

Article

Deep Reinforcement Learning Control for Path Following and Static Obstacle Avoidance for Autonomous Surface Vessels

Nam Tran ¹, Hung Duc Nguyen ¹, Peter King ^{1,*} and Minh Tran ^{1,2}

¹ Australian Maritime College, University of Tasmania, Maritime Way, Newnham, Tasmania 7250, Australia; hoangnam.tran@utas.edu.au (N.T.); h.d.nguyen@utas.edu.au (H.D.N.); minh.tranquang@rmit.edu.vn (M.T.)

² School of Science, Engineering & Technology, RMIT University Vietnam, 702 Nguyen Van Linh Boulevard, Tan Hung Ward, Ho Chi Minh City, Vietnam

* Correspondence: p.d.king@utas.edu.au

Highlights

What are the main findings?

- A LiDAR-based deep reinforcement learning policy (Soft Actor-Critic) enables an underactuated ASV to follow a reference path while avoiding static obstacles in a narrow environment, evaluated statistically in simulation and demonstrated in single-trial model-scale field experiments.
- A feasibility-inspired LiDAR sector pooling design provides compact obstacle features for policy input and supports evaluation of LiDAR configuration.

What are the implications of the main findings?

- The study provides a practical baseline for ASV navigation using local SLAM pose and LiDAR sensing, suitable for constrained waterways.
- The sim-to-field evaluation protocol and modular observation design can be extended towards dynamic obstacle avoidance and COLREG-based decision-making in future work.

Abstract

Autonomous surface vessels (ASVs) operating in narrow and restricted waterways must follow a planned path while avoiding nearby static hazards and maintaining safe clearance from boundaries. This paper presents a LiDAR-based deep reinforcement learning framework for path following and static obstacle avoidance of an underactuated ASV. The vessel receives local pose information from a localization system and surrounding environment through a 2D LiDAR scan, which is converted into compact sector features using feasibility-inspired pooling method. A Soft Actor-Critic (SAC) policy is trained in simulation to output continuous rudder and propulsion commands, based on LiDAR features, estimated motion states, and path-relative errors. The policy is evaluated over 500 randomized simulation episodes ranging from 0–4 obstacles. The trained policy achieved an overall success rate of 95.0%, with an average cross-track error of 0.66 m. Obstacle and border collision rates are 3.80% and 1.20%, respectively; indicating that the policy can perform path tracking and collision avoidance in constrained layouts. A single field trial was then conducted in each of three fixed obstacle layouts using the model-scale Bluefin vessel. In these trials the policy executed on the physical platform and avoided static obstacles, with minimum obstacle clearances of 0.50–1.17 m. However, the field trajectories exhibit larger oscillations and longer path lengths than simulation, with an average RMS cross-track error of 1.14 m compared to 0.69 m in simulation.

Academic Editor: Lu Liu

Received: 27 July 2026

Revised: 31 August 2026

Accepted: 4 September 2026

Published: 7 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).

Keywords: autonomous surface vessel; LiDAR; deep reinforcement learning; Soft Actor-Critic; path following; collision avoidance; restricted waterways; field experiments

1. Introduction

Autonomous Surface Vessels (ASVs) are increasingly deployed for monitoring, inspection, and logistics in coastal and inland waters. Many practical missions take place in confined or restricted waterways, such as harbors, canals and narrow channels, where available maneuvering space is limited and small navigation errors can quickly lead to collision with quay walls or grounding. Recent work on autonomous navigation in confined waters emphasizes that boundary constraints and nearby static structures change the decision-making problem compared with open-water operation, motivating specific methods to narrow environments [1,2]. In restricted waterways, static obstacle avoidance is not simply a local detour problem. Because the available clearance is limited, avoiding an obstacle can immediately increase the risk of boundary contact or excessive deviation from the survey path. Therefore, path following, obstacle avoidance, and boundary clearance must be treated as a coupled control problem rather than independent modules. This motivates the present focus on static-obstacle navigation in narrow environments as a necessary step toward reliable ASV autonomy before introducing dynamic obstacle encounters.

Surveys of ASV autonomy consistently highlight that collision avoidance and motion planning remain as central bottlenecks, emphasizing that practical solutions must be robust, real-time, and compatible with marine vehicle dynamics and sensing limitations [3–5]. Recent literature on marine collision avoidance is driven by dynamic encounters and COLREGs compliance, where an autonomous vessel must navigate among multiple moving targets and rule-based behaviors [6,7]. This focus is well justified for open-water autonomy, however, for many restricted-water missions, the most frequent and immediate risk is collision with static obstacles, such as quay walls or narrow channel boundaries, where even small tracking errors can lead to contact. Static-obstacle navigation is also a practical foundation: a system that cannot reliably avoid fixed hazards in confined corridors is unlikely to be dependable once multi-vessel reasoning is added.

Classical ASV navigation methods combine model-based guidance laws with local planners or reactive avoidance layers. Marine guidance and control references, such as line-of-sight (LOS), offer convergence properties for path following, but they rely on a controller that can realize the commanded course and speed changes under hydrodynamics and actuator constraints [8]. When obstacles are introduced, an additional layer is often required to select avoidance maneuvers while maintaining progress along a reference path. Implementations in confined waters often utilize onboard ranging sensors to construct a local obstacle representation and then integrate path following with collision avoidance logic. For example, Villa et al. demonstrated path following with LiDAR-based obstacle avoidance for a surface vessel operating under harbor conditions, using a structured guidance formulation and sensor-driven safety logic [9]. The avoidance layer in that work depends on hand-tuned safety logic, which must be re-tuned when the corridor geometry or the vessel changes. More optimization-based pipelines integrate collision constraints directly into the control problem: Gonzales-Garcia et al. formulated a nonlinear model predictive control (NMPC) strategy that simultaneously tracks a path and enforces obstacle clearance using LiDAR measurements [10]. Performance in that formulation depends directly on the fidelity of the internal prediction model, and obstacle constraints must first be extracted from the scan as explicit geometry before the optimization can be posed. Experimental studies using 2-dimensional LiDAR further illustrate practical issues, such as spurious returns or partial occlusions, that can dominate the performance

gap between simulation and field deployment [11]. While these methods can be effective, they also expose recurring limitations, including reliance on accurate models and tuning, and low adaptability in dense environments [4,5,9].

Machine learning-based methods offer another route where the control policy is learned from data rather than manually tuned parameters. Deep reinforcement learning (DRL) provides a direct way to treat navigation and control as a sequential decision problem: the policy selects control actions that balance between progress, tracking accuracy, and safety [12]. In marine robotics, DRL has shown promise in both path-following control and collision avoidance. Woo et al. demonstrated a DRL controller for USV path following using an actor-critic continuous-control algorithm, known as deep deterministic policy gradient (DDPG), framing path following as a closed-loop control policy learned through interaction rather than explicit controller tuning [13]. Subsequent work has refined state design and training structure for underactuated marine vehicles, including DRL path-following formulations targeting robustness and smoother convergence. Qu et al. proposed a DRL-based path-following control scheme for an underactuated autonomous marine vehicle under uncertainty, illustrating how detailed reward shaping and state construction can stabilize learning and improve tracking behavior [14].

Obstacle avoidance in marine settings is harder than in many ground-robot cases because yaw response is slow, stopping distance is large, and feasible actions are constrained by rudder and thrust limits. Early DRL work on marine vessels showed that a learned policy can produce avoidance actions under disturbances when the reward encourages both safety and progress. Cheng and Zhang proposed a concise DRL approach for obstacle avoidance in underactuated unmanned marine vessels, focusing on usable policies without complex control laws [15]. Meyer et al. trained an agent, using proximal policy optimization (PPO), for an underactuated ASV to follow a known path while avoiding static obstacles, using multiple rangefinder sensors as perception [16]. Their study demonstrates that the combined objective between path tracking and collision avoidance can be achieved when observations and rewards are designed carefully. That evaluation is nevertheless confined to simulation and to open-water geometry, so boundary clearance in a bounded corridor is not addressed.

Woo and Kim targeted the decision-making stage of USV collision avoidance and proposed a DRL approach that uses a grid-map representation of encounter situations [17]. They designed a semi-Markov decision process and a neural network architecture tailored to collision avoidance, then evaluated the trained network in both simulations and collision avoidance experiments. This work shows the value of structured spatial inputs when the environment geometry is complex, although the grid representation grows with workspace size, which constrains its use for onboard execution in narrow channels. More recent works also use spatial encodings to improve learning in crowded settings. Teitgen et al. proposed a collision-grid representation for dense environments with multiple obstacles, illustrating the broader trend of using geometry-aware inputs rather than raw state vectors [18]. In restricted waters, PPO variants have been explored to improve stability and decision-making quality. Hao et al. introduced an improved PPO-based strategy for USV collision avoidance in restricted waters, using a recurrent network (GRU) to improve convergence compared with standard feed-forward structures [19].

Beyond pure simulation training, recent work increasingly targets robustness and transfer from simulation to field validation. Slawik et al. studied domain randomization for DRL path-following control of an ASV by training over a distribution of dynamics and sensing parameters rather than a single model, and compared popular DRL algorithms, including DDPG, PPO, and Soft Actor-Critic (SAC), to improve sim-to-real readiness [20]. Evaluation in that study nevertheless remains in simulation, so the effectiveness of the randomisation for physical transfer is not measured directly. Cross-domain training has

been proposed as another way to reduce the data burden and improve generalization when marine data are expensive. Lambert et al. reported a ground-to-water cross-domain DRL approach for ASV navigation to address training sparsity in marine environments [21]. Across this recent body of work the emphasis has shifted towards transfer to physical platforms and towards richer perception encodings, yet evaluation in the majority of these studies remains confined to simulation, and reported deployments are typically limited in number.

Transfer from simulation to a physical platform has been addressed directly in several recent studies of underactuated surface vessels. Slawik et al. [20] examine domain randomization as a means of improving sim-to-real readiness, Wang et al. [22] report a mapless navigation policy transferred to a physical USV, and Wang et al. [23] deploy a learned tracking controller on a full-scale vessel in natural waters. These studies establish that transfer is achievable for this class of vessel while also showing that residual dynamics mismatch remains the principal obstacle, which is consistent with the field behavior reported in Section 3.4.

Despite this progress, two gaps remain relevant to static obstacle avoidance in narrow waterways. First, while most ML-based marine literature targets dynamic obstacles and COLREGs compliance, fewer studies isolate and deeply evaluate the static-hazard problem in constrained environments. Second, sim-to-real transfer continues to be a practical bottleneck for DRL marine autonomy, motivating cross-domain training and domain randomization strategies. The paper addresses these gaps by developing and validating DRL-based navigation and control system for an ASV operating in a narrow environment, focusing on path following with static obstacle avoidance.

This work extends a prior conference study [24] that established the algorithmic baseline for this task by comparing PPO and SAC under a sweep of the reward weighting coefficient. The present paper differs from that study in the observation representation, the action space, the treatment of boundary clearance, the training procedure, and the experimental scope. Table 1 sets out the differences point by point, so that the incremental contribution of the present work is explicit rather than implied.

Table 1. Differences between the prior conference study [24] and the present work.

Aspect	Prior Conference Study [21]	This Work
LiDAR observation	Raw per-beam ranges supplied directly to the policy	Feasibility-inspired pooling into $M = 25$ sector closeness features that encode whether each sector admits the vessel footprint
Action space	Continuous rudder only; propeller speed held fixed	Continuous rudder and propulsion, using a residual parameterization around a nominal cruise setting
Speed handling	Fixed throughout training and evaluation	Staged three-phase curriculum widening the propulsion band to 25–75% of maximum
Algorithm scope	PPO and SAC compared across reward weighting settings	SAC adopted on the basis of [24]; emphasis shifted to a deployment-oriented formulation
Evaluation protocol	Randomized episodes and fixed qualitative test scenarios	Randomized episodes stratified by obstacle count (0–4), with obstacle and border collision rates reported separately
Platform	Simulation only	Simulation, followed by single-trial deployment on a model-scale Bluefin vessel
Vessel model	3-DOF model with continuous rudder control	3-DOF model calibrated against straight-line and turning-circle maneuvers of the physical Bluefin vessel

Taken together, these differences move the work from an algorithm-selection study conducted entirely in simulation to a deployment-oriented framework in which

perception encoding, actuation, and boundary clearance are treated jointly and the resulting policy is executed on a physical platform.

1. A deployment-oriented DRL navigation framework for an underactuated ASV in restricted waterways, in which path following, static obstacle avoidance, and boundary clearance are treated as a single coupled control problem rather than as independent modules.
2. A feasibility-inspired LiDAR sector pooling scheme that compresses a raw scan into a fixed 25-element closeness vector while preserving whether each sector admits the vessel footprint, a property that neither minimum nor maximum pooling retains.
3. Extension of the action space from rudder-only control to joint rudder and propulsion control, together with a staged propulsion curriculum that stabilizes learning once speed becomes an action rather than a fixed parameter.
4. A two-part evaluation protocol comprising statistical evaluation in simulation across obstacle densities and a single-trial feasibility demonstration on a model-scale platform in matched scenarios, with an explicit statement of what each part can and cannot establish.

The remainder of the paper is organized as follows. Section 2 describes the materials and methods for reproducibility. Section 3 presents the results and discussion, covering simulation study and field experiments. Section 4 concludes the paper and discusses future extensions.

2. Materials and Methods

2.1. System Model and Problem Setup

The ASV operates in a two-dimensional bounded workspace representing a narrow operating area. The environment contains static obstacles that remain fixed during an episode. A collision is declared when the ASV footprint intersects an obstacle or the workspace boundary. During operation, the platform provides a local pose estimate (x, y, ψ) in a resettable SLAM/localization frame, together with a 2D LiDAR scan. A reference path from start to destination point is assumed available from a global path planner. The controller is not provided with the full path as a map. Instead, it receives compact path-relative quantities computed from the local pose and path geometry, including cross-track error, course error, and look-ahead course error. These terms provide a low-dimensional representation of tracking performance and are used to guide the learned policy toward smooth convergence back to the reference path after avoidance maneuvers.

The control policy outputs rudder and propulsion commands at a fixed control period. The commanded actions are constrained by actuator limits and mapped into feasible setpoints before being applied to the vessel. Episodes are finite and truncated after a maximum duration if neither the goal nor a collision event occurs. The objective is to reach the goal region while tracking the reference path and maintaining safe clearance from obstacles and boundaries.

An episode terminates under exactly one of three conditions. First, success, when the vessel enters the goal region within the episode horizon. Second, collision, when the vessel footprint intersects a static obstacle or the workspace boundary. Boundary contact is a terminating collision event, not merely a penalized state, and it is recorded separately from obstacle collision so that the two failure modes can be distinguished in the results. Third, timeout, when the episode horizon is reached without success or collision.

Boundary safety is therefore enforced by two distinct mechanisms operating at different ranges: a graded soft penalty, defined in Section 2.4.4, that shapes behavior as clearance decreases; and the hard termination condition above, which ends the episode on contact. The soft penalty supplements the termination condition and does not replace it.

Figure 1 illustrates the proposed navigation and control framework as a closed-loop pipeline that maps onboard sensing to continuous rudder and propulsion commands.

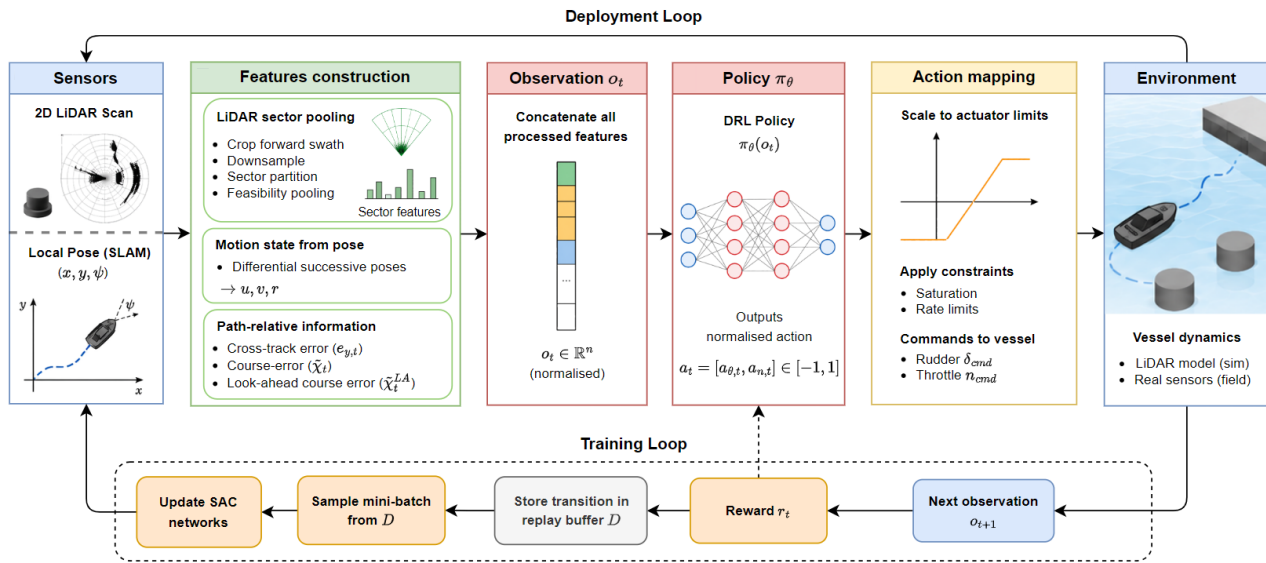


Figure 1. ASV navigation and control workflow.

During training, the environment produces a scalar reward r_t based on the reward function. The transition (o_t, a_t, r_t, o_{t+1}) is stored in a replay buffer to support off-policy SAC updates. For evaluation and field trials, the same closed-loop pipeline is executed, except for learning updates and replay-buffer storage as only the trained policy is used for decision making.

2.2. Vessel Model and Control

The autonomous surface vessel is represented by a physics-informed maneuvering model with three degrees of freedom (3-DOF): surge, sway, and yaw. This formulation is widely used for low-speed surface vessel guidance and collision avoidance, where motion is dominated by the horizontal plane and the effects of heave, roll, and pitch can be neglected without compromising the local navigation objective (Figure 2) [25]. The model combines a rigid-body formulation with calibrated hydrodynamic terms so that it remains computationally efficient for RL simulations, while still reproducing the measured speed and turning behavior of the Bluefin vessel.

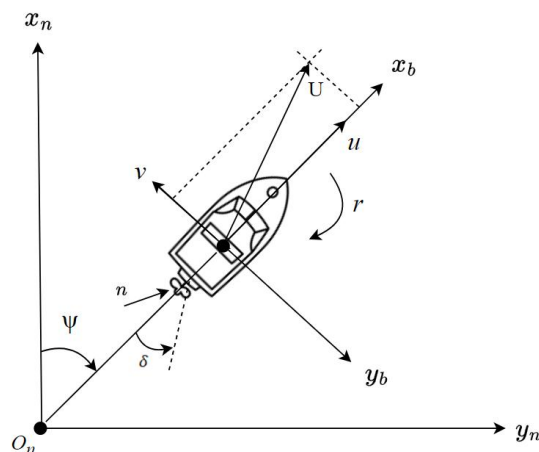


Figure 2. Local earth-fixed frame $\{n\}$ and body-fixed frame $\{b\}$ for the 3-DOF vessel maneuvering model.

Let the vessel pose in the local earth-fixed frame be

$$\eta = [x, y, \psi]^T, \quad (1)$$

where x and y denote the planar position and ψ is the heading angle. The body-fixed velocity vector is written as

$$v = [u, v, r]^T, \quad (2)$$

where u is the surge velocity, v the sway velocity, and r the yaw rate. The kinematic transformation between body-fixed motion and earth-fixed motion is written as

$$\dot{\eta} = R(\psi)v, \quad R(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3)$$

The 3-DOF dynamics are expressed as

$$M\dot{v} + C(v)v + D(v)v = \tau(\delta, n), \quad (4)$$

where M is the effective inertia matrix, $C(v)$ contains Coriolis and centripetal terms, and $D(v)$ represents hydrodynamic damping. The generalized force/moment vector $\tau = [X, Y, N]^T$ is generated by the propeller and rudder, which is commonly decomposed as

$$\tau = \tau_H(v) + \tau_P(u, n) + \tau_R(v, \delta, n), \quad (5)$$

corresponding to hull, propeller, and rudder contributions [25]. The commanded control inputs are rudder angle δ and propeller speed n , while lateral force Y and yaw moment N arise through hydrodynamic interaction. Actuator commands are bounded by $\delta \in [-\delta_{max}, \delta_{max}]$ and $n \in [n_{min}, n_{max}]$. To avoid aggressive maneuvers during simulation rollouts, the rudder and propulsion commands are limited to the allowable range. The equations of motion are integrated with a fixed-step 4th order Runge-Kutta method using the control period Δt . Model parameters are calibrated against straight-line and turning-circle maneuvers of the model-scale vessel such that the surge speed build-up and turning performance in simulation reflect the experimental platform.

The parameters of the vessel model are listed in Appendix A—Table A1.

2.3. LiDAR Observation Design

To enable local path following and static obstacle avoidance in a narrow area, the ASV relies on a forward-looking 2D scanning LiDAR to provide surrounding geometric data without requiring a pre-built map. LiDAR-based mapless policies are widely used in DRL navigation because they offer a compact and meaningful observation that reduces dependency on global localization accuracy [22,26]. Moreover, they can transfer more robustly across simulation and deployment domains when compared to map-dependent planners.

2.3.1. LiDAR Configuration

At time step t , the LiDAR returns a raw scan vector

$$d_t = [d_{t,1}, d_{t,2}, \dots, d_{t,N}]^T, \quad (6)$$

where $d_{t,i} \in [D_{min}, D_{max}]$ is the measured range of beam i and N is the number of raw beams. The physical LiDAR has a valid sensing interval bounded by a measurable range $[D_{min}, D_{max}]$. Measurements outside of this range are treated as invalid and replaced by D_{max} before sector pooling. This configuration follows the practical constraint that the forward sector contains the most relevant collision information for local obstacle collision avoidance in limited environments, while still providing sufficient lateral coverage to detect corridor-like structures [27].

Let $\alpha_i \in \left[-\frac{\Phi}{2}, \frac{\Phi}{2}\right]$ denote the bearing angle of beam i expressed in the ASV body frame, measured relative to the forward axis. In the simulation, each $d_{t,i}$ is computed by ray-casting against obstacle polygons and boundary segments, then clipped to $[0, D_{max}]$.

To ensure consistency when transferring from sim to field, the parameters used in the simulation are kept the same as the real LiDAR used in the field experiment. Details of the LiDAR configuration are listed in Section 3.1.

2.3.2. Sector Pooling

Directly feeding the raw scan vector into the policy increases input dimension and can make learning less stable, particularly when the training distribution includes varied obstacle layouts and narrow passages. Therefore, the raw scan is compressed into a fixed number of angular sectors:

$$S_1, S_2, \dots, S_M, \quad M = 25, \quad (7)$$

where each sector S_j is an adjacent set of beam indices spanning a constant angular interval (Figure 3).

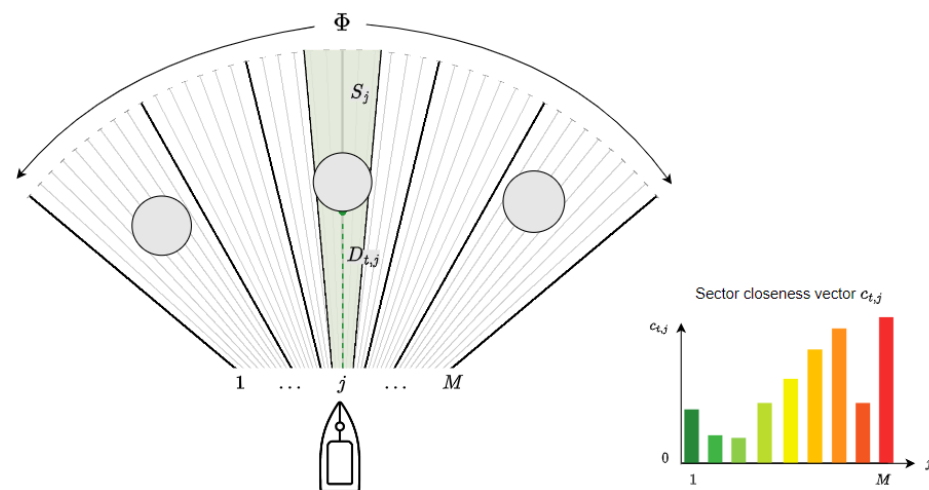


Figure 3. LiDAR sector pooling and closeness vector.

For consistency, the raw scan is denoted d_t throughout the paper and in Algorithm 1; the symbols x_t and x that appeared in earlier statements of the algorithm refer to the same quantity and have been replaced. Individual beam ranges are written $d_{t,i}$, pooled sector distances $D_{t,j}$, and the normalized closeness vector c_t .

Algorithm 1: Feasibility sector pooling for LiDAR

Require:

W_s : safety-adjusted vessel width

Φ : LiDAR swath (FOV)

D_{max} : maximum LiDAR range

N : total number of LiDAR beams

M : number of sectors

x_t : raw LiDAR scan at time t , $x_t = \{x_{t,1}, \dots, x_{t,N}\}$

S_j : sector partitions, $j = 1 \dots M$

Ensure:

$D_t = \{D_{t,1}, \dots, D_{t,M}\}$: pooled sector distances

$c_t = \{c_{t,1}, \dots, c_{t,M}\}$: sector closeness vector

Function: SectorPooling(d_t)

$$\theta \leftarrow \frac{\Phi}{N-1}$$

```

for each sector  $S_j$  do
   $x \leftarrow$  raw beam ranges in  $S_j$ 
   $I \leftarrow$  indices of  $x$  sorted in ascending range
   $D_{t,j} \leftarrow \max(x)$ 
  for  $i$  in  $I$  do
     $x_i \leftarrow x[i]$ 
    if  $x_i \leq 0$  then
      continue
     $d_i \leftarrow \theta x_i$ 
     $y \leftarrow 0.5 d_i$ 
    opening_found  $\leftarrow$  false
    for each beam range  $x_k$  in  $x$ , traversed in angular order do
      if  $x_k > x_i$  then
         $y \leftarrow y + d_i$ 
        if  $y > W_s$  then
          opening_found  $\leftarrow$  true
          break
      else
         $y \leftarrow y + 0.5 d_i$ 
        if  $y > W_s$  then
          opening_found  $\leftarrow$  true
          break
         $y \leftarrow 0$ 
    if opening_found is false then
       $D_{t,j} \leftarrow x_i$ 
      break
   $c_{t,j} \leftarrow \text{clip}(1 - D_{t,j}/D_{max}, 0, 1)$ 
return  $D_t$  and  $c_t$ 

```

The choice of pooling operator is not neutral with respect to safety, and the reason is structural rather than empirical. Minimum pooling returns the smallest range within a sector and maximum pooling the largest; both are statistics of the multiset of ranges falling in that sector, computed without reference to the order in which those ranges occur across its angular span. The quantity the policy requires is not a summary of the ranges but whether the sector admits the vessel, and a sector is traversable at range d only if the returns beyond d leave a contiguous angular opening whose arc width at that range exceeds the safety-adjusted width W_s . That property depends on the angular arrangement of the returns, not only on their values.

The two operators fail in opposite directions, and the consequences are asymmetric. Minimum pooling reports a sector as obstructed whenever any beam within it returns a short range, and so discards traversable headings whenever a thin obstacle, a corner, or a single spurious return falls within the sector; in a confined channel, where the set of feasible headings is already narrow, this is a substantive loss. Maximum pooling reports a sector as clear whenever any beam returns a long range, and so can indicate free passage through a sector the vessel cannot enter. The first failure costs maneuvering room; the second removes the indication of an obstruction the policy is required to avoid. Feasibility-inspired pooling evaluates traversability directly and is subject to neither failure by construction. It reduces to minimum pooling in the limit where W_s approaches the sector arc width, since every return then closes the sector, and to maximum pooling in the limit W_s approaches zero, since no return does; between these limits it is equivalent to neither.

$$D_{t,j} = \text{Pool}(\{d_{t,i} | i \in S_j\}), \quad j = 1, 2, \dots, M. \quad (8)$$

Instead of using minimum or maximum pooling, feasibility-inspired pooling principle is adopted so that each sector feature reflects not only the nearest return, but also whether the sector can accommodate the vessel's footprint at increasing distances. The intuition is that a single short return, such as a corner or a thin obstacle, should not automatically dominate the entire sector if there exists a sufficiently wide free area in a neighboring direction. Similar motivations appear in low-dimensional range representations for DRL navigation, where sparse or compressed scans are designed to preserve actionable structure while reducing learning complexity [27].

Algorithm 1 describes the feasibility sector pooling used to compress a raw LiDAR scan into fixed-length sector features. For each sector, candidate range levels are examined in ascending order. At each candidate level, beams farther than the candidate range are treated as free, while beams at or nearer than the candidate range close the current opening. The pooled distance $D_{t,j}$ is the first candidate range at which no opening wider than the safety-adjusted vessel width W_s remains. If all candidate levels remain feasible, the pooled distance is set to the farthest measured range in the sector. The pooled distance is then normalized into the bounded closeness feature $c_{t,j}$.

2.3.3. Boundary Visibility

A key sim-to-field consideration in narrow environments is that the boundary may not appear symmetrically in the LiDAR scan due to sensor placement and platform geometry, where one wall is visible while the opposite wall is partially occluded. To emulate this and prevent policies from overfitting to a single boundary sensing, the environment supports different border visibility modes for the observation LiDAR scan. However, the true collision boundary is still enforced geometrically for termination and safety penalties. This separation of what the policy sees versus what counts as collision is consistent with sensor-level mapless USV navigation studies that emphasize realism in perception while maintaining strict safety evaluation [22,26].

In addition, when shaping the obstacle avoidance reward, the obstacle proximity term is computed from a LiDAR scan that excludes borders, ensuring that obstacle avoidance learning is not dominated by persistent wall returns. Boundary interactions are instead handled by a border-clearance penalty term, using the minimum geometric distance between the vessel and the map boundary. This penalty operates in addition to, and not in place of, the hard termination condition defined in Section 2.1: contact with the boundary terminates the episode and is counted as a border collision.

2.4. MDP Elements for DRL Guidance Navigation and Control

2.4.1. Reinforcement Learning Algorithm

To formalize the closed-loop interaction, the navigation task is modelled as a partially observable Markov decision process (POMDP) $\langle S, A, P, R, \Omega, O, \gamma \rangle$, where S is the state space, A is the action space, P is the transition model, R is the reward function, Ω is the observation space, O is the observation model, and $\gamma \in (0, 1]$ is a discount factor [12]. The underlying state s_t captures the vessel motion and surrounding obstacle configuration, while the agent receives an observation o_t constructed from processed LiDAR features, estimated motion variables and path-relative errors. At each time step t , the policy $\pi_\theta(a_t|o_t)$ selects a continuous action $a_t \in A$. The environment transitions according to $P(s_{t+1}|s_t, a_t)$ and returns a scalar reward $r_t = r(s_t, a_t, s_{t+1})$. The discounted return from time t is defined as

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1}, \quad (9)$$

where $\gamma \in (0, 1]$ is the discount factor and T is the episode horizon determined by success, collision, or timeout. The learning objective is to find policy parameters θ that maximise the expected return $J(\pi_\theta) = \mathbb{E}[G_0]$.

For continuous-control problems, DRL commonly represents the policy and value functions using function approximation, leading to policy-gradient and actor-critic formulations that can handle high-dimensional observations and continuous actions [28]. A widely adopted strategy is to learn a critic alongside the actor to reduce gradient variance and stabilize training. Practical deep actor-critic implementations often use parallelized data collection to improve stability [29]. In on-policy actor-critic methods, generalized advantage estimation (GAE) is frequently used to balance bias and variance in the advantage signal and improve learning stability [30]. While these approaches provide important foundations, the navigation task involves long horizons and continuous actuation.

In this work, an off-policy actor-critic algorithm, Soft Actor-Critic (SAC), is selected for two reasons. First, the ASV navigation problem considered here has a continuous action space, where the policy must regulate rudder and propulsion commands rather than choosing from a set of discrete maneuvers. SAC directly learns a stochastic continuous control policy, which is suitable for this control interface. Second, the training environment contains randomized start/goal positions and obstacle layouts so sample efficiency and exploration are important. SAC is off-policy and reuses stored transitions through a replay buffer, which improves data efficiency compared to on-policy methods, while its entropy-regularized objective encourages exploration and reduces premature convergence to a narrow avoidance strategy [31]. This choice is also supported by our previous study, where SAC outperformed PPO in success rate, cross-track error, trajectory smoothness, and robustness across varied static-obstacle layouts [24]. Therefore, SAC is used as the final algorithm in this paper, while present study focuses on deployment-oriented observation design, restricted-waterway evaluation, and field validation.

SAC learns a stochastic policy π_θ for continuous control using a maximum-entropy objective, which encourages exploration and improves robustness [32]. The objective is

$$J(\pi) = \sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))], \quad (10)$$

where α is a temperature parameter. SAC is trained off-policy using a replay buffer of transitions (o_t, a_t, r_t, o_{t+1}) . In a standard implementation, two critics Q_{ϕ_1} and Q_{ϕ_2} are used and the target uses the minimum to reduce overestimation bias. The critic target can be written as

$$y_t = r_t + \gamma \left(\min_{i \in \{1, 2\}} Q_{\phi_i}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\theta(a_{t+1} | s_{t+1}) \right), \quad a_{t+1} \sim \pi_\theta(\cdot | s_{t+1}), \quad (11)$$

and the policy update maximizes expected soft value

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_\theta} \left[\alpha \log \pi_\theta(a_t | s_t) - \min_{i \in \{1, 2\}} Q_{\phi_i}(s_t, a_t) \right]. \quad (12)$$

In our implementation, the entropy temperature α is tuned automatically to match a target entropy, reducing sensitivity to continuous action scaling.

The actor and both critics receive the same observation vector o_t . No privileged state information is supplied to the critic during training, since the Stable-Baselines3 implementation does not provide a separate state input. The distinction between s_t and o_t is therefore conceptual, describing the information available to the agent relative to the underlying environment state, and does not correspond to two different network inputs.

In Equation (11) the bootstrapped term is masked at terminal states, so that no value is propagated past a transition ending in success or collision. Truncation at the episode horizon is treated separately from termination and does bootstrap, following standard practice for time-limited environments.

2.4.2. Observation Vector

The policy input is constructed as a compact observation vector

$$o_t = [c_t, u_t, v_t, r_t, e_{y,t}, \tilde{\chi}_t, \tilde{\chi}_t^{LA}], \quad (13)$$

where:

- $c_t \in \mathbb{R}^M$ is the LiDAR sector closeness vector.
- u_t and v_t are the surge and sway velocities in the body frame.
- r_t is the yaw rate.
- $e_{y,t}$ is the cross-track error (CTE), signed lateral deviation from the closest point on the reference path.
- $\tilde{\chi}_t$ is the course or heading error relative to the local path tangent.
- $\tilde{\chi}_t^{LA}$ is the look-ahead course error relative to a forward point on the path, used to encourage smoother convergence and reduce oscillations.

Equation (13) gives the concatenated form of the observation for notational compactness. In the implementation the observation is exposed as a dictionary with separate entries for the sector closeness vector and for the kinematic and path-relative quantities, and is processed with a multi-input policy network; the two forms are equivalent in content and differ only in how the components are presented to the network.

Let $p(s)$ denote the reference path parameterised by arc length s , $p(s_c)$ be the closest point on the path to the vessel position $p_{xy} = [x, y]^T$, and t_c is the unit tangent at s_c . The signed cross-track error is defined as

$$e_y = \text{sign}\left(t_c \times (p_{xy} - p(s_c))\right) \|p_{xy} - p(s_c)\|_2 \quad (14)$$

This error definition is consistent with standard path following guidance formulations that penalize lateral deviation from the desired track [8]. The course error $\tilde{\chi}_t$ measures alignment between the vessel's direction of motion and the local path direction, while a look-ahead course error $\tilde{\chi}_t^{LA}$ provides anticipative steering toward a forward point on the path. Let p_t^{LA} be a path point located ahead of $p(s_c)$ by a look-ahead distance D_{LA} . We adopt a path-length-proportional look-ahead:

$$D_{LA} = \max(D_{min}, \kappa L_{path}), \quad (15)$$

where L_{path} is the current path length, $D_{min} > 0$ prevents degeneracy on short paths, and $\kappa > 0$ is the tuning coefficient. Then $\tilde{\chi}_t^{LA}$ is the angular difference between the vessel's course and the bearing angle from p_t to p_t^{LA} . These path-relative error signals are common in marine path following guidance because they encode task progress and convergence geometry in low dimension [33]. The geometric interpretation of the path-relative quantities is illustrated in Figure 4.

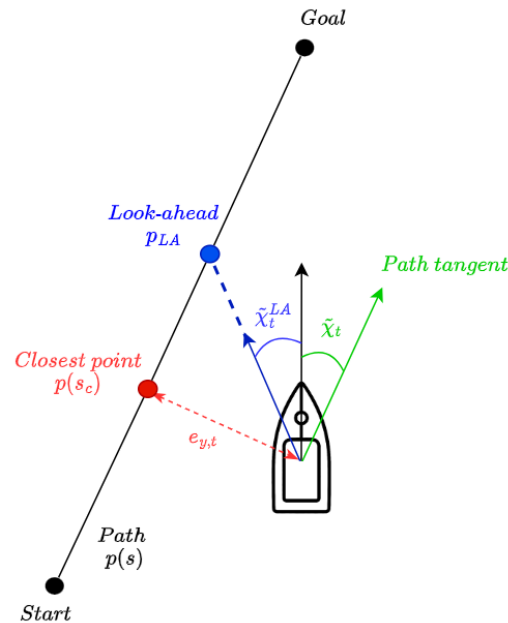


Figure 4. Path-relative tracking quantities.

2.4.3. Action Space

The agent outputs a continuous action corresponding to rudder and propulsion commands (Equation (16)).

$$a_t = [a_{\delta,t}, a_{n,t}]^T, \quad a_{\delta,t}, a_{n,t} \in [-1, 1] \quad (16)$$

The rudder command is mapped linearly to a bounded physical rudder angle (Equation (17)).

$$\delta_t = \delta_{max} a_{\delta,t} \quad (17)$$

For propulsion, two operating modes are supported:

- Fixed-speed mode is used to stabilize early learning. The propeller speed is held constant $n_t = n_c$ and $a_{n,t}$ is ignored.
- Speed-control mode outputs a residual throttle around a nominal cruise RPM (Equation (18))

$$n_t = \text{clip}(n_c + \Delta n a_{n,t}, n_{min}, n_{max}), \quad (18)$$

where n_c is the cruise RPM, Δn is the maximum residual authority, and $[n_{min}, n_{max}]$ bounds the actuation. This residual parameterisation keeps the action bounded, avoids requiring the policy to rediscover a nominal cruising point from scratch, and improves training stability in continuous control settings.

2.4.4. Reward Function

The reward is shaped to encode the two coupled objectives of this work: (i) follow the reference path with small tracking error and stable steering, and (ii) avoid collisions with static obstacles and boundaries, while preventing stationary behaviors such as stopping or circling without progress. The total step reward combines path tracking, heading alignment, obstacle proximity, boundary clearance, and progress/speed shaping terms:

$$r_t = \lambda g_{u,t} r_{pf,t} + (1 - \lambda) r_{oa,t} + w_\chi g_{u,t} r_{\chi,t} + r_{exist} + r_{border,t} + r_{spd,t} + r_{term,t}. \quad (19)$$

This type of reward design is common in DRL-based ASV navigation, where the policy must learn a practical balance between safety and task completion efficiency rather than optimizing only a single objective [16]. In the present configuration intended for field

deployment, the weighting coefficient $\lambda = 0.5$ is fixed following the sweep reported in [24]. The sensitivity of learned navigation behaviour to reward parameterisation has itself become a subject of quantitative study [34], and the analysis below is offered in that spirit.

Because the terms of Equation (19) differ in natural scale, their nominal coefficients do not by themselves describe the effective balance between objectives. The path-following term is bounded in $(0, 1]$ and, over the reachable range of cross-track error, varies by less than 10% of its own value, contributing approximately 0.48 per step after mixing. The obstacle-avoidance term is a beam-weighted mean of inverse distances, evaluating to roughly -0.03 in free space and approaching -1.0 with all beams at the penalty floor, so its typical contribution after mixing is of order -0.016 . The per-step signal is therefore dominated by path following, while the terminal penalty of -1000 outweighs the cumulative shaped return of a typical episode by roughly a factor of five, leaving the episode-level signal dominated by collision avoidance.

This separation is deliberate. Dense shaping supplies the continuous gradient required for tracking, the sparse terminal signal enforces safety, and the border term, a squared hinge, engages only within the soft safety distance. The clearances reported in Section 3.3 indicate that the trained policy rarely enters that band, so the term is largely inactive at evaluation while having shaped behavior during training, when the policy did enter it. The coefficient is retained at the value selected in [24] because it was chosen on measured performance. More generally, in shaped rewards whose terms differ in natural scale, reporting nominal coefficients without their achievable ranges can misrepresent how objectives are balanced during learning.

Path tracking terms

The path following component penalizes lateral deviation from the reference path using an exponential penalty on the cross-track error (Equation (20)).

$$r_{pf,t} = \exp(-\gamma_e |e_{y,t}|) \quad (20)$$

This choice provides a smooth learning signal near the path while reducing reward for large deviations, discouraging wide detours in narrow corridors. The sensitivity parameter γ_e controls how tightly the policy tracks the path: larger γ_e enforces stricter tracking, while smaller γ_e allows the vessel to temporarily deviate when obstacle avoidance requires it.

A common failure mode in safety-shaped rewards is that the agent may stand still to avoid collisions. To resolve this, the tracking and heading terms are multiplied by a speed gate (Equation (21)).

$$g_{u,t} = \text{clip}\left(\frac{\max(u_t, 0)}{U_{ref}}, 0, 1\right). \quad (21)$$

When the vessel is not moving forward ($u_t \leq 0$), the gated terms contribute little to the return, which encourages the agent to achieve safety through feasible motion rather than stopping indefinitely. U_{ref} is a reference surge speed used for normalisation.

In addition to minimizing CTE, the reward encourages heading alignment to reduce oscillations and promote smooth steering. A combined heading error is formed using the instantaneous course error $\tilde{\chi}_t$ and a look-ahead course error $\tilde{\chi}_t^{LA}$ (Equation (22)).

$$\tilde{\chi}_t^* = \omega_{LA} \tilde{\chi}_t^{LA} + (1 - \omega_{LA}) \tilde{\chi}_t, \quad r_{\chi,t} = \cos(\tilde{\chi}_t^i), \quad (22)$$

where $\omega_{LA} \in [0, 1]$ weights the look-ahead course error relative to the instantaneous course error. The alignment term is scaled by w_χ and gated by forward speed through $g_{u,t}$. The look-ahead component provides anticipation when approaching corners or tight passages and helps reduce high-frequency corrections that can occur when minimising CTE alone [33].

Safety terms

Obstacle proximity is penalized using a LiDAR scan that excludes border returns, preventing persistent wall measurements from dominating the learning signal. The obstacle penalty is computed using the raw beam distances to preserve directional sensitivity and to provide a smooth gradient for learning (Equation (23)).

$$r_{oa,t} = -\frac{1}{N} \sum_{i=1}^N \frac{w_i}{\max(d_{t,i}, d_{min})}, \quad w_i = \frac{1}{1 + |\alpha_i|}. \quad (23)$$

Here, $d_{t,i}$ is the range of beam i , α_i is its bearing angle relative to the forward direction, and d_{min} prevents numerical blow-up for very small ranges. The weighting w_i emphasizes forward-facing beams, reflecting the fact that collision risk is dominated by obstacles ahead of the vessel.

Since boundary sensing can be asymmetric in the observation scan due to sensor placement, boundary safety is enforced separately through a geometric clearance penalty computed from the true map boundary (Equation (24)).

$$r_{border,t} = -K_b \left[\max\left(0, 1 - \frac{d_{b,t}}{d_{safe}}\right) \right]^2, \quad (24)$$

$d_{b,t}$ is the minimum distance from the vessel to the boundary, d_{safe} is a soft safety distance, and $K_b > 0$ controls penalty strength. This squared hinge form yields zero penalty when clearance is sufficient and increases smoothly as the vessel approaches the boundary.

Efficiency terms

A constant living penalty $r_{exist} < 0$ encourages efficient completion and further discourages wandering. Terminal rewards enforce a clear separation between success and failure:

- if the vessel reaches destination point, a terminal bonus $R_{goal} > 0$ is applied;
- if a collision occurs or the episode ends due to timeout, a terminal penalty $R_{collision} < 0$ is applied.

$$r_{term,t} = \begin{cases} R_{goal} > 0, & \text{if reach goal} \\ R_{collision} < 0, & \text{if collision/timeout} \end{cases} \quad (25)$$

When propulsion is controlled by the policy, extra speed shaping encourages progress and prevents persistent stalling or unnecessary thruster variation (Equation (26)).

$$r_{spd,t} = K_p(d_{g,t-1} - d_{g,t}) - K_s \max(0, U_{min} - \max(u_t, 0)) - K_n \frac{|n_t - n_c|}{\Delta n}, \quad (26)$$

The first term rewards progress (or reduction in distance-to-goal), the second penalizes low forward speed below U_{min} , and the third regularises the propulsion command around a nominal cruise RPM n_c with authority Δn . These terms are only active in speed-control mode and can be omitted when the propulsion is fixed.

Although the timeout penalty is defined for completeness, no timeout event occurred in the evaluation episodes reported in Section 3.3; in every episode the outcome was success, obstacle collision, or border collision. The timeout penalty is retained because it prevents degenerate stalling behavior during early training, when the policy has not yet learned goal-directed motion.

Overall, the reward is structured so that safety-related terms are not gated by forward progress, while tracking and heading rewards are emphasized only when the vessel makes forward progress. The obstacle avoidance penalty is active at every step, since it accumulates over all beams regardless of range, whereas the border penalty is a squared hinge that engages only within the soft safety distance d_{safe} ; both discourage unsafe proximity independently of speed, and the speed gate $g_{u,t}$ prevents a trivial strategy of stopping to avoid collisions. The cross-track term encourages the vessel to remain close to the reference path in corridor-like environments, and the combined heading term with a look-ahead component reduces oscillatory steering and promotes anticipatory turning when approaching narrow passages. A constant living penalty promotes efficient

solutions and discourages aimless wandering. Finally, terminal rewards dominate the return at success or failure, ensuring that reaching the goal and avoiding collisions remain the primary objectives. The speed-shaping terms are enabled only when speed control is part of the action space.

2.5. Baseline Controllers

Two baseline controllers are implemented for comparison. Both operate in the same environment, receive the same observation vector, produce commands in the same action space, and are evaluated on the same 500 randomized episodes as the proposed policy. No baseline receives information that the learned policy does not also receive; in particular, neither is given ground-truth obstacle geometry, and both consume the same pooled sector closeness vector defined in Section 2.3.

2.5.1. Line-of-Sight Guidance with PID Heading Control and Potential-Field Avoidance

The classical baseline combines line-of-sight (LOS) guidance for path following with a proportional-integral-derivative (PID) heading controller and an artificial potential field (APF) avoidance layer. LOS guidance produces a desired course from the signed cross-track error and a look-ahead distance, using the same path-relative quantities that form part of the policy observation. The avoidance layer computes a repulsive course deflection from the pooled sector closeness values: each sector contributes a repulsive term that increases with closeness and decays with angular separation from the vessel heading, with sectors below an influence threshold contributing nothing. The commanded course is the LOS course plus the weighted repulsive deflection. Heading error is regulated by a PID controller whose output is mapped to the rudder command and subject to the same saturation and rate limits applied to the learned policy. Commanded speed is reduced as the minimum sector closeness and the absolute heading error increases, so that the baseline exercises the same speed authority as the policy.

Because an inadequately tuned classical controller would not constitute a fair comparison, the baseline parameters were tuned by random search over 250 configurations on a separate set of 100 layouts generated with disjoint seeds. The search covered the look-ahead distance, the three PID gains, the repulsion weight, the influence threshold and decay exponent, and the two speed-reduction gains. Configurations were ranked by success rate, with mean cross-track error used to break ties among configurations within one percentage point of the best. Only the single best configuration was subsequently evaluated on the 500-episode set; the tuning layouts were never used for evaluation.

2.5.2. Proximal Policy Optimization Baseline

The learned baseline is a policy trained with proximal policy optimization (PPO) in the same environment as the proposed method. Although a PPO and SAC comparison was reported in the prior conference study [24], that comparison used the earlier environment formulation with raw LiDAR beams and rudder-only control, and therefore does not satisfy a like-for-like comparison under the present formulation. The PPO baseline here uses the same observation vector, action space, reward function, termination conditions, and staged propulsion curriculum as the SAC policy, and is trained for the same total number of environment interactions. The curriculum is scheduled against total environment steps rather than per-environment steps, so that both algorithms encounter identical curriculum transitions.

3. Results and Discussion

This section presents the simulation and field results used to evaluate the proposed DRL framework in a restricted operating area. First, the simulation training configuration and evaluation protocol is described, followed by quantitative simulation results. Then, the model-scale field experiments conducted under predefined obstacle layouts are presented. Finally, the consistency and differences between simulations and real-world performance are discussed.

3.1. Training Configuration and Parameters

The SAC policy is trained in the simulation using the Stable-Baselines3 (SB3) implementation of Soft Actor-Critic [35]. The environment is constructed as a Gymnasium interface with a dictionary observation space (MultiInputPolicy), continuous actions, and scalar rewards. Parallel rollout collection is performed using a vectorized environment (SubprocVecEnv) with N_{env} worker processes to improve sample throughput and stabilize off-policy data collection.

The simulation study has two purposes: to quantify navigation performance across obstacle density under randomized conditions, and to produce a trained policy that can be deployed on the physical platform without further tuning. This subsection describes the training environment, the randomization procedure, and the parameter settings; Section 3.2 defines the evaluation metrics and Section 3.3 reports the results.

The workspace is a bounded rectangular corridor of 10.0 m by 25.0 m, representing a narrow operating area on a model scale. At the start of each episode, between zero and four 1.0 m by 1.0 m obstacles are placed randomly from the reference path endpoints. The obstacle count is drawn from the distribution [0.15, 0.15, 0.45, 0.15, 0.10] for zero through four obstacles respectively. The reference path is a straight segment of length 20.0 m along the corridor centerline, and the episode horizon is 700 control steps, equivalent to 70 s at the 10 Hz control period.

The principal parameter choices are motivated as follows. The sector count $M = 25$ gives an angular resolution of 10.8 degrees per sector, which subtends approximately 3.0 m at the maximum LiDAR range and is therefore comparable to the vessel beam of 0.50 m, so that a single sector approximately corresponds to one vessel width at the distance at which an avoidance decision must be initiated. The soft safety distance $d_{safe} = 0.7$ m corresponds to 1.4 vessel beams, allowing the boundary penalty to activate before the geometric collision condition. The tracking sensitivity $\gamma_e = 0.20$ sets the cross-track error at which the path-following reward halves to 3.5 m. The reference speed $U_{ref} = 0.8$ m/s corresponds to the nominal cruise propeller setting used during field deployment.

To assess training repeatability, the policy configuration reported here was trained with three independent random seeds. Seeds control network initialization, action sampling, and environment randomization. All results in Section 3.3 are aggregated across the three seeds.

The training environment randomizes the start/goal positions and static obstacle layouts within a bounded narrow workspace. During training, the obstacle count is sampled from a predefined distribution of 0 to 4 obstacles. This distribution exposes the agent to both obstacle-free path-following and obstacle-avoidance scenarios, so the learned policy does not overfit to either pure path tracking or constant off-track avoidance. Obstacles are generated near the path to ensure that they create meaningful local planning challenges.

The policy is trained for 1 million timesteps. The first 700,000 timesteps are used for fixed-speed training, allowing the agent to first learn stable steering, path-following, and obstacle-avoidance behavior without simultaneously adapting propulsion. The remaining 300,000 timesteps introduce speed control in three stages of 100,000 steps each. In these

stages, the propulsion command is represented as a residual action around a nominal cruise setting:

$$n = \text{clip}(n_c + \Delta n a_n, n_{\min}, n_{\max}), \quad (27)$$

where $a_n \in [-1, 1]$ is the normalized propulsion action, n_c is the cruise command, and Δn defines the allowable residual authority. The speed-control range is gradually widened from a narrow band around cruise speed to a final range corresponding to 25–75% of the maximum propulsion command. The full propulsion range is not used because near-zero propulsion can lead to ineffective steering or standing-still behavior, while full-speed operation is less reliable in narrow waterways due to larger turning radius and increased overshoot. The staged residual formulation therefore allows the vessel to accelerate and decelerate around a safe cruise region without learning degenerate behaviors or overly aggressive high-speed maneuvers.

Table 2 summarizes the parameters used throughout training and evaluation. The same observation construction and action mapping are used consistently during training, simulation evaluation, and field deployment to minimize implementation gaps.

Table 2. Parameters and training configurations.

Category	Parameter	Notation	Value
LiDAR	Measurable range	$[D_{\min}, D_{\max}]$	[1, 16] m
	Field of view/Swath	ϕ	270.0 deg
	Number of beams	N	225
	Number of sectors	M	25
	Safety-adjusted vessel width	W_s	0.50
Reward	Collision penalty	$R_{\text{collision}}$	−1000
	Timeout penalty	R_{timeout}	−1000
	Goal bonus	R_{goal}	50
	Step penalty	r_{exist}	−0.5
	Path tracking sensitivity	γ_e	0.05
	Speed gate reference	U_{ref}	0.8 m/s
	Weighting parameter	λ	0.5
	Obstacle penalty distance floor	$D_{\min}$	2.0 m
	Heading mix	ω_{LA}	0.7
	Heading multiplier	ω_{χ}	0.35
	Border soft penalty	K_b, d_{safe}	0.25, 0.7 m
	Progress reward weight	K_p	0.7
	Low-speed penalty weight	K_s	0.10
	Propulsion regularization weight	K_n	0.025
	Look-ahead coefficient	κ	0.25
	Minimum speed threshold	$U_{\min}$	0.30 m/s
Training configuration	Total timesteps	N_{steps}	1,000,000
	Parallel environments	N_{env}	8
	Learning rate	η	5×10^{-5}
	Batch size	B	512
	Discount factor	γ	0.99
	Replay buffer size	$\mathcal{D}$	1,000,000
	Training frequency	f_{train}	1
	Gradient steps	G	1
	Entropy coefficient	α	auto
	Evaluation episodes per seed	N_{eval}	500
Speed-control curriculum	Cruise propulsion	n_c	50% of max. speed
	Stage 1 propulsion range	$[n_{1\min}, n_{1\max}]$	[37.5%, 62.5%]

	Stage 2 propulsion range	$[n_{2min}, n_{2max}]$	[33.3%, 66.6%]
	Stage 3 propulsion range	$[n_{3min}, n_{3max}]$	[25.0%, 75.0%]
	Residual propulsion	$\Delta n_1, \Delta n_2, \Delta n_3$	12.5%, 16.5%, 25.0%
Actuation	Maximum rudder angle	δ_{max}	40.0 deg
	Rudder rate limit	$\dot{\delta}_{max}$	20.0 deg/s
	Control period	δ_t	0.1 s

3.2. Evaluation Metrics

To quantify navigation performance, task-level outcomes and path-following quality are reported. The success rate is defined as the fraction of episodes in which the ASV reaches the goal region within the episode horizon. Path tracking performance is measured by the mean and standard deviation of cross-track error, which reflect tracking accuracy and oscillatory behavior. Efficiency is reported through the elapsed time of each successful episode.

Two clearance metrics are used when comparing individual trajectories. Minimum obstacle clearance is the smallest distance recorded over an episode between the vessel footprint and the nearest static obstacle surface. Minimum border clearance is the smallest distance recorded over an episode between the vessel footprint and the nearest workspace boundary. Both are minima over all control steps of a single trajectory rather than averages, and a positive value indicates that no contact occurred at any point. Root-mean-square cross-track error (RMS CTE) is computed over all control steps of a trajectory and is reported for individual scenarios, whereas the mean and standard deviation of cross-track error are reported when aggregating over randomized episodes.

Because navigation in restricted waters involves both obstacle avoidance and boundary clearance, it is informative to report collision outcomes separately as obstacle collision rate and border collision rate. These rates help diagnose whether failures are dominated by obstacle interaction or boundary contact. These collision-rate metrics are recommended for interpreting failure modes in this study.

3.3. Simulation Evaluation

3.3.1. Training Process

To visualize learning progress, Figure 5 plots the training signal in two forms: panel (a) shows the raw episode reward recorded during training, and panel (b) the mean evaluation return measured at fixed checkpoints on held-out episodes. The solid line represents the mean episode reward at each evaluation checkpoint, while the shaded region indicates one standard deviation across evaluation episodes. Since the training environment randomizes the start/goal positions and obstacle layouts, the reward variance is expected to remain relatively large, especially during early training when the policy has not yet learned consistent avoidance behavior.

The learning curve shows a rapid improvement during the initial phase of training, where the mean reward increases from large negative values to near-zero values. This indicates that the policy first learns basic goal-directed motion and begins to reduce terminal failure events such as collision or timeout. After approximately the middle stage of training, the reward curve reaches a plateau and fluctuates around a small positive value, suggesting that the policy has converged to a stable behavior under the sampled training distribution. The remaining reward variation is attributed to differences in obstacle number and placement across evaluation episodes.

The raw training reward in panel (a) records every training episode and therefore retains structure that the checkpoint curve in panel (b) smooths away. Two features are relevant to the training protocol. First, the per-episode trace continues to show excursions to the terminal penalty throughout training, including in the final phase. These reflect

exploration under the stochastic training policy rather than failures of the deployed controller, which is evaluated deterministically; the evaluation results of Section 3.3.2 are obtained without exploration noise.

Second, the rolling mean is not monotonic in the later phase of training. The propulsion range was widened in stages by restarting training at fixed step boundaries rather than by an automated scheduler, and the rolling mean shows a transient reduction following the stage transitions before recovering to its previous level. This is the behavior the staged protocol is intended to produce: each widening of the propulsion range exposes the policy to speed settings it has not previously encountered, degrading performance briefly until the policy adapts. The magnitude of the transient decreases across successive transitions, which is consistent with the policy acquiring speed-regulation behavior that generalizes beyond the specific range in which it was acquired. Because the stages were advanced manually, the transitions are visible in the raw training record rather than logged as a separate schedule.

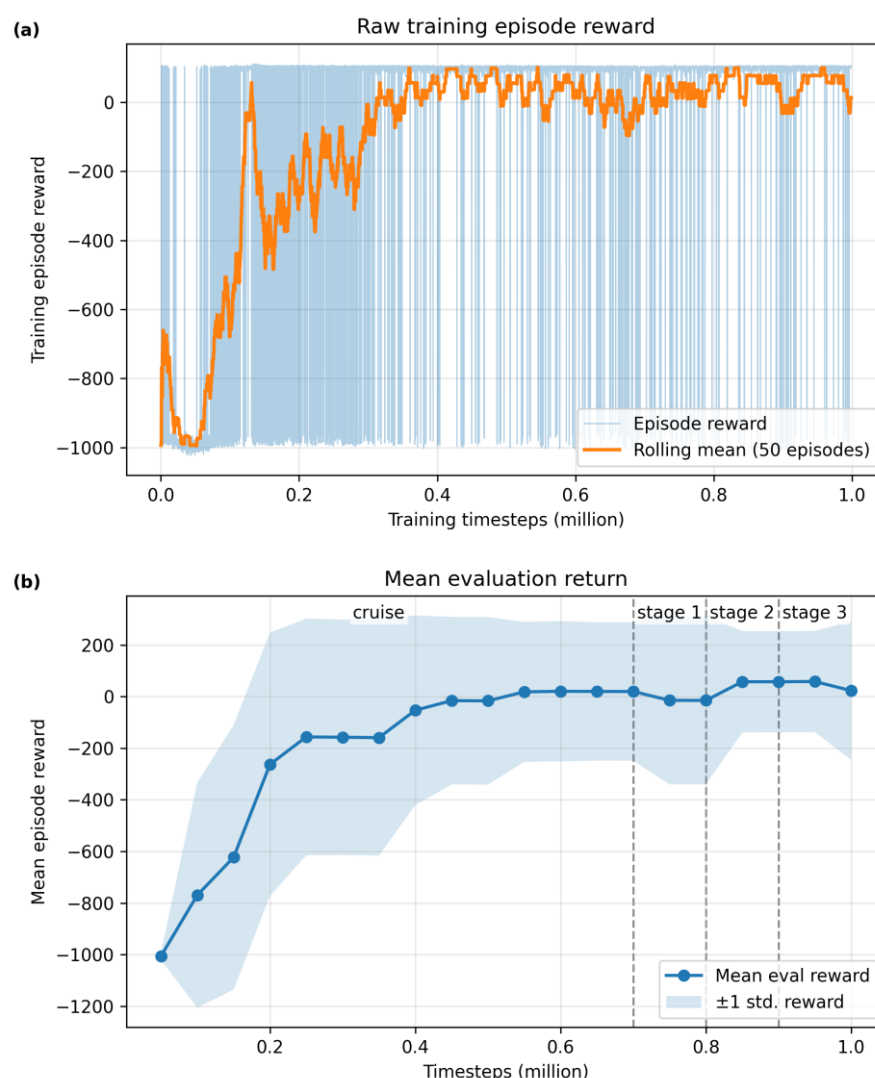


Figure 5. Training signal for the deployed policy. (a) Raw training episode reward; the light trace is the per-episode reward recorded during training and the heavy trace its rolling mean over 50 episodes. (b) Mean evaluation return versus training timesteps; the solid line is the mean episode return at each evaluation checkpoint and the shaded band is one standard deviation across evaluation episodes. Reward and return are dimensionless, being sums of shaped reward terms; the horizontal axis is the cumulative count of environment interaction steps.

3.3.2. Quantitative Evaluation Across Obstacle Density

In our previous study, both PPO and SAC were trained and evaluated over 1000 randomized episodes [21]. Across various weighting coefficients λ settings, PPO achieved success rates between 49.4% and 67.4%, whereas SAC achieved success rates between 90.0% and 95.5%. Under the same comparison, SAC yielded higher efficiency with average completion time and lower average cross-track errors. These results established SAC as the most suitable learning algorithm for this navigation and control problem, which is why the present paper focuses on the selected SAC policy and evaluates it under a more deployment-oriented formulation.

In this work, we conduct $N_{eval} = 500$ evaluation episodes, evenly distributed across 0 to 4 obstacles. Table 3 summarizes the simulation evaluation results. Reporting results by obstacle count provides a compact and interpretable representation of robustness: as obstacle density increases, the policy must trade off tracking precision against avoidance maneuvers, which may increase cross-track deviations and elapsed time while maintaining a high success rate if avoidance remains reliable.

Table 3. Simulation evaluation of the trained SAC policy over 500 randomly generated episodes.

Number of Obstacles	Success Rate	Avg. Cross-Track Error	RMS Cross-Track Error	Avg. Time Elapsed	Obstacle Collision Rate	Border Collision Rate
0	100%	0.12 m	0.15 m	19.06 s	0%	0%
1	96%	0.56 m	0.75 m	20.23 s	4%	0%
2	95%	0.80 m	1.02 m	20.91 s	3%	2%
3	95%	0.86 m	1.07 m	20.98 s	5%	0%
4	89%	0.92 m	1.17 m	21.30 s	7%	4%
Overall	95%	0.66 m	0.83 m	20.39 s	3.8%	1.2%

Because a single training run cannot establish repeatability, the reported configuration was trained with three independent seeds and evaluated on a common set of 500 randomized episodes. The same 500 layouts, initial poses, and obstacle configurations were used for every seed and for every comparison reported in this section, so that all differences between configurations are measured on identical episodes. Table 4 shows the results aggregated across seeds.

Table 4. Aggregate performance across three independent training seeds, evaluated on a common set of 500 randomized episodes. Confidence intervals are 95% stratified bootstrap intervals over seeds and episodes.

Metric	Mean	95% CI	Seed Spread (Min–Max)
Success rate (%)	92.4	[91.10, 93.70]	90.0–94.0
Obstacle collision rate (%)	4.7	[3.70, 5.80]	1.60–8.60
Border collision rate (%)	2.9	[2.10, 3.70]	1.40–4.40
Mean cross-track error (m)	0.87	[0.85, 0.88]	0.75–0.90
Min. obstacle clearance (m)	0.44	[0.42, 0.46]	0.43–0.54
Episode duration (s)	20.3	[20.20, 20.40]	19.30–20.60

The spread across seeds is 4.0 percentage points in success rate, indicating that the training procedure is reproducible in a sense that all three seeds fall within four percentage points. Three seeds is a small sample, and the confidence intervals should be read as indicative rather than tight; the interval widths are reported so that subsequent comparisons in this section can be interpreted against the run-to-run variation of the method itself.

The deployed SAC policy achieves an overall success rate of 95.0% across the 500 evaluation episodes, with an obstacle collision rate of 3.8% and a border collision rate of

1.2%. No episode reached the horizon without terminating, so the timeout outcome does not appear in Table 3. In the obstacle-free case, the policy reaches the destination in all episodes and maintains the smallest average cross-track error, which confirms that the learned controller can perform the basic path-following task. When obstacles are introduced, the average cross-track error increases from 0.12 m in obstacle-free scenarios to 0.92 m in the four-obstacle cases. This trend is expected because successful avoidance in a narrow environment requires temporary deviations from the reference path.

Despite the increasing obstacle density, the success rate remains above 90% for all tested obstacle counts. The average elapsed time increases once obstacles are present but remain within a narrow range. This suggests that the policy does not rely on excessive waiting behavior to avoid obstacles. Instead, it performs avoidance maneuvers while continuing to make progress toward the goal.

The collision-rate breakdown also provides useful information about the failure modes. Obstacle collisions are most noticeable in single and two-obstacle cases, while border collisions become more prominent as the number of obstacles increases. This indicates that in denser layouts, the policy is able to avoid obstacles, but avoidance maneuvers can push the vessel closer to the workspace boundary. Overall, the simulation results show that the proposed observation design and reward formulation lead to robust path-following and static obstacle avoidance across a range of randomly generated environments.

From this perspective, the 95.0% success rate obtained in the present study is significant for two reasons. First, the result is evaluated over 500 randomized episodes spanning from 0 to 4 obstacles, which reflects generalization across obstacle density and spatial arrangement. Second, the task setting is more restrictive than an open-water avoidance problem because obstacle avoidance and boundary clearance are coupled. This is reflected in the separate reporting of obstacle and border collision rates, which are 3.8% and 1.2% respectively. The policy therefore does not simply avoid obstacles at the cost of drifting into the boundary; instead, it maintains a useful balance between path tracking, static obstacle avoidance, and boundary clearance in confined waterways.

3.3.3. Comparison with Baseline Controllers

Table 5 compares the deployed SAC policy against the retrained SAC seeds, the PPO baseline and the LOS-PID with APF baseline on the identical set of 500 randomized episodes. The learned columns are averaged over three training seeds and the classical column over three independent tuning searches; the deployed policy is a single controller and its column reports that controller alone. Because every method is evaluated on the same episodes, differences between columns are not confounded by layout variation.

Table 5. Comparison of the proposed method against a classical and a learned baseline on a common set of 500 randomized episodes. The values are averaged over 3 seeds.

Metric	LOS-APF (3 Searches)	PPO (3 Seeds)	SAC (3 Seeds)	SAC (Deployed Policy)
Success rate (%)	92.20	90.50	92.40	95.00
Obstacle collision rate (%)	3.50	7.90	4.70	3.80
Border collision rate (%)	4.30	1.70	2.90	1.20
RMS cross-track error (m)	1.33	1.29	1.10	0.91
Min. obstacle clearance (m)	0.67	0.53	0.44	0.23
Episode duration (s)	25.40	22.00	20.30	20.39

The comparison shows a consistent pattern rather than a uniform ranking. The deployed policy attains the highest success rate of the four controllers at 95.0%, against 92.4% for the retrained SAC seeds, 92.2% for the classical baseline and 90.5% for PPO. The

margin over the classical baseline is 2.8 percentage points and the margin over PPO is 4.5, so the learned controller is ahead on task completion but not by a decisive amount relative to a properly tuned classical stack. The retrained seeds and the classical baseline are separated by 0.2 percentage points and should be read as equivalent on this metric.

The clearest and most consistent advantage of the learned policy lies in path-following accuracy rather than in task completion. The deployed policy attains an RMS cross-track error of 0.91 m against 1.33 m for the classical baseline and 1.29 m for PPO, a reduction of approximately 30% in both cases, and it completes episodes in 20.9 s against 25.4 s for the classical stack, a reduction of 18%. The retrained seeds sit between the two at 1.10 m, which indicates that part of the deployed policy's advantage derives from its staged training protocol rather than from the algorithm alone. For a survey vessel, where remaining on the reference line is the mission objective rather than a means to it, this is the operationally significant difference between the two families of controller.

The ordering reverses for obstacle clearance. The classical baseline passes obstacles at a minimum clearance of 0.67 m against 0.23 m for the deployed policy, a factor of approximately three, with PPO intermediate at 0.53 m. The learned controllers therefore purchase tracking accuracy and speed at the cost of margin, while the potential-field baseline makes the opposite trade and pays for its clearance in elapsed time. Neither controller dominates, and the comparison is better understood as two positions on a single accuracy-against-margin trade-off than as a ranking. Which position is preferable depends on the deployment: a wider margin is worth the additional transit time in cluttered or poorly charted water, whereas tighter line-keeping is worth the reduced margin on a surveyed route where the hazards are known. The failure modes differ in the same way. The classical baseline records the lowest obstacle collision rate of the learned-and-classical set at 3.50% but the highest border collision rate at 4.30%, since its repulsive term acts on obstacles and pushes the vessel toward the channel walls, whereas the deployed policy records 3.80% and 1.20% respectively and treats the two hazards within a single learned response.

Figure 6 resolves these aggregate figures by obstacle density. All four controllers succeed in every obstacle-free episode, so the differences between them arise entirely from obstacle interaction. The deployed policy retains a success rate of 95% or above from one to three obstacles and falls to 89% at four, while the classical baseline declines more steadily across the same range. The separation between controllers is therefore not uniform across the evaluation set but concentrated in the denser layouts, which is where the compressed sector representation and the learned avoidance response are most exercised.

One structural difference is worth noting independently of the numerical outcome. The classical stack requires the guidance and avoidance layers to be tuned separately, and their interaction is governed by a fixed blending weight that does not adapt to the local obstacle configuration. The learned policy maps the pooled scan directly to actuator commands, with no explicit blending stage and no online optimization, at the cost of requiring training and of providing no formal constraint-satisfaction guarantee.

Placing these figures alongside published results requires care, because reported success rates depend on workspace geometry, obstacle density, episode termination criteria and the definition of success, none of which are standardized across the literature. With that qualification, the present results are broadly consistent with recent work on learned ASV navigation. Meyer et al. [16] report successful path following with collision avoidance for a simulated vessel using multiple rangefinder sensors, and Hao et al. [19] report improved convergence for a recurrent PPO variant in multi-vessel encounters; both evaluate in open water rather than in a bounded channel, so boundary contact does not appear as a distinct failure mode and the border collision rate reported here has no counterpart in their results. Paulig and Okhrin [36] address the closest setting to the present work, learned path following for an underactuated vessel on inland waterways, and likewise

benchmark against a tuned PID controller rather than against an optimization-based method; their finding that the learned controller generalizes to unseen river sections while retaining navigational accuracy is consistent with the pattern reported here, in which the advantage of the learned policy appears in tracking accuracy rather than in task completion.

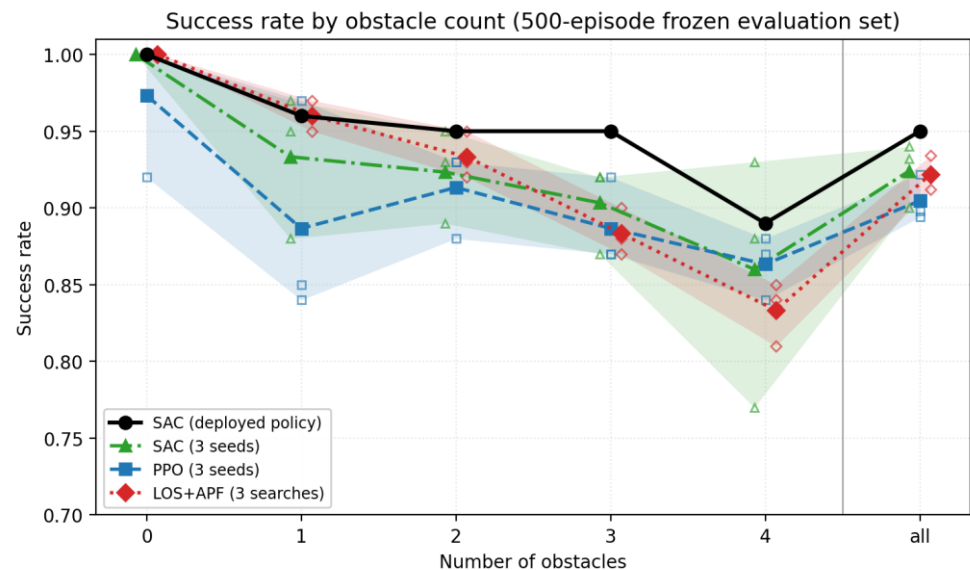


Figure 6. Success rate by obstacle count for the four evaluated controllers on the common set of 500 randomized episodes, 100 episodes per obstacle count. All controllers succeed in every obstacle-free episode; the separation between them is concentrated in the denser layouts.

The comparison against optimization-based control has also been examined experimentally. Wang et al. [23] compare a DRL tracking controller against a nonlinear model predictive controller on a physical surface vessel in natural waters and report lower tracking error for the learned controller in both simulation and field conditions, attributing the difference to disturbance rejection. That study required physical deployment precisely because a simulation-only comparison would evaluate the predictive controller against the model on which it was built. The same consideration applies here and is the reason a model predictive baseline is not included in the present evaluation, as discussed in Section 3.5.

3.4. Field Experiments

3.4.1. Experiment Setup

Field experiments are conducted using a model-scaled Bluefin vessel equipped with a 2D LiDAR sensor and a SLAM-based localization system. The trained SAC policy is executed on an offboard computer, while the vessel provides real-time telemetry containing local pose, heading, and LiDAR range measurements. The field implementation follows the same observation and action interface used in simulation: LiDAR sector features, motion states and path-relative errors. The trained policy outputs continuous rudder and propulsion actions. Table 6 specifies the dimensions of the Bluefin vessel and the LiDAR sensor package used in this experiment.

Communication between the vessel and the offboard computer is implemented through UDP protocol. At the start of a trial, the offboard computer sends a message to initiate telemetry streaming and receives LiDAR and localization data from the vessel. For each valid telemetry frame, the computer constructs the observation vector and evaluates the trained policy deterministically. The policy output is mapped to physical command values before transmission to the vessel. The rudder action is converted into a bounded

rudder command and rate limiting applied to match the real actuator. The propulsion action is mapped using the same residual cruise-speed formulation in the simulation. Commands are sent back to the vessel through UDP at 10 Hz control rate.

Table 6. Technical details of the Bluefin vessel and LiDAR sensor.

Component	Parameter	Value
Bluefin Vessel	Mass	64.55 kg
	Length Over All	1.73 m
	Length Between Perpendicular	1.57 m
	Breadth	0.50 m
	Draft	0.19 m
	Yaw inertia moment, I	10.45 kg·m ²
LiDAR	LiDAR Model	RPLidar C1
	Navigation	ROS2 package rf2o_laser odometry

To enable direct comparison between simulation and field behavior, three test scenarios are defined with three obstacles arranged in a confined waterway layout as shown in Figure 7. The same scenario settings are reproduced in simulation for trajectory overlay comparison. Each scenario was executed once. No repeated trials were performed, because access to the test basin was limited to a single testing window and could not be extended. The field results reported below therefore constitute a feasibility demonstration for the three layouts tested; they do not provide an estimate of trial-to-trial variability and are not used here to support a statistical claim about sim-to-field transfer.

The test basin measures 25 m in length by 10 m in width, and each scenario uses three static obstacles with an approximately 1 m by 1 m footprint. The obstacles occupy the central portion of the basin, roughly between 8 m and 17 m along its length, and are spread across the 10 m width and offset on alternating sides so that the vessel must make successive port and starboard passes. They are grouped at about one-third and two-thirds of the way along the basin, separated by roughly 7–8 m along-track, leaving navigable gaps on the order of 1.5–3 m. Exact positions vary between the three scenarios and are approximate. All trials were conducted in calm water with no generated waves or current, so the results do not test disturbance rejection.

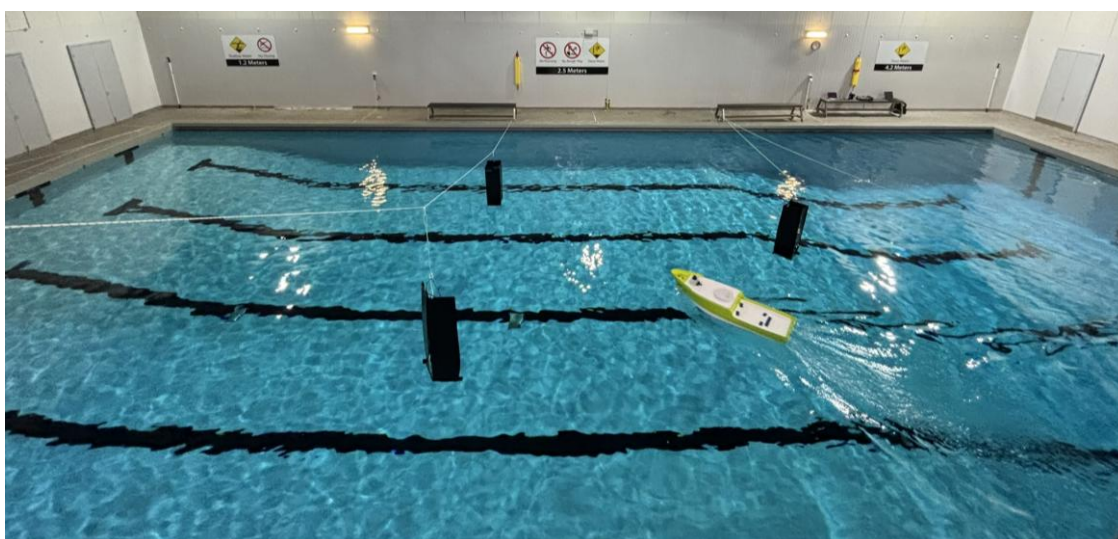


Figure 7. Model-scale Bluefin vessel operating in the test basin with three static obstacles. Obstacle dimensions, spacing, and positions relative to the basin are given in Section 3.4.1.

3.4.2. Comparison Between Simulation and Field

Figure 8 overlays the trajectories obtained in simulation and field trials for three scenarios. Table 7 provides the corresponding trajectory-based metrics. The purpose of this comparison is not to obtain identical trajectories, but to determine whether the policy trained in simulation can reproduce the same task-level behavior on the Bluefin vessel.

Overall, the single trial conducted in each scenario shows task-level transfer of the trained SAC policy under the tested conditions. In all three scenarios, the field trajectory avoids static obstacles and progresses toward the goal. The minimum obstacle clearance and boundary clearance in the field trials remain positive in all cases, which indicates that the policy is able to maintain collision-free motion in the tested environment. At the same time, the field trajectories show stronger oscillation and wider recovery maneuvers than the simulated trajectories. Averaged over the three scenarios, the field RMS cross-track error is 1.14 m, compared to 0.69 m in simulation. The field trajectories are also longer, with an average path length of 21.11 m over 19.49 m in simulation. This confirms the visual observation in Figure 7, where the real vessel generally follows the same direction as the simulated vessel, but requires additional correction and larger deviations.

In the first scenario, the field result is close to the simulation result in terms of cross-track error. This suggests that the vessel completed the same avoidance task but with more corrective motion. In the second scenario, the field trajectory shows a larger tracking deviation than simulation. The RMS cross-track error increases from 0.79 m in simulation to 1.17 m in the field deployment, and the standard deviation of cross-track error increases from 0.61 m to 1.02 m. This shows that the deployed vessel adopted the same avoidance strategy as simulated with higher overshoot, resulting in a larger path deviation and closer to the boundary. The sim-to-field gap is more noticeable in the final test scenario, where the field RMS cross-track error is 1.53 m and path length increases to 22.69 m. This test case highlights the main limitation observed in the field experiments: the learned policy achieves the task, but the real vessel response is more oscillatory and less smooth than the simulated response.

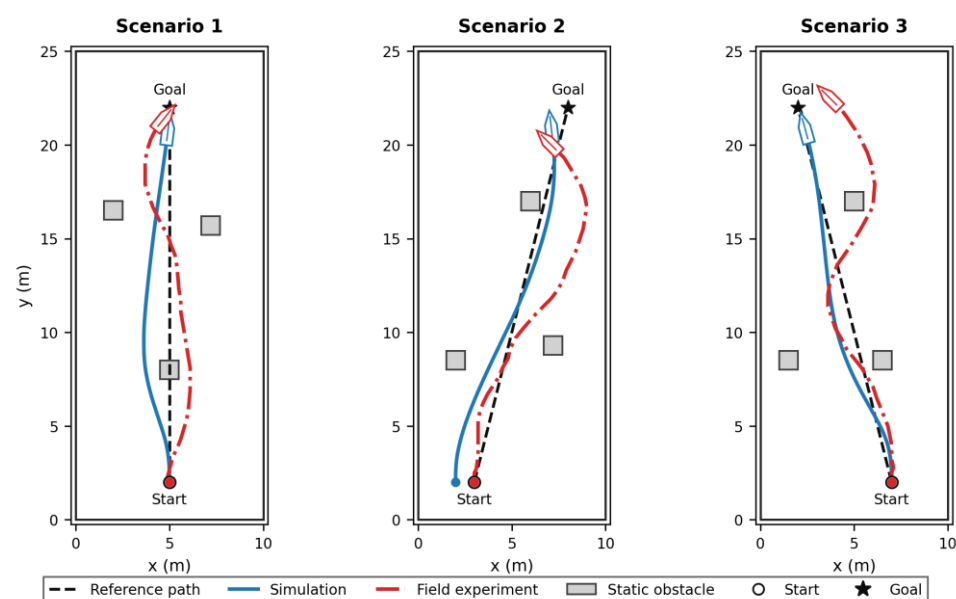


Figure 8. Simulated and field trajectories for the three test scenarios. In each panel the dashed line is the reference path, the solid line is the simulated trajectory, and the dash-dot line is the trajectory recorded in the single field trial. Filled circles mark the start position and black stars the goal region; shaded squares are static obstacles and the outer rectangle is the workspace boundary.

Table 7. Trajectory-based comparison between simulation and field experiments.

Scenario	Platform	Std. Cross-Track Error	RMS Cross-Track Error	Path Length	Min. Obstacle Clearance	Min. Border Clearance
1	Simulation	0.50 m	0.81 m	19.00 m	0.68 m	2.00 m
	Field	0.72 m	0.73 m	20.26 m	0.54 m	2.00 m
2	Simulation	0.61 m	0.79 m	19.88 m	0.52 m	2.00 m
	Field	1.02 m	1.17 m	20.38 m	1.17 m	1.00 m
3	Simulation	0.39 m	0.46 m	19.60 m	1.09 m	2.00 m
	Field	1.30 m	1.53 m	22.69 m	0.50 m	2.00 m
Overall	Simulation	0.50 m	0.69 m	19.49 m	0.76 m	2.00 m
	Field	1.01 m	1.14 m	21.11 m	0.74 m	1.67 m

Minimum obstacle clearance is comparable between simulation and field when aggregated over the three scenarios, at 0.76 m and 0.74 m respectively, but which of the two platforms records the larger clearance changes from scenario to scenario. In Scenario 2 the field trajectory passes further from the nearest obstacle than the simulated one, at 1.17 m against 0.52 m, whereas in Scenarios 1 and 3 the simulated trajectory holds the larger margin, at 0.68 m against 0.54 m and 1.09 m against 0.50 m respectively. This pattern is consistent with the field trajectories exhibiting larger excursions in both directions relative to the reference path, rather than a systematic reduction in safety margin.

The comparison suggests that the main limitation is not the decision-making of the policy, but the mismatch between the simulated and real vessel responses. The field trajectories preserve the intended behavior of path following, obstacle avoidance, and boundary clearance, but with larger oscillations and wider turns. These differences are caused by unmodelled hydrodynamic effects, actuator response delay, localization noise, LiDAR measurement variation, and water disturbances. Since the ASV is underactuated, small discrepancies in heading response or propulsion mapping can produce visible differences in cross-track error, especially in narrow waterways where clearance is limited.

3.5. Discussion and Limitations

The simulation and field results together demonstrate that the proposed LiDAR-based SAC framework can produce feasible navigation behavior in a restricted environment, simultaneously tracking a reference path and avoiding static obstacles. Simulation evaluation over randomized episodes provides evidence of performance across varying obstacle densities, while the three single-trial field scenarios demonstrate that the learned policy can be executed on the physical platform and complete the task under the tested conditions. Because each scenario was run once, these trials establish feasibility rather than robustness.

The separate reporting of obstacle and border collisions is important in restricted waterways. In open-water settings, the main failure mode is often collision with another object; however, in a narrow channel environment, a successful avoidance maneuver can still fail if it pushes the vessel towards the boundary. The simulation results show that as obstacle density increases, cross-track error increases and border collisions become more frequent. This supports the central motivation of the paper: static obstacle avoidance in confined waterways must be evaluated together with path-tracking and boundary-clearance performance.

The field experiments also reveal the main limitation of the current framework. Although the policy transfers successfully at the task level, the field trajectories are less smooth than the simulated trajectories, with larger cross-track variation and longer path lengths. This suggests that the primary sim-to-field gap is not the high-level decision-making module, but the mismatch in dynamic responses. Likely causes include

unmodelled hydrodynamic effects, actuator delay, localization jitter, and water disturbances. These effects are amplified in narrow waterways, where small heading or actuation errors can quickly produce larger lateral deviations.

Therefore, future work will focus on reducing this simulation-to-field gap by refining the vessel dynamics model through additional system identification, improving actuator and propulsion modeling, and introducing disturbance variations during training. Furthermore, the method will be extended toward dynamic obstacle avoidance and COLREG-compliant navigation. This staged deployment is necessary because reliable path recovery and static obstacle avoidance in restricted waterways provide the foundation for more complex multi-vessel encounter scenarios.

4. Conclusions

This paper presented a LiDAR-based Soft Actor-Critic guidance navigation control framework for path following and static obstacle avoidance of an underactuated ASV in narrow waterways. The proposed method differs from conventional modular approaches by learning a unified observation-to-action policy that combines LiDAR sector features, local motion estimates, and path-relative errors to generate continuous rudder and propulsion commands. This is important in restricted environments, where path following, static obstacle avoidance, and boundary clearance must be handled together rather than as independent objectives.

In simulation, the trained policy achieved a 95% success rate over 500 randomized episodes, with obstacle collision rate of 3.80% and border collision rates of 1.20%. These results show that the policy can generalize across different obstacle densities while balancing path tracking and avoidance behavior.

A single field trial in each of three fixed layouts on the model-scaled Bluefin vessel demonstrated task-level transfer: the vessel avoided static obstacles and maintained positive obstacle and boundary clearance in every scenario tested. However, field trajectories showed larger oscillations and longer paths than simulation, with average RMS cross-track error increasing from 0.69 m to 1.14 m. On the same 500 episodes, the deployed policy exceeded the success rate of a tuned LOS-PID controller with potential-field avoidance (95.0% against 92.2%), while reducing RMS cross-track error from 1.33 m to 0.91 m and mean episode duration from 25.4 s to 20.9 s. The dominant distinction, however, is structural rather than a uniform performance advantage: the classical stack must extract obstacle geometry from the scan and tune its guidance and avoidance layers separately, blending them with a fixed weight that does not adapt to the local configuration, while an NMPC formulation additionally requires an accurate internal prediction model and an online optimization at every control step. The proposed policy instead maps a fixed-dimension pooled scan directly to rudder and propulsion commands in a single forward pass, at fixed and predictable computational cost. The trade-off is that it requires training, offers no formal constraint-satisfaction guarantee, and operates with a smaller minimum obstacle clearance than the classical baseline, achieving its lower tracking error by passing closer to obstacles rather than by detouring further around them.

Two limitations qualify the findings. First, the field trials were conducted in calm water, so disturbance rejection was not tested. Second, the residual mismatch between simulated and measured vessel response, evident in the larger oscillations and longer path lengths observed in the field, indicates that the identified dynamics model does not fully capture the transient behavior of the physical vessel.

The results demonstrate that the proposed framework can perform path following and static obstacle avoidance under restricted-waterway conditions, but also reveal that the current simulation model does not fully capture the real vessel response. Future work will refine the vessel dynamics and actuator models through further system identification,

incorporate sensing and disturbance variations during training, and extend the framework toward dynamic obstacle avoidance and COLREG-compliant navigation.

Author Contributions: Conceptualization, N.T., H.D.N. and P.K.; Methodology, N.T. and P.K.; Software, P.K.; Validation, P.K.; Formal analysis, N.T.; Investigation, N.T.; Writing—original draft, N.T.; Writing—review & editing, H.D.N., P.K. and M.T.; Supervision, H.D.N., P.K. and M.T.; Funding acquisition, H.D.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets presented in this article are not readily available because the data are part of an ongoing study. Requests to access the datasets should be directed to the corresponding author, Dr Peter King, by email at pdking@utas.edu.au.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

APF	Artificial potential field
ASV	Autonomous surface vessel
COLREGs	International Regulations for Preventing Collisions at Sea
CTE	Cross-track error
DDPG	Deep deterministic policy gradient
DOF	Degrees of freedom
DRL	Deep reinforcement learning
DWA	Dynamic window approach
FOV	Field of view
GAE	Generalized advantage estimation
GRU	Gated recurrent unit
LiDAR	Light detection and ranging
LOS	Line of sight
MDP	Markov decision process
NMPC	Nonlinear model predictive control
PID	Proportional-integral-derivative
POMDP	Partially observable Markov decision process
PPO	Proximal policy optimization
RL	Reinforcement learning
RMS	Root mean square
ROS	Robot Operating System
SAC	Soft actor-critic
SB3	Stable-Baselines3
SLAM	Simultaneous localization and mapping
UDP	User Datagram Protocol
USV	Unmanned surface vehicle

Nomenclature

The following symbols are used in this manuscript. Where a symbol is reused between the vessel model and the learning formulation, both meanings are listed so that the intended quantity is unambiguous at each point of use.

$\eta = [x, y, \psi]^T$	Vessel pose in the local earth-fixed frame	Equation (1)
$v = [u, v, r]^T$	Body-fixed velocity vector: surge, sway, yaw rate	Equation (2)
$R(\psi)$	Rotation matrix from the body-fixed to the earth-fixed frame	Equation (3)
$M, C(v), D(v)$	Inertia, Coriolis/centripetal, and hydrodynamic damping matrices	Equation (4)

$\tau = [X, Y, N]^T$	Generalized force and moment vector; N denotes yaw moment	Equation (5)
δ, n	Commanded rudder angle and propeller speed	Equations (5), (17) and (18)
$d_{t,i}$	Measured range of LiDAR beam i	Equation (6)
$D_{\min}, D_{\max}$	Minimum and maximum measurable LiDAR range	Equation (6)
N	Number of raw LiDAR beams (distinct from the yaw moment N in Equation (5))	Equation (6)
α_i	Bearing angle of LiDAR beam i in the body frame (distinct from the entropy temperature α)	Equation (23)
S_j, M	Angular sector j and the number of sectors (M is distinct from the inertia matrix M in Equation (4))	Equation (7)
$D_{t,j}, c_{t,j}$	Pooled sector distance and normalized sector closeness	Equation (8), Algorithm 1
W_s	Safety-adjusted vessel width used in feasibility pooling	Algorithm 1
o_t, a_t, r_t	Observation, action, and scalar reward at step t (r_t is the reward, distinct from the yaw rate r in Equation (2))	Equations (9), (13) and (16)
γ	Discount factor	Equation (9)
α	SAC entropy temperature (distinct from the beam bearing angle α_i)	Equations (10)–(12)
$Q\phi_1, Q\phi_2$	Twin critic networks	Equation (11)
e_t	Signed cross-track error	Equation (14)
χ_t, χ_t^{LA}	Instantaneous and look-ahead course error	Equations (13) and (22)
D^{LA}	Look-ahead distance	Equation (15)
λ	Nominal mixing coefficient between path-following and obstacle-avoidance reward terms	Equation (19)
$g_{u,t}, U_{ref}$	Forward-speed gate and reference surge speed	Equation (21)
$\omega\chi, \omega^{LA}$	Heading-term multiplier and look-ahead mixing weight	Equations (19) and (22)
K^b, d_{safety}^e	Border penalty gain and soft safety distance	Equation (24)
$R_{goal}, R_{collision}, R_{timeout}$	Terminal rewards for success, collision, and timeout	Equation (25)
K_p, K_s, K_n, U_{min}	Progress, low-speed, and propulsion-regularisation gains, and minimum speed threshold	Equation (26)
$n_c, \Delta n$	Cruise propeller speed and residual propulsion authority	Equations (18) and (27)

Appendix A. Vessel Model Parameters and System Identification

The 3-DOF model of Section 2.2 uses the numerical values below. Added-mass terms are included in the inertia matrix.

Table A1. Numerical values of the vessel model matrices and hydrodynamic coefficients.

Quantity	Symbol	Value	Unit
Mass	m	64.55	kg
Yaw inertia	I_z	10.45	kg·m ²
Added mass in surge	$X_{\dot{u}}$	3.66	kg
Added mass in sway	$Y_{\dot{v}}$	62.74	kg
Added yaw inertia	$N_{\dot{r}}$	0.63	kg·m ²
Linear surge damping	X_u	2.00	kg/s
Linear sway damping	Y_v	18.00	kg/s
Linear yaw damping	N_r	1.50	kg·m ² /s
Rudder moment arm	x_r	1.05	m
Maximum rudder angle	δ_{max}	40.00	deg
Rudder rate limit	$\dot{\delta}_{max}$	20.00	deg/s

References

1. Hansen, P.N.; Enevoldsen, T.T.; Papageorgiou, D.; Blanke, M. Autonomous Navigation in Confined Waters—A COLREGs Rule 9 Compliant Framework. *IFAC-PapersOnLine* **2022**, *55*, 222–228. <https://doi.org/10.1016/j.ifacol.2022.10.435>.

2. Enevoldsen, T.T.; Galeazzi, R. Grounding-aware RRT* for Path Planning and Safe Navigation of Marine Crafts in Confined Waters. *IFAC-PapersOnLine* **2021**, *54*, 195–201. <https://doi.org/10.1016/j.ifacol.2021.10.093>.
3. Qiao, Y.; Yin, J.; Wang, W.; Duarte, F.; Yang, J.; Ratti, C. Survey of Deep Learning for Autonomous Surface Vehicles in Marine Environments. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 3678–3701. <https://doi.org/10.1109/TITS.2023.3235911>.
4. Vagale, A.; Oucheikh, R.; Bye, R.T.; Osen, O.L.; Fossen, T.I. Path planning and collision avoidance for autonomous surface vehicles I: A review. *J. Mar. Sci. Technol.* **2021**, *26*, 1292–1306. <https://doi.org/10.1007/s00773-020-00787-6>.
5. Burmeister, H.-C.; Constapel, M. Autonomous Collision Avoidance at Sea: A Survey. *Front. Robot. AI* **2021**, *8*, 592060.
6. Hu, L.; Hu, H.; Naeem, W.; Wang, Z. A review on COLREGs-compliant navigation of autonomous surface vehicles: From traditional to learning-based approaches. *J. Autom. Intell.* **2022**, *1*, 100003. <https://doi.org/10.1016/j.jai.2022.100003>.
7. Meyer, E.; Heiberg, A.; Rasheed, A.; San, O. COLREG-Compliant Collision Avoidance for Unmanned Surface Vehicle Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 165344–165364. <https://doi.org/10.1109/ACCESS.2020.3022600>.
8. Fossen, T.I. *Handbook of Marine Craft Hydrodynamics and Motion Control*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
9. Villa, J.; Aaltonen, J.; Koskinen, K.T. Path-Following with LiDAR-Based Obstacle Avoidance of an Unmanned Surface Vehicle in Harbor Conditions. *IEEE/ASME Trans. Mechatron.* **2020**, *25*, 1812–1820. <https://doi.org/10.1109/TMECH.2020.2997970>.
10. Gonzalez-Garcia, A.; Collado-Gonzalez, I.; Cuan-Urquiza, R.; Sotelo, C.; Sotelo, D.; Castañeda, H. Path-following and LiDAR-based obstacle avoidance via NMPC for an autonomous surface vehicle. *Ocean. Eng.* **2022**, *266*, 112900. <https://doi.org/10.1016/j.oceaneng.2022.112900>.
11. Kim, J.-S.; Lee, D.-H.; Kim, D.-W.; Park, H.; Paik, K.-J.; Kim, S. A numerical and experimental study on the obstacle collision avoidance system using a 2D LiDAR sensor for an autonomous surface vehicle. *Ocean. Eng.* **2022**, *257*, 111508. <https://doi.org/10.1016/j.oceaneng.2022.111508>.
12. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
13. Woo, J.; Yu, C.; Kim, N. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. *Ocean. Eng.* **2019**, *183*, 155–166. <https://doi.org/10.1016/j.oceaneng.2019.04.099>.
14. Qu, X.; Jiang, Y.; Zhang, R.; Long, F. A Deep Reinforcement Learning-Based Path-Following Control Scheme for an Uncertain Under-Actuated Autonomous Marine Vehicle. *J. Mar. Sci. Eng.* **2023**, *11*, 1762. <https://doi.org/10.3390/jmse11091762>.
15. Cheng, Y.; Zhang, W. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* **2018**, *272*, 63–73. <https://doi.org/10.1016/j.neucom.2017.06.066>.
16. Meyer, E.; Robinson, H.; Rasheed, A.; San, O. Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 41466–41481. <https://doi.org/10.1109/ACCESS.2020.2976586>.
17. Woo, J.; Kim, N. Collision avoidance for an unmanned surface vehicle using deep reinforcement learning. *Ocean. Eng.* **2020**, *199*, 107001. <https://doi.org/10.1016/j.oceaneng.2020.107001>.
18. Teitgen, R.; Monsuez, B.; Kukla, R.; Pasquier, R.; Foinet, G. Dynamic trajectory planning for ships in dense environment using collision grid with deep reinforcement learning. *Ocean. Eng.* **2023**, *281*, 114807. <https://doi.org/10.1016/j.oceaneng.2023.114807>.
19. Hao, S.; Guan, W.; Cui, Z.; Lu, J. USV Collision Avoidance Decision-Making Based on the Improved PPO Algorithm in Restricted Waters. *J. Mar. Sci. Eng.* **2024**, *12*, 1428. <https://doi.org/10.3390/jmse12081428>.
20. Slawik, T.; Wehbe, B.; Christensen, L.; Kirchner, F. Deep Reinforcement Learning for Path-Following Control of an Autonomous Surface Vehicle using Domain Randomization. *IFAC-PapersOnLine* **2024**, *58*, 21–26. <https://doi.org/10.1016/j.ifacol.2024.10.027>.
21. Lambert, R.; Li, J.; Wu, L.-F.; Mahmoudian, N. Robust ASV Navigation Through Ground to Water Cross-Domain Deep Reinforcement Learning. *Front. Robot. AI* **2021**, *8*, 592060.
22. Wang, N.; Wang, Y.; Zhao, Y.; Wang, Y.; Li, Z. Sim-to-Real: Mapless Navigation for USVs Using Deep Reinforcement Learning. *J. Mar. Sci. Eng.* **2022**, *10*, 895. <https://doi.org/10.3390/jmse10070895>.
23. Wang, W.; Cao, X.; Gonzalez-Garcia, A.; Yin, L.; Hagemann, N.; Qiao, Y.; Ratti, C.; Rus, D. Deep Reinforcement Learning Based Tracking Control of an Autonomous Surface Vessel in Natural Waters. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 3109–3115.
24. Tran, N.H.; Nguyen, H.D.; King, P.; Tran, M. Autonomous Surface Vessels Path Following and Static Obstacle Collision Avoidance using Deep Reinforcement Learning. In Proceedings of the 2026 4th International Conference on Mechatronics, Control and Robotics (ICMCR), Tokyo, Japan, 20–22 March 2026; pp. 81–87.
25. Fossen, T.I. *Marine Control System—Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*; Marine Cybernetics: Tiller, Norway, 2002.
26. Wu, X.; Wei, C.; Guan, D.; Ji, Z. Risk-aware deep reinforcement learning for mapless navigation of unmanned surface vehicles in uncertain and congested environments. *Ocean. Eng.* **2025**, *322*, 120446. <https://doi.org/10.1016/j.oceaneng.2025.120446>.

27. Choi, J.; Park, K.; Kim, M.; Seok, S. Deep Reinforcement Learning of Navigation in a Complex and Crowded Environment with a Limited Field of View. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 5993–6000.
28. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12*; MIT Press: Cambridge, MA, USA, 1999.
29. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
30. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
31. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
32. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
33. Breivik, M.; Fossen, T.I. Path following for marine surface vessels. In Proceedings of the Oceans '04 MTS/IEEE Techno-Ocean '04 (IEEE Cat. No.04CH37600), Kobe, Japan, 9–12 November 2004; pp. 2282–2289.
34. Krautwig, B.; Wans, D.; Li, L.; Temmen, T.; Koch, L.; Eisenbarth, M.; Andert, J. Navigating the Trade-Offs: A Quantitative Analysis of Reinforcement Learning Reward Functions for Autonomous Maritime Collision Avoidance. *J. Mar. Sci. Eng.* **2025**, *13*, 2233. <https://doi.org/10.3390/jmse13122233>.
35. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
36. Paulig, N.; Okhrin, O. Robust path following on rivers using bootstrapped reinforcement learning. *Ocean. Eng.* **2024**, *298*, 117207. <https://doi.org/10.1016/j.oceaneng.2024.117207>.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.