Proceedings of the Thirty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2026)

SaferSAC: A Deep Reinforcement Learning Framework for Autonomous Obstacle Avoidance Navigation in UAVs

Kaibo Su, Kun Zhu*

Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
{sukaibo, zhukun}@nuaa.edu.cn

Abstract

Autonomous obstacle avoidance and navigation for Unmanned Aerial Vehicles (UAVs) in dynamic environments presents a significant challenge in the field of autonomous planning. Traditional sampling- and optimization-based methods typically rely on global maps and necessitate frequent replanning, rendering them often ill-suited for real-time applications in dynamic scenarios. While deep reinforcement learning (DRL) approaches offer end-to-end perception-decision mapping, they often suffer from limitations in three-dimensional (3D) perception accuracy, sample efficiency, and action space adaptability. To address these challenges, this paper proposes SaferSAC, a DRL framework specifically tailored for robust UAV obstacle avoidance and navigation. First, we design a depth-semantic fusion-based 3D obstacle detection module that achieves precise spatial awareness by jointly processing depth images and semantic segmentation results. Second, we introduce a four-buffer prioritized experience replay mechanism that differentially stores and samples experiences based on distinct categories (e.g., success, obstacle, and precision), thereby significantly enhancing sample efficiency. Finally, we propose an optimization-based adaptive action space planning method. By solving a constrained optimization problem to dynamically adjust action boundaries for velocity and yaw rate, this method enhances the safety and flexibility of local avoidance maneuvers. Experimental results demonstrate that, compared to baseline methods, our approach yields more robust navigation strategies and substantially improves success rates in complex environments.

Code — https://github.com/Sukb24/SaferSAC

Introduction

UAVs are a crucial part of modern aviation and have shown significant potential for applications in various fields, including payload transport (Li et al. 2023), post-disaster emergency communications (Yang et al. 2024b), and exploration (Zhou, Xu, and Shen 2023). Traditional planning-based methods, such as Rapidly Exploring Random Trees (RRT) and Model Predictive Control (MPC), perform well in structured environments but face substantial challenges in scenarios with dynamic obstacles.

In recent years, DRL has offered innovative solutions for the autonomous navigation of UAVs. DRL methods address environmental uncertainty and dynamics (e.g., (Loquercio et al. 2021; Kaufmann et al. 2023)) by directly mapping perceptual data to control commands through end-to-end learning. Notably, the Soft Actor-Critic (SAC) algorithm (Haarnoja et al. 2018) has shown exceptional performance in continuous control tasks, owing to its superior exploration capabilities and training stability, significantly surpassing traditional methods.

However, applying DRL to dynamic obstacle avoidance for UAVs still faces three core challenges. The first challenge is the trade-off between accuracy and real-time performance in 3D obstacle perception. Existing end-to-end DRL methods primarily use single-modal inputs, such as RGB images or depth maps, which makes it challenging to obtain both precise geometric information and semantic understanding simultaneously. Monocular depth estimation, though lightweight, suffers from scale ambiguity (Yang et al. 2024a). Estimating the 3D bounding box and velocity of obstacles in real time through depth images helps achieve a balance between the accuracy and efficiency of geometric perception (Lin, Zhu, and Alonso-Mora 2020). However, performance can degrade under strong light conditions. The second challenge is the tension between sample efficiency and partial observability. Prioritised Experience Replay (PER) (Schaul et al. 2015) samples important experiences based on temporal difference errors, but its effectiveness is limited in partially observable and sparsely rewarded UAV navigation environments. The third challenge is the limited adaptability of fixed action spaces to complex environments. Continuous action masking methods focus on learning by dynamically constraining the action space to state-relevant action sets, which leads to faster convergence across four control tasks (Stolz et al. 2024). However, existing methods cannot differentially adjust the UAV's action dimensions based on environmental conditions. Furthermore, due to the black-box nature of neural networks, it is challenging to provide an intuitive explanation for an agent's decisions, which complicates ensuring safety during real-time obstacle avoidance.

This paper proposes the Semantic-Aware Four-buffer Experience Replay with Soft Actor-Critic (SaferSAC) framework to address the aforementioned challenges. It is specif-

*Corresponding author.

ically designed for UAV obstacle avoidance navigation tasks. The framework integrates semantic awareness, a four-buffer hierarchical experience replay mechanism, and optimization-based adaptive action space planning, enabling efficient obstacle avoidance and navigation in complex environments. The contributions of our work are summarized as follows:

- **A 3D Obstacle Detection Method with Depth-Semantic Fusion.** We designed a lightweight multi-modal fusion architecture that processes depth images and semantic segmentation results together to accurately compute the 3D bounding boxes of obstacles.
- **Four-Buffer Prioritized Experience Replay Mechanism.** We propose a four-buffer design specifically tailored to the UAV obstacle avoidance navigation tasks, including a main experience buffer, a success experience buffer, an obstacle experience buffer, and a precision experience buffer.
- **An Optimization-Based Adaptive Action Space Planning Method.** At each decision step, we dynamically adjust the action space boundaries for horizontal velocity and yaw rate by solving a constrained optimization problem, utilizing obstacle distribution information analyzed from LiDAR data.

Related Work

The technical evolution of UAV navigation and obstacle avoidance has shifted from planning-based methods to data-driven intelligent approaches. This section categorizes UAV navigation methods into planning-based and learning-based methods.

Planning-based methods: Planning-based algorithms represent classical approaches to UAV path planning. The RRT (Karaman and Frazzoli 2011) ensures asymptotic optimality by introducing a rewiring mechanism. (Mellinger and Kumar 2011) proposed a minimum-snap trajectory generation method for quadcopters based on differential flatness theory. By optimizing the fourth-order derivative of position, it generates smooth trajectories and uses a nonlinear controller for precise tracking. EGO-Planner (Zhou et al. 2020) proposes a gradient-based planning framework based on voxel maps that eliminates the need for ESDF, significantly reducing computational time. MADER (Tordesillas and How 2021) proposes a decentralized asynchronous planner that ensures safety in dynamic environments by optimizing separating planes and employing the MINVO basis for tighter trajectory representations. Although these methods demonstrate excellent theoretical completeness, this layered architecture faces significant challenges in complex, constrained real-world environments. The core issue with these approaches is the cascaded structure of perception, planning, and control, which leads to error accumulation and challenges real-time optimization in high-dimensional state spaces.

Learning-based methods: In recent years, learning-based methods have directly mapped raw sensory inputs into control commands. Some approaches employ supervised learning for training, while others utilize reinforcement learning to explore optimal strategies. For example, (Loquercio et al. 2021) employed a supervised learning paradigm to train visual-motion strategies by mimicking expert algorithms possessing privileged information, thereby enabling high-speed autonomous flight of UAV in complex outdoor environments. Policy gradient methods, such as PPO, are widely used in multi-UAV coordination due to their training stability. (Han, Chen, and Hao 2020) proposed a collaborative navigation framework based on PPO, which achieves policy sharing among multiple robots through target allocation and dynamic randomization training, effectively mitigating performance degradation from simulation to real-world environments. Actor-Critic methods have made significant breakthroughs in continuous control. TD3 (Fujimoto, Hoof, and Meger 2018) significantly improves stability through three enhancements: Clipped Double-Q Learning, delayed policy updates, and target policy smoothing. SAC (Haarnoja et al. 2018) provides optimal exploration and robustness. GTrXL-SAC (Huang et al. 2025) introduced Gated Transformer-XL to implement self-attention mechanisms and multimodal fusion, converging 20% faster than PPO and SAC in AirSim. YOPO (Lu et al. 2024) integrates perception and mapping, front-end path search, and back-end optimization into a single network, enabling direct prediction of trajectory deviations based on motion primitives without explicit mapping. This approach introduces a novel unsupervised learning strategy called guidance learning, which provides numerical gradients as guidance for training.

Semantic-Aware Four-buffer Experience Replay with Soft Actor-Critic(SaferSAC)

The proposed SaferSAC autonomous obstacle avoidance framework for UAVs is shown in Figure 1. In the perception module, we designed a depth-semantic fusion method for 3D obstacle detection. After acquiring the 3D bounding box of an obstacle, Kalman filtering differentiates between static and dynamic obstacles, with feature extraction performed separately for each category. During the reinforcement learning training phase, we designed four specialized buffers based on experience type and importance. At the decision-making level, we dynamically adjust the action space boundaries for horizontal velocity and yaw rate by solving constrained optimization problems. The following sections will elaborate on each component of SaferSAC.

Markov Decision Process Formulation

The UAV navigation task is defined as a Markov Decision Process (MDP), represented as a tuple (S, A, P, R, γ). Here, S denotes the state space, A represents the action space, $P(s_{t+1}|s_t, a_t)$ is the state transition function, $R(s_t, a_t)$ is the reward function. γ serves as the discount factor for future rewards. The objective of the DRL agent is to learn an optimal policy $\pi^*(a_t|s_t)$ that maximises the expected cumulative reward.

State Space S: Within the framework shown in Figure 1, the state in this paper is composed of four concatenated components: the UAV's own state features $S_{self} \in \mathbb{R}^{1\times 6}$,

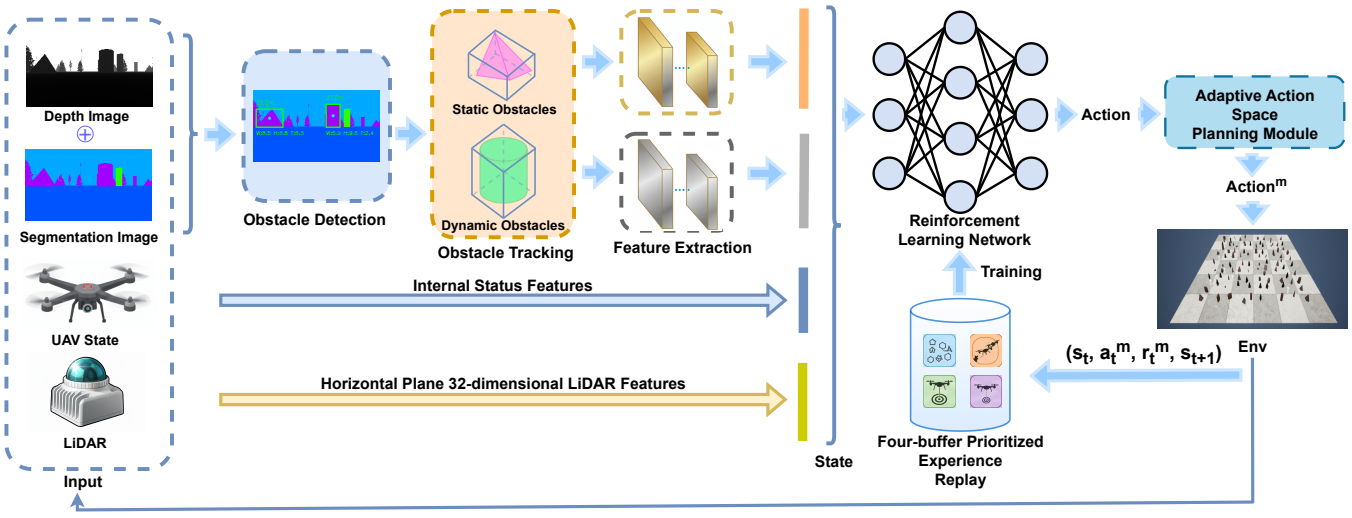


Figure 1: The overview of the proposed SaferSAC framework.

point cloud data features from the horizontal plane LiDAR $S_{lidar} \in \mathbb{R}^{1\times32}$, dynamic obstacle features $S_{dyn} \in \mathbb{R}^{1\times16}$, and static obstacle features $S_{stat} \in \mathbb{R}^{1\times32}$.

We uniformly partition the $180°$ field of view directly ahead of the UAV (from $-90°$ to $+90°$) into 32 non-overlapping angular intervals, each with an angular resolution of $5.625°$. For the 3D LiDAR point cloud within each angular interval, we select the distance to the point closest to the UAV as the representative value for that interval and normalize the distance to the range $[0, 1]$. This process yields a 32-dimensional feature vector.

The raw data for dynamic obstacles is represented as a two-dimensional array of size $N_\mathrm{d} \times 8$, where N_d denotes the predefined number of dynamic obstacles. Each row corresponds to the characteristics of a single dynamic obstacle, consisting of eight dimensions: the obstacle's 3D directional unit vector relative to the UAV, its distance from the UAV, the magnitude of its velocity relative to the UAV, and its dimensions. A two-layer convolutional neural network is then used to extract high-level semantic features from the array, resulting in a 16-dimensional feature vector.

The raw data for static obstacles is represented as a two-dimensional array of size $N_s \times 7$, where N_s denotes the predefined number of static obstacles. Each row corresponds to the features of a single static obstacle, consisting of seven dimensions: the 3D unit direction vector of the obstacle relative to the UAV, the distance of the obstacle from the UAV, and the dimensions of the obstacle. A two-layer convolutional neural network is then used to extract a 32-dimensional feature vector.

Action Space A: We use a continuous action space, where the action output by the policy network is defined as $a = (v_{xy}, v_z, \omega)$. Specifically, $v_{xy} \in [v_{xy,\mathrm{base,min}}, v_{xy,\mathrm{base,max}}]$, $v_z \in [-v_{z,\max}, v_{z,\max}]$, and $\omega \in [-\omega_{\mathrm{base,max}}, \omega_{\mathrm{base,max}}]$. v_{xy} represents the UAV's velocity in the x-y plane, v_z denotes the UAV's velocity along the z-axis, and ω represents the UAV's yaw angular velocity. The action a is then mapped to a^m, which serves as the UAV's actual control command.

Reward R: To achieve the dual objectives of obstacle avoidance and navigation within the proposed framework, the reward function consists of two components: continuous rewards and constant rewards. The reward function used in this study is as follows:

$$R = \begin{cases} 10 & if\ goal\ reached, \\ -20 & if\ crashed, \\ -10 & if\ outside\ the\ workspace, \\ R_c & otherwise. \end{cases} \tag{1}$$

The continuous reward function R_c consists of five components, as follows:

- **Goal reaching reward**: $R_{goal} = \frac{d_{t-1}-d_t}{D_{total}}$, where d_t is the Euclidean distance to the goal at time t, and D_{total} is the initial distance to the target. This component encourages the UAV to reduce the distance to the target.
- **Collision avoidance punishment**: $R_{col} = \begin{cases} 1 - \mathrm{clip}\left(\frac{d_{obs}-d_{crash}}{5}, 0, 1\right) & if\ d_{obs} < 10, \\ 0 & otherwise, \end{cases}$ where d_{obs} is the minimum distance to obstacles and d_{crash} is the collision boundary. This factor encourages the UAV to maintain a safe distance from obstacles.
- **Position punishment**: $R_{pose} = \mathrm{clip}\left(\frac{|P_{xy}-G_{xy}|}{10}, 0, 1\right) + 0.5 \cdot \mathrm{clip}\left(\frac{z-z_g}{5}, 0, 1\right) + \left(\frac{|z|}{z,max}\right)^2$, where P_{xy} represents the current position of the UAV, Gxy denotes the straight line formed by the starting point and the target point. This component prevents unexpected positions.
- **Velocity punishment**: $R_{vel} = \left(\frac{|v_z|}{v_{z,max}}\right)^2 + \frac{|\omega|}{\omega_{max}}$. This component optimizes action efficiency while penalizing excessive speed.
- **Yaw angle error punishment**: $R_{yaw} = \frac{|\psi|}{90}$. This com-

ponent helping the UAV maintain the correct heading relative to the target.

R_c is defined as the weighted sum of these five components, with α_1 to α_5 representing the weighting factors for each component:

$$R_c = \alpha_1 R_{goal} - \alpha_2 R_{col} - \alpha_3 R_{pose} - \alpha_4 R_{vel} - \alpha_5 R_{yaw}. \tag{2}$$

3D Obstacle Detection Module

The lightweight depth-semantic fusion method proposed here achieves precise 3D localization of both static and dynamic obstacles with low latency by integrating complementary information from depth cameras and semantic segmentation. This module is designed as a universal, platform-independent solution that can run on any robotic navigation system equipped with a depth camera (such as the RealSense D435i). We use depth cameras in the environment to acquire two types of depth images: Depth Perspective and Depth Planar. Depth Perspective represents the Euclidean distance from each pixel to the camera center, while Depth Planar denotes the perpendicular projection distance of each pixel along the camera's optical axis.

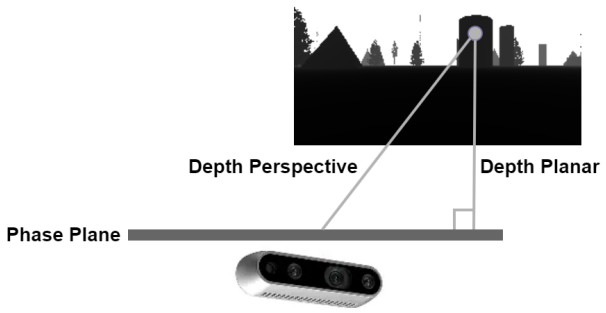


Figure 2: Geometric relationships between different distance information.

The coordinates of the central pixel (C_x, C_y) are $(\frac{W-1}{2}, \frac{H-1}{2})$. For each pixel (u, v) in the depth image, the offset relative to the image center is calculated:

$$\begin{aligned} y_{off} &= u - C_x, \\ z_{off} &= v - C_y. \end{aligned} \tag{3}$$

Next, calculate the Euclidean distance between the pixel and the image center: $r = \sqrt{{y_{off}}^2 + {z_{off}}^2}$. Given the distance information for each pixel in the depth perspective image d_{persp} and the depth plane image d_{pla}, and based on the geometric relationships shown in Figure 2, we can calculate the 3D position of a pixel in the camera coordinate system:

$$\begin{aligned} x_{cam} &= d_{pla}, \\ y_{cam} &= \sqrt{d_{persp}^2 - d_{pla}^2} \cdot \tfrac{y_{off}}{r}, \\ z_{cam} &= \sqrt{d_{persp}^2 - d_{pla}^2} \cdot \tfrac{z_{off}}{r}. \end{aligned} \tag{4}$$

Inverting z_{cam} yields the NED coordinates in the camera coordinate system.

Each pixel's grey value in the semantically segmented image represents a unique object identifier. Connected component analysis is performed on the segmented image to group pixels belonging to the same object into an obstacle candidate. This grouping method based on semantic segmentation has two advantages: first, it naturally decomposes scenes into distinct objects, preventing the misclassification of different objects as a single obstacle; second, it effectively filters out non-obstacle categories, such as ground and sky, reducing false detections.

Semantic segmentation allows us to obtain masks for different obstacles. However, noise points and edge uncertainties in depth measurements can significantly affect bounding box accuracy. To address this issue, inspired by (Xu et al. 2023), we introduce a robust estimation method that combines the interquartile range (IQR) and median absolute deviation (MAD). IQR and MAD are established statistical methods. Our contribution lies in innovatively applying this robust statistical filtering process to this depth-semantic fusion method, thereby effectively filtering depth noise at object edges. This approach operates in two stages: first, global coarse filtering using IQR, followed by local fine filtering with MAD, achieving efficient outlier suppression.

Given the set of valid depth values within the obstacle mask, we first calculate the first quartile Q_1 and third quartile Q_3, then compute the interquartile range: $IQR = Q_3 - Q_1$. Using the IQR, we initially filter out extreme outliers. We then apply the MAD method for further noise removal. MAD is defined as the median of the absolute deviations of data points from the median value. We compute MAD based on the median depth value d_{med} within the mask region M_{reg}:

$$MAD = median(|d_i - d_{med}|),\ d_i \in M_{reg}. \tag{5}$$

Based on the MAD value, we can determine the valid range of depth values, $d_{\min}$ and d_{max}. Ultimately, as shown in Figure 3, the width, height, and thickness of the obstacle can be calculated using the minimum depth $d_{\min}$ and maximum depth d_{max} in the corresponding direction, thereby obtaining the obstacle's 3D bounding box.

Four-buffer Prioritized Experience Replay

Our four-buffer architecture consists of the following four specialized experience storage units. Each experience in the buffer units is assigned a priority value based on the temporal difference (TD) error. The four buffer are defined as follows:

- **Main Buffer**: Capacity C, storing all types of experience.
- **Obstacle Buffer**: Stores the experience (s, a, r, s') of UAV approaching obstacles, with a capacity of $\frac{C}{4}$. When the minimum distance between the UAV and an obstacle falls below the threshold θ_{obs} or a collision occurs, the corresponding experience is added to this buffer.
- **Success Buffer**: Stores the experience (s, a, r, s') from episodes that end in a "goal achieved" status, with a capacity of $\frac{C}{8}$.

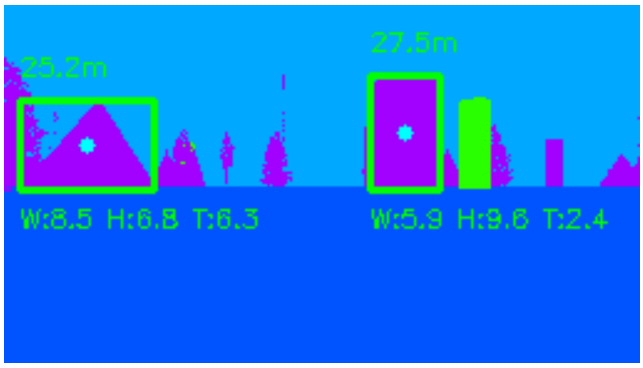


Figure 3: The 3D bounding box detected by the obstacle detection module. The camera's detection depth is 30 meters. The figure above the bounding box indicates the obstacle's current distance from the UAV in depth, while the figures below show the obstacle's width, height, and thickness (in meters).

- **Precision Buffer**: Stores experiences (s, a, r, s') when the UAV approaches the target point, with a capacity of $\frac{C}{8}$. When the UAV's distance to the target point falls below the threshold θ_{prec} without a collision, the experience is added to this buffer.

In this article, θ_{obs} is set to 8 meters, and θ_{prec} is set to 10 meters.

Optimization-Based Adaptive Action Space Planning Module

Due to the black-box nature of neural networks and the limitations of fixed action spaces, we propose an optimization-based adaptive action space planning method. The method dynamically adjusts the boundaries of the action space by performing real-time analysis of obstacles surrounding the UAV based on LiDAR perception data.

The LiDAR provides 180° scanning data across the horizontal plane, divided into two zones: the left zone (-90° to 0°) and the right zone (0° to $+90^\circ$). For each zone, we track two key metrics: the current minimum distance $d_{\min}$ to any obstacle and the current obstacle density ρ. The density is defined as the number of laser points P_{thr} within the distance threshold range, weighted by the inverse of the average distance d_{avg} to obstacles: $\rho = \frac{P_{thr}}{d_{avg}}$. Using these metrics, we calculate the obstacle density ρ_{left} for the left region and ρ_{right} for the right region. This information helps the system steer towards the direction with fewer obstacles. For example, when there are many obstacles on the left, the system directs the UAV to turn right. This enhances the interpretability of the UAV's actual control actions.

Based on the real-time analysis by the LiDAR, we model the computation of the adaptive action space as a constrained optimization problem. The optimization variables include the upper bound $v_{xy,\max}$ for the horizontal velocity action space, and the lower bound $\omega_{\min}$ and upper bound $\omega_{\max}$ for the yaw rate action space, denoted as $x = [v_{xy,\max}, \omega_{\min}, \omega_{\max}]$. The objective function aims to maximize obstacle avoidance safety. We define the objective function as follows:

$$J(x) = \begin{cases} (r_v + r_\psi) \cdot \varphi & if\ d_{\min} \le \theta_{obs}, \\ 0 & otherwise. \end{cases} \tag{6}$$

When $d_{\min}$ is less than or equal to the obstacle distance threshold θ_{obs}, an obstacle factor $\varphi = 1 - \frac{d_{\min}}{\theta_{\text{obs}}}$ is introduced. r_v represents the reward for reducing horizontal velocity. This encourages the UAV to decrease horizontal velocity in its action space:

$$r_v = \frac{v_{xy,base,\max} - v_{xy,\max}}{v_{xy,base,\max}}. \tag{7}$$

r_ψ is the steering adjustment reward, dynamically adjusting the steering range based on the obstacle density on either side:

$$r_\psi = \begin{cases} \frac{-\omega_{base,\max} - \omega_{\min}}{\omega_{base,\max}} & if\ \rho_{left} < \rho_{right}, \\ \frac{\omega_{\max} - \omega_{base,\max}}{\omega_{base,\max}} & otherwise. \end{cases} \tag{8}$$

In summary, the adaptive action space planning method proposed in this paper is formulated as the following constrained optimization problem:

$$\begin{aligned} \max_x \ & J(x), \\ s.t. \ & v_{xy,\max} - v_{xy,base,\min} > 0 \\ & \omega_{\max} - \omega_{\min} > 0 \\ & v_{xy,\max} \le \tfrac{d_{\min} - \varepsilon}{\Delta t} \\ & 0.5 \cdot v_{xy,base,\max} \le v_{xy,\max} \le v_{xy,base,\max} \\ & -2.0 \cdot \omega_{base,\max} \le \omega_{\min} \le -\omega_{base,\max} \\ & \omega_{base,\max} \le \omega_{\max} \le 2 \cdot \omega_{base,\max}. \end{aligned} \tag{9}$$

The first two constraints ensure that the upper bound of the optimized action space exceeds the lower bound. The last three constraints impose fundamental range limitations, ensuring that the optimized action space stays within a reasonable operational envelope, satisfying the UAV's physical constraints. The third constraint is critical: to maximize collision avoidance, the UAV's maximum displacement within a time step Δt must be less than the current minimum distance $d_{\min}$ to obstacles, minus a safety margin ε. The safety margin is set to 1 meter.

Given the original action space A, after obtaining the optimized action space A^m, we map the action a output by the reinforcement learning policy into A^m. Inspired by (Stolz et al. 2024), we use a ray-mapping method. However, unlike their approach, our method independently maps each dimension of the action space, making it suitable for scenarios where UAV action dimensions have low coupling. Since the horizontal plane LiDAR cannot directly detect obstacles in the vertical direction, the mapping process is applied only to a_0 (xy-axis velocity) and a_2 (yaw angular velocity). We use the following mapping function $f(a_i)$ to transform action a_i into a_i^m:

$$a_i^m = f(a_i) = c_i^m + \frac{\lambda_{A_i^m}}{\lambda_{A_i}}(a_i - c_i),\ i \in \{0, 2\}. \tag{10}$$

Here, c_i and c_i^m denote the centers of action spaces A_i and A_i^m, respectively, while λ_{A_i} and $\lambda_{A_i^m}$ represent the distances

from c_i and c_i^m to their respective action space boundaries. For the one-dimensional action space $\lambda_{A_i^m}$, $f(a_i)$ is bijective. Using the properties of bijection, we can deduce that:

$$\nabla_\theta \log \pi_\theta^m(a^m|s) = \nabla_\theta \log \pi_\theta(a|s). \tag{11}$$

Action a^m from $\pi_\theta^m(a^m|s)$ is obtained by ray-mapping the action a sampled from the original policy, with a^m serving as the final action command for controlling the UAV. Therefore, the policy gradient formula using ray-mapping is:

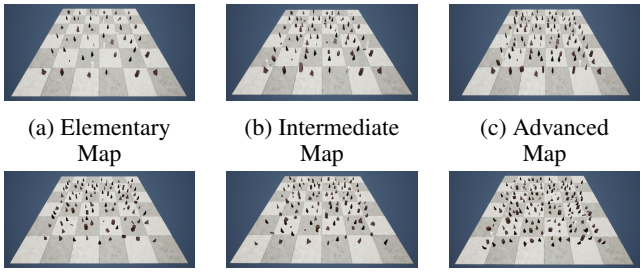
$$\nabla_\theta J(\pi_\theta^m(a^m|s)) = E_{\pi_\theta^m}[\nabla_\theta Q(s, a^m) - \alpha \nabla_\theta \log \pi_\theta(a|s)]. \tag{12}$$

In other words, the policy gradient after ray-mapping can directly use the logarithmic gradient of the original policy, requiring only the replacement of the action with a^m when computing the value function. This allows our module to be directly integrated into existing SAC frameworks without substantial modifications to the algorithmic logic.

Experiments and Results

Simulation Environment and Experimental Setup

To evaluate the proposed SaferSAC method, we built an AirSim simulation environment based on Unreal Engine 4.27 and compared it with two state-of-the-art benchmarks, YOPO (Lu et al. 2024) and EGO-Planner(Zhou et al. 2020), as well as baseline algorithms SAC and TD3. As shown in Figure 4, we created three training scenarios and three validation scenario. In the training scenarios, the environment was categorized into elementary, intermediate, and advanced map levels based on the nums of static and dynamic obstacles. As the map difficulty increased, the number of obstacles grew progressively. Validation map 1 is a purely static environment containing 100 static obstacles. Validation map 2 contains 113 dynamic and static obstacles (equivalent in difficulty to advanced maps). Validation map 3 contains 175 dynamic and static obstacles. All maps had a uniform size of $264m \times 264m$.



(a) Elementary Map (b) Intermediate Map (c) Advanced Map

(d) Validation Map 1 (e) Validation Map 2 (f) Validation Map 3

Figure 4: Training and validation environment. The dynamic white cylinder depicted in the image represents a high-speed dynamic obstacle.

The initial action space for the UAV is set as follows: $v_{xy} \in [0.5, 5], v_z \in [-2, 2], \omega \in \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]$. Expanding the angular velocity action space allows the UAV's angular velocity limit to reach $\frac{\pi}{3}$/s, which is within the physical constraints for maximum angular velocity imposed by the PX4 UAV. At the start of each episode, the UAV's initial position is set at coordinates $(0, 0, 5)$ with a randomly generated yaw angle. The target position is randomly generated on the boundary of a rectangular region R, defined as: $R = \{(x, y)|\ x \in [-120, 120],\ y \in [-120, 120]\}$. After completing a reinforcement learning episode (reaching the destination, collision, or exceeding the working area), the simulation environment is reset, and the UAV returns to the starting point to begin a new episode.

During training, we use a curriculum learning approach, gradually advancing the UAV from elementary to intermediate and advanced stages. After 120,000 training steps in the elementary stage, the UAV transitions to the intermediate stage. After 50,000 steps in the intermediate stage, the UAV moves to the advanced stage, concluding with 50,000 training steps in this final phase. During the first 10,000 steps of training in each environment, randomly sampled actions control the UAV's flight to enrich buffer experience. Learning and updating of the decision module begin only after 10,000 steps. All experiments were conducted on a computer with an Intel i7 13th CPU and an NVIDIA GeForce RTX 4060 GPU. The Adam optimizer was used, with an initial learning rate of 3×10^{-4} and a reward discount factor γ set to 0.99.

Performance Metrics

We use the cumulative reward and the Actor network's loss value for evaluation during the UAV's autonomous navigation training phase. The average cumulative reward directly quantifies the UAV's overall performance in navigation tasks. Furthermore, trends observed during training offer an intuitive assessment of the algorithm's convergence. The Actor network loss value characterizes the stability and optimization efficiency of policy learning. Its convergence trend and fluctuation amplitude indicate whether the control policy optimization process is reasonable and whether gradient updates are effective.

During the validation phase, we performed thirty navigation tasks across three validation scenarios and evaluated each algorithm's performance using the following metrics:

- **Average Cumulative Reward (ACR):** Calculate the average cumulative reward obtained across all episodes.
- **Success Rate (SR):** Percentage of episodes where the UAV successfully reaches the target within the specified number of steps without a collision.
- **Collision Rate (CR):** Percentage of episodes in which the UAV collides with obstacles.
- **Navigation Time (NT):** Average time taken for the UAV to successfully navigate to the target without a collision.
- **Success-Weighted Path Length (SPL):** The calculation formula is $SPL = \frac{1}{N} \sum_{i=1}^{N} s_i \frac{l_i}{\max(l_i, p_i)}$. Here, N denotes the total number of navigation tasks, $S_i \in \{0, 1\}$ indicates whether the i-th navigation task was successful, l_i represents the ideal shortest path length, i.e., the Euclidean shortest distance between the starting point and the target point, and p_i denotes the actual path length navigated by the UAV. Generally, p_i is always greater than l_i.

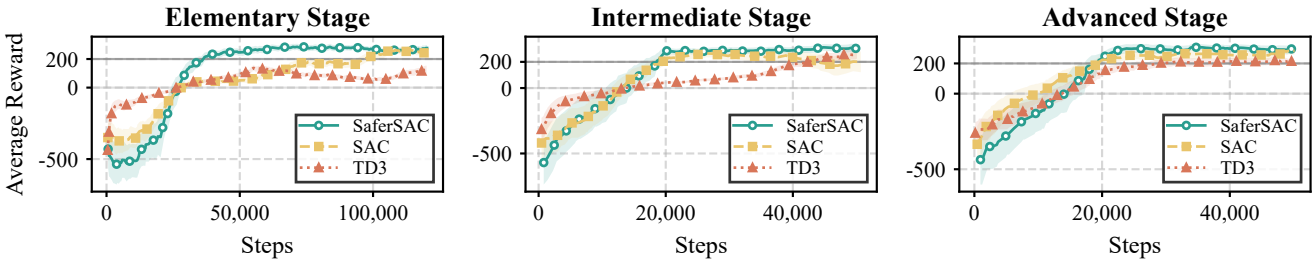


Figure 5: The cumulative rewards accumulated by SaferSAC, SAC, and TD3 during the training period. The shaded area represents the standard deviation.

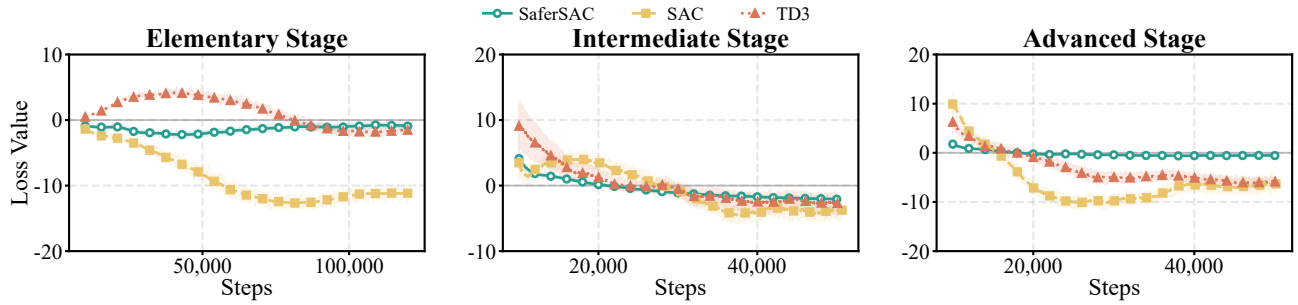


Figure 6: Comparison of actor network loss during training across SaferSAC, SAC, and TD3. The shaded area represents the standard deviation.

Experimental Results

Validation Scenarios	Method	ACR	SR(%)	CR(%)	NT	SPL
Map 1	SaferSAC	**324.72**	**96.67**	**3.33**	**29.27**	0.81
	YOPO	N/A	96.67	3.33	30.03	**0.95**
	EGO-Planner	N/A	46.67	53.33	39.61	0.41
	SAC	311.90	80.00	20.00	35.21	0.70
	TD3	301.97	76.67	13.33	32.89	0.69
Map 2	SaferSAC	**331.33**	**93.33**	**6.67**	30.92	**0.77**
	YOPO	N/A	50.00	50.00	**29.37**	0.49
	EGO-Planner	N/A	N/A	N/A	N/A	N/A
	SAC	313.73	63.33	36.67	35.33	0.56
	TD3	247.23	56.67	16.67	34.14	0.48
Map 3	SaferSAC	**315.24**	**90.00**	**10.00**	31.75	**0.75**
	YOPO	N/A	43.33	56.67	**29.31**	0.43
	EGO-Planner	N/A	N/A	N/A	N/A	N/A
	SAC	305.65	66.67	33.33	35.42	0.59
	TD3	270.27	60.00	36.67	34.89	0.52

Table 1: Comparison of metrics across different methods in three validation scenarios.

Training Performance As shown in Figure 5, during the elementary phase, TD3 suffers from local optima, while SAC fails to match the final convergence of SaferSAC. In contrast, SaferSAC undergoes an exploratory phase in the initial period, but its performance surges rapidly after around 25,000 steps, stabilizing at a high reward value. Entering the intermediate stage, SaferSAC showed exceptional adaptability, converging rapidly to the highest reward level. In the advanced environment stage, TD3's performance decreased further. While SAC maintained a certain level of performance, its rewards remained lower than those achieved by SaferSAC. As shown in Figure 6, SaferSAC demonstrates exceptionally high stability and smoothness across all stages. The loss curve of SaferSAC consistently converges smoothly near zero with minimal variance. This indicates exceptionally robust policy updates, mainly due to the adaptive planning action space proposed here. By dynamically constraining the action space through the optimization problem, the algorithm limits the Actor's output of extreme actions, preventing the policy network from undergoing drastic gradient updates on complex value surfaces.

Validation Performance Table 1 presents the performance metrics of the three algorithms in the validation scenario. Maps 1 and 3 provide varying levels of complexity to comprehensively evaluate the algorithm's generalization capability. EGO-planner displays N/A in maps 2 and 3 because it is designed for static environments and frequently gets stuck in replanning, performing poorly in dynamic environments. Experimental results show that SaferSAC significantly outperforms the baseline algorithms across all key metrics under various environmental conditions. Across all scenarios, SaferSAC consistently maintains the highest average cumulative reward compared to SAC and TD3. SaferSAC has demonstrated exceptional security, achieving the highest success rate across all maps. Although YOPO achieves high success rates on static maps, its performance plummets on dynamic maps due to the lack of explicit mod-

eling for moving obstacles, resulting in a sharp increase in collision rates. SPL serves as the gold standard for evaluating navigation quality, considering both success rate and path length. Although SaferSAC's navigation time on Maps 2 and 3 was slightly slower than YOPO's, it maintained top-tier and stable SPL scores across all maps, demonstrating an excellent balance between safety and path efficiency.

Ablation Study To validate the individual contributions of each component within our proposed method, we conducted a comprehensive ablation study on validation map 2. Specifically, we systematically removed each component from SaferSAC. First, for the variant lacking the depth-semantic fusion 3D obstacle detection module (denoted as SaferSAC w/o SI), we relied exclusively on depth maps to generate U-depth (Lin, Zhu, and Alonso-Mora 2020) for 3D obstacle bounding boxes. Second, to evaluate the four-buffer prioritized experience replay, the SaferSAC w/o FPER configuration substituted this component with the standard prioritized experience replay method (Schaul et al. 2015). Finally, the optimization-based adaptive action space planning method was omitted entirely in the SaferSAC w/o AASP setup.

Method	ACR	SR(%)	CR(%)	NT	SPL
SaferSAC	**331.33**	**93.33**	**6.67**	30.92	**0.77**
SaferSAC w/o SI	319.65	90.00	10.00	31.99	0.71
SaferSAC w/o FPER	309.35	83.33	10.00	32.77	0.66
SaferSAC w/o AASP	301.58	63.33	36.67	**30.91**	0.54

Table 2: Ablation studies in validation scenario 2.

Table 2 summarizes the quantitative results of the ablation study. First, the Optimization-based Adaptive Action Space Planning (AASP) module is critical for system safety. Excluding this module caused the most severe performance degradation, with the Success Rate (SR) dropping significantly from 93.33% to 63.33%. This result suggests that simple end-to-end policies often lack the interpretability required to guarantee safety in complex environments. In contrast, our AASP module enables the UAV to decelerate and adjust its heading in hazardous zones, thereby ensuring safety. Second, the Four-buffer Prioritized Experience Replay (FPER) module significantly enhances sample efficiency and policy robustness. Replacing FPER with the standard PER method resulted in a 10% decrease in SR, while the SPL dropped to 0.66. Finally, the depth-semantic fusion module further enhances obstacle detection accuracy and obstacle avoidance success rates. Compared to the depth-only baseline, incorporating semantic information improved SR by 3.33% and increased SPL from 0.71 to 0.77.

Sensitivity Analysis of the Obstacle Distance Threshold To evaluate the impact of obstacle distance threshold θ_{obs} on the navigation performance of the SaferSAC algorithm, we conducted a comprehensive parameter sensitivity analysis. We adjusted this parameter from $5m$ to $10m$ and performed thirty navigation tasks on validation map 2, assessing performance through key metrics. As shown in Table 3, the success rate and collision rate are highly sensitive to the value of θ_{obs}. When the threshold is small, the UAV lacks sufficient reaction space to avoid complex obstacles, resulting in a low success rate. When θ_{obs} was increased to $8m$, the algorithm achieved its highest success rate (93.33%). However, further increasing the threshold leads to a decrease in success rate. Furthermore, the table indicates that at $\theta_{obs} = 8m$, the SPL reaches a peak value of 0.77, indicating that the UAV successfully reached the target point while maintaining extremely high path efficiency. Correspondingly, the average cumulative reward (ACR) also peaks simultaneously at 331.33 at this threshold. This substantial gain reflects the agent's ability to consistently execute high-quality trajectories, thereby demonstrating the algorithm's exceptional robustness and policy stability in complex environments. Additionally, the navigation time (NT) data reveals that a larger threshold prompts the UAV to adopt a more cautious flight trajectory, thereby monotonically increasing the overall navigation time. After comprehensive evaluation of all metrics, the algorithm performs best when $\theta_{obs} = 8m$.

θ_{obs} (m)	ACR	SR (%)	CR (%)	NT	SPL
5	231.02	50.00	40.00	27.40	0.36
6	264.82	66.67	26.67	27.76	0.49
7	269.95	70.00	23.33	29.31	0.53
8	331.33	93.33	6.67	30.92	0.77
9	306.04	83.33	13.33	31.57	0.71
10	316.73	76.67	20.00	32.25	0.65

Table 3: Sensitivity analysis of obstacle distance threshold θ_{obs} on navigation performance.

Conclusion

This paper proposes SaferSAC, a deep reinforcement learning framework for UAV obstacle avoidance that effectively addresses challenges related to safety and sample efficiency in complex dynamic environments. Experimental results demonstrate SaferSAC's superiority. In quantitative evaluations across all validation maps, SaferSAC achieved the highest average cumulative reward and success rate while maintaining a high SPL score. The ablation analysis confirmed the critical role of each component. Among them, the adaptive action space planning module plays a critical role in ensuring system safety. However, SaferSAC exhibits slightly slower navigation time than YOPO in certain scenarios. Therefore, in future research, we will further explore how to enhance navigation efficiency while maintaining security.

References

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep rein-

forcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. Pmlr.

Han, R.; Chen, S.; and Hao, Q. 2020. Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 448–454. IEEE.

Huang, J.; Cui, Y.; Xi, G.; Bai, S.; Li, B.; Wang, G.; and Neretin, E. 2025. GTrXL-SAC-Based Path Planning and Obstacle-Aware Control Decision-Making for UAV Autonomous Control. *Drones*, 9(4): 275.

Karaman, S.; and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7): 846–894.

Kaufmann, E.; Bauersfeld, L.; Loquercio, A.; Müller, M.; Koltun, V.; and Scaramuzza, D. 2023. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976): 982–987.

Li, H.; Wang, H.; Feng, C.; Gao, F.; Zhou, B.; and Shen, S. 2023. Autotrans: A complete planning and control framework for autonomous uav payload transportation. *IEEE Robotics and Automation Letters*, 8(10): 6859–6866.

Lin, J.; Zhu, H.; and Alonso-Mora, J. 2020. Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2682–2688. IEEE.

Loquercio, A.; Kaufmann, E.; Ranftl, R.; Müller, M.; Koltun, V.; and Scaramuzza, D. 2021. Learning high-speed flight in the wild. *Science Robotics*, 6(59): eabg5810.

Lu, J.; Zhang, X.; Shen, H.; Xu, L.; and Tian, B. 2024. You only plan once: A learning-based one-stage planner with guidance learning. *IEEE Robotics and Automation Letters*, 9(7): 6083–6090.

Mellinger, D.; and Kumar, V. 2011. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, 2520–2525. IEEE.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Stolz, R.; Krasowski, H.; Thumm, J.; Eichelbeck, M.; Gassert, P.; and Althoff, M. 2024. Excluding the irrelevant: Focusing reinforcement learning through continuous action masking. *Advances in Neural Information Processing Systems*, 37: 95067–95094.

Tordesillas, J.; and How, J. P. 2021. MADER: Trajectory planner in multiagent and dynamic environments. *IEEE Transactions on Robotics*, 38(1): 463–476.

Xu, Z.; Zhan, X.; Xiu, Y.; Suzuki, C.; and Shimada, K. 2023. Onboard dynamic-object detection and tracking for autonomous robot navigation with rgb-d camera. *IEEE Robotics and Automation Letters*, 9(1): 651–658.

Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; and Zhao, H. 2024a. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10371–10381.

Yang, X.; Wang, C.; Liang, J.; Zhao, J.; Yue, K.; and Li, W. 2024b. Research on path planning in UAV-assisted emergency communication. *EURASIP Journal on Wireless Communications and Networking*, 2024(1): 86.

Zhou, B.; Xu, H.; and Shen, S. 2023. Racer: Rapid collaborative exploration with a decentralized multi-uav system. *IEEE Transactions on Robotics*, 39(3): 1816–1835.

Zhou, X.; Wang, Z.; Ye, H.; Xu, C.; and Gao, F. 2020. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2): 478–485.