.1201/9781420073775.

Larsen, T.N., Teigen, H.Ø., Laache, T., Varagnolo, D., Rasheed, A., 2021. Comparing deep reinforcement learning algorithms' ability to safely navigate challenging waters. Front. Robot. AI 8, 738113. http://dx.doi.org/10.3389/FROBT.2021.738113.

Lemaire, S., Cao, Y., Kluyver, T., Hausner, D., Vasilovici, C., Lee, Z.Y., Varbaro, U.J., Schillai, S.M., 2018. Adaptive probabilistic tack manoeuvre decision for sailing vessels. Proc. Int. Robot. Sail. Conf. 95–103, URL: https://eprints.soton.ac.uk/429158/.

Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., Scaramuzza, D., 2020. Deep drone racing: From simulation to reality with domain randomization. IEEE Trans. Robot. 36 (1), 1–14. http://dx.doi.org/10.1109/TRO.2019.2942989.

NDI, 2024. Legacy products: NDI's 40-year history and transition. URL: https://www.ndigital.com/products/legacy-products/.

Øvereng, S., Nguyen, D., Hamre, G., 2021. Dynamic positioning using deep reinforcement learning. Ocean Eng. 235, 109433. http://dx.doi.org/10.1016/j.oceaneng.2021.109433.

Quadvlieg, F.H.H.A., Rapuc, S., 2019. A pragmatic method to simulate maneuvering in waves. In: SNAME Maritime Convention. Tacoma, Washington, USA.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N., 2021. Stable-Baselines3: Reliable reinforcement learning implementations. J. Mach. Learn. Res. 22 (268), 1–8, URL: http://jmlr.org/papers/v22/20-1364.html.

Saoud, H., Hua, M., Plumet, F., Ben Amar, F., 2015. Routing and course control of an autonomous sailboat. In: European Conference on Mobile Robots, ECMR. http://dx.doi.org/10.1109/ECMR.2015.7324218.

Stelzer, R., Pröll, T., John, R.I., 2007. Fuzzy logic control system for autonomous sailboats. IEEE Int. Conf. Fuzzy Syst. http://dx.doi.org/10.1109/FUZZY.2007.4295347.

Suda, T., Nikovski, D., 2022. Deep reinforcement learning for optimal sailing upwind. In: International Joint Conference on Neural Networks. IJCNN, pp. 1–8. http://dx.doi.org/10.1109/IJCNN55064.2022.9892369.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., Vanhoucke, V., 2018. Sim-to-real: Learning agile locomotion for quadruped robots. Robot.: Sci. Syst. http://dx.doi.org/10.15607/RSS.2018.XIV.010.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P., 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, pp. 23–30. http://dx.doi.org/10.1109/IROS.2017.8202133.

Ulysse, D., Niklas, R., Jakob, K., 2019. Energy efficient self-steering mechanism for an autonomous sailing vessel. In: OCEANS 2019 - Marseille. Marseille, pp. 1–6. http://dx.doi.org/10.1109/OCEANSE.2019.8867310.

Wada, D., Araujo-Estrada, S., Windsor, S., 2022. Sim-to-real transfer for fixed-wing uncrewed aerial vehicle: Pitch control by high-fidelity modelling and domain randomization. IEEE Robot. Autom. Lett. 7 (4), 11735–11742. http://dx.doi.org/10.1109/LRA.2022.3205442.

Wang, N., Wang, Y., Zhao, Y., Wang, Y., Li, Z., 2022. Sim-to-real: Mapless navigation for USVs using deep reinforcement learning. J. Mar. Sci. Eng. 10 (7), http://dx.doi.org/10.3390/jmse10070895.

Xu, J., Huang, F., Wu, D., Cui, Y., Yan, Z., Du, X., 2022. A learning method for AUV collision avoidance through deep reinforcement learning. Ocean Eng. 260, 112038. http://dx.doi.org/10.1016/j.oceaneng.2022.112038.

Zhao, W., Queralta, J.P., Westerlund, T., 2020. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In: IEEE Symposium Series on Computational Intelligence. SSCI, Institute of Electrical and Electronics Engineers Inc., pp. 737–744. http://dx.doi.org/10.1109/SSCI47803.2020.9308468.

Zheng, Y., Tao, J., Sun, Q., Sun, H., Chen, Z., Sun, M., Xie, G., 2022. Soft Actor–Critic based active disturbance rejection path following control for unmanned surface vessel under wind and wave disturbances. Ocean Eng. 247, 110631. http://dx.doi.org/10.1016/j.oceaneng.2022.110631.