applied sciences

MDPI

Article

Deep Reinforcement Learning for Sim-to-Real Robot Navigation with a Minimal Sensor Suite for Beach-Cleaning Applications

Guillermo Cid Ampuero [1], Gabriel Hermosilla [1,*], Germán Varas [2] and Matías Toribio Clark [1]

[1] Escuela de Ingeniería Eléctrica, Pontificia Universidad Católica de Valparaíso (PUCV), Valparaíso 2340025, Chile; guillermo.cid@pucv.cl (G.C.A.); matias.toribio.c@mail.pucv.cl (M.T.C.)
[2] Instituto de Física, Pontificia Universidad Católica de Valparaíso (PUCV), Valparaíso 2340025, Chile; german.varas@pucv.cl
* Correspondence: gabriel.hermosilla@pucv.cl

Abstract

Autonomous beach-cleaning robots require reliable, low-cost navigation on sand. We study Sim-to-Real transfer of deep reinforcement learning (DRL) policies using a minimal sensor suite—wheel-encoder odometry and a single 2-D LiDAR—on a 30 kg differential-drive platform (Raspberry Pi 4). Two policies, Proximal Policy Optimization (PPO) and a masked-action variant (PPO-Mask), were trained in Gazebo + Gymnasium and deployed on the physical robot without hyperparameter retuning. Field trials on firm sand and on a natural loose-sand beach show that PPO-Mask reduces tracking error versus PPO on firm ground (16.6% ISE reduction; 5.2% IAE reduction) and executes multi-waypoint paths faster (square path: 112.48 s vs. 103.46 s). On beach sand, all waypoints were reached within a 1 m tolerance, with mission times of 115.72 s (square) and 81.77 s (triangle). These results indicate that DRL-based navigation with minimal sensing and low-cost compute is feasible in beach settings.

Keywords: sim-to-real transfer; deep reinforcement learning; minimal sensor suite; PPO-mask; low-cost mobile robotics; autonomous navigation

Academic Editor: Rui Araújo

Received: 4 September 2025
Revised: 30 September 2025
Accepted: 3 October 2025
Published: 5 October 2025

Citation: Ampuero, G.C.; Hermosilla, G.; Varas, G.; Clark, M.T. Deep Reinforcement Learning for Sim-to-Real Robot Navigation with a Minimal Sensor Suite for Beach-Cleaning Applications. *Appl. Sci.* **2025**, *15*, 10719. https://doi.org/10.3390/app151910719

1. Introduction

Managing plastic waste on beaches is an operational challenge: about 8 Mt of plastics enter the oceans each year, directly impacting coastal areas [1]. Conventional cleaning methods are labor-intensive and inefficient, motivating robotic solutions that operate on sand with greater autonomy, efficiency, and coverage [2,3]. Existing robotic systems have demonstrated the ability to cover large areas effectively; however, most of them are heavy, industrial-scale machines that weigh several tons, making them costly, energy-intensive, and difficult to deploy flexibly in diverse coastal environments.

This motivates the exploration of developing lighter, more autonomous, and efficient robotic platforms requires careful choices in both sensing and algorithms. From the sensing perspective, two main approaches have emerged: (i) advanced sensors such as 3-D LiDAR and stereo cameras, which provide dense point clouds and semantic maps but entail high cost, energy demand, and computational load that limit long-term operation in coastal environments [4]; and (ii) low-cost configurations based on 2-D LiDAR and wheel encoders, whose effectiveness on granular substrates remains debated [5].

From an algorithmic perspective, DRL can produce adaptive navigation policies without explicit kinematic models, but two challenges remain: (i) the large number of

interactions required for training, which makes the process computationally expensive; and (ii) the simulation-to-real (Sim-to-Real) transfer gap, which is particularly acute on sandy terrains exposed to intense solar reflectance [6,7].

Despite these challenges, recent advances in DRL are encouraging. Quiroga et al. [8] showed that agents trained with *PPO* in CoppeliaSim could navigate controlled beaches while avoiding obstacles. Kiran et al. [9] emphasized the importance of photorealistic simulators and domain randomization in improving generalization, whereas Weerakoon et al. [10] integrated elevation maps and visual-attention mechanisms into a hybrid DRL–classical-control framework, improving navigation performance.

Building on these advances, we present a lightweight four-wheeled differential-drive robot (30 kg) equipped with a minimal, low-cost sensor suite (2-D LiDAR and wheel encoders) and on-board computation on a Raspberry Pi 4 (Ubuntu 22.04, ROS 2 Humble [11]). The platform is controlled by DRL policies—specifically, PPO and its masked-action variant PPO-Mask, trained in simulation and transferred to the real robot without hyperparameter retuning. This integration of accessible hardware and DRL-based navigation enables autonomous operation on sandy terrains under real beach conditions.

In this context, the main contributions of this work are

- A complete autonomous navigation pipeline combining minimal sensing and DRL control;
- End-to-end Sim-to-Real transfer of PPO and PPO-Mask without hyperparameter retuning;
- Quantitative outdoor validation on both firm sand and natural loose-sand beaches.

This paper is organized as follows. Section 2 reviews the relevant literature on coastal cleaning robots, Sim-to-Real navigation, and DRL. Section 3 describes the robot platform and control architecture. Section 5 presents the DRL methods and training setup, while Section 7 reports the evaluation and field results. Section 8 discusses the findings, and Section 9 concludes the work with future directions.

2. Related Work

Currently, nearly 140 million tons of plastic are accumulated in water bodies such as rivers, lakes, and oceans, and projections indicate that, without mitigation, this figure could rise to approximately 500 million tons by 2060 [12]. The accumulation of plastic waste on beaches poses a growing threat to coastal ecosystems, affecting not only biodiversity but also economic activities such as tourism and fishing [13,14]. In this context, it is essential to address three complementary lines of research: (i) the development and implementation of robots specialized in coastal cleaning, (ii) the application of advanced autonomous navigation methods using Sim-to-Real strategies with minimal sensory configurations, and (iii) the incorporation of adaptive and safety-oriented frameworks in DRL.

2.1. Coastal Cleaning Robots

Coastal robotics has advanced significantly through solutions that improve collection efficiency and operational autonomy. Current developments include intelligent detection systems such as EcoBot, which integrates YOLO-based computer vision on accessible hardware [15], and robots equipped with YOLOv5 detection that combine raking and sifting [16]. UmiBeach employs vision and LiDAR sensors for safe operation under adverse conditions [17]. In autonomous navigation, Binman combines GPS with ROS to cover large coastal areas [18], while Hirottaro uses pole-based positioning and a rangefinder [19]. SMURF implements nonlinear predictive control for surface cleaning of water bodies [20]. Specialized mechanisms include a spiral-propelled amphibious robot for intertidal zones [21], VERO with a quadruped design and vacuum cleaner [22], BeBot as a

silent electric solution [3], SURF RAKE as a high-capacity towed system [2], and ProjectBB for robotic swarm coordination [23].

Based on these trends, our approach adopts accessible hardware such as a Raspberry Pi 4, which ensures compatibility with the ROS 2 Humble middleware and maintains low energy consumption. Following the widespread use of LiDAR in coastal robotics, we employ a single 2-D LiDAR as the primary perception sensor, combined with wheel encoders for odometry. In addition, the variety of mechanical architectures described—from amphibious platforms to towed systems—has informed the design of a more compact and lightweight configuration, optimized to operate efficiently on loose sandy terrain and intended for future integration into autonomous beach-cleaning tasks.

2.2. Sim-to-Real Navigation with Minimal Sensor Approaches

The transfer of navigation policies from simulated environments to real systems is a promising paradigm for reducing costs and accelerating robotic implementation with minimal sensory configurations. Recent studies have validated the use of inexpensive 2-D LiDAR for mapless navigation, combining experience collection and fusion techniques that accelerate convergence under localization uncertainty [24]. Successful transfers from NVIDIA Isaac Sim to real platforms with ROS 2 have achieved performance comparable to Nav2 [25]. Parallel actor–critic distributional architectures with exclusive laser data input and target vectors improve generalization [26], while hierarchical reinforcement learning (HRL) approaches implement dynamic subgoals validated on TurtleBot3 [27]. In coastal cleanup specifically, Sim-to-Real strategies have enabled direct transfer of trained behaviors, achieving safe navigation without post-retuning [8]. Platform-agnostic frameworks further minimize simulation–reality discrepancies, ensuring consistent performance [6].

These advances confirm the feasibility of efficient, low-cost navigation systems, although challenges persist on loose sand surfaces and under variable environmental conditions.

2.3. Adaptive and Safety-Oriented RL Frameworks for Dynamic Environments and Constrained Hardware

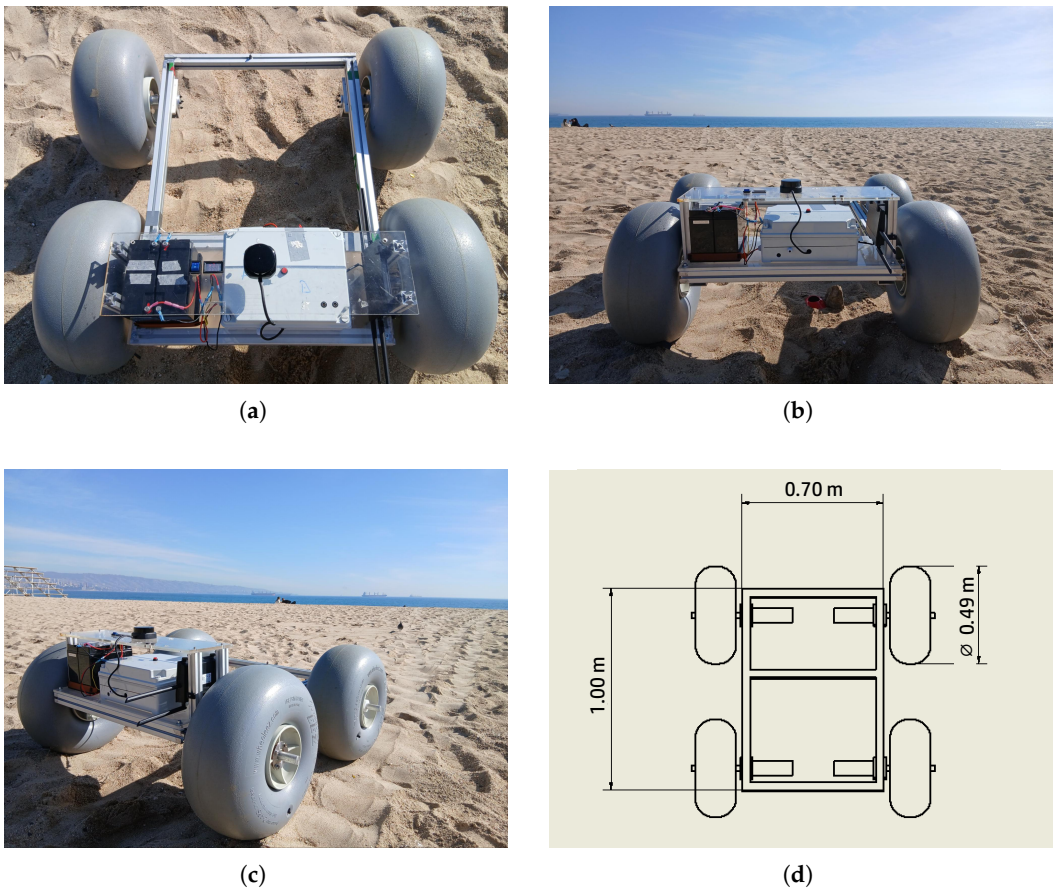
Coastal environments present particular challenges, including dynamic terrain variations and embedded hardware constraints. Adaptive and safety-oriented frameworks have emerged as fundamental solutions for robust operations. Continuous domain adaptation through randomization sustains performance in the face of friction and tilt variations. Conflict-Averse Safe Reinforcement Learning (CASRL) [28] reduces collisions by separating critical subtasks [29]. The SafeDPA framework combines barrier functions with predictive control to ensure safety under unmodeled perturbations [30], while barrier certificate approaches guarantee stability in nonstationary dynamics [31]. Predictive safety filters adapt policies in real time [32], and hierarchical strategies optimize multi-agent cooperation [33]. Fault-tolerant control preserves performance under partial actuator failures [34], while event-triggered control reduces computational load without compromising stability [35]. Methods that integrate GANs with attention networks enhance detection under adverse visual conditions [36], a critical aspect in coastal environments with frequent light variations.

Recent advances have demonstrated the feasibility of applying Sim-to-Real strategies to coastal navigation. However, most existing approaches still rely on costly sensors, photorealistic simulators, or multi-sensor configurations, which restrict their scalability and practical use in natural beach environments. This highlights the need for lightweight and accessible platforms that can be directly validated under real conditions. In response to this gap, our work introduces a compact and low-cost robot that combines a minimal sensor suite (2-D LiDAR and wheel encoders) with DRL policies. By employing PPO and its

masked variant, the system achieves autonomous navigation on both firm and loose sand, demonstrating that reliable Sim-to-Real transfer can be accomplished without expensive hardware or complex sensing pipelines.

3. Robot Hardware and Software Architecture

The platform is a four-wheeled differential-drive robot designed to operate on sandy terrain. Its welded structural-aluminum chassis weighs approximately 30 kg and has a footprint of 1.00 m × 0.70 m, combining rigidity with low weight and maneuverability. Mobility is provided by low-pressure polyurethane wheels (0.49 m diameter and 0.23 m width) that enhance traction while reducing ground pressure; each wheel supports up to 120 kg, allowing future payload integration. All electronic components are housed in an IP55-protected compartment, and a dedicated space is reserved for the cleaning module to be incorporated in the next stage of the project. Figure 1 shows the prototype during its first field tests on the beach, together with its CAD schematic.



(**a**) (**b**) (**c**) (**d**)

Figure 1. Four-wheel differential-drive robotic platform during its initial beach trials: (**a**) top view showing the aluminum chassis and the layout of the electronic components; (**b**) rear view highlighting the electronics compartment and sensor mounting; (**c**) perspective view illustrating the overall design and the low-pressure wheels for sand mobility; (**d**) CAD drawing with dimensions and wheel arrangement.

While Figure 1 provides a visual overview of the prototype, Table 1 complements it by consolidating the main technical specifications. The table lists key parameters such as dimensions, weight, motors, batteries, sensors, and computation units, offering a concise reference for the detailed description that follows.

Table 1. Technical specifications of the beach-cleaning robot prototype.

Parameter	Specification
Chassis	Welded structural-aluminum frame, IP55 electronics compartment
Dimensions	1.00 m × 0.70 m footprint
Weight	≈30 kg (without payload)
Wheels	Low-pressure polyurethane, diameter 0.49 m, width 0.23 m, traction optimized for sand
Motors	Two 24 V DC gear motors, 100 kg·cm torque, front-wheel differential drive
Encoders	Integrated optical encoders on each motor
Motor drivers	BTS7960 H-bridges, one per motor
Low-level	ESP32 with micro-ROS for motor control and sensor feedback
High-level	Raspberry Pi 4 (Ubuntu 22.04, ROS 2 Humble)
Sensors	RPLIDAR S2 (2-D LiDAR, 30 m range, 32 kHz, 0.1125° resolution) + wheel encoders
Power supply	Two 12 V, 18 Ah batteries in series (24 V nominal), ≈3 h autonomy
Power distribution	Three DC/DC converters: (i) 5 V, 20 A for Raspberry Pi 4; (ii) 12 V, 3 A for Wi-Fi router; (iii) 5 V, 3 A for ESP32 and electronics
Connectivity	2.4 GHz Wi-Fi router, external PC for mission dispatch and monitoring
Future expansion	Reserved compartment for cleaning module

To provide a clearer understanding of the platform, the following subsections describe the hardware and software components in detail, highlighting how sensing, actuation, computation, and communication are integrated into a unified architecture.

(a) **Hardware:**
The traction system relies on two 24 V DC gear motors (100 kg·cm torque) with integrated optical encoders, selected for their balance between torque and efficiency in sandy environments. Motor control is performed by two BTS7960 H-bridge drivers, which offer robustness against high current peaks while remaining cost-effective. An ESP32 microcontroller was adopted as the low-level controller due to its native support for micro-ROS [37], ensuring seamless communication with ROS 2 nodes. The power supply is based on two 12 V, 18 Ah batteries in series, providing a 24 V nominal voltage and approximately 3 h of autonomy, which was deemed sufficient for experimental field trials while maintaining a manageable total weight of 30 kg. The power distribution unit incorporates three DC/DC converters to decouple loads, preventing voltage drops in critical modules such as the Raspberry Pi 4. Environmental perception relies on an RPLIDAR S2 2-D LiDAR [38], offering a 30 m range, 32 kHz sampling rate, and 0.1125° angular resolution. This model is specifically designed for outdoor use, ensuring robust performance under direct sunlight, airborne dust, and reflective sandy surfaces, which makes it well-suited for real beach conditions. High-level processing is carried out by a Raspberry Pi 4, which executes the trained DRL policies, while wireless connectivity via a 2.4 GHz router allows remote mission dispatch and monitoring.

(b) **Software:**
The high-level computation is handled by a Raspberry Pi 4 running Ubuntu 22.04 with ROS 2 Humble, which coordinates sensing, state estimation, and autonomous navigation. The trained DRL policies (PPO and PPO-Mask) are deployed as ROS 2 nodes, publishing `geometry_msgs/Twist` commands derived from 2-D LiDAR scans and wheel-encoder odometry. Low-level actuation and motor feedback are

executed on the ESP32 using micro-ROS, enabling real-time communication with the motor drivers and ensuring precise control. This division between high-level decision-making (navigation policies) and low-level execution (actuation and sensing) provides both modularity and reliability. Wireless connectivity via a 2.4 GHz router allows remote operators to dispatch missions and monitor robot status in real time, facilitating field experiments in beach environments.

Figure 2 illustrates the overall hardware and software architecture of the robot. The diagram highlights the interaction between three main subsystems: (i) the **low-level controller**, composed of the ESP32 microcontroller and the BTS7960 motor drivers, responsible for actuation and encoder feedback; (ii) the **high-level controller**, implemented on the Raspberry Pi 4, which processes LiDAR scans, integrates odometry, and executes the DRL navigation policies; and (iii) the **power distribution unit**, which regulates the 24 V battery supply through multiple DC/DC converters to feed each component with its required voltage level. In addition, the figure emphasizes the role of the Wi-Fi router in establishing wireless communication with an external computer for mission dispatch and monitoring.

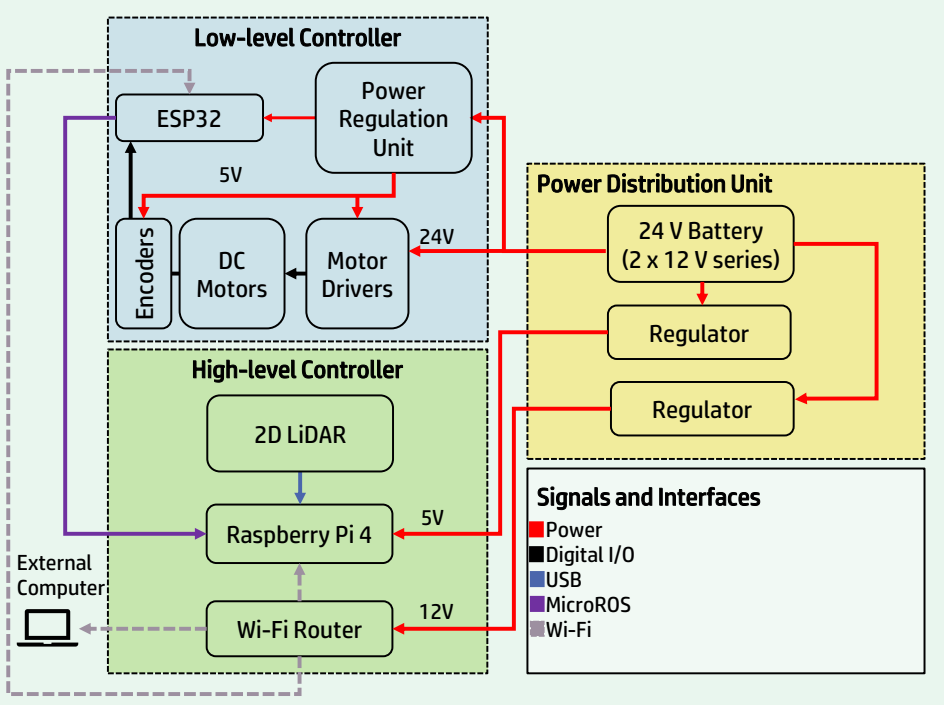


Figure 2. General hardware and software architecture of the robot, including the low-level controller (ESP32 and BTS7960 driver), the high-level controller (Raspberry Pi 4 and 2D LiDAR), and the power distribution unit (batteries and DC/DC regulators).

4. Differential Kinematic Model and Navigation Geometry

The robot's motion follows a differential-drive kinematic model [39], which relates the commanded linear and angular velocities to the evolution of its planar position and heading. The resulting state–space relations are expressed in Equations (1)–(3).

$$\dot{x} = v\cos\theta, \tag{1}$$

$$\dot{y} = v\sin\theta, \tag{2}$$

$$\dot{\theta} = \omega, \tag{3}$$

where (x, y) denotes the robot's position in the global frame, θ is the heading angle measured counter-clockwise, v is the linear velocity $(\mathrm{m\,s^{-1}})$, and ω is the angular velocity $(\mathrm{rad\,s^{-1}})$.

To guide the robot toward a target point $(x_{\mathrm{tp}}, y_{\mathrm{tp}})$, the geometric errors that quantify the remaining distance and the misalignment with the goal are defined in Equations (4)–(6) [40]:

$$\alpha = \tan^{-1}\left(\frac{y_{\mathrm{tp}} - y_s}{x_{\mathrm{tp}} - x_s}\right), \tag{4}$$

$$d = \sqrt{(x_{\mathrm{tp}} - x_s)^2 + (y_{\mathrm{tp}} - y_s)^2}, \tag{5}$$

$$e = \alpha - \theta, \tag{6}$$

where (x_s, y_s) denotes the current robot position, d is the Euclidean distance to the target (m), α is the bearing angle, and e represents the difference between the desired and current headings.

These expressions form the mathematical backbone of the navigation system and are embedded in the DRL environment. Kinematics model and sensor validation was first carried out by teleoperating the platform connected to ROS 2 , confirming real-time motor response as well as correct encoder and LiDAR readings.

5. Deep Reinforcement Learning

In deep reinforcement learning, an agent interacts with an environment modeled as a Markov decision process $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$. At each discrete time step t, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to its policy $\pi_\theta(a_t|s_t)$, receives a reward $r_t = R(s_t, a_t)$, and transitions to s_{t+1} with probability $P(\cdot|s_t, a_t)$, as illustrated in Figure 3.

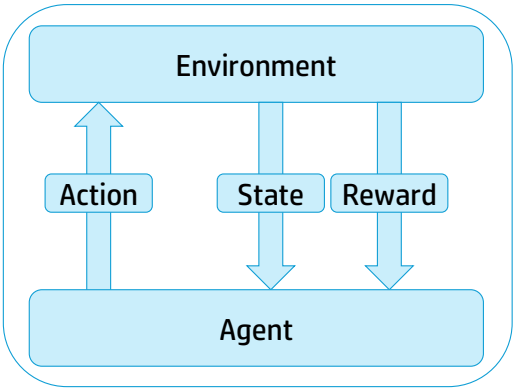


Figure 3. Agent-environment interaction loop in DRL.

The objective is to maximize the expected return G_t defined in Equation (7):

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \qquad 0 < \gamma < 1, \tag{7}$$

where γ is the *discount factor*. A value of γ close to 1 makes the agent far-sighted, placing nearly equal emphasis on distant rewards, whereas a smaller γ induces short-sighted (myopic) behavior by weighting immediate rewards more heavily. The discount factor also ensures convergence of the infinite series in Equation (7), resulting in a well-defined optimization problem.

For the beach-cleaning robot—where wheel–sand dynamics and moving obstacles are difficult to model—DRL is particularly suitable, as it learns policies directly from experience and adapts to unpredictable variations.

5.1. Proximal Policy Optimization

PPO [41] was adopted for its balance between training stability and computational efficiency. The clipped surrogate objective, shown in Equation (8), constrains the magnitude of each policy update:

$$L_{\text{clip}}(\theta) = \mathbb{E}_t\big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}\big(r_t(\theta),\,1\pm\varepsilon\big)\,\hat{A}_t\big)\big], \tag{8}$$

where $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the likelihood ratio between the new and old policies; $\hat{A}_t$ is the advantage estimated via Generalized Advantage Estimation (GAE); and ε is the clipping threshold. GAE interpolates between Monte Carlo and TD(1) returns through the parameter $\lambda \in [0, 1]$, reducing variance without introducing excessive bias, while the clip operator constrains the updated policy to a proximal region that preserves training stability.

The training loop comprises four stages:

(a) **Collection:** Generate trajectories with the current policy for T steps.
(b) **Estimation:** Compute $\hat{A}_t$ from the rewards and the value critic.
(c) **Optimization:** Perform stochastic gradient descent over mini-batches and multiple epochs, applying early stopping if the KL divergence exceeds a threshold.
(d) **Iteration:** Repeat the process until the average return converges.

5.2. PPO-Mask: Masking Invalid Actions

The discrete action space includes commands that, in certain states, are physically infeasible (e.g., a turn exceeding the robot's minimum turning radius). The *PPO-Mask* variant [42] addresses this issue by subtracting a large negative constant from the logits of invalid actions before the softmax is applied, as shown in Equation (9):

$$\pi_\theta(a \mid s) = \frac{\exp\big(z_a - M\,\mathbb{1}_{\{a\notin\mathcal{A}_{\text{val}}(s)\}}\big)}{\sum_{b\in\mathcal{A}}\exp\big(z_b - M\,\mathbb{1}_{\{b\notin\mathcal{A}_{\text{val}}(s)\}}\big)}, \tag{9}$$

where $M = 10^8$ [42], ensuring that masked actions receive negligible probability without risking floating-point precision errors. Here, $\mathcal{A}_{\text{val}}(s)$ denotes the set of admissible actions in state s, z_a is the pre-softmax logit for action a, and $\mathbb{1}_{\{\cdot\}}$ is the indicator function. Because the term $-M$ is extremely negative, the softmax assigns an effectively zero probability to invalid actions, and their gradients vanish $\big(\nabla_{z_a}\log\pi_\theta(a \mid s) = 0\big)$.

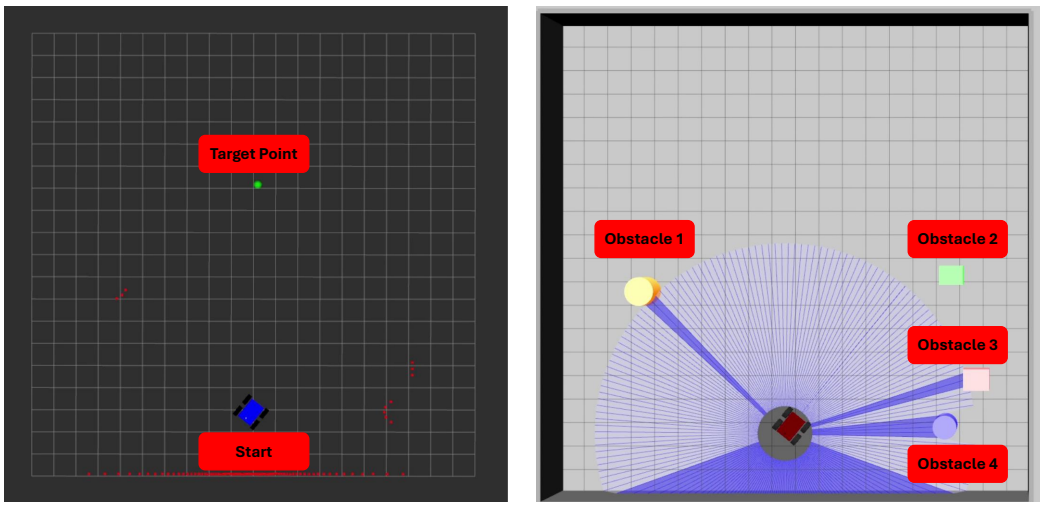
Both PPO and its masked variant were selected as the learning algorithms for this work. PPO is widely recognized for its training stability and strong performance in robotic navigation tasks [8,9], and it had already shown favorable results in our previous studies on autonomous navigation [8]. In contrast, PPO-Mask was incorporated based on its reported ability to filter out infeasible actions and improve reliability in complex terrains, making it a promising candidate for Sim-to-Real transfer in sandy environments.

6. Experimental Setup for Agent Training

The training environment consisted of a 20 m × 20 m Gazebo world, as illustrated in Figure 4. The world was procedurally configured to increase task complexity and improve the robustness of the learned policies, for which it was populated with two types of static obstacles, cylinders (0.60 m in diameter and 0.80 m in height) and cuboids (0.80 m × 1.00 m × 0.60 m), all sufficiently large to be reliably detected by the 2-D LiDAR. At the beginning of each training episode, these obstacles were randomly placed within the environment to ensure variability across episodes and to prevent the agent from memorizing their locations.

To approximate the interaction between the wheels and sand, the simulation employed friction and contact stiffness parameters rather than granular deformation or terrain height variations. The robot model preserved physical properties consistent with the real prototype (mass, dimensions, and wheel geometry), using static and kinetic friction coefficients $\mu_1 = 1.2$ and $\mu_2 = 1.0$, contact stiffness $k_p = 500{,}000$, and damping $k_d = 1.0$.

Within this environment, policies were trained entirely offline in Gazebo and later deployed on the Raspberry Pi 4, thereby keeping on-board computation minimal while ensuring consistency between simulated and real-world dynamics.



(**a**) (**b**)

Figure 4. Training environment visualization in RViz and Gazebo. (**a**) RViz interface showing the robot's starting position and target point. (**b**) Gazebo scene with LiDAR-based obstacle detection and four labeled obstacles within the sensor's perception area.

6.1. Gymnasium Environment

The environment was designed to encourage obstacle-avoidance behavior. Target points were spawned behind obstacles relative to the robot's initial pose, forcing it to detour around them to reach the goal. This strategy prevented trivial straight-line paths and promoted more robust navigation policies.

The implementation followed the `Gymnasium` API, providing `reset()`, `step()`, `render()`, and `close()` methods to ensure reproducibility and enable comparison with other DRL platforms. We employed the *PPO* algorithm from Stable-Baselines3 (SB3) [43], recognized for its robustness and efficiency in continuous control tasks, together with the *PPO-Mask* variant from SB3-Contrib [44]. To clarify how these algorithms interact with the training setup, the following subsections describe (a) the observation and action spaces that define the agent's interaction with the environment, and (b) the reward function that guides learning by balancing goal-reaching efficiency with obstacle avoidance.

(a) **Observation and Action Spaces:** The observation space provides a rich set of sensory and contextual data that enables the robot to make informed decisions during navigation. By combining LiDAR readings, the direction toward the goal, and the robot's kinematic states, it yields a comprehensive representation of both the environment and the agent itself.
The action space is discrete because the physical robot demonstrated greater stability with quantized commands, as confirmed by teleoperation tests. This choice facilitates the direct transfer of policies from simulation to hardware, thereby improving robustness. The complete definitions of both spaces are given in Tables 2 and 3.

Table 2. Observation space used by the DRL policy.

Component	Description	Dimension	Range
LiDAR data	Sector-based distance readings	36	Normalized [0, 1]
Distance to goal	Euclidean distance to the target	1	Normalized [0, 1]
Direction vector	x and y components of the goal vector	2	Normalized $[-1, 1]$
Linear velocity	Current forward speed	1	Fraction of $v_{\max} = 0.8\ \mathrm{m\,s^{-1}}$
Angular velocity	Current rotational speed	1	Fraction of $\omega_{\max} = 0.6\ \mathrm{rad\,s^{-1}}$
Total		**41**	

Table 3. Action space used by the DRL policy.

ID	Description	Command	v $(\mathrm{m\,s^{-1}})$	ω $(\mathrm{rad\,s^{-1}})$
0	Move forward	Forward translation	0.8	0
1	Turn left	Counter-clockwise turn	0	+0.6
2	Turn right	Clockwise turn	0	−0.6

(b) **Reward function:** The reward function combines terminal bonuses, which are triggered by the completion of an episode, with dense components that guide the robot during navigation. Terminal rewards provide clear learning signals at the end of an episode, while dense rewards shape the trajectory by penalizing unsafe or inefficient behaviors and reinforcing desirable actions. This hybrid design accelerates policy convergence and promotes reliable navigation in challenging environments, as summarized in Equation (10).

$$\mathrm{Reward}_t = \begin{cases} +200, & d_t < d_{\mathrm{goal}}, \\ -150, & d_{\min} < d_{\mathrm{coll}}, \\ -100, & t \geq \text{max\_steps}, \\ 8\,p_t + 3\,v_t \cos\theta_t - 1.5\,|\omega_t| - 0.8\,\mathbb{1}\{|v_t| < 0.03\} & \\ \quad -5\left(\frac{d_{\mathrm{coll}} - d_{\min}}{d_{\mathrm{coll}}}\right) + 2\,\mathbb{1}\{d_{\min} > d_{\mathrm{safe}}\} + 3\,\mathbb{1}\{|\theta_t| < 0.1\}, & \text{otherwise.} \end{cases} \tag{10}$$

The terminal terms correspond to a success reward of +200 when the robot reaches the goal, a collision penalty of −150, and a timeout penalty of −100 when the episode ends without success. The dense terms act at each timestep: the progress term p_t and the alignment term $v_t \cos\theta_t$ encourage forward motion and goal-oriented heading; the angular-rate penalty $|\omega_t|$ discourages sharp turns; and the idling penalty $\mathbb{1}\{|v_t| < 0.03\}$ prevents stagnation. Collision avoidance is promoted by the shaping term $\frac{d_{\mathrm{coll}} - d_{\min}}{d_{\mathrm{coll}}}$, which penalizes proximity to obstacles, while the safety bonus $\mathbb{1}\{d_{\min} > d_{\mathrm{safe}}\}$ rewards maintaining a safe distance. Finally, the orientation bonus $\mathbb{1}\{|\theta_t| < 0.1\}$ reinforces accurate heading alignment.

Together, these components balance efficiency, safety, and robustness, enabling successful Sim-to-Real transfer in both firm ground and beach environments.

6.2. Training

The simulation environment was implemented entirely in Python 3, combining Gazebo for physics, ROS 2 Humble for inter-node and sensor communication, and Gymnasium to expose an API compatible with DRL algorithms.

On this infrastructure, we employed Stable-Baselines3 to train the standard *PPO* algorithm, and SB3-Contrib to train the masked variant, *PPO-Mask*. All experiments were carried out on an Ubuntu 22.04 workstation equipped with a Ryzen 7 3700X CPU and an RTX 3060 GPU.

Table 4 summarizes the hyperparameter ranges explored during training; each agent completed 600,000 training steps in approximately three hours.

Table 4. Explored hyperparameter ranges.

Parameter	Criterion	PPO Range	Final PPO	PPO-Mask Range	Final PPO-Mask
`n_steps`	Max. reward	{1024, 2048, 4096}	4096	Inherited	4096
`batch_size`	Max. reward	{128, 256, 512}	512	Inherited	512
`gae_lambda`	Max. reward	0.85–0.96	0.85	Inherited	0.85
`gamma`	Max. reward	0.90–0.99	0.99	Inherited	0.99
`learning_rate`	Max. reward	1×10^{-4}–3×10^{-4}	1×10^{-4}	Inherited	1×10^{-4}
`ent_coef`	Max. reward	0.01–0.05	0.05	Inherited	0.05
`clip_range`	Default	-	0.20	Default	0.20
`n_epochs`	Default	-	10	Default	10

Table 5 compares the performance of the two algorithms explored during training: *PPO* and *PPO-Mask*. When comparing the two methods, an apparent discrepancy arises: PPO achieved the highest success rate (94.3%), yet its average reward was lower than that of PPO-Mask. This outcome is explained by the reward function defined in Equation (10), which evaluates not only goal completion but also trajectory efficiency, motion stability, and mission time. PPO reached the destination more frequently, but at the cost of longer trajectories and higher control effort, leading to greater penalties. In contrast, PPO-Mask obtained more direct and stable routes with fewer oscillations and unnecessary corrections, which favored higher cumulative rewards per episode.

Table 5. Comparison of training performance metrics for PPO and PPO-Mask.

Algorithm	Episodes	Success Rate	Mean Reward	Max Reward	Std. Dev.
PPO	103,923	94.3%	203.97	496.82	84.14
PPO-Mask	15,900	86.9%	277.74	1585.85	190.75

Figure 5 shows that the standard *PPO* policy converged at approximately +200 points per episode, whereas *PPO-Mask* achieved about +300 points with lower variance, confirming that action masking accelerated convergence and improved stability.

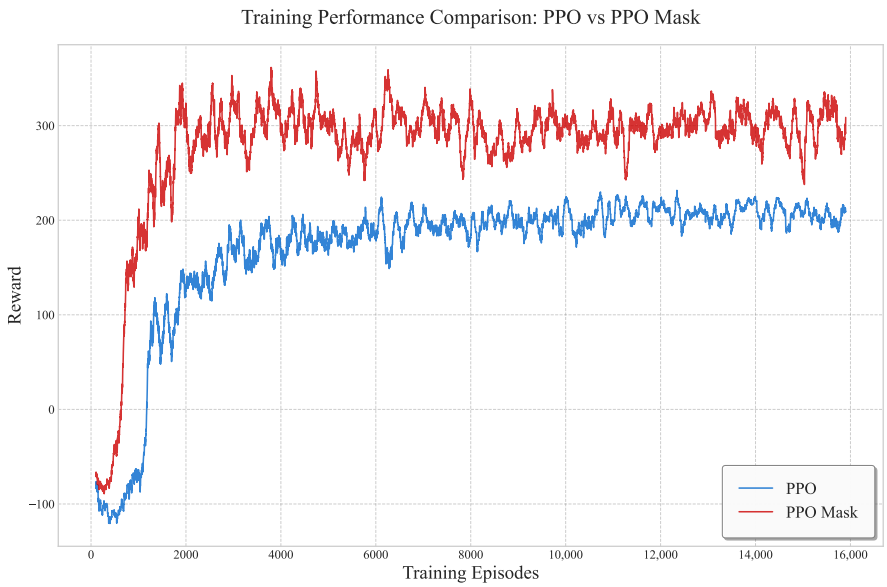


Figure 5. Training reward curves: *PPO* versus *PPO-Mask*.

7. Experiments and Results

This section reports the experiments conducted to evaluate the Sim-to-Real transfer of the proposed policies. The trained policies were deployed on the physical robot without modifying any hyperparameters, the only adjustment concerned the episode duration: in simulation, timesteps per episode were shortened to accelerate training, whereas in reality they were extended to account for the slower dynamics of physical execution.

7.1. Field Experiments on Firm Sand

To verify whether the learned policy could be effectively transferred from simulation to reality, experiments were first conducted on firm sand. Figure 6 shows the firm-ground test environment where these initial trials took place, aimed at providing a preliminary assessment of the robot's performance and validating the proper transfer of the policy before deployment in more demanding scenarios such as the beach.

(**a**) (**b**)

Figure 6. Views of the firm-ground test environment and the robotic platform. (**a**) Front view of the robot positioned on the terrain. (**b**) Wide perspective of the test area, showing the ground conditions and available space for evaluation.

7.1.1. Experiment 1—Simple Navigation Task

The first experiment evaluated the policies trained in simulation, with moving the robot from the origin (0, 0) m to a target located at (3, 3) m on an obstacle-free plane.

This experiment aimed to verify whether the policies learned in simulation could be successfully transferred to the real robot, assessing their ability to generate stable and accurate trajectories in the absence of elements interfering with its movement.

Figure 7 presents the trajectories obtained in this experiment for both the simulation and the physical robotic platform, enabling a direct comparison between the *PPO* and *PPO-Mask* policies.

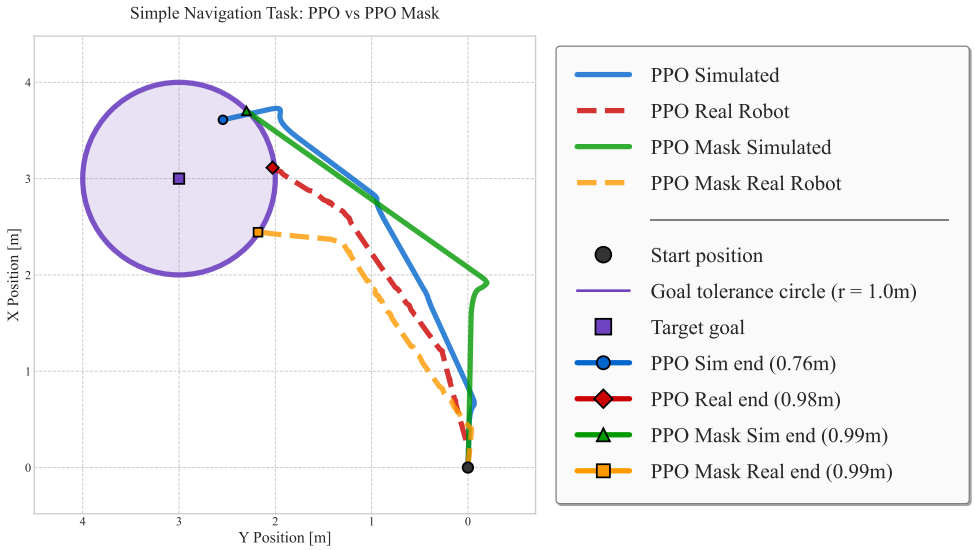


Figure 7. Simple navigation task: comparison between *PPO* and *PPO-Mask* (simulation vs. real robot).

The performance gap is evident in the trajectories of Figure 7. In both simulation and the physical robotic platform, *PPO-Mask* produced straighter paths with less lateral oscillation than *PPO*. This improvement is attributed to masking kinematically infeasible actions during sampling, which prevented spasmodic corrections and yielded more coherent control commands.

To quantify these differences, four classical control indices were used—the Integral of Absolute Error (IAE), the Integral of Squared Error (ISE), the Integral of Time-weighted Absolute Error (ITAE), and the Integral of Time-weighted Squared Error (ITSE)—defined in Equations (11)–(14) [45]. In all expressions, the error $e(t)$ denotes the instantaneous distance between the robot center and the target point. These indices quantify cumulative tracking accuracy: IAE and ISE measure overall position error, whereas ITAE and ITSE additionally weight that error by time, penalizing late or persistent deviations.

$$\mathrm{IAE} = \int_0^\infty |e(t)|\, dt, \tag{11}$$

$$\mathrm{ISE} = \int_0^\infty e^2(t)\, dt, \tag{12}$$

$$\mathrm{ITAE} = \int_0^\infty t\,|e(t)|\, dt, \tag{13}$$

$$\mathrm{ITSE} = \int_0^\infty t\, e^2(t)\, dt. \tag{14}$$

As shown in Table 6, the simulation results indicate that both policies followed nearly straight trajectories; however, *PPO-Mask* achieved the lowest IAE, ITAE, and ITSE, whereas *PPO* attained the minimum ISE. On the real robot, the absolute values increased—owing to slippage and sensor noise—yet the relative trend persisted: *PPO-Mask* reduced ISE by 17%, IAE by 5%, and ITSE by 15% compared with *PPO*, confirming more accurate goal tracking. *PPO* retained a slight (≈2.6%) advantage in ITAE, suggesting marginally faster initial

corrections. Overall, these results confirm that action masking improves smoothness and *Sim-to-Real* robustness, although the magnitude of the benefit depends on the chosen metric.

Table 6. Performance index comparison between PPO and PPO-Mask in real and simulated environments.

Index	Real		Simulated	
	PPO	PPO-Mask	PPO	PPO-Mask
ISE	434.99	**362.85**	**73.35**	73.79
IAE	135.93	**128.92**	24.51	**22.91**
ITSE	6932.79	**5869.30**	203.90	**200.16**
ITAE	**2601.61**	2669.86	89.73	**73.77**

Bold values indicate the best performance (a lower index indicates better performance).

7.1.2. Experiment 2—Waypoint-Based Path Following

This second experiment evaluated a more demanding task: closed-loop path following defined solely by discrete waypoint sequences. Each waypoint was assigned a tolerance radius of 1 m, equivalent to the robot's effective clean-up area; once inside this radius, the controller automatically advanced to the next goal. The entire route was completed without reinitializing the policy, enabling assessment of navigation consistency over extended trajectories. Two reference geometries were tested, a square and a triangle, imposing 90° and 120° turns, respectively, with different leg lengths.

(a) **Square path:** The results in Figure 8 show that, in simulation, *PPO-Mask* generated straighter legs between waypoints and turns with slightly larger radius, achieving smoother transitions than the *PPO* policy. After transfer to the physical robotic platform, firm sand introduced additional slip—particularly after each 90° turn—yet the relative superiority of *PPO-Mask* persisted: it avoided overshooting the vertices, maintained straighter lines, and spent less time within each waypoint's tolerance circle, resulting in more efficient navigation.

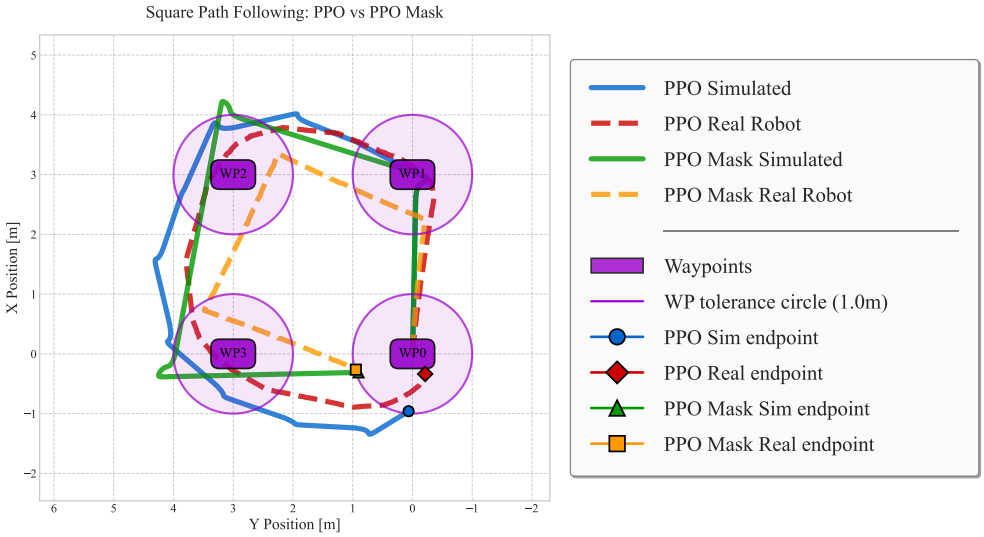


Figure 8. Square path following: comparison between *PPO* and *PPO-Mask* (simulation vs. real robot).

Table 7 quantitatively confirms the advantage of *PPO-Mask*. In simulation, it completed the square in 24.42 s with 235 steps, whereas *PPO* required 27.89 s and 270 steps—improvements of 12.4% in time and 13% in step efficiency. The trend held in the real environment, where *PPO-Mask* finished in 103.46 s compared to 112.48 s for *PPO* (≈8%

faster). Moreover, *PPO-Mask* achieved higher mean rewards in both settings, namely, 7.94 in simulation and 4.13 on hardware, compared to 7.12 and 3.75 for *PPO*, reflecting more effective and consistent navigation.

Table 7. Performance comparison in simulated and real environments for the square route.

Policy	Metric	Simulated	Real
PPO	Time (s)	27.89	112.48
	Steps	270	594
	Mean reward	7.12	3.75
PPO-Mask	Time (s)	**24.42**	**103.46**
	Steps	**235**	**569**
	Mean reward	**7.94**	**4.13**

The best value for each metric is highlighted in bold (lowest values are better, except for mean reward where higher is better).

(b) **Triangular path:** In this case, the triangular route consisted of three vertices spaced approximately 4 m apart, resulting in wide turns of about 120°. This geometry posed a different challenge from the square path, as the wider corners required prolonged maneuvers and more precise control of angular velocity during each transition. The results in Figure 9 show that, in both simulation and real-world testing, *PPO-Mask* achieved lower integrated errors, demonstrating better handling of wide turns. The masked policy produced commands that enabled smoother transitions without compromising tracking accuracy—a benefit that became more evident on hardware, where sand dynamics amplified control errors.

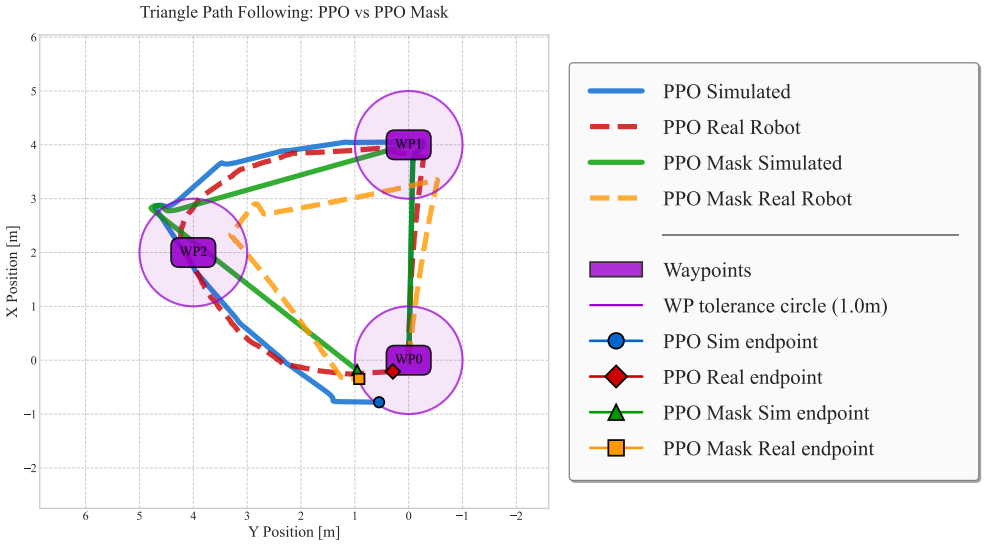


Figure 9. Triangle path following: comparison between *PPO* and *PPO-Mask* (simulation vs. real robot).

Table 8 quantifies these findings. In simulation, *PPO-Mask* completed the triangle in 22.75 s and 219 steps, outperforming *PPO*—26.20 s and 254 steps—equivalent to 13.2% faster and 13.8% more step-efficient.

In the real environment, however, *PPO* achieved a slightly shorter total time (104.37 s vs. 116.71 s) and fewer steps. *PPO-Mask* nonetheless yielded smoother, more curvilinear trajectories. This smoothness proved advantageous during field tests on physically uneven sand, even though the simulator remained flat. The higher mean reward of *PPO-Mask* in both settings (8.25 and 4.45 vs. 7.38 and 3.92) further confirmed its robustness on demanding paths.

Table 8. Performance comparison in simulated and real environments for the triangular route.

Policy	Metric	Simulated	Real
PPO	Time (s)	26.20	**104.37**
	Steps	254	**581**
	Mean reward	7.38	3.92
PPO-Mask	Time (s)	**22.75**	116.71
	Steps	**219**	638
	Mean reward	**8.25**	**4.45**

The best value for each metric is highlighted in bold (lowest values are better, except for mean reward where higher is better).

7.1.3. Experiment 3—Dynamic Obstacle Avoidance

The third experiment evaluated the policy's capability to react to unforeseen obstacles that suddenly appeared in the planned path. In simulation, the term *dynamic obstacle* was used in a spawn-only sense (the obstacle appeared after the episode started but then remained stationary); in the real-world test, the obstacle was truly moving (a pedestrian crossing the path). This sudden appearance forced the policy to detect it solely from the 360° LiDAR range data, assess collision risk, and replan its trajectory to avoid impact.

The robot started at position $(0, 0)$ m with a target at $(10, 0)$ m, initially following a straight path. After advancing a few meters, the obstacle appeared directly in its route, as illustrated in Figure 10. The policy had no prior information about the obstacle's presence or location and had to adapt its trajectory in real time to reach the goal safely.

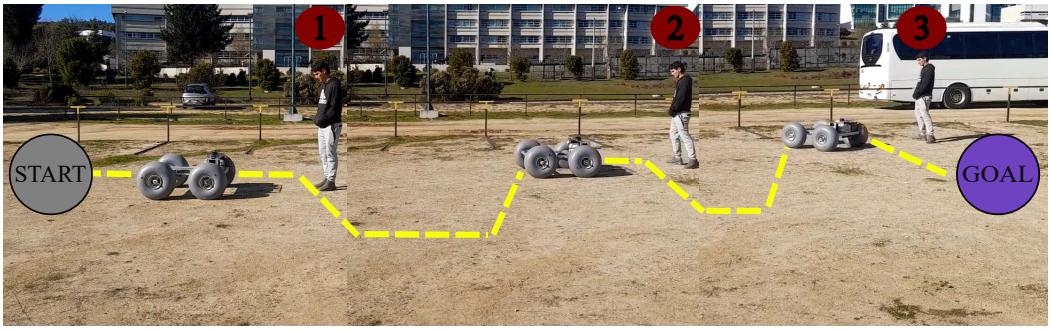


Figure 10. Real-world sequence in which a pedestrian interrupts the robot's path (illustrative, not to scale).

Figure 11 compares the resulting trajectories. In simulation, *PPO* skirted the obstacle via the positive y half-plane, reaching the goal with a slightly oscillatory path; by contrast, *PPO-Mask* executed an earlier avoidance maneuver in the negative half-plane, maintaining a larger clearance. When the policies were transferred to the robotic platform, *PPO* broadly reproduced its simulated trajectory, whereas *PPO-Mask* behaved more stably and smoothly, avoiding the obstacle without abrupt corrections.

Although both policies reached the goal within the 1.0 m threshold, they exhibited different navigation styles: *PPO-Mask* favored wider, safer paths, prioritizing stability and distance from obstacles—an advantage in real beach scenarios where operational safety is paramount.

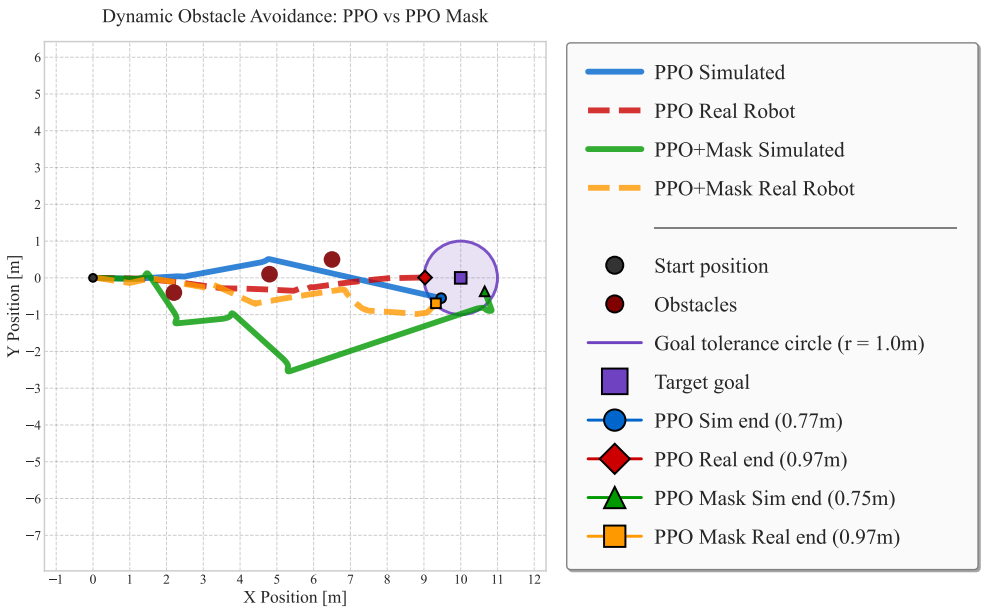
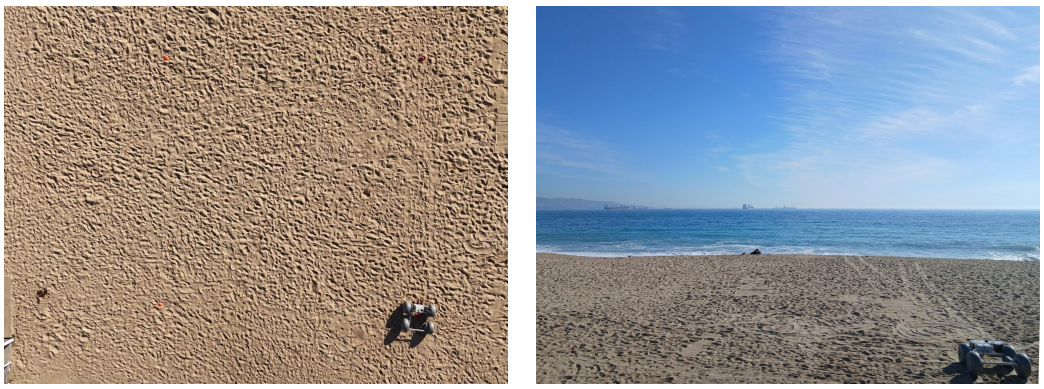


Figure 11. Dynamic obstacle avoidance with *PPO* and *PPO-Mask* in simulation and real-world tests.

Overall, the experiments demonstrated that the integration of Gazebo, Gymnasium, and Stable-Baselines3 enabled the successful transfer of DRL-based navigation policies to the physical robotic platform. *PPO-Mask* achieved the best integral scores, followed more direct trajectories, and maintained its superiority even on firm sand terrain, reducing the Sim-to-Real gap without additional tuning. These results validated the feasibility of achieving reliable autonomous navigation on soft substrates using DRL in combination with a minimal sensor suite.

7.2. Field Experiments on the Beach

From the firm-ground experiments, *PPO-Mask* emerged as the policy with the best performance and was therefore selected for field testing under real beach conditions. This stage is particularly important as it evaluates the system in the environment for which it was originally designed, where additional phenomena such as variable humidity, irregular compaction, and intermittent slippage directly affect traction and odometry accuracy. Figure 12 illustrates the real-world environment where the field experiments were conducted.



(**a**) (**b**)

Figure 12. Beach test scenario used for the evaluation of *PPO-Mask*. (**a**) Aerial view of the *Playa del Deporte* in Viña del Mar. (**b**) Perspective view of the surroundings where the field experiments were conducted.

The experiments were carried out at *Playa del Deporte* in Viña del Mar, Chile, because it is a representative coastal area of the region and for its logistical accessibility, which facilitated field trials under realistic operating conditions.

Waypoint-Based Path Following Experiments

To validate the system, two waypoint-based navigation patterns were tested—a square and a triangle—replicating the experiments previously conducted on firm ground but now under real beach conditions with reduced traction and a higher likelihood of slippage. These trajectories enabled evaluation of the robot's ability to handle sharp 90° (square) and 120° (triangle) turns when irregular terrain affected odometry accuracy. They were also chosen because such routes will later represent the typical cleaning paths to be executed once the cleaning module is integrated.

(a) **Square path**

Unlike the firm-ground experiments, which evaluated a 3 m × 3 m area, in this case, the robot followed a 4 m × 4 m square path with four waypoints and an arrival tolerance of 1 m, corresponding to the target area for future cleaning tasks. As in the firm-ground trials, reaching this area was considered successful mission completion. Figure 13 shows the trajectory executed on sand. The accumulation of odometry drift and slight skidding at the corners produced small deviations from the ideal route, although the robot successfully reached all waypoints within the tolerance zone. As reported in Table 9, under ideal conditions, the total path length would be 16.0 m; however, the actual distance traveled was 17.32 m, representing an 8.25% excess. This increase reflected corrective maneuvers and reduced tracking precision due to the interaction with loose sand.

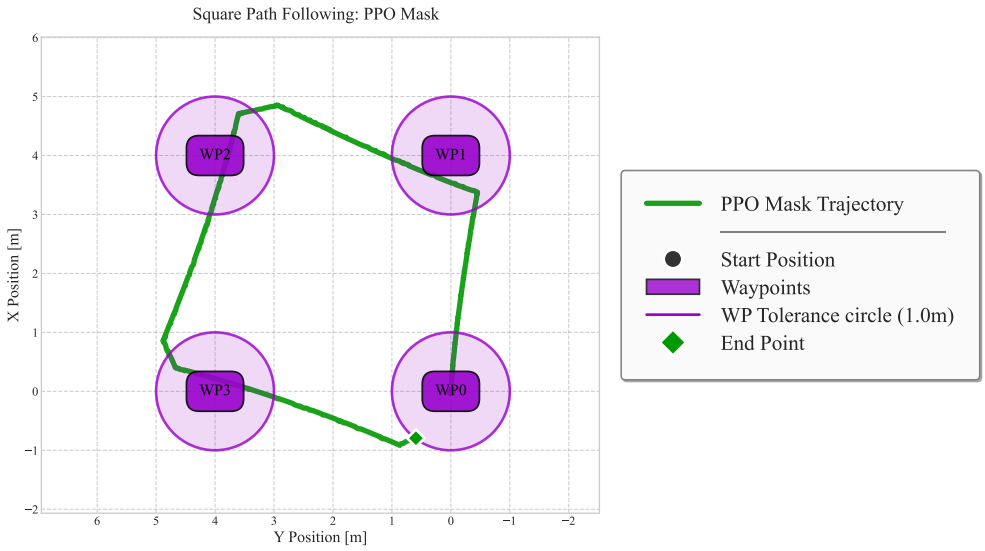


Figure 13. Square Path Following on the Beach.

(b) **Triangular path**

In this case, the triangular pattern used on firm ground was replicated, with turns of approximately 120° and three waypoints, maintaining the same 1 m arrival tolerance corresponding to the target area for future cleaning tasks. As in the firm-ground trials, reaching this area was considered successful mission completion. Figure 14 presents the trajectory executed by the robot, showing that although occasional skidding occurred during the turns, the system completed the route without leaving the operational area. As shown in Table 9, the ideal path length for this configuration was 10.25 m, whereas the robot traveled 12.44 m, representing a 21.4% excess. This increase was explained by

trajectory adjustments required to maintain orientation and compensate for deviations during the inclined segments.

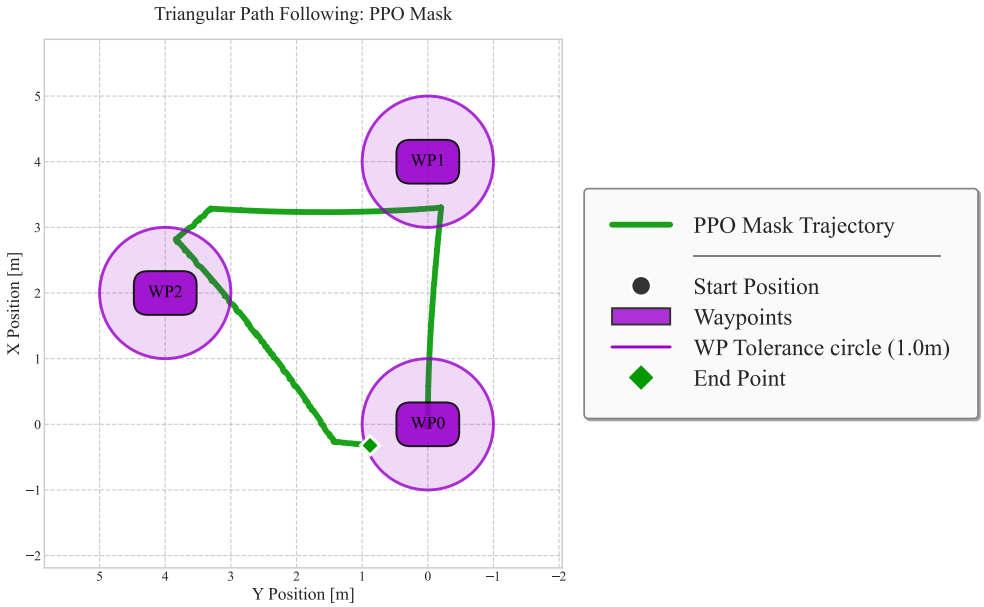


Figure 14. Triangle path following on the beach.

Table 9 summarizes the average results obtained for each trajectory pattern. The reported metrics include (i) total mission time from the start until the last waypoint was reached, (ii) the distance traveled by the robot along the executed trajectory, (iii) the ideal distance according to the geometric reference path, and (iv) the final error, defined as the Euclidean distance between the robot's final position and the corresponding target point in the planned route.

Table 9. Mission performance metrics for the beach experiments.

Pattern	Mission Time [s]	Distance [m]	Ideal Distance [m]	Final Error [m]
Square	115.72	17.32	16.00	0.99
Triangle	81.77	12.44	10.25	0.94

Final error is computed as the Euclidean distance between the robot's final position and the target point of the planned route.

As observed in the beach tests, the system was able to accurately execute waypoint-based routes even under loose sand and low-traction conditions, validating one of the project's key objectives: controlled and repeatable motion in real beach environments. Despite challenges such as slippage and odometry drift, all routes were successfully completed. As in the firm-sand experiments, these results confirm that effective Sim-to-Real transfer can be achieved with a minimal sensor suite and DRL-based navigation policies, providing a solid foundation for the future integration of the cleaning module.

8. Discussion

In this work, *PPO* and *PPO-Mask* were selected as the main training algorithms due to their stability, computational efficiency, and our team's prior experience in Sim-to-Real applications. As an initial reference, classical controllers inspired by Braitenberg and Villela were also explored. Although simple and computationally inexpensive, these controllers exhibited low stability and adaptability on sand and were therefore discarded. Consistently with this decision, the system design prioritized a minimal sensor suite (wheel encoders

and a single 2-D LiDAR), with the objective of demonstrating that robust Sim-to-Real transfer can be achieved without relying on more costly and complex sensor configurations.

Simulation experiments showed that *PPO-Mask* outperformed *PPO* in most error indices, significantly reducing mission times and the number of control steps, while generating more stable and safer trajectories when navigating around obstacles. These results were replicated on the real robot operating on firm ground, confirming that action masking provides additional regularization and robustness.

Based on these results, *PPO-Mask* was selected as the best-performing model for beach validation. Under real conditions, with variations in sand moisture and compaction, the system successfully completed all planned trajectories in a stable manner, confirming the feasibility of defining work cells and systematically covering them for beach-cleaning applications.

From this point, two directions are identified to further extend the system's scope: (i) incorporating more realistic terrain models (granular dynamics or digital elevation models) to capture slope- and compaction-dependent effects and improve the fidelity of Sim-to-Real transfer; (ii) integrating the cleaning module developed by the physics and chemistry team, which will apply a liquid agent onto the sand surface in each cleaning cell of approximately 1 m^2, with each waypoint defining the center of such a cell. This modular structure enables the robot to evolve from validated autonomous navigation toward the effective execution of beach-cleaning tasks.

Overall, these results consolidate the feasibility of autonomous beach navigation with accessible, low-cost hardware and outline a path toward more complete systems that integrate both greater simulation realism and effective field-cleaning capabilities.

9. Conclusions

This work demonstrated that reliable Sim-to-Real transfer is achievable for autonomous beach navigation using only wheel encoders and a 2-D LiDAR. Policies trained entirely in Gazebo were deployed on the physical robot without hyperparameter retuning, achieving stable trajectories on both firm sand and natural beach conditions.

Among the algorithms tested, *PPO-Mask* consistently outperformed *PPO*, producing smoother trajectories, higher stability, and improved robustness under challenging sandy conditions. These improvements were evident not only in simulation but also during real-world experiments on both compact terrain and natural beach sand, validating the ability of action masking to enhance policy regularization and safety.

Beyond these algorithmic results, this study highlights that autonomous navigation in beach environments does not require complex or expensive sensor configurations. Instead, the proposed framework shows that accessible components, combined with carefully designed DRL strategies, can deliver reliable performance under demanding outdoor conditions. This minimalist philosophy reduces costs, power consumption, and integration complexity, while still enabling reliable Sim-to-Real transfer.

Looking ahead, future developments will build on this foundation by integrating the cleaning module and exploring complementary sensor configurations, always preserving the minimalist design philosophy. In this way, the contribution consolidates the viability of scalable robotic solutions to address coastal pollution challenges.

Author Contributions: Conceptualization, G.V., G.H., and G.C.A.; methodology, G.C.A. and G.H.; software, G.C.A. and M.T.C.; validation, G.C.A.; formal analysis, G.C.A.; investigation, G.C.A.; resources, G.V. and G.H.; data curation, G.C.A. and M.T.C.; writing—original draft preparation, G.C.A. and G.H.; writing—review and editing, G.H. and G.C.A.; visualization, G.C.A.; funding acquisition, G.V. and G.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by FONDEF under Grant ID23I10249 and FONDECYT under Grant 1240573.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: The author thanks the team of the Robotics and Vision Laboratory at PUCV for their support during system development, with special recognition to Amira Gallegos for her collaboration during field testing. Additional thanks are extended to Giovanni Cocca-Guardia, Gabriel Olmos, and Manuel Silva for their valuable comments and suggestions in preparing the manuscript. The author is also deeply grateful to his family—Erika Ampuero Nuñez, Eduardo Cid Álvarez, Eduardo Cid Ampuero, and Christopher Cid Ampuero—for their constant support, patience, collaboration during field testing, and motivation throughout the entire process.

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

References

1. Parker, L. Por qué la Contaminación por Plásticos se Convirtió en una Crisis Mundial. National Geographic LA. 2024. Available online: https://www.nationalgeographicla.com/medio-ambiente/2024/04/por-que-la-contaminacion-por-plasticos-se-convirtio-en-una-crisis-mundial (accessed on 3 September 2025).
2. H. Barber & Sons, Inc. Limpiadoras de Playa. Available online: https://www.hbarber.com/es/surf-rake-limpiador-de-la-playa/los-modelos/ (accessed on 3 September 2025).
3. The Searial Cleaners. BeBot—Beach Cleaning Robot. Searial Cleaners. Available online: https://searial-cleaners.com/our-cleaners/bebot-the-beach-cleaner (accessed on 3 September 2025).
4. Teng, H.; Wang, Y.; Chatziparaschis, D.; Karydis, K. Adaptive LiDAR odometry and mapping for autonomous agricultural mobile robots in unmanned farms. *Comput. Electron. Agric.* **2025**, *232*, 110023. [CrossRef]
5. Cai, G.S.; Lin, H.Y.; Kao, S.F. Mobile Robot Localization using GPS, IMU and Visual Odometry. In Proceedings of the 2019 International Automatic Control Conference (CACS), Keelung, Taiwan, 13–16 November 2019; pp. 1–6. [CrossRef]
6. Li, D.; Okhrin, O. A Platform-Agnostic Deep Reinforcement Learning Framework for Effective Sim2Real Transfer towards Autonomous Driving. *Commun. Eng.* **2024**, *3*, 147. [CrossRef]
7. Yin, Y.; Chen, Z.; Liu, G.; Yin, J.; Guo, J. Autonomous navigation of mobile robots in unknown environments using off-policy reinforcement learning with curriculum learning. *Expert Syst. Appl.* **2024**, *247*, 123202. [CrossRef]
8. Quiroga, F.; Hermosilla, G.; Varas, G.; Alonso, F.; Schröder, K. RL-Based Sim2Real Enhancements for Autonomous Beach-Cleaning Agents. *Appl. Sci.* **2024**, *14*, 4602. [CrossRef]
9. Kiran, B.R.; Sobh, I.; Talpaert, V.; Mannion, P.; Sallab, A.A.A.; Yogamani, S.; Pérez, P. Deep Reinforcement Learning for Autonomous Driving: A Survey. *arXiv* **2021**, arXiv:2002.00444. [CrossRef]
10. Weerakoon, K.; Sathyamoorthy, A.J.; Patel, U.; Manocha, D. TERP: Reliable Planning in Uneven Outdoor Environments using Deep Reinforcement Learning. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 9447–9453. [CrossRef]
11. ROS 2 Documentation. Doc ROS. 2025. Available online: https://docs.ros.org/en/humble/index.html (accessed on 3 September 2025).
12. Contaminación Marina. Oceana. 2025. Available online: https://chile.oceana.org/campanas/contaminacion-marina/ (accessed on 3 September 2025).
13. Mazurkiewicz, M.; Martinez, P.S.; Konwent, W.; Deja, K.; Kotwicki, L.; Węsławski, J.M. Plastic contamination of sandy beaches along the southern Baltic—A one season field survey results. *Oceanologia* **2022**, *64*, 769–780. [CrossRef]
14. Espiritu, E.Q.; Rodolfo, R.S.; Evangelista, S.M.J.; Feliciano, J.J.G.; Sumaway, A.M.N.; Pauco, J.L.R.; Alvarez, K.V.N.; Enriquez, E.P. Microplastics contamination in the fishes of selected sites in Pasig River and Marikina River in the Philippines. *Mar. Pollut. Bull.* **2023**, *187*, 114573. [CrossRef]
15. Fatma M.T.; Morshedy, A.; Khaled, M.; Salem, M. EcoBot: An Autonomous Beach-Cleaning Robot for Environmental Sustainability Applications. Available online: https://www.researchgate.net/publication/391451003_EcoBot_An_Autonomous_Beach-Cleaning_Robot_for_Environmental_Sustainability_Applications (accessed on 3 September 2025).

16. Barathraj, M.; Yathunanthanasarma, B.; Mahiliny, J.; Jayasekara, A.G.B.P. Intelligent Beach Cleaning Robot with Dual Modes of Refuse Collection and YOLO-based Detection. In Proceedings of the 2024 4th International Conference on Electrical Engineering (EECon), Colombo, Sri Lanka, 12 December 2024; pp. 89–94. [CrossRef]
17. UmiBeach Robótica Autónoma y Eléctrica. Available online: https://umibeach.es/ (accessed on 3 September 2025).
18. Varghese, D.; Mohan, A. Binman: An Autonomous Beach Cleaning Robot. In Proceedings of the 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 16–17 October 2022; pp. 1–5. [CrossRef]
19. Ichimura, T.; Nakajima, S.I. Development of an autonomous beach cleaning robot "Hirottaro". In Proceedings of the 2016 IEEE International Conference on Mechatronics and Automation, Harbin, China, 7–10 August 2016; pp. 868–872. [CrossRef]
20. Zhu, J.; Yang, Y.; Cheng, Y. SMURF: A Fully Autonomous Water Surface Cleaning Robot with A Novel Coverage Path Planning Method. *J. Mar. Sci. Eng.* **2022**, *10*, 1620. [CrossRef]
21. Zhang, Y.; Huang, Z.; Chen, C.; Wu, X.; Xie, S.; Zhou, H.; Gou, Y.; Gu, L.; Ma, M. A Spiral-Propulsion Amphibious Intelligent Robot for Land Garbage Cleaning and Sea Garbage Cleaning. *J. Mar. Sci. Eng.* **2023**, *11*, 1482. [CrossRef]
22. Amatucci, L.; Turrisi, G.; Bratta, A.; Barasuol, V.; Semini, C. VERO: A vacuum-cleaner-equipped quadruped robot for efficient litter removal. *J. Field Robot.* **2024**, *41*, 1829–1842. [CrossRef]
23. ProjectBB. 2025. Available online: https://project.bb/ (accessed on 3 September 2025).
24. He, N.; Yang, Z.; Bu, C.; Fan, X.; Wu, J.; Sui, Y.; Que, W. Learning Autonomous Navigation in Unmapped and Unknown Environments. *Sensors* **2024**, *24*, 5925. [CrossRef]
25. Salimpour, S.; Peña-Queralta, J.; Paez-Granados, D.; Heikkonen, J.; Westerlund, T. Sim-to-Real Transfer for Mobile Robots with Reinforcement Learning: From NVIDIA Isaac Sim to Gazebo and Real ROS 2 Robots. *arXiv* **2025**, arXiv:2501.02902.
26. Kich, V.A.; Kolling, A.H.; Jesus, J.C.d.; Heisler, G.V.; Jacobs, H.; Bottega, J.A.; Kelbouscas, A.L.d.S.; Ohya, A.; Grando, R.B.; Drews, P.L.J., Jr.; et al. Parallel Distributional Deep Reinforcement Learning for Mapless Navigation of Terrestrial Mobile Robots. *arXiv* **2024**, arXiv:2408.05744. [CrossRef]
27. Gao, J.; Pang, X.; Liu, Q.; Li, Y. Hierarchical Reinforcement Learning for Safe Mapless Navigation with Congestion Estimation. *arXiv* **2025**, arXiv:2503.12036. [CrossRef]
28. Josifovski, J.; Gu, S.; Malmir, M.; Huang, H.; Auddy, S.; Navarro-Guerrero, N.; Spanos, C.; Knoll, A. Safe Continual Domain Adaptation after Sim2Real Transfer of Reinforcement Learning Policies in Robotics. *arXiv* **2025**, arXiv:2503.10949. [CrossRef]
29. Zhou, Z.; Ren, J.; Zeng, Z.; Xiao, J.; Zhang, X.; Guo, X.; Zhou, Z.; Lu, H. A safe reinforcement learning approach for autonomous navigation of mobile robots in dynamic environments. *CAAI Trans. Intell. Technol.* **2023**, 1–16. [CrossRef]
30. Xiao, W.; He, T.; Dolan, J.; Shi, G. Safe Deep Policy Adaptation. *arXiv* **2024**, arXiv:2310.08602. [CrossRef]
31. Ohnishi, M.; Wang, L.; Notomista, G.; Egerstedt, M. Barrier-Certified Adaptive Reinforcement Learning with Applications to Brushbot Navigation. *IEEE Trans. Robot.* **2019**, *35*, 1186–1205. [CrossRef]
32. Wabersich, K.P.; Zeilinger, M.N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *arXiv* **2021**, arXiv:1812.05506. [CrossRef]
33. Sun, H.; Jiang, H.; Zhang, L.; Wu, C.; Qian, S. Multi-robot hierarchical safe reinforcement learning autonomous decision-making strategy based on uniformly ultimate boundedness constraints. *Sci. Rep.* **2025**, *15*, 5990. [CrossRef]
34. Zhang, D.; Wang, Y.; Meng, L.; Yan, J.; Qin, C. Adaptive critic design for safety-optimal FTC of unknown nonlinear systems with asymmetric constrained-input. *ISA Trans.* **2024**, *155*, 309–318. [CrossRef] [PubMed]
35. Qin, C.; Jiang, K.; Wang, Y.; Zhu, T.; Wu, Y.; Zhang, D. Event-triggered H∞ control for unknown constrained nonlinear systems with application to robot arm. *Appl. Math. Model.* **2025**, *144*, 116089. [CrossRef]
36. Qin, C.; Ran, X.; Zhang, D. Unsupervised image stitching based on Generative Adversarial Networks and feature frequency awareness algorithm. *Appl. Soft Comput.* **2025**, *183*, 113466. [CrossRef]
37. micro-ROS Developers. Micro-ROS: ROS 2 on Microcontrollers. Micro-ROS Project. 2025. Available online: https://micro.ros.org/ (accessed on 29 September 2025).
38. RPLIDAR S2. Slamtec. 2025. Available online: https://www.slamtec.com/en/s2 (accessed on 3 September 2025).
39. Reza Nourbakhsh, I.; Siegwart, R. *Introduction to Autonomous Mobile Robots*; MIT Press: Cambridge, MA, USA, 2004.
40. Quiroga, F.; Hermosilla, G.; Farias, G.; Fabregas, E.; Montenegro, G. Position Control of a Mobile Robot through Deep Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 7194. [CrossRef]
41. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
42. Huang, S.; Ontañón, S. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *Int. Flairs Conf. Proc.* **2022**, *35*, 130584. [CrossRef]
43. Raffin, A.; Contributors, S.B. Stable-Baselines3 Docs—Reliable Reinforcement Learning Implementations. Stable Baselines. 2025. Available online: https://stable-baselines3.readthedocs.io/en/master/ (accessed on 3 September 2025).

44. Raffin, A.; Contributors, S.B. Stable Baselines3 Contrib Documentation. SB3-Contrib. 2025. Available online: https://sb3-contrib.readthedocs.io/en/master/ (accessed on 3 September 2025).
45. Shinners, S.M. *Modern Control System Theory and Design*, 2nd ed.; J. Wiley: New York, NY, USA, 1998.