machines

MDPI

Article

LiDAR-Aided Human–Machine Shared Control Optimization for Unknown Complex Environments via Model Predictive Control and Deep Reinforcement Learning

Zhiao Cheng [1], Qianqian Zhang [1,*] and Zerui Li [2]

[1] School of Artificial Intelligence, Anhui University, Hefei 230601, China
[2] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei 230026, China
* Correspondence: zqq789@mail.ustc.edu.cn

Abstract

Intelligent navigation of mobile robots in unknown environments has become a key enabling technology for service and logistics applications. However, in unstructured scenarios, perception noise and environmental uncertainty can significantly degrade system performance, making it a major challenge to balance safety and efficiency. In this work, we propose a Human–Machine Shared Control method with Model Predictive Control constraints (HMSC). HMSC establishes a confidence-driven human–machine shared control mechanism that maximizes collaborative efficiency by dynamically assessing the reliability of agent decisions to regulate control weights. Simultaneously, the method introduces a composite confidence evaluation model which, by fusing epistemic uncertainty with geometric feasibility from lightweight LiDAR measurements, achieves a robust quantification of policy risk. To ensure safe execution, we develop a multi-trajectory prediction mechanism which, after validating kinematic constraints, minimally intervenes to safely adjust control commands. We conducted Gazebo-based simulation experiments on obstacle avoidance and target navigation using a LiDAR-equipped mobile robot model and validated the rationality of the confidence model. The results demonstrate that the proposed shared control strategy, which combines confidence assessment with deterministic safety boundaries, significantly improves the success rate and robustness of the system in uncertain environments.

Keywords: human–machine shared control; model predictive control; deep reinforcement learning; intelligent navigation

Check for updates

Academic Editor: Luca Bruzzone

Received: 4 August 2026
Revised: 13 September 2026
Accepted: 17 September 2026
Published: 19 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.

1. Introduction

The widespread application of mobile robots in real-world scenarios such as services, inspections, and logistics has imposed higher demands on their operational reliability and execution capabilities [1–3]. However, when confronting complex, dynamic environments that cannot be fully modeled (such as narrow passages, multi-robot interactions, or dense moving crowds), traditional control methods often face severe challenges in balancing execution efficiency with system safety.

Traditional planning and control methods for mobile robots have long held a dominant position in industrial and engineering applications. Foundational search-based techniques, such as A* and Dijkstra, alongside sampling-based frameworks like RRT and PRM, have set the standard for finding globally feasible paths through state-space discretization or

topological mapping [4–6]. While reactive methods—namely Artificial Potential Fields (APF) and the Dynamic Window Approach (DWA)—facilitate immediate obstacle avoidance via real-time feedback [7,8], Model Predictive Control (MPC) pushes these capabilities further by integrating system dynamics into a predictive optimization framework [9]. Nevertheless, these traditional frameworks often falter when faced with the unpredictability of dynamic environments. Search and sampling algorithms frequently succumb to the "curse of dimensionality," where computational demands spike in high-dimensional spaces. Meanwhile, reactive controllers remain vulnerable to local minima and lack long-term foresight. Perhaps most critically, the efficacy of MPC hinges on the precision of environment models; it is notoriously sensitive to perceptual noise and the stochastic nature of human behavior, which frequently leads to overly cautious or mathematically infeasible trajectories. Compared with image-based perception or dense semantic reconstruction, LiDAR measurements provide lightweight geometric information that can be directly transformed into obstacle distances, safety margins, and constraints compatible with model-based control frameworks [10–12]. Therefore, LiDAR offers a practical perception interface for real-time constrained control in resource-limited mobile robot platforms.

In recent years, the rapid development of deep learning and reinforcement learning has provided a new paradigm for solving navigation problems in complex and unknown environments. Deep Reinforcement Learning (DRL) merges the representational power of neural networks with the strategic decision-making of RL, allowing agents to distill raw sensory input into optimal policies without the need for exhaustive environment modeling [13–15]. Early breakthroughs like the Deep Q-Network (DQN) demonstrated this potential by mastering high-dimensional Atari environments [16]. This was later extended to the continuous control domains essential for UAVs and robotics through the Deep Deterministic Policy Gradient (DDPG) [17]. More recent refinements, including Proximal Policy Optimization (PPO) [18] and Soft Actor-Critic (SAC) [19], have further bolstered training stability and sample efficiency across complex tasks. However, applying these algorithms to real-world scenarios still presents numerous challenges [20–22]. Typically, these algorithms can only learn effective control policies after millions of steps of poor performance in a simulated environment. This might be acceptable if perfectly accurate simulators were available. However, for many real-world problems, such simulators do not exist. In such cases, the agent must learn in the real environment where its actions have actual consequences, requiring it to exhibit excellent online performance right from the beginning of the learning process.

Human operators still possess irreplaceable advantages in experience-based judgment, anomaly detection, and critical decision-making, but their control precision is relatively low [23,24]. Consequently, the integration of human experience, learning-based decision-making capabilities, and model-based safety guarantees has become a crucial direction in current research. When realizing Human–Machine Shared Control (HMSC), two core issues are how to reliably determine whether an autonomous policy is trustworthy, and under what conditions to allow human intervention or share control with the autonomous policy [25]. While fuzzy logic and Control Barrier Functions (CBFs) offer mechanisms for control shifting [26,27], they remain insufficient. Most current frameworks treat reliability as a singular, external metric (e.g., tracking error), overlooking the critical role of epistemic uncertainty. This omission makes the system vulnerable to Out-of-Distribution (OOD) risks [28], undermining the practical validity of theoretical safety models. Traditional "hard-switching" intervention modes often overlook the continuity of control commands and dynamic constraints, which can easily induce system oscillation and disrupt human operational intent [29].

This study proposes an adaptive HMSC framework to address the safety and reliability challenges in obstacle avoidance and navigation tasks within unknown environments. The core idea of the proposed architecture is to integrate the long-horizon planning capability of deep reinforcement learning with human experiential judgment, while introducing a prediction-based online safety layer to enforce physical constraints. Unlike traditional rigid switching strategies, the proposed system dynamically and smoothly allocates control authority between the human operator and the autonomous agent according to the estimated reliability of the autonomous policy. Specifically, epistemic uncertainty is estimated through MC-Dropout and mapped to a Bayesian confidence, which is further calibrated using geometric feasibility information obtained from the MPC safety layer to form a composite confidence measure. The resulting confidence regulates the human–machine authority allocation: lower confidence increases human involvement, whereas higher confidence assigns greater authority to the autonomous policy. When the confidence falls below a predefined threshold, human-priority arbitration is activated subject to the safety constraints. The detailed mathematical formulation is presented in Section 3. Meanwhile, the proposed framework exploits lightweight geometric features from LiDAR measurements, enabling the MPC layer to perform online collision prediction and minimally invasive safety correction. This work provides a simulation-based proof-of-concept for integrating learning-based decision making, confidence-aware human–machine cooperation, and model-based safety verification, and establishes a basis for future validation in safety-critical real-world applications. The main contributions of this paper are as follows:

- A HMSC framework with adaptive control authority allocation is proposed. The system dynamically adjusts fusion weights based on the agent's confidence level regarding the current environmental state. The proposed HMSC method was implemented and validated, with its performance and the rationality of the confidence model verified through both local and global control tasks.
- This paper designed and implemented an online safety correction mechanism based on parallel multi-trajectory prediction and constraint projection. MPC is employed to generate multiple candidate trajectories for online safety evaluation of the policy. When potential collision risks are detected, a constrained optimization problem is solved to project the control action onto the nearest feasible region on the safe manifold, thereby preserving the human or DRL intent to the greatest extent possible.
- A quantitative evaluation model for jointly optimized machine decision confidence is proposed. By performing randomized forward passes of the policy network during inference, an action sample distribution is obtained, from which statistical measures are used to construct a decision confidence metric. The confidence is further fused with geometric feasibility information derived from LiDAR measurements to obtain a comprehensive composite confidence measure.

2. Problem Description and Related Work

2.1. Problem Description

In unknown and complex environments, autonomous navigation of mobile robots requires not only planning collision-free paths, but also balancing execution efficiency with strict physical safety constraints. To achieve more robust decision-making, introducing a HMSC mechanism has become an effective approach. In this study, the problem of HMSC and safety control can be formally described as a continuous Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \gamma)$. The goal is a process of maximizing cumulative rewards under the conditions of satisfying dynamic safety constraints and

human intent. Here, the state $s(t) \in \mathcal{S}$ incorporates environmental perception information and goal-related information, and is defined as

$$s(t) = [s_{lidar}, d_{goal}, \theta_{goal}], \tag{1}$$

where s_{lidar} denotes the compressed LiDAR observation vector, and d_{goal} and θ_{goal} represent the relative distance and relative bearing angle between the robot and the target location, respectively. The action $u(t) \in \mathcal{U}$ denotes the velocity commands, and the transition dynamics $\mathcal{P}$ are governed by the robot's kinematics. Specifically, the objective model is defined as follow:

$$\begin{aligned} \max_{u(t)\in\mathcal{U}} J((s(t),u(t)) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s(t),u(t))\right] \\ \text{s.t.} \quad u(t) &= \mathcal{F}(s(t),u^m(t),u^h(t),C) \\ u^m(t) &= \pi(s(t)) \\ u^h(t) &= \text{Human-Input} \\ s(t+1) &= \mathbb{F}(s(t),u(t)) \\ D_{\min}(u(t)) &> d_{\text{safe}} \end{aligned} \tag{2}$$

In this model, $J((s(t),u(t))$ represents the cumulative optimization objective over the task horizon. The final executed control command $u(t)$ is determined by the human–machine share function $\mathcal{F}$. This function depends on the current integrated confidence C, and performs dynamic arbitration and weighting between the machine-generated action $u^m(t)$, produced by the deep reinforcement learning policy π, and the human input $u^h(t)$. The system state evolves according to the dynamics model $\mathbb{F}(s(t),u(t))$, and all actions must strictly satisfy safety constraints; that is, the minimum distance to obstacles along the predicted trajectory, D_{min}, must be greater than the safety threshold d_{safe}.

2.2. Related Work

2.2.1. Human–Machine Shared Control

The paradigm of HMSC has drawn considerable interest across disciplines such as robotics and intelligent transportation. A number of studies utilize model-based or game-theoretic frameworks to dynamically allocate control authority, achieved by assessing the risks of human and autonomous decisions in real time. In human–vehicle cooperative driving, for instance, it is imperative for the system to correct hazardous driving behaviors promptly. Prevalent strategies for this include mathematical function construction, fuzzy control, and game theory [30]. From a reinforcement learning perspective, representative human-in-the-loop approaches such as Deep TAMER, ELI, and Hug-DRL [31–33] incorporate human feedback, intervention, or guidance into the learning and decision-making process. Confidence-based and adaptive shared-control strategies have also been investigated [34,35]. These studies demonstrate that human knowledge can be incorporated through interactive feedback, intervention, confidence assessment, or adaptive authority allocation. Compared with these representative approaches, the proposed HMSC framework focuses on coupling confidence-aware authority allocation with an explicit model-based safety layer. Specifically, epistemic uncertainty of the autonomous policy is estimated through MC-Dropout and further calibrated using the geometric feasibility information provided by MPC. The resulting composite confidence regulates the human–machine authority allocation, while the MPC layer independently verifies the geometric safety of candidate actions before execution. Therefore, the proposed framework jointly considers policy reliability, human intervention, and environmental feasibility within a unified control architecture.

2.2.2. Model Predictive Control and Safety Filtering

MPC achieves online optimization and constraint handling of future trajectories by solving a finite-horizon optimal control problem at each control step. It is a widely used method in industrial applications and mobile robot trajectory planning. The standard discrete-time MPC optimization problem can be formulated as:

$$\min_{\{u_k\}_{k=0}^{N-1}} \sum_{k=0}^{N-1} \ell(s_k, u_k) + V_{\mathbb{F}}(s_N) \tag{3a}$$

$$\text{s.t.} \quad s_{k+1} = \mathbb{F}(s_k, u_k), \tag{3b}$$

$$s_k \in \mathcal{S},\ u_k \in \mathcal{U},\ \forall k = 0, \ldots, N-1, \tag{3c}$$

$$s_0 = s_{\text{current}}, \tag{3d}$$

where N denotes the prediction horizon, ℓ is the stage cost, and $\mathcal{S}$,$\mathcal{U}$ are the state and control constraint sets. The strength of MPC lies in its explicit consideration of future constraints, but its computational feasibility becomes challenging in high-dimensional or nonconvex settings, such as complex dynamic crowd environments. Recent studies have shown that integrating Control Barrier Functions (CBFs) with MPC or real-time safety filters is an effective way to ensure safe system operation [27,36]. A common continuous-time CBF constraint for a scalar safety function $h(s)$ (with safe set $\mathcal{C} = s : h(s) \geq 0$) is formulated as:

$$\dot{h}(s, u) = L_f h(s) + L_g h(s) u \geq -\alpha(h(s)), \tag{4}$$

where L_f and L_g represent the corresponding Lie derivatives, and α is an appropriately chosen class-$\mathcal{K}$ function that guarantees forward invariance of the safe set. Within discrete-time formulations or MPC-based schemes, the CBF condition may be reformulated as linear or nonlinear inequality constraints, or alternatively realized as a control-filtering mechanism via a quadratic program (QP) that projects a nominal input onto the admissible set with minimal deviation:

$$\begin{aligned} &\min_{u} \|u - u_{\text{des}}\|^2 \\ &\text{s.t. } A(s)u \leq B(s) \quad (\text{CBF/Collision constraints}). \end{aligned} \tag{5}$$

This safety filter ensures that the executed action stays as close as possible to the desired action while satisfying safety constraints, and has become a mainstream approach in recent years for enforcing safety in learning-based controllers [37–39]. Nevertheless, in unfamiliar, unstructured, or rapidly evolving crowd environments, the system model $\mathbb{F}(s_k, u_k)$ is frequently arduous to ascertain with precision. The behavior of obstacles is stochastic, and the constraint boundaries are time-varying, making it challenging for standard MPC to ensure feasibility and optimality in real time. Moreover, since MPC relies solely on deterministic model predictions, it cannot explicitly represent perception noise, model errors, or policy uncertainty. As a result, it tends to produce overly conservative control or suffer from local failures in complex scenarios.

2.2.3. Bayesian Neural Networks

BNN treat network weights as random variables, aiming to incorporate both data uncertainty and model uncertainty into the posterior inference of the predictive distribution [40]. The predictive distribution of a BNN is given by:

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, w), p(w|\mathcal{D}) dw, \tag{6}$$

but directly solving this posterior integral is computationally expensive. Gal and Ghahramani proposed interpreting Dropout as a form of variational Bayesian approximation: by keeping Dropout active during inference and performing T stochastic forward passes for the same input x^*, the posterior predictive distribution can be approximated—this approach is known as MC-Dropout [41]. For a given state x, a set of sampled actions $\left\{a^{(i)}\right\}_{i=1}^{T}$ is obtained, from which the sample mean and sample variance are used as the predictive mean and uncertainty estimate, respectively:

$$\hat{\mu} = \frac{1}{T}\sum_{i=1}^{T} a^{(i)}, \quad \widehat{\mathrm{Var}} = \frac{1}{T}\sum_{i=1}^{T}(a^{(i)} - \hat{\mu})^2. \tag{7}$$

The sample variance mainly reflects epistemic uncertainty, while if the network additionally predicts a variance head, aleatoric uncertainty can also be estimated [42]. The advantages of MC-Dropout lie in its simplicity and seamless integration with existing neural architectures, while its limitations stem from the approximate nature of the method and the sampling cost.

2.2.4. Twin Delayed Deep Deterministic Policy Gradient

The Twin Delayed Deep Deterministic Policy Gradient(TD3) algorithm, an improved variant of the Deep Deterministic Policy Gradient method, enhances performance in continuous control tasks by introducing techniques such as double-Q learning and delayed updates [43].

During training, the data generated through interactions between the agent and the environment are stored in an experience replay buffer $\mathcal{D}$. The action output by the network, denoted as a_t, corresponds to the machine-side control decision $u^m(t)$ in the proposed framework. At each training step, a mini-batch of samples is drawn from the replay buffer to update the network parameters. The twin Critic networks, parameterized by θ_1 and θ_2, are updated by minimizing the mean squared Bellman error:

$$L(\theta_i) = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\left(y_t - Q_{\theta_i}(s_t,a_t)\right)^2\right], \quad i = 1,2 \tag{8}$$

To mitigate overestimation bias, the target value y_t is computed using the minimum of the outputs from two target Critic networks, parameterized by θ_1' and θ_2', respectively. In addition, a smoothing noise term $\tilde{\epsilon}$, sampled from a clipped Gaussian distribution $\tilde{\epsilon} \sim \mathrm{clip}(\mathcal{N}(0,\tilde{\sigma}), -c, c)$, is added to the action generated by the target Actor policy:

$$y_t = r_t + \gamma \min_{j=1,2} Q_{\theta_j'}\left(s_{t+1}, \pi_{\phi'}(s_{t+1}) + \tilde{\epsilon}\right), \tag{9}$$

where γ denotes the long-term reward discount factor defined in Equation (9). The smoothing noise $\tilde{\epsilon}$ is clipped to ensure that the resulting action remains within the valid physical kinematic constraints.To improve training stability, the Actor network is updated less frequently than the Critic networks through a delayed policy update mechanism. Furthermore, the target networks are updated using a soft update strategy, which enables smooth parameter transitions and enhances training robustness.

Many robotic navigation tasks adopt TD3 to train policy networks for efficient path control. Building on this foundation, the present work introduces a human–machine integration framework along with an MPC-based safety layer, enabling the RL policy to not only learn navigation efficiently but also maintain real-time obstacle avoidance capability.

3. Method

This study proposes a hybrid control framework that integrates deep reinforcement learning, a MPC safety layer, and a HMSC strategy to achieve safe and efficient robot navigation in unknown environments.The overall framework of this study is illustrated in Figure 1.

The robot acquires observations S through onboard sensors. A DRL policy generates control actions and can also accept operator commands u^h. These commands are first passed through an MPC-based safety verification stage. If the commands fail the safety check, the MPC corrects them via online constrained optimization and re-verifies the result; if they pass, the commands are forwarded to the HMSC module, where they are dynamically fused according to the policy confidence and the human input to generate the fused command u. The fused command is then subjected to a second MPC safety verification. If safety constraints are still violated, the correction process is repeated until all constraints are satisfied. Finally, the verified safe command is executed on the robot, and the execution outcome is fed back to the perception module, forming a closed-loop system.

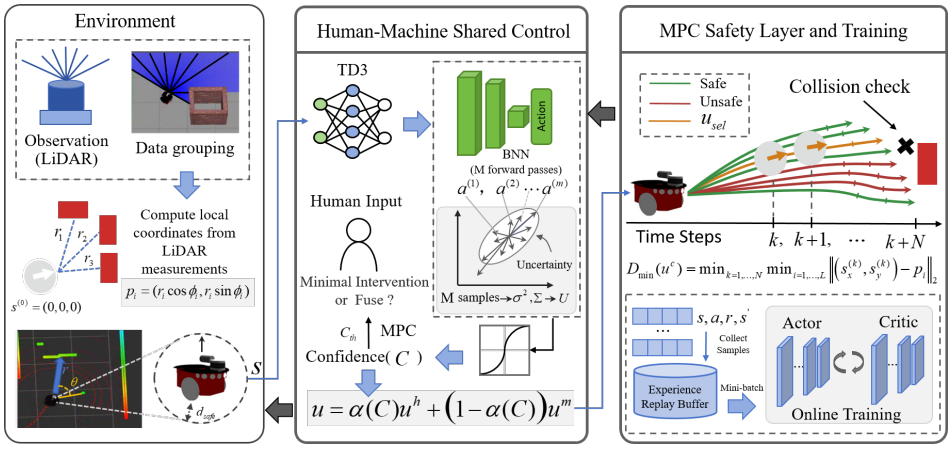


Figure 1. Overview of the proposed HMSC framework. The architecture integrates a deep reinforcement learning policy with human teleoperation through a HMSC module. This module employs MC-Dropout for uncertainty estimation and accordingly determines the fusion weight $\alpha(C)$. The fused control command u is then fed into a MPC safety layer, which performs collision checking and solves a constrained optimization problem to ensure system safety before the final command is executed by the robot.

To achieve effective coordination between human operation and autonomous policy, this study designs and implements a comprehensive HMSC framework within the control loop. The objective of this module is to dynamically allocate control authority and fuse human commands with DRL policy outputs while ensuring safety. Within this control framework, BNN are introduced to characterize the confidence levels of the autonomous policy in real-time. This provides a quantitative basis for dynamic control authority allocation and enables joint verification with a model-predictive safety layer.

In the HMSC strategy, the weight of the human command is represented as $\alpha(C) \in [0, 1]$, which is a monotonically decreasing function of the confidence level C:

$$\alpha(C) = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})(1 - C)^{\gamma}, \tag{10}$$

here, $\alpha_{\min}$ and $\alpha_{\max}$ define the operational bounds of the human weight, while $\gamma > 0$ is a curvature adjustment factor. As confidence $C \to 1$, the human contribution decays smoothly toward $\alpha_{\min}$, effectively prioritizing the autonomous policy. A drop in C triggers a rapid escalation in human authority, restoring operator control during high-uncertainty

or volatile conditions. Unlike rigid binary handoff mechanisms, this adaptive formulation establishes a progressive weighting mechanism, which is essential for mitigating control chattering and ensuring a well-bounded authority transition. The fused control command is then expressed as:

$$u = \alpha(C)u^h + \big(1 - \alpha(C)\big)u^m, \tag{11}$$

where u^h denotes the human input, and u^m denotes the action output of the TD3 policy.

In the practical implementation, a Bayesian neural network constructed via Monte Carlo Dropout is employed to estimate the uncertainty of the DRL's action outputs. Specifically, for a given state s, the network performs M forward passes during inference with randomly sampled Dropout masks, yielding a set of action samples $\{\hat{a}^m\}_{m=1}^{M}$:

$$\hat{a}^{(1)} = f_{w_1}(s),\ \hat{a}^{(2)} = f_{w_2}(s),\ \ldots,\ \hat{a}^{(M)} = f_{w_M}(s). \tag{12}$$

Based on these samples, the sample mean and variance of each action dimension can be computed as

$$\bar{a} = \frac{1}{M}\sum_{m}\hat{a}^{(m)}, \tag{13}$$

$$\mathrm{var}_i = \frac{1}{M}\sum_{m=1}^{M}\left(\hat{a}_i^{(m)} - \bar{a}_i\right)^2, \quad i = 1, \ldots, d, \tag{14}$$

where d denotes the dimensionality of the action space. The variances across all action dimensions are then averaged to obtain a scalar uncertainty metric that characterizes the overall uncertainty of the policy at the current state:

$$U = \frac{1}{d}\sum_{i=1}^{d}\mathrm{var}_i\,. \tag{15}$$

Based on U, a monotonically decreasing mapping function $g(\cdot)$ is designed to project the uncertainty value onto a Bayesian confidence component $C_{\mathrm{B}} = g(U) \in (0, 1)$. In this work, a sigmoid function is adopted:

$$C_{\mathrm{B}} = g(U) = \frac{1}{1 + \exp(\zeta U - b)} \in (0, 1), \tag{16}$$

where the hyperparameters ζ and b control the slope and offset of the mapping, respectively. This Bayesian confidence reflects the model's degree of confidence in its action outputs at the current state: when the action distribution is concentrated and the sample variance is small, C_{B} approaches 1. Conversely, when the action distribution is more dispersed, C_{B} decreases accordingly. The parameters ζ and b determine the sensitivity and transition location of the uncertainty-to-confidence mapping. Specifically, the midpoint of the sigmoid, where $C_B = 0.5$, occurs at $U = b/\zeta$. With $\zeta = 10$ and $b = 3$, this transition point is located at $U = 0.3$. Moreover, the maximum magnitude of the confidence sensitivity with respect to uncertainty is $|\partial C_B/\partial U|_{\max} = \zeta/4$. Therefore, increasing ζ produces a steeper confidence decrease around the transition region and consequently causes a more rapid increase in human authority, whereas decreasing ζ results in a smoother authority transition. For a fixed ζ, increasing b shifts the transition toward a higher uncertainty level and delays the increase in human authority, while decreasing b leads to earlier intervention. The adopted values $\zeta = 10$ and $b = 3$ were selected to provide a sufficiently responsive but continuous transition, avoiding both excessive sensitivity to small uncertainty fluctuations and excessively delayed human intervention.

Confidence assessment does not rely solely on the Bayesian component; to obtain a more robust decision metric, the system fuses the Bayesian confidence with the number of feasible safe actions n_{safe} returned by the MPC safety layer to form a composite confidence C:

$$C = \begin{cases} 0, & n_{\text{safe}} = 0, \\ \max(\eta, \frac{n_{\text{safe}}}{\beta}) C_B, & 0 < n_{\text{safe}} < \beta, \\ C_B, & n_{\text{safe}} \geq \beta, \end{cases} \tag{17}$$

where η denotes the confidence lower-bound factor, and β represents the threshold on the number of feasible safe actions.

In this work, MPC is not employed as the primary controller replacing DRL or human operation; Instead, it functions as an online safety layer. Its primary responsibility is to perform collision detection and generate safe actions at each control step based on the current LiDAR data.

During the collision detection phase, model prediction is carried out in the robot's local coordinate frame. The local obstacle point set is obtained from the LiDAR distance readings r_i and the corresponding scanning angles ϕ_i:

$$p_i = (r_i \cos \phi_i, r_i \sin \phi_i) \quad i = 1, \ldots, L. \tag{18}$$

The sum of the robot radius R_{robot} and the safety buffer δ is used as the safety threshold:

$$d_{\text{safe}} = R_{\text{robot}} + \delta. \tag{19}$$

As shown in Figure 2, the specific prediction model uses difference-driven discretization. Let the robot's initial pose in the local coordinate frame be $s^{(0)} = (0, 0, 0)$. For a constant control input $u^c = [v, \omega]^\top$, forward integration is performed for $N = \lfloor T/\Delta t \rfloor$ steps within the prediction horizon T using a time step Δt. The trajectory update formula is as follows:

$$\begin{aligned} s_x^{(k+1)} &= s_x^{(k)} + v \cos(\varphi^{(k)}) \Delta t, \\ s_y^{(k+1)} &= s_y^{(k)} + v \sin(\varphi^{(k)}) \Delta t, \quad k = 0, \ldots, N-1. \\ \varphi^{(k+1)} &= \varphi^{(k)} + \omega \Delta t. \end{aligned} \tag{20}$$

We define the minimum obstacle distance corresponding to a control input u^c based on the predicted discrete trajectory as:

$$D_{\min}(u^c) = \min_{k=1,\ldots,N} \min_{i=1,\ldots,L} \left\| \left(s_x^{(k)}, s_y^{(k)} \right) - p_i \right\|_2. \tag{21}$$

If $D_{\min}(u^c) > d_{\text{safe}}$, then u^c is regarded as collision-free within the prediction horizon; Otherwise, u^c is considered unsafe.

To generate safe actions, a candidate set is first obtained by uniformly sampling the action space:

$$\mathcal{U}_{\text{cand}} = \{(v_j, \omega_\ell) \mid v_j \in \mathcal{V}, \omega_\ell \in \Omega\}. \tag{22}$$

Here, $\mathcal{V}$ and Ω are the sampling spaces. In each control cycle, the MPC evaluates the feasibility of each candidate control pair (v_j, ω_ℓ). At each step, a collision detection function is invoked to assess the geometric overlap between the robot's occupied footprint and any obstacles. Once a potential collision is detected, the control pair is deemed infeasible and is not evaluated further. If no collision occurs throughout the entire prediction horizon, the control pair is added to the safe action set:

$$A_{\text{safe}} = \{u^c \in \mathcal{U}_{\text{cand}} \mid D_{\min}(u^c) > d_{\text{safe}}\}. \tag{23}$$

In practical operation, when the system obtains a target action u (derived from the DRL policy, manual input, or their dynamic fusion), the action is first checked for safety: if $D_{\min}(u) > d_{\text{safe}}$, it is executed directly; else, the system proactively solves the following minimum-perturbation optimization problem:

$$\min_{u_{\text{sel}} \in \mathcal{U}_{\text{cand}}} \quad J(u_{\text{sel}}) = \|u_{\text{sel}} - u\|_2^2 \tag{24a}$$

$$\text{s.t.} \quad D_{\min}(u_{\text{sel}}) > d_{\text{safe}} \tag{24b}$$

$$0 \le v \le v_{\max} \tag{24c}$$

$$-\omega_{\max} \le \omega \le \omega_{\max} \tag{24d}$$

By discretizing the continuous control space, the non-convex collision avoidance constraints are effectively transformed into a tractable feasible-set search problem.

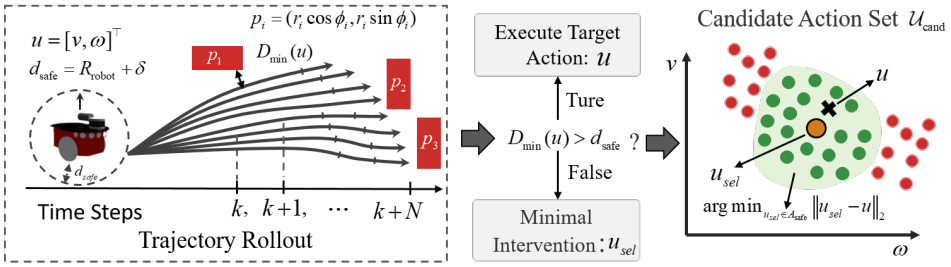


Figure 2. Working principle of an MPC-based safety layer.

The autonomous decision-making branch is built upon a TD3 architecture integrated with a BNN mechanism. The Actor network takes the fused state vector (comprising the compressed 20-dimensional LiDAR data and odometry information) as input. This state is sequentially processed by three Fully Connected (FC) layers with hidden dimensions of 800, 600, and 2, respectively. The first two hidden layers utilize ReLU activation functions, while the final output layer employs bounded activation to restrict the continuous action components—linear velocity a_1 and angular velocity a_2—within the robot's physical kinematic limits. To construct the BNN for epistemic uncertainty estimation, MC-Dropout layers are systematically embedded after each hidden layer within the Actor network. These layers remain active during the inference phase to perform M stochastic forward passes, generating the action distribution required for confidence evaluation. For evaluating the value of a state-action pair (s, a), two critic networks with identical architectures but staggered parameter updates are employed to mitigate Q-value overestimation. The state s is initially fed into a FC and activated by ReLU to extract the feature representation L_s. This feature and the action a are then independently processed through two transformation layers of equal size, denoted as τ_1 and τ_2, and combined as follows:

$$L_c = L_s W_{\tau_1} + a W_{\tau_2} + b_{\tau_2}. \tag{25}$$

Let W_{τ_1} and W_{τ_2} be the weight matrices of the first and second transformation layers, respectively, and let b_{τ_2} denote the bias vector for the τ_2 layer. These transformed features are summed and passed through a ReLU activation, followed by a final FC that outputs the Q-value. To prevent overly optimistic estimates, the smaller of the two Q-values produced by the twin critic networks is chosen as the ultimate value estimate. The complete network architecture is shown in Figure 3.

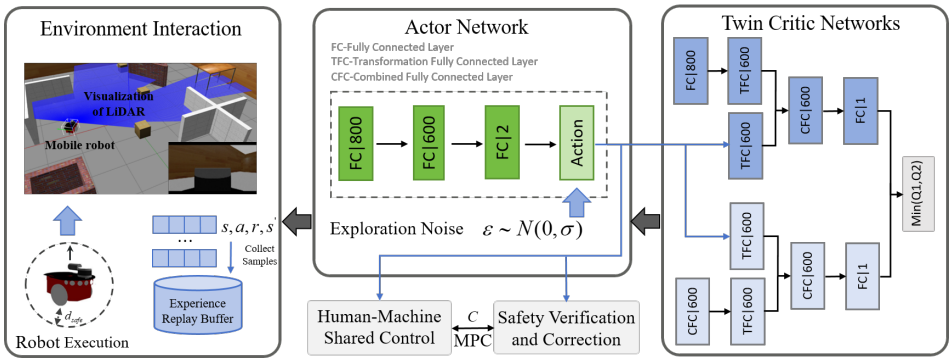


Figure 3. TD3 system architecture diagram integrating human–machine shared control and MPC-based safety layer.

Algorithm 1 presents the proposed HMSC framework. The algorithm first checks whether both the human action and the DRL action satisfy the minimum obstacle distance requirement $D_{\min}(\cdot) > d_{\text{safe}}$. If both actions are safe, the human weight $\alpha(C)$ is computed based on the confidence, and the two actions are linearly fused to generate u. If the fused action remains within the safe region, it is executed directly; otherwise, an alternative action u_{sel} is selected from the safe action set A_{safe} that is closest to u. When either action is unsafe, the system decides whether to prioritize the human or the DRL action based on the confidence value: if the confidence is below the threshold C_{th}, the human decision is preferred; when the confidence is high, the system favors the DRL policy output. When both actions violate the safety constraints, the system still follows the principle of minimal intervention by selecting from A_{safe} the safe alternative that is closest to the human action and executing it.

The advantage of this integrated strategy lies in the complementary combination of probabilistic uncertainty estimation (Bayesian inference) and deterministic geometric safety assessment (MPC verification). In completely unknown or unstructured environments, relying solely on the Bayesian cognitive confidence C_B, which is extracted through Monte Carlo dropout in a neural network, can indeed lead the system to dangerous overconfidence due to perception noise or encounters with OOD scenarios. To fundamentally ensure the correctness and robustness of confidence assessment, this study does not depend on a standalone probabilistic model. Instead, we propose a novel composite confidence evaluation mechanism (Equation (17)). This mechanism introduces the number of geometrically feasible and safe actions, n_{safe}, obtained from the MPC safety layer through multi-trajectory forward prediction in the current local coordinate frame. This quantity serves as a deterministic physical verification metric that continuously calibrates the Bayesian confidence online. When the robot operates in an open and well-defined environment, a large number of physically feasible solutions exist; therefore, the system places greater trust in the Bayesian cognitive component to maintain efficient navigation. However, when the robot enters highly uncertain, constrained, or high-risk environments, even if the neural network produces an artificially high confidence value because it has never encountered such scenarios before, the MPC layer can objectively predict a sharp reduction in the number of n_{safe}.

Under these circumstances, our mechanism disregards the network's "blind confidence" and forces the final composite confidence C to decrease accordingly. This immediately triggers either human intervention or a conservative control strategy. By combining probabilistic reasoning with deterministic physical validation, the proposed framework enhances the reliability of the confidence assessment and helps maintain a safer control behavior in unknown environments through MPC-based geometric feasibility verification.

Algorithm 1: Human–Machine Shared Control

Input: Human control command u^h, DRL control command u^m, safe action set A_{safe}, integrated confidence C, confidence threshold C_{th}, weighting function $\alpha(C)$

Output: Final control command u

while human input u^h is available **do**
  Compute the minimum predicted obstacle distances corresponding to the human and machine control commands: $D_{\min}(u^h)$ and $D_{\min}(u^m)$
  if $D_{\min}(u^h) > d_{\text{safe}}$ and $D_{\min}(u^m) > d_{\text{safe}}$ **then**
    Compute the human control weight $\alpha(C)$ according to (10)
    Fuse the human and machine control commands: $u = \alpha(C)u^h + (1 - \alpha(C))u^m$
  else
    if $C \geq C_{\text{th}}$ and $D_{\min}(u^m) > d_{\text{safe}}$ **then**
      Select the autonomous control command: $u = u^m$
    else
      Select the human control command: $u = u^h$
    end
  end
  if $D_{\min}(u) > d_{\text{safe}}$ **then**
    Execute the control command directly: $u_{\text{sel}} = u$
  else
    Obtain the safe control command by solving the optimization problem in (24): u_{sel}
  end
  Execute the final safe control command: $u \leftarrow u_{\text{sel}}$
end

4. Experiments

To verify the effectiveness of the proposed method, we conduct experiments in unknown environments. The experimental design focuses on three core aspects: (1) obstacle avoidance tasks; (2) goal-driven navigation tasks; and (3) rationality analysis of the confidence model. The proposed method is comprehensively evaluated from both performance and mechanism perspectives.

4.1. Experimental Setup

Our experimental framework is grounded in the Robot Operating System (ROS) and the Gazebo simulator, with Rviz providing a real-time visualization of the gathered sensor data. The robot's perceptual suite consists of a calibrated LiDAR unit and an odometry sensor. To maintain computational efficiency and filter out noise, we compress the LiDAR's 180° forward-facing data into a 20-element vector by extracting the minimum values from 20 equal segments. This streamlined spatial information, when fused with the position and pose metrics from the odometer, forms the robot's complete state representation. Controlled through linear and angular velocity commands, the robot navigates an environment populated by various obstacles. Training episodes are governed by three terminal conditions: reaching the target, encountering a collision, or exhausting the maximum allotted time steps. The target network parameters were only updated every two episodes. The training was completed in a simulated environment of 10 × 10 m as shown in Figure 4.

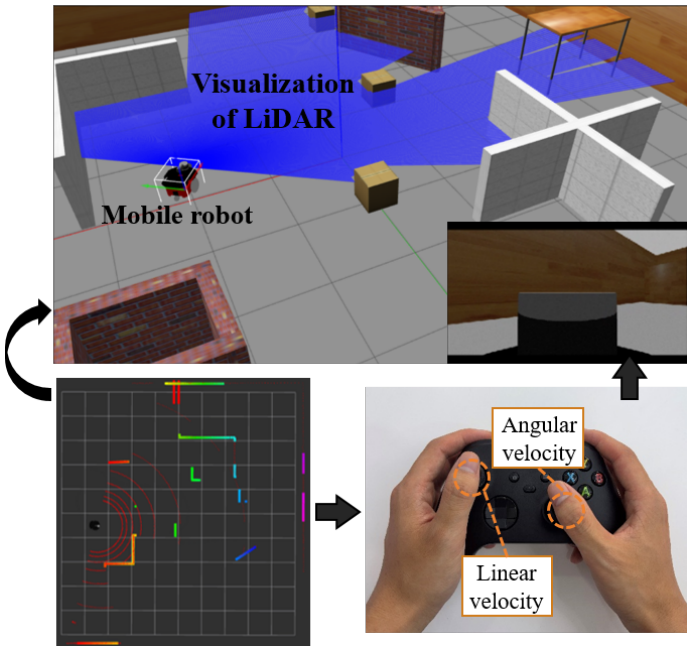


Figure 4. Example of training environments.

It should be explicitly noted that during the human-in-the-loop experiments, the human input (u^h) was provided by real human operators in real time via a standard ROS teleoperation node. Specifically, the shared-control experiments involved five human participants (aged 23–25). The five participants were used for controlled proof-of-concept evaluation of the shared-control mechanism rather than for population-level human-factors inference. Before testing, all participants received standardized instructions and practice sessions to become familiar with the teleoperation interface and velocity command inputs. During operation, participants received the same sensory feedback, including LiDAR observations and robot state information, through the teleoperation interface. Human commands were generated online according to a predefined operation protocol, where participants controlled the robot velocity based on the perceived environment and navigation objectives. Each participant conducted multiple independent trials under identical environment configurations and initial conditions, and the reported results were averaged over all participants. Since human commands were generated interactively during the shared-control process, the exact input sequences naturally varied across trials. It should also be clarified that only HMSC involved human input, whereas TD3 and MPC-TD3 were fully autonomous baselines without human intervention.

To promote policy generalization and encourage the exploration of new strategies, Gaussian noise is added to the action values. The exploration noise follows a distribution with a mean of 0 and a standard deviation that linearly decays from 1.0 to 0.1. The policy noise follows a distribution with a mean of 0, a standard deviation of 0.2, and is clipped within ±0.5. In addition, in each training episode, the robot's starting position, the target position, and the positions of the box-shaped obstacles are all randomly assigned. To ensure reproducibility, all experiments, network training, and baseline comparisons were conducted on the same high-performance workstation. The hardware platform was equipped with an Intel Xeon Silver 4210 CPU, 1 TB of system RAM, and four NVIDIA GeForce RTX 3090 GPUs, each providing 24 GB of GPU memory. The software environment consisted of Ubuntu 20.04 LTS, ROS Noetic for robot–environment interaction, and PyTorch 1.12.1 as the deep-learning backend. The global random seed was fixed to 0 for PyTorch network initialization, NumPy random-number generation, and replay-buffer sampling. The detailed parameter settings are listed in Table 1. The confidence-related parameters

in Table 1 were selected according to their functional roles in the proposed shared-control mechanism and were kept fixed throughout all experiments. The general design follows confidence-based shared-control principles, where control authority is adjusted according to the estimated reliability of autonomous decisions [34,35]. The specific numerical values are application-dependent implementation hyperparameters rather than universal constants. The confidence threshold $C_{\text{th}} = 0.15$ defines a low-confidence condition for control arbitration. Since the composite confidence C is normalized to $0, 1$, the relatively low threshold avoids frequent discrete switching when the confidence remains in an intermediate range, where continuous human–machine weighting is used instead. The curvature factor $\gamma = 2$ in Equation (10) produces a smooth nonlinear authority transition, progressively increasing human influence as the confidence decreases. For the uncertainty-to-confidence mapping in Equation (16), $\zeta = 10$ determines the sigmoid slope and $b = 3$ determines its offset. These values place the midpoint of the Bayesian confidence mapping, $C_B = 0.5$, at $U = b/\zeta = 0.3$, while maintaining a sufficiently smooth transition around this uncertainty level. The action dimension $d = 2$ is determined directly by the linear- and angular-velocity control outputs and is therefore not a tunable confidence parameter.

The candidate action space used by the MPC safety layer is discretized independently from the temporal prediction step. In the implementation, the linear velocity is uniformly sampled using 20 points over the interval $[0, 1]$ m/s, while the angular velocity is uniformly sampled using 50 points over $[-1, 1]$ rad/s. Therefore, the corresponding action-space resolutions are approximately $\Delta v = 0.05263$ m/s and $\Delta\omega = 0.04082$ rad/s, respectively. This results in at most $20 \times 50 = 1000$ candidate control actions at each control cycle. A fixed prediction time step of $\Delta t = 0.1$ s is used for all forward integration steps within the MPC horizon. Therefore, for $T = 1$ s, each candidate action is evaluated over $N = T/\Delta t = 10$ prediction steps.

Table 1. Parameter settings.

Parameter	Symbol	Value
Robot Kinematics		
Max. Linear Velocity	$v_{\max}$	1.0 m/s
Max. Angular Velocity	$\omega_{\max}$	1.0 rad/s
Robot Radius	R_{robot}	0.35 m
Safety Buffer	δ	0.1 m
Goal Reward	r_g	100
Collision Penalty	r_c	−100
MPC Safety Layer		
Prediction Horizon	T	1
Time Step	Δt	0.1 s
Horizon Steps	N	10
Human–Machine Shared Control		
MC-Dropout Samples	M	20
Curvature Factor	γ	2
Action Dimension	d	2
Confidence Threshold	C_{th}	0.15
Mapping Slope	ζ	10
Mapping Offset	b	3

To separately evaluate the contributions of the reinforcement learning controller, the MPC-based safety mechanism, and the proposed shared-control strategy, three methods are considered in the experiments: TD3, MPC-TD3, and the proposed HMSC framework. TD3 is used as the basic learning-based baseline, where the control command is generated

directly by the trained TD3 policy without any explicit safety filtering or human intervention. MPC-TD3 is constructed by integrating the same MPC-based safety layer used in the proposed HMSC framework with the TD3 controller, while excluding human input, MC-Dropout-based uncertainty estimation, confidence evaluation, and adaptive authority allocation. Specifically, the robot state observations are first processed by the TD3 policy to generate a nominal control action, which is then checked and, if necessary, corrected by the MPC safety layer before execution. Therefore, MPC-TD3 serves as an intermediate baseline for isolating the contribution of the MPC-based safety mechanism. The framework of MPC-TD3 is illustrated in Figure 5. In contrast, the proposed HMSC framework further introduces human input, confidence-aware authority allocation, and MC-Dropout-based uncertainty estimation, thereby enabling adaptive fusion between human and autonomous control under safety constraints.

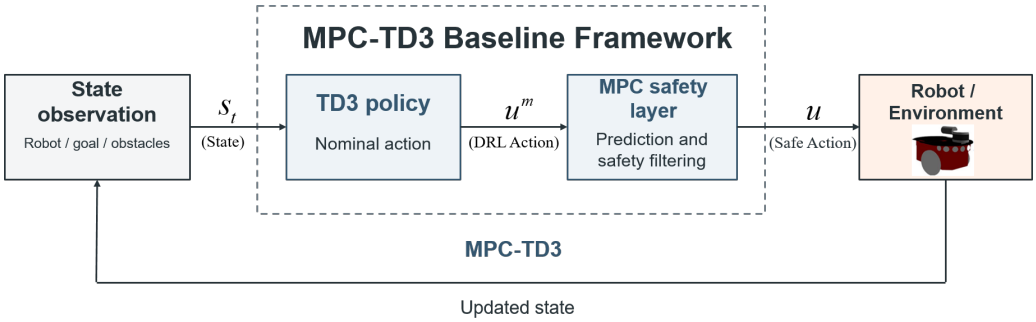


Figure 5. Framework of the MPC-TD3 baseline.

4.2. Local Safety Capability Verification: Obstacle Avoidance Task

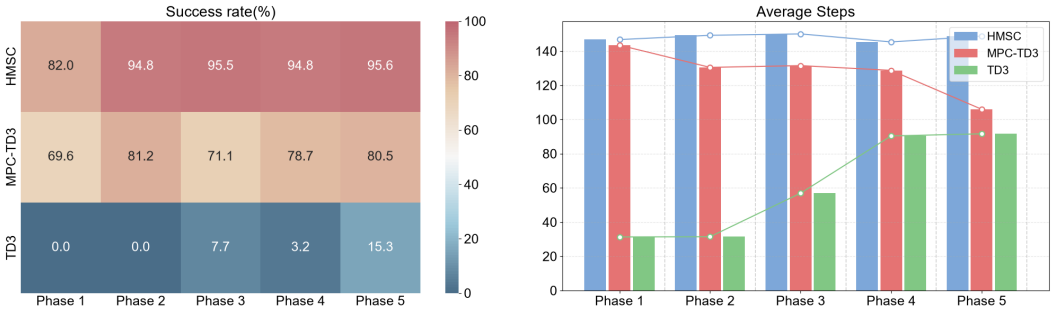
The obstacle avoidance task is designed to evaluate an algorithm's instantaneous safety decision-making capability in the absence of explicit goal constraints, with particular emphasis on its ability to move continuously and avoid collisions in unknown and complex environments. The task requires the robot to maintain continuous motion while maximizing the duration of collision-free operation. In this task, the robot's initial position is fixed, and multiple obstacles are randomly distributed in the environment. Each episode is assigned a maximum of 150 control steps and terminates either when a collision occurs or when the step limit is reached. Throughout the experiments, the robot makes decisions solely based on real-time sensor information, without relying on any global planning or map information. The obstacle-avoidance task adopts the following piecewise reward function:

$$r(s_t, a_t) = \begin{cases} r_c & \text{if a collision occurs,} \\ v - |\omega| - r_3(m) & \text{otherwise.} \end{cases} \tag{26}$$

At time step (t), the reward $r(s_t, a_t)$ is assigned according to the following rules. When a collision is detected, a negative penalty r_c is imposed. In all other cases, the instantaneous reward is determined by the current forward velocity (v), the absolute angular velocity $|\omega|$, and an additional penalty term based on the minimum LiDAR range measurement (m). Here, $r_3(\cdot)$ is a discriminative function defined as:

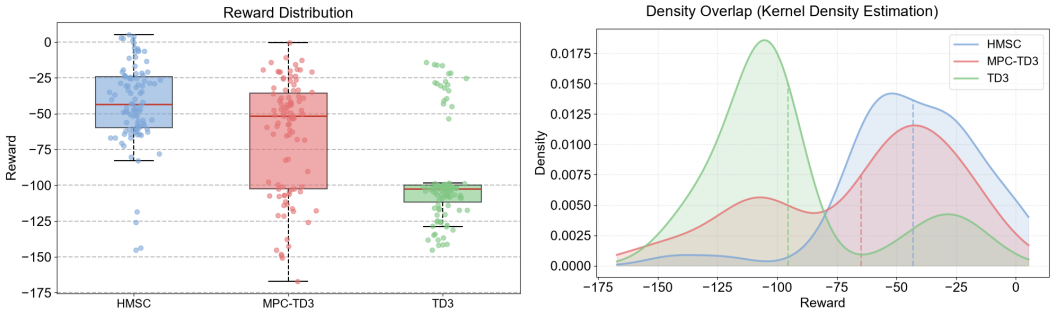
$$r_3(x) = \begin{cases} 1 - x, & x < 1, \\ 0, & x \geq 1. \end{cases} \tag{27}$$

To comprehensively evaluate obstacle avoidance performance, three metrics are considered: task success rate, survival steps, and reward distribution. The experimental results are shown in Figures 6 and 7:



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

Figure 6. Performance comparison of the proposed HMSC against MPC-TD3 and TD3 baselines. (**a**) Heatmap of success rates (%). (**b**) Average steps per episode. Phase 1–5 denote five training phases, each consisting of 20 episodes. HMSC demonstrates higher success rates and stability across all phases.



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

Figure 7. Comparative analysis of reward statistics for HMSC, MPC-TD3, and TD3. (**a**) Reward distribution illustrated via box plots. (**b**) Density overlap visualization using Kernel Density Estimation (KDE). The blue, red, and green colors correspond to HMSC, MPC-TD3, and TD3, respectively, highlighting the superior reward accumulation of the HMSC method.

The success-rate heatmap in Figure 6a underscores the superior adaptability and robustness of HMSC. The proposed framework maintains a consistently high success rate across all test scenarios, averaging over 90% and peaking at 95.5% in Phase 3. The baseline TD3 algorithm fails to complete any trials in Phases 1 or 2, reflecting the fundamental limitations of pure reinforcement learning when navigating complex environmental constraints. These shortcomings are further evidenced by the step-count data in Figure 6b; the exceptionally low averages for TD3 suggest frequent premature terminations caused by safety violations or collisions. HMSC sustains significantly longer trajectories, validating its capacity for stable, long-term planning. This reliability is corroborated by the reward distribution in Figure 7a, where HMSC yields the highest median reward with a narrow interquartile range. While MPC-TD3 occasionally outperforms the baseline, its erratic performance—marked by high variance and frequent low-value outliers—suggests a lack of the consistency required for robust operation. Finally, the KDE curves (Figure 7b) reinforce this conclusion: the reward probability density of HMSC is significantly right-skewed, with its peak concentrated in the high-reward region, in sharp contrast to TD3's sharply unimodal distribution centered in the low-reward region. Overall, HMSC significantly outperforms the baseline algorithms in terms of task completion rate, safety, and long-term reward quality.

4.3. Global Capability Verification: Goal-Driven Navigation Task

In the goal-driven navigation task, the robot is required to move from a start position to a randomly generated target point in each episode within an unknown environment, while continuously avoiding static and dynamic obstacles along the path. This task simultaneously evaluates the policy's long-term planning capability (reaching randomly placed goals) and its real-time safety control performance (collision avoidance and trajectory smoothness). To guide policy learning and balance navigation efficiency with driving safety, the following piecewise reward function is adopted:

$$r(s_t, a_t) = \begin{cases} r_g & \text{if } D_t < \eta_D, \\ r_c & \text{if a collision occurs,} \\ v - |\omega| - r_3(m) & \text{otherwise.} \end{cases} \tag{28}$$

To balance obstacle avoidance and navigation capabilities, a goal-reaching reward is incorporated on top of the original obstacle-avoidance reward. Specifically, when the distance between the robot and the target, denoted as D_t, falls below a predefined threshold η_D, a positive goal reward r_g is assigned. To evaluate the decision-making capability of the proposed method in goal-driven scenarios with randomly sampled targets, we conducted systematic training and testing of three approaches (HMSC, MPC-TD3, and TD3) under a unified experimental protocol.

Specifically for the proposed HMSC framework, to accommodate its unique shared control mechanism while preventing operator fatigue, we implemented a specific three-stage curriculum training process across the 4000 episodes. In Phase 1 (Episodes 1–500), the TD3 agent trains purely autonomously using exploration noise to learn fundamental obstacle avoidance. In Phase 2 (Episodes 501–510), the robot was controlled collaboratively by human operators and the system. This brief intervention injects high-quality demonstrations to help the agent escape local optima and calibrates the confidence model. Finally, in Phase 3 (Episodes 511–4000), human input is removed, allowing the agent to autonomously converge using the enriched experience.

During training, online evaluations were conducted at fixed intervals to track the learning dynamics and training stability. Multiple performance metrics were recorded throughout the evaluation process to assess the learning effectiveness of the proposed algorithm. Figure 8a–c illustrate the learning trajectories of the three evaluated methods. HMSC demonstrates a clear advantage in convergence, particularly regarding the mean Q-value (Figure 8a) and maximum Q-value (Figure 8c), which serve as proxies for policy value estimation. The Loss curves in Figure 8b show that HMSC maintains a smoother profile and lower convergence values, confirming superior gradient stability during optimization. The success rate heatmap (Figure 8d) provides a direct visual of HMSC's consistent lead; it sustains a success rate above 85% throughout the latter half of the training. These results are corroborated by the reward distributions in Figure 8e and collision statistics in Figure 8f. Specifically, HMSC achieves the highest median cumulative reward with the tightest distribution, while successfully suppressing collisions to a minimal level. Overall, HMSC outperforms both MPC-TD3 and pure TD3 in key dimensions, including training stability, sample efficiency, and safety.

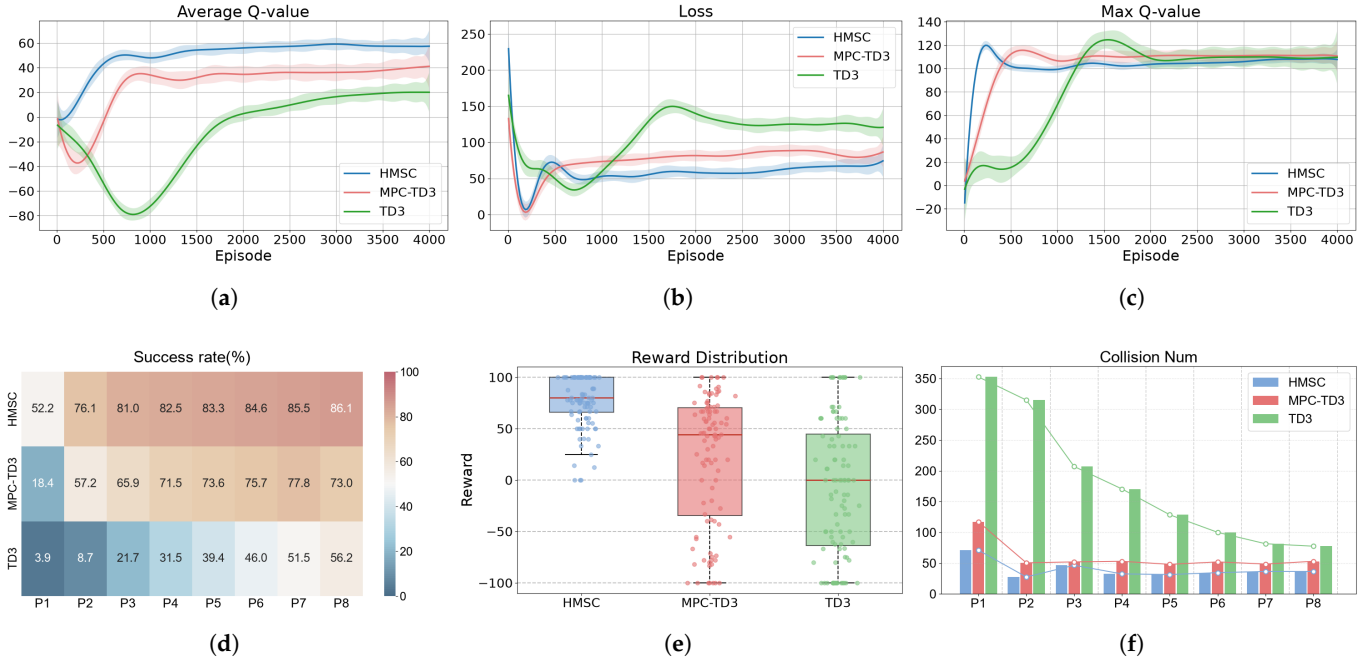
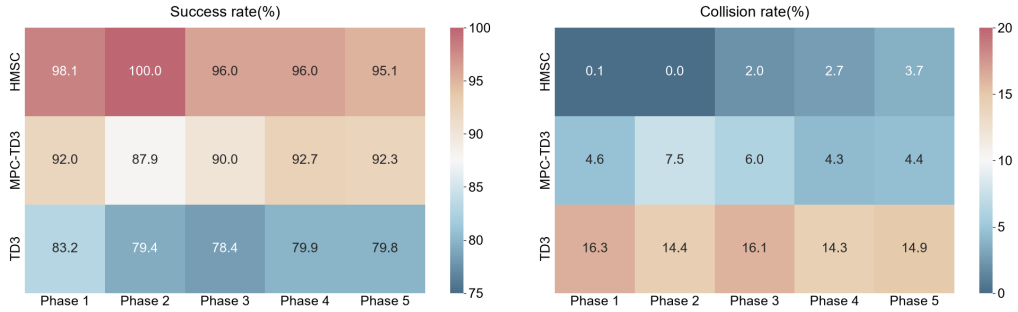


Figure 8. Training process and performance comparison of the HMSC, MPC-TD3, and TD3 methods. Subplots (**a**–**c**) show the evolution of the average Q-value, loss, and maximum Q-value over 4000 episodes, respectively. (**d**) The success rate (%) presented as a heatmap. (**e**) The statistical distribution of rewards shown via box plots. (**f**) The number of collisions recorded in each phase. Phases 1–8 denote eight distinct training phases, with each phase consisting of 500 episodes.

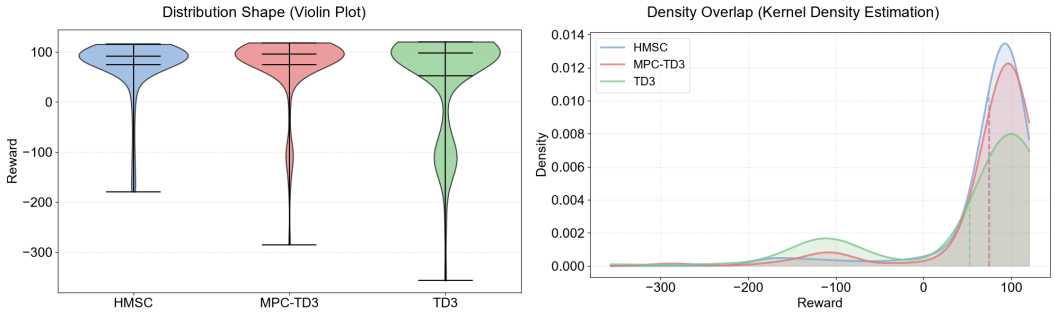
During the testing phase, the human operators fully and continuously participated in the shared control loop throughout the entire duration of each trial. These inputs were dynamically weighted against the autonomous policy based on the composite confidence and safely filtered by the MPC layer at each control step. The testing performance, captured through deterministic indicators (Figure 9) and reward statistics (Figure 10), highlights the advantages of HMSC. As shown in Figure 9, HMSC achieves the best overall results, maintaining a success rate over 95% and a remarkably low collision rate (under 3.7%). This performance confirms the model's reliability in dynamic environments. The standard TD3 algorithm exhibits poor safety, with collision rates surpassing 14%, failing to meet the requirements for effective obstacle avoidance. MPC-TD3 outperforms TD3 but still falls significantly short of HMSC across all metrics. The violin plot in Figure 10a and the KDA in Figure 10b further reveal the convergence quality of the algorithms. The reward distribution of HMSC exhibits a pronounced "high-mean, low-variance" characteristic, with probability density sharply concentrated in the high-reward region and no evident long tail in the low-reward range. This indicates that the model can stably reproduce near-optimal policies. The TD3 algorithm shows a much wider distribution with a secondary peak in the low-reward region, suggesting high policy uncertainty and a tendency to fall into local optima or experience task failures.

To complement the graphical results in Figures 9 and 10, Table 2 reports the corresponding numerical statistics. The success and collision rates are summarized as the mean ± standard deviation across the five evaluation phases, while the cumulative reward is reported as the mean ± standard deviation over the 100 evaluation episodes. HMSC achieves the highest average success rate of 97.04% and the lowest average collision rate of 1.71%. Its cumulative reward is comparable to that of MPC-TD3, while exhibiting lower variability. In contrast, TD3 shows a substantially lower success rate, a considerably higher collision rate, and the largest reward variability.



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

Figure 9. Operational performance metrics across five experimental phases. (**a**) Success rate (%) heatmap illustrating the task completion capability of HMSC, MPC-TD3, and TD3. (**b**) Collision rate (%) heatmap characterizing the safety performance of each algorithm.



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

Figure 10. Characterizing reward distribution and convergence across different approaches. (**a**) Violin plots provide a granular view of reward ranges and central tendencies. (**b**) Policy robustness is evaluated through KDE curves, which highlight the concentration and overlap of reward probabilities.

Table 2. Quantitative summary of the testing performance.

Method	Success Rate (%)	Collision Rate (%)	Cumulative Reward
HMSC	97.04 ± 1.99	1.71 ± 1.62	75.71 ± 57.80
MPC-TD3	90.98 ± 2.01	5.36 ± 1.38	74.23 ± 67.77
TD3	80.14 ± 1.81	15.20 ± 0.94	52.49 ± 96.95

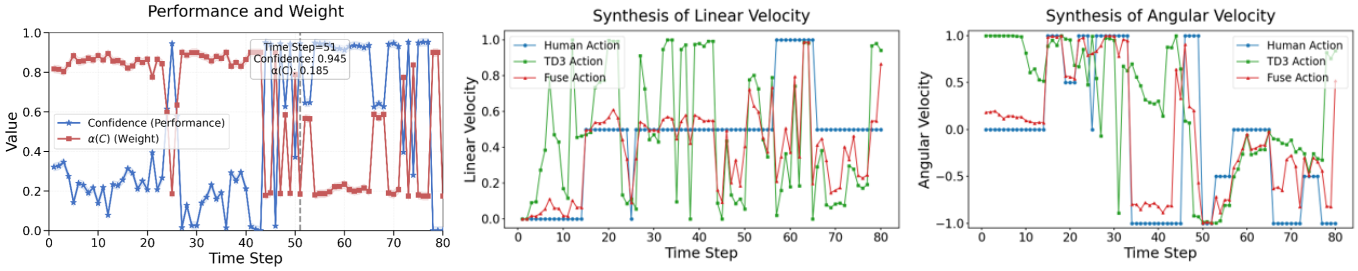
HMSC excels in managing intricate control scenarios, consistently surpassing baseline benchmarks. Beyond simply boosting success rates, the algorithm maintains a robust policy and ensures high safety standards during interactions—addressing a common weakness in standard models.

4.4. Analysis of the Rationality of the Confidence Model

The confidence model serves as a critical bridge between autonomous decision-making and human intervention, and the accuracy of its evaluation as well as the rationality of its modulation mechanism directly determine the robustness of the overall system. To validate this mechanism, we recorded the evolution of confidence, the corresponding weight allocation, and the final action synthesis during a single representative experimental run.

Figure 11a intuitively reveals a dynamic negative correlation between the confidence value (Confidence) and the fusion weight ($\alpha(C)$). From the curves, it can be clearly observed that the system is capable of adaptively adjusting the weight according to the current state safety and policy reliability. Taking the annotated Step = 51 as an example, the agent is in a

relatively familiar or safe state, where the estimated confidence reaches as high as 0.945. Accordingly, the system reduces the weight of external intervention ($\alpha(C)$) to 0.185. Such a low weight allows the TD3 agent to dominate the control authority and fully exploit the efficient policy learned during training. In contrast, within the intervals of Steps 5–20 and around Step 35, the confidence exhibits severe fluctuations and drops to a low level, indicating high uncertainty or potential collision risks. In these cases, the model responds rapidly by significantly increasing the fusion weight, forcibly introducing human guidance to prevent unsafe actions.



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

Figure 11. Confidence-based Dynamic Action Fusion Analysis. (**a**) Performance evaluation and weight allocation. (**b**) Linear velocity synthesis results. (**c**) Angular velocity synthesis results.

This confidence-based weight adjustment is directly mapped to the low-level action execution layer. Figure 11b and 11c plot the fusion processes for angular and linear velocities, respectively. Comparing the human, TD3, and fused action curves reveals that the final output is not a mere linear combination. Instead, it adaptively shifts between the two inputs based on real-time weights. When confidence levels are high, the fused action converges with the TD3 output to drive the system autonomously. In contrast, during periods of low-confidence fluctuations, the fused action shifts toward the human input. This mechanism effectively strikes a balance between exploration efficiency and operational safety, thereby demonstrating the rationality and effectiveness of the proposed confidence model when handling unknown and complex environments.

To qualitatively evaluate the performance of the HMSC method, we recorded the robot's motion trajectories during navigation using the trained HMSC model and plotted the resulting paths, as shown in Figure 12. The results intuitively demonstrate that the proposed method can achieve efficient, safe, and smooth trajectory planning and navigation in complex 3D unknown environments.

To quantitatively evaluate the online computational efficiency of the proposed HMSC framework, computational latency was recorded over (N = 493) control-cycle samples during navigation. Each sample corresponds to one complete online decision cycle, including TD3 policy inference, MPC-based safety evaluation, MC-Dropout uncertainty estimation, and confidence-based human–machine control fusion. As shown in Table 3, the proposed HMSC framework requires an average of 53.476 ms to complete one decision cycle on the experimental computing platform. Let $\bar{T}$cycle denote the average computational time of one complete HMSC decision cycle. Since $\bar{T}$cycle = 53.476 ms, the corresponding average online execution frequency can be calculated as

$$f_{\text{online}} = \frac{1}{\bar{T}_{\text{cycle}}} = \frac{1000}{53.476} \approx 18.70 \text{ Hz}. \tag{29}$$

Here, the factor 1000 is used to convert the computational time from milliseconds to seconds. Among the major computational components, TD3 policy inference, the MPC safety layer,

MC-Dropout uncertainty estimation with (M = 20), and confidence-based human–machine control fusion require average computational times of 4.697 ms, 25.361 ms, 22.993 ms, and 0.345 ms, respectively. Therefore, based on the 493 measured control-cycle samples, the proposed HMSC framework achieves an average online decision-making capability of approximately 18.70 Hz on the experimental computing platform.

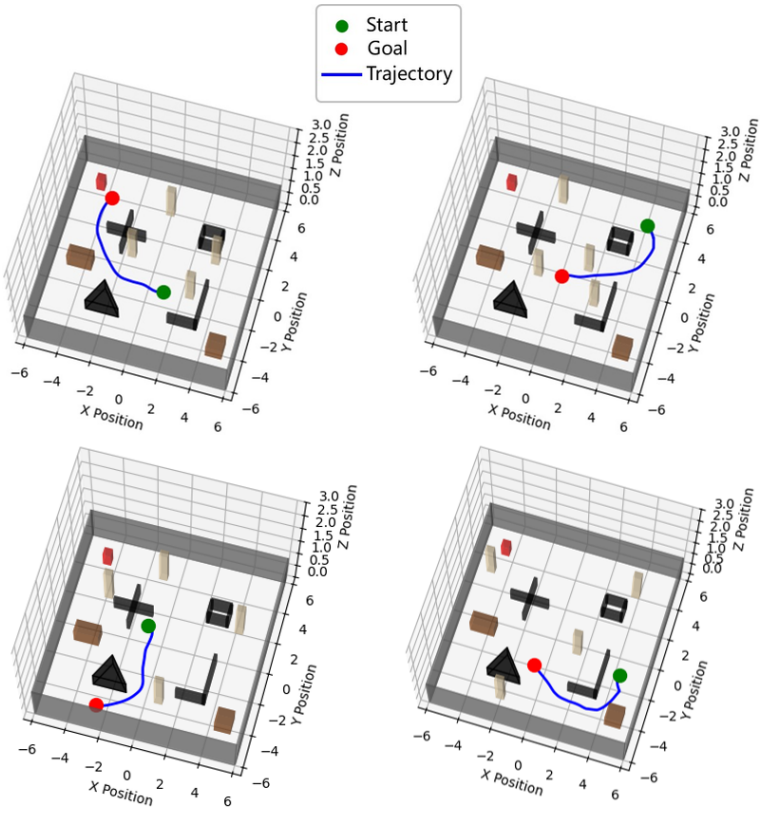


Figure 12. Trajectory generation results under different initial configurations in three-dimensional space.

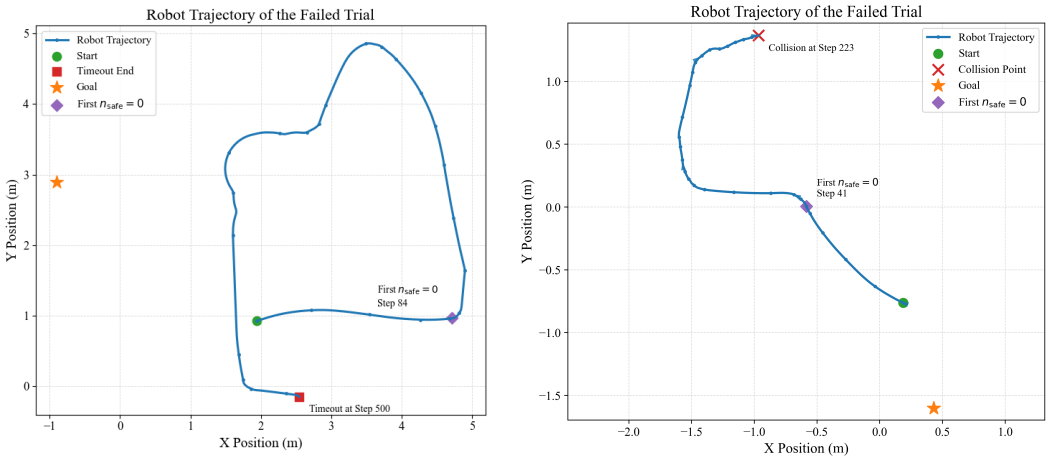
The MC-Dropout sample number was empirically set to $M = 20$ to balance sampling variability and computational cost. MC-Dropout estimates predictive moments through repeated stochastic forward passes [41]. The selected sample number is consistent with Loquercio et al. [28], who used 20 samples and discussed the trade-off between uncertainty estimation quality and inference time. For our implementation, the mean MC-Dropout computation time is 22.993 ms, and the mean complete decision-cycle time is 53.476 ms, as reported in Table 3. These measurements characterize the selected configuration;

Table 3. Computational performance of the proposed HMSC framework.

Component	Mean (ms)	Std. (ms)	Max. (ms)
TD3 policy inference	4.697	6.307	48.029
MPC safety layer	25.361	11.884	76.038
MC-Dropout ($M = 20$)	22.993	26.917	180.166
Confidence and control fusion	0.345	1.819	35.242
Complete HMSC cycle	53.476	30.882	218.874

To further examine the non-ideal operating conditions of the proposed framework, representative failure cases are presented in Figure 13. Figure 13a shows a timeout case in which the robot fails to reach the target within the maximum limit of 500 control steps. During this trial, the sampled safe-action set becomes empty ($n_{\text{safe}} = 0$) at several stages,

indicating temporary loss of locally feasible candidate actions. Although the robot continues to move without collision, repeated local corrections reduce its goal-directed progress and eventually result in timeout. Figure 13b presents a collision case. The sampled safe-action set first becomes empty at Step 41 and subsequently recovers; however, a more critical condition occurs near the end of the trial, where persistent local infeasibility is observed before the collision at Step 223.



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

Figure 13. Representative failure trajectories of HMSC. (**a**) Timeout failure trajectory, where the robot does not reach the target within the maximum number of control steps. (**b**) Collision failure trajectory, where the robot eventually collides with an obstacle at Step 223. The start position, target position, first occurrence of $n_{\text{safe}} = 0$, and final failure position are indicated.

To further illustrate this process, Figure 14 shows the evolution of n_{safe} and the minimum LiDAR distance in the collision trial. From Step 173 to Step 223, n_{safe} remains zero for 51 consecutive control cycles, while the minimum LiDAR distance decreases from approximately 0.460 m to 0.352 m. When no feasible sampled action is available, the current implementation invokes an emergency low-speed turning strategy; nevertheless, this fallback behavior does not restore a feasible local configuration before the collision occurs. These observations indicate that the proposed HMSC framework substantially improves navigation safety but cannot completely eliminate failure cases. In particular, persistent local infeasibility and repeated conservative corrections remain important limitations of the current finite-horizon and discretized MPC safety mechanism.

Table 4 reports the wall-clock training cost of the evaluated methods under the same experimental computing environment. HMSC requires a longer overall training time than the baseline methods. This difference should not be interpreted solely as additional per-step computational overhead. In episodic navigation, the total wall-clock training time is also affected by the number of environment-interaction steps completed in each episode. Since HMSC exhibits improved obstacle-avoidance performance and fewer premature terminations, its training episodes can remain active for longer durations, leading to a larger cumulative number of simulation and control steps. Therefore, the longer wall-clock training time reflects both the additional computations introduced by MPC-based safety evaluation and uncertainty estimation, and the longer effective episode durations resulting from improved navigation performance.

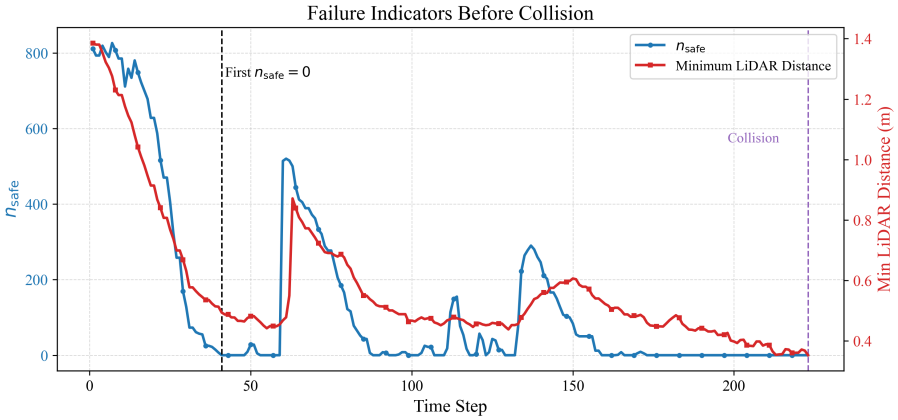


Figure 14. Evolution of the number of feasible sampled actions n_{safe} and the minimum LiDAR distance in the representative collision trial. The first occurrence of $n_{\text{safe}} = 0$ and the collision at Step 223 are indicated.

Table 4. Model training time on the experimental computing platform.

Method	Training Episodes	Total Training Time (h)	Average Time/Episode (s)
HMSC	4000	29.21	26.29
MPC-TD3	4000	26.87	24.18
TD3	4000	17.58	15.82

5. Conclusions

This paper presents and implements a confidence-aware human–machine shared control framework, termed HMSC. The method augments the TD3 policy with Monte Carlo Dropout to construct a Bayesian TD3 controller, enabling online estimation of policy uncertainty and confidence-driven allocation of control authority between the human operator and the autonomous agent. MPC is incorporated as an online safety layer to perform short-horizon trajectory prediction and geometric safety checking for all candidate actions.

From a design perspective, the proposed approach couples deep reinforcement learning with MPC to achieve effective planning in unknown environments at the policy level, while enforcing online safety verification during control execution. By jointly considering policy confidence and MPC feasibility, the framework proactively safeguards high-risk states and grants human intervention in a principled manner. Beyond merely preserving efficiency, HMSC offers significant improvements in training stability and system safety. Our Gazebo-based evaluations, which included complex navigation and obstacle avoidance scenarios, confirm that HMSC holds a clear advantage over traditional TD3-based methods, including those augmented by MPC. HMSC exhibits faster convergence, higher sample efficiency, improved success rates, and reduced collision rates. The visualization of confidence evolution over time shows that when confidence levels are low, more conservative control strategies or human intervention are activated. These results confirm the rationality of the proposed confidence model in HMSC and unknown scenarios. The main advantages of HMSC arise from the complementary integration of learning-based decision making, confidence-aware human–machine authority allocation, and model-based geometric safety verification. The proposed framework can adaptively increase human involvement when autonomous-policy confidence decreases, while the MPC safety layer provides an additional verification of candidate actions before execution. Moreover, the use of compressed LiDAR observations maintains a relatively lightweight perception representation, and the current implementation achieves an average online decision frequency of approximately 18.70 Hz on the experimental computing platform. Nevertheless, several limitations re-

main. First, the MPC safety layer relies on a finite prediction horizon and a discretized candidate-action set; therefore, persistent local infeasibility may not always be resolved, as demonstrated by the failure cases analyzed in Section 4.4. Second, the current safety evaluation mainly depends on locally observed LiDAR geometry and does not explicitly predict future obstacle motion. Third, MC-Dropout uncertainty estimation introduces additional computational cost and latency variation. In addition, the shared-control performance remains dependent on the quality and timeliness of human input. Finally, the present evaluation is conducted in simulation and mainly compares against TD3-based baselines, which limits conclusions regarding real-world deployment and broader comparisons with other HMSC approaches.

Future work will focus on improving the MPC-based safety layer through enhanced recovery strategies and dynamic-obstacle prediction for locally constrained situations. Meanwhile, the computational efficiency of uncertainty estimation and safety evaluation will be further optimized, together with standardized comparisons against additional HMSC methods under a unified experimental protocol.

Author Contributions: Conceptualization, Z.C.; methodology, Q.Z.; software, Z.C.; validation, Z.C.; investigation, Z.C. and Q.Z.; resources, Z.L.; writing—original draft preparation, Z.C. and Q.Z.; writing—review and editing, Z.C. and Q.Z.; supervision, Z.L.; funding acquisition, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 62203006, Grant No. 62173317, Grant No. 92467302, Grant No. U25A20454).

Data Availability Statement: The original contributions presented in the study are included in the article.

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

References

1. Bernardo, R.; Sousa, J.M.; Gonçalves, P.J. Survey on robotic systems for internal logistics. *J. Manuf. Syst.* **2022**, *65*, 339–350. [CrossRef]
2. Koung, D.; Kermorgant, O.; Fantoni, I.; Belouaer, L. Cooperative multi-robot object transportation system based on hierarchical quadratic programming. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6466–6472. [CrossRef]
3. Liu, Y.; Wang, S.; Xie, Y.; Xiong, T.; Wu, M. A review of sensing technologies for indoor autonomous mobile robots. *Sensors* **2024**, *24*, 1222. [CrossRef] [PubMed]
4. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
5. Dijkstra, E.W. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His life, Work, and Legacy*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 287–290.
6. Orthey, A.; Chamzas, C.; Kavraki, L.E. Sampling-based motion planning: A comparative review. *Annu. Rev. Control. Robot. Auton. Syst.* **2023**, *7*, 285–310. [CrossRef]
7. Lee, D.H.; Lee, S.S.; Ahn, C.K.; Shi, P.; Lim, C.C. Finite distribution estimation-based dynamic window approach to reliable obstacle avoidance of mobile robot. *IEEE Trans. Ind. Electron.* **2020**, *68*, 9998–10006. [CrossRef]
8. Yang, W.; Wu, P.; Zhou, X.; Lv, H.; Liu, X.; Zhang, G.; Hou, Z.; Wang, W. Improved artificial potential field and dynamic window method for amphibious robot fish path planning. *Appl. Sci.* **2021**, *11*, 2114. [CrossRef]
9. Nascimento, T.P.; Dórea, C.E.; Gonçalves, L.M.G. Nonholonomic mobile robots' trajectory tracking model predictive control: A survey. *Robotica* **2018**, *36*, 676–696. [CrossRef]
10. Engelsman, D.; Klein, I. Information-aided inertial navigation: A review. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–18. [CrossRef]
11. Luo, Y.; Lu, F.; Guo, C.; Liu, J. Matrix lie group-based extended Kalman filtering for inertial-integrated navigation in the navigation frame. *IEEE Trans. Instrum. Meas.* **2023**, *73*, 1–16. [CrossRef]
12. Wu, Z.; Yue, Y.; Wen, M.; Zhang, J.; Yi, J.; Wang, D. Infrastructure-free hierarchical mobile robot global localization in repetitive environments. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–12. [CrossRef]
13. Cimurs, R.; Suh, I.H.; Lee, J.H. Goal-driven autonomous exploration through deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2021**, *7*, 730–737. [CrossRef]

14. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
15. Kang, Y.; Di, J.; Li, M.; Zhao, Y.; Wang, Y. Autonomous multi-drone racing method based on deep reinforcement learning. *Sci. China Inf. Sci.* **2024**, *67*, 180203. [CrossRef]
16. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
17. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971 .
18. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347 .
19. 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 International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Proceedings of Machine Learning Research; pp. 1861–1870.
20. Chiang, H.T.L.; Faust, A.; Fiser, M.; Francis, A. Learning navigation behaviors end-to-end with autorl. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2007–2014. [CrossRef]
21. Linial, O.; Tennenholtz, G.; Shalit, U. Benchmarks for Reinforcement Learning with Biased Offline Data and Imperfect Simulators. *arXiv* **2024**, arXiv:2407.00806 .
22. Wagenmaker, A.; Huang, K.; Ke, L.; Jamieson, K.; Gupta, A. Overcoming the Sim-to-Real Gap: Leveraging Simulation to Learn to Explore for Real-World RL. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 78715–78765. [CrossRef]
23. Losey, D.P.; McDonald, C.G.; Battaglia, E.; O'Malley, M.K. A review of intent detection, arbitration, and communication aspects of shared control for physical human–robot interaction. *Appl. Mech. Rev.* **2018**, *70*, 010804. [CrossRef]
24. Li, X.; Wang, X.; Zheng, X.; Jin, J.; Huang, Y.; Zhang, J.J.; Wang, F.Y. SADRL: Merging human experience with machine intelligence via supervised assisted deep reinforcement learning. *Neurocomputing* **2022**, *467*, 300–309. [CrossRef]
25. Li, X.; Wang, Y.; Su, C.; Gong, X.; Huang, J.; Yang, D. Adaptive authority allocation approach for shared steering control system. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 19428–19439. [CrossRef]
26. Marcano, M.; Díaz, S.; Pérez, J.; Irigoyen, E. A review of shared control for automated vehicles: Theory and applications. *IEEE Trans. Hum. Mach. Syst.* **2020**, *50*, 475–491. [CrossRef]
27. Ames, A.D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; Tabuada, P. Control barrier functions: Theory and applications. In *Proceedings of the 2019 18th European Control Conference (ECC), Naples, Italy, 25–28 June 2019*; IEEE: New York, NY, USA, 2019; pp. 3420–3431.
28. Loquercio, A.; Segu, M.; Scaramuzza, D. A general framework for uncertainty estimation in deep learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3153–3160. [CrossRef]
29. Abbink, D.A.; Mulder, M.; Boer, E.R. Haptic shared control: Smoothly shifting control authority? *Cogn. Technol. Work.* **2012**, *14*, 19–28. [CrossRef]
30. Flad, M.; Fröhlich, L.; Hohmann, S. Cooperative shared control driver assistance systems based on motion primitives and differential games. *IEEE Trans.-Hum.-Mach. Syst.* **2017**, *47*, 711–722. [CrossRef]
31. Warnell, G.; Waytowich, N.; Lawhern, V.; Stone, P. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018 ; Volume 32.
32. Mandel, T.; Liu, Y.E.; Brunskill, E.; Popović, Z. Where to add actions in human-in-the-loop reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017 ; Volume 31.
33. Wu, J.; Huang, Z.; Huang, C.; Hu, Z.; Hang, P.; Xing, Y.; Lv, C. Human-in-the-loop deep reinforcement learning with application to autonomous driving. *arXiv* **2021**, arXiv:2104.07246.
34. Saeidi, H.; Opfermann, J.D.; Kam, M.; Raghunathan, S.; Léonard, S.; Krieger, A. A confidence-based shared control strategy for the smart tissue autonomous robot (STAR). In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018* ; IEEE: New York, NY, USA, 2018; pp. 1268–1275.
35. Amirshirzad, N.; Kumru, A.; Oztop, E. Human adaptation to human–robot shared control. *IEEE Trans. -Hum.-Mach. Syst.* **2019**, *49*, 126–136. [CrossRef]
36. Zeng, J.; Zhang, B.; Sreenath, K. Safety-critical model predictive control with discrete-time control barrier function. In *Proceedings of the 2021 American Control Conference (ACC), Virtual, 25–28 May 2021* ; IEEE: New York, NY, USA, 2021; pp. 3882–3889.
37. Wabersich, K.P.; Zeilinger, M.N. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica* **2021**, *129*, 109597. [CrossRef]
38. Ames, A.D.; Xu, X.; Grizzle, J.W.; Tabuada, P. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Autom. Control* **2016**, *62*, 3861–3876. [CrossRef]

39. Fisac, J.F.; Akametalu, A.K.; Zeilinger, M.N.; Kaynama, S.; Gillula, J.; Tomlin, C.J. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Trans. Autom. Control* **2018**, *64*, 2737–2752. [CrossRef]
40. Gal, Y. Uncertainty in Deep Learning. Ph.D. Thesis, University of Oxford, Oxford, UK, 2016.
41. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016 ; Proceedings of Machine Learning Research; pp. 1050–1059.
42. Kendall, A.; Gal, Y. What uncertainties do we need in bayesian deep learning for computer vision? *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5580–5590.
43. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, Stockholm Sweden, 10–15 July 2018; Proceedings of Machine Learning Research; pp. 1587–1596.

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.