UBG: An Unreal BattleGround Benchmark With Object-Aware Hierarchical Proximal Policy Optimization

Longyu Niu, Baihui Li, Xingjian Fan, Hao Fang, Jun Li, Junliang Xing, *Senior Member, IEEE*, Jun Wan, *Senior Member, IEEE*, and Zhen Lei, *Fellow, IEEE*

***Abstract*—The deep reinforcement learning (DRL) has made significant progress in various simulation environments. However, applying DRL methods to real-world scenarios poses certain challenges due to limitations in visual fidelity, scene complexity, and task diversity within existing environments. To address limitations and explore the potential ability of DRL, we developed a 3-D open-world first-person shooter (FPS) game called Unreal BattleGround (UBG) using the unreal engine (UE). UBG provides a realistic 3-D environment with variable complexity, random scenes, diverse tasks, and multiple scene interaction methods. This benchmark involves far more complex state-action spaces than classic pseudo-3-D FPS games (e.g., ViZDoom), making it challenging for DRL to learn human-level decision sequences. Then, we propose the object-aware hierarchically proximal policy optimization (OaH-PPO) method in the UBG. It involves a two-level hierarchy, where the high-level controller is tasked with learning option control, and the low-level workers focus on mastering subtasks. To boost the learning of subtasks, we propose three modules: an object-aware module for extracting depth detection information from the environment, potential-based intrinsic reward shaping for efficient exploration, and annealing imitation learning (IL) to guide the initialization. Experimental results have demonstrated the broad applicability of the UBG and the effectiveness of the OaH-PPO. We will release the code of the UBG and OaH-PPO after publication.**

Received 24 June 2024; revised 25 January 2025; accepted 27 April 2025. Date of publication 20 May 2025; date of current version 4 September 2025. This work was supported in part by Beijing Natural Science Foundation under Grant JQ23016 and in part by the Chinese National Natural Science Foundation Projects under Grant 62476273. *(Corresponding author: Jun Wan.)*

Longyu Niu is with the State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China, also with the School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing 100049, China, and also with Shanghai Radio Equipment Research Institute, Shanghai 201109, China.

Baihui Li, Hao Fang, Jun Li, Jun Wan, and Zhen Lei are with the State Key Laboratory of Multimodal Artificial Intelligence Systems (MAIS), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing 100049, China (e-mail: jun.wan@ia.ac.cn).

Xingjian Fan is with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications (NJUPT), Nanjing, Jiangsu 210023, China, also with Nanjing Artificial Intelligence Research of IA, Nanjing, Jiangsu 211135, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Nanjing, Jiangsu 211135, China.

Junliang Xing is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2025.3567001, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2025.3567001

***Index Terms*—Deep reinforcement learning (DRL), first-person shooter (FPS) benchmark, hierarchical reinforcement learning (HRL), imitation learning (IL).**

I. Introduction

DEEP reinforcement learning (DRL) has achieved significant progress in specific tasks within some game scenarios [1]. For example, DeepMind used the deep Q-network [2] to train an agent that exceeded human levels in the Atari game, JueWu-MC [3] completed the diamond mining task in MineCraft [4], which is also a difficult task for humanity. However, these game scenes and tasks differ significantly from the real world. Even in virtual highway environments [5], algorithms that perform well in driving tasks in games using DQN and DDAC for lane tracking may face challenges in real-world applications due to insufficient fidelity. Algorithmic research based on these benchmarks can only be applied to specific gaming tasks, and it is challenging to extend this further to real-life applications. Therefore, there is a pressing need within the RL research community for more challenging benchmarks.

As shown in Table I, we summarize some commonly used research benchmarks and highlight the limitations associated with them.

1) *Low-Visual Fidelity:* Some benchmarks only feature low-quality rendering textures and toy-like visual styles, such as Quake III and ViZDoom (shown in Fig. 1). This can pose challenges for applying certain detection and recognition methods developed on these benchmarks to real-life scenarios.
2) *Starved State-Action Space:* The low complexity of the scene and limited interaction with the environment lead to an insufficient state-action space. For example, ViZDoom only features 13 common key combinations and an average of 2000 frames per play [6]. As a result, the trajectory space created by these combinations [7] is relatively limited in scope. A larger state-action space can introduce more challenging tasks, providing a better platform for developing more intelligent DRL algorithms.
3) *Inconvenient Script Editing:* As indicated in the "scripting" column of Table I, certain benchmarks lack

TABLE I

OVERVIEW OF SOME BENCHMARKS: "MED." IS SHORTHAND FOR MEDIUM. "COMPL.", "INTER.," "T-D.," AND "GRAPH.," DENOTE "COMPLEXITY," "INTERACTION," "TOP-DOWN VIEW," AND "GRAPHICAL," RESPECTIVELY. THE VALUE OF TRAJECTORY SPACE IS ESTIMATED BASED ON THE NUMBER OF POSSIBLE ACTIONS AND THE GAME'S DURATION. "GAME COMPLEXITY" IS ASSESSED BY CONSIDERING THE DIVERSITY OF GAME MODES AND THE MAGNITUDE OF STATE-ACTION SPACE. AT THE SAME TIME "VISUAL REALISM" IS EVALUATED THROUGH THE PRECISION OF IMAGE SIMULATION AND THE FIDELITY OF THE PHYSICAL REPRESENTATION

Game	Engine	Scene Compl.	Scene Inter.	Game Compl.	Visual Realism	Screen Buffer	Depth Buffer	T-D. map	Graph. Editor	Free Assets	Scripting	System Require	Disk Space	Trajectory Space
ViZDoom 1993	ZDoom	Low	✗	Low	Low	✓	✓	✓	✗	✗	✓	Low	40MB	$13^{2000} \approx 10^{2200}$
Half-Life 2 2004	Source	High	✓	Med.	High	✗	✗	✗	✗	✗	✓	Med.	4.5GB	-
Quake III 1999	ioquake3	Low	✗	Low	Low	✓	✓	✗	✓	✗	✗	Low	70MB	-
Malmo 2016	Mojang	Med.	✓	Med.	Low	✓	✗	✗	✗	✗	✓	Low	40MB	-
Unreal Tournament 2014	UE4	High	✗	High	High	✓	✗	✗	✓	✓	✓	High	>10GB	-
CS:GO 2012	Source	Med.	✓	High	High	✗	✗	✓	✗	✗	✗	High	>10GB	-
Wild-Scav 2022	Unity	Med.	✗	Low	Med.	✓	✓	✗	✓	✓	✓	Med.	1.2GB	$30^{600} \approx 10^{900}$
UBG 2023	**UE5**	**High**	✓	**High**	**High**	✓	✓	✓	✓	✓	✓	**High**	**>10GB**	$100^{3000} \approx 10^{6000}$

Fig. 1. Illustrations of different game scenes. Compared with previous benchmarks, UBG exhibits higher visual realism and more complex scenes. (a) Quake III. (b) ViZDoom. (c) UBG.

support for convenient script editing. This makes it challenging for researchers to set up game components.

4) *No Access to Environment Information:* The lack of access to essential information, such as screen buffer, depth buffer, and top-down view map makes it challenging to develop more flexible and adaptive DRL algorithms that are better suited to real-world environments.
5) *Insufficiency of the Auxiliary Function:* Certain games may not be explicitly developed with research purposes in mind, which can result in a lack of adequate auxiliary functions. These functions, such as the recording function and access to more abstract information, are crucial for addressing complex control problems. The combination of these various factors presents a growing challenge in meeting the needs of the research community.

To address the above issues, we developed Unreal Battle-Ground (UBG), a 3-D open-world first-person shooter (FPS) benchmark. We utilize the most popular game engine, UE to develop UBG for its impressive rendering capabilities. As shown in Table I, the proposed UBG boasts higher visual fidelity, a larger trajectory space (over 100 discrete actions and 3000 frames per game), rich environment information access, numerous auxiliary features, and will be entirely open-source. Similar to the battle royale game setup, where multiple characters compete in an ever-shrinking play area until only one remains, UBG includes multiple subtasks, such as navigation, firing, and supply collection. In addition, the procedural content generation (PCG) technique was incorporated to generate special components, such as enemies, effectively enhancing the diversity of the game environment. UBG is specifically designed for agent learning in open-world environments and offers various scene interaction actions, such as breaking through windows, vaulting, and crawling. Possible scene changes brought by interaction provide more spaces for exploration, greater flexibility in task design, and the possibility of generalization with diverse scenarios. We believe UBG can effectively serve the research in open-world agent learning.

Based on UBG, we investigated several issues that RL faces in 3-D open-world environments, such as partial observability and sparse rewards [3]. As a result, we propose the object-aware hierarchically proximal policy optimization (OaH-PPO) to enable efficient explorations in UBG. OaH-PPO is a hierarchical RL framework with a novel imitation learning (IL) method. Inspired by Huang et al.'s [8] work, we also incorporate object detection and depth images to fully leverage the high-fidelity environment of UBG. In response to the

limitations on aiming accuracy and transferability posed by the supervised grid-like auxiliary input, we opt for high-resolution coupled depth detection maps as augmented input. To adapt to the dynamic nature of game situations, we incorporated potential-based intrinsic rewards that direct the actions of low-level workers. Furthermore, we propose the use of margin-clip PPO, an efficient method for utilizing human demonstrations (Demo.) in the learning process. This comprehensive approach ensures the robustness and adaptability of the network in various shooter games [9]. The contributions of this article are as follows.

1) We release UBG, a 3-D open-world FPS benchmark. It includes a high-fidelity environment that can be easily customized. Its intricate and varied game scenes will present novel challenges for the RL community.
2) We propose the OaH-PPO approach, which incorporates an object-aware module to augment input, intrinsic reward shaping to address reward sparsity, and the integration of demonstrations to guide the initialization.
3) The conducted experiments in UBG and ViZDoom showcase the control capabilities and generalization of our framework. Our model consistently outperforms existing algorithms in various game environments.

II. Related Works

A. Shooter Games

As indicated in Table I, the most classic FPS benchmark, ViZDoom [10], utilizes low-resolution textures while having the most lightweight architecture. Three Visual Doom AI Competitions (VDAIC) [11] were organized to promote the application of RL in shooter games [12], [13]. The augmented DRQN model [9] exploited both visual input and the game feature information to augment the network. Also, Bhatti et al. [6] used object detection and 3-D-scene reconstruction to extract different component information.

Half-life two lacks screen buffer access support. Although similar problems are shared by Unreal Tournament (UT) and CS:GO, Dawes [14] deemed UT a potential AI research test bed. Also, Pearce [15] employed large-scale behavioral cloning (BC) [16] to play CS:GO.

Quake III's scripting capabilities were greatly extended by DeepMind [17]. As a result, there has been a substantial amount of research built upon it in recent years [18], [19]. WILD-SCAV [20] is a recently launched environment that also has huge exploration spaces with over 30 unique actions. However, to reduce rendering pressure, the game significantly reduces the rendering batches, so the trajectory space in Table I is not considered very large. In addition, its toy graphics may not be conducive to expanding into the real world.

In contrast to classic arcade games (i.e., ALE [21] and StarCraft II [22]), FPS games feature 3D graphics and partially observable states [23]. Their flexible and complex action controls make them a more realistic research environment. Many modern 3D FPS game environments have emerged for visual RL agents [24]. Also, it is a pity that the latest FPS video games, such as PlayerUnknow's BattleGrounds (PUBG) and Grand Theft Auto (GTA), do not provide adequate application programming interfaces (APIs) to access their internal data and sometimes have licensing issues. In parallel with these efforts, we are investigating further advancements in the field of AI agent learning through the use of large-scale, open-world FPS games.

B. Reinforcement Learning

Hierarchical reinforcement learning (HRL) [25], [26], related to our work, improves learning efficiency by building and leveraging a hierarchical structure of cognitive and decision-making processes. Inspired by Dayan [27], Vezhnevets [28] proposed a feudal network capable of automatically discovering subgoals. Based on the concept of options [29], option-critic (OC) [30] extended the policy gradient theory to options for learning options that scale to vast domains. H-DQN [31] learned hierarchical work values at different time scales, while the MaxQ architecture [32] decomposed tasks by decomposing value functions. Barto [33] believed HRL provides a natural framework for incorporating principles of intrinsic motivation. Existing work [34], [35] primarily relies on sparse environmental feedback for task decomposition guidance. In contrast, our work suggests that organically combining intrinsic rewards with hierarchical structures through reward shaping can significantly improve training efficiency.

IL introduces additional information sources, such as expert demonstrations, to initialize or guide agent training. This effectively alleviates the problem of low sample efficiency in DRL. However, directly using BC can result in insufficient generalization. Dataset aggregation (DAgger) [36] was a more useful and efficient online IL method. Inverse reinforcement learning (IRL) [37] extracted a reward function from expert policy, which was then used for forward RL. Gupta [38] proposed a two-phase approach consisting of an IL stage and an RL phase. While existing studies on IL [39] predominantly employ off-policy algorithms, the integration of demonstrations into on-policy algorithms [40], facilitated by the use of additional regularization terms and clipping, can more naturally establish a balance between exploration and exploitation.

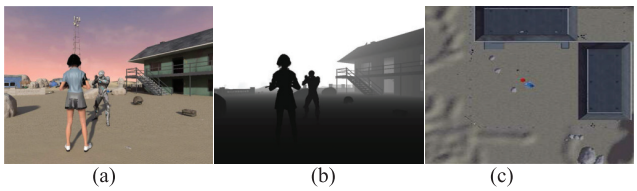
III. Preliminaries

The PPO [41] algorithm implements a direct clipping mechanism on the objective function utilized for policy gradients. This approach leads to a more conservative update strategy and simplifies the computational process. Let $r_t(\theta)$ be the probability ratio $\prod_{i=0}^{M-1}(\pi_\theta(a^{(i)}|s)/\pi_{\theta_{\text{old}}}(a^{(i)}|s))$. The standard PPO algorithm uses a ratio clip function as follows to discipline extreme changes to the policy:

$$\mathcal{J}_\pi^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{(s,a)\in\mathcal{B}_\pi}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{1}$$

where $\hat{A}_t(s,a) = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V$ is the advantage value computed by generalized advantage estimation (GAE) [42], and $\delta_t^V = R_t + \gamma V(s_{t+1}) - V(s_t)$.

DQfD [43] and DDPGfD [44] encounter limitations in fully leveraging expert trajectory learning, particularly when



(a) (b) (c)

Fig. 2. UBG provides access to the regular screen buffer, the depth buffer, and the top-down view map. (a) Regular screen. (b) Depth buffer. (c) Top-down view map.

there are few demonstrations available. Furthermore, the rewards obtained from demonstrations that deviate from the current environmental rewards may render them ineffective. To address this, Kang et al. [40] proposed an additional regularization term, denoted by $D_{\mathrm{JS}}(\pi_\theta, \pi_E)$, by leveraging the one-to-one correspondence between policy and occupancy measures. This term encourages exploration around the confidence region of the demonstration trajectory ξ_D, thereby demonstrating the feasibility and effectiveness of incorporating demonstrations into the policy algorithm PPO. However, the computation of this additional term, which is based on the Jensen-Shannon divergence, poses a challenge due to its complexity, leading to substantial computational costs, especially when dealing with large state-action spaces. Reflecting on the simplification of the PPO algorithm from the TRPO [45] algorithm, a first-order approximation of the regularization term is chosen, and simple additional terms are constructed using importance sampling and clipping techniques.

IV. Game Design

A. Advantages of UBG

One of the critical features of UBG is the ability to customize complex scenes easily. Utilizing the UE editor, a real-time graphical editor, individuals without programming can readily edit map scenes, set particular components (such as ammo and guns), and modify game settings (such as terminal conditions and rewards). With the support of its large community and a variety of free original assets, this mechanism can accommodate the diverse needs of researchers. In particular, it enables the creation of scenarios that are sufficiently challenging to evaluate the capabilities of learning algorithms effectively.

UBG is supported by UE's animation blueprint system, which offers a well-structured and comprehensive action sequence space. It allows agents to simulate human keyboard and mouse real-time control through various action combinations, enabling them to perform complex interactions within the scene. This also means that UBG can facilitate the testing of algorithms that involve continuous actions [46] or operate in continuous time [47]. In addition, multiple scene capture components provide information about the agent and its surrounding environment, including RGB maps with adjustable resolution, depth images, and top-down view maps, as depicted in Fig. 2. Access is also provided to various game variable information, such as the agent's current position and orientation, health, backpack information, playtime, and more. This information is critical for the agent's overall understanding of its environment. At the same time, UBG supports free switching between various perspectives. To enhance the presentation of high-fidelity digital humans and scenes, this article uses third-person perspective images, but subsequent experiments follow the convention of using the first-person perspective. We introduced the latest technologies, such as "PCG," and advantages applied in the four modules of UBG in the appendix: pawns (characters), maps, functions, and interactions.

B. Application Programming Interface

Refer to UnrealCV [48], UBG's server communicates with the client using TCP. The server utilizes UE's C++ API to access information about the virtual world. Meanwhile, the client is an integrated library containing both Python and C++ bindings that communicates with the server by sending commands and parsing responses. The communication protocol is defined by a set of Jycomm commands, with further details available in the Appendix. UBG is supported on major platforms, including Windows, Linux, and Mac, and provides complete control through its rich Python and C++ API. Game settings, such as game difficulty, can be directly configured through the profile. The game data logging feature also suggests the potential for applying reinforcement, imitation, and curriculum learning [49] algorithms. We have provided detailed information on the communication process in the Appendix.

C. Game Settings

UBG currently offers a variety of scenes, including desert, city, and forest, as well as two modes: training and deathmatch. In the training mode, you can experience the real physics of the UE, such as gun recoil, and pretrain agents to aim or collect supplies. In deathmatch mode, you will face a controllable number of enemies and must collect scattered supplies to defeat as many enemies as possible. Please refer to the Appendix for more details about game modes and map environment settings. In addition, the supplementary materials feature a demonstration video showcasing UBG.

Furthermore, it is essential to take into account the *computational costs* associated with high-fidelity environments such as UBG. In the Appendix, a thorough quantitative analysis is provided to offer a detailed understanding of these costs.

V. Algorithm Design

In consideration of the phases of the shooter games, we design a hierarchical network consisting of a controller and workers [50]. Also, we define n subtasks based on human cognitive patterns and structure demonstrations accordingly $(\mathcal{B}_D \rightarrow \{\mathcal{B}_{D_0}, \mathcal{B}_{D_1}, \ldots, \mathcal{B}_{D_{n-1}}\})$ [51]. Fig. 3 illustrates the architecture. Specifically, the controller determines abstract subtasks, or options, at a lower temporal resolution. The worker specified by the option executes basic game actions at a higher temporal resolution to fulfill the received subtasks. Following human habits, the action space is divided to allow

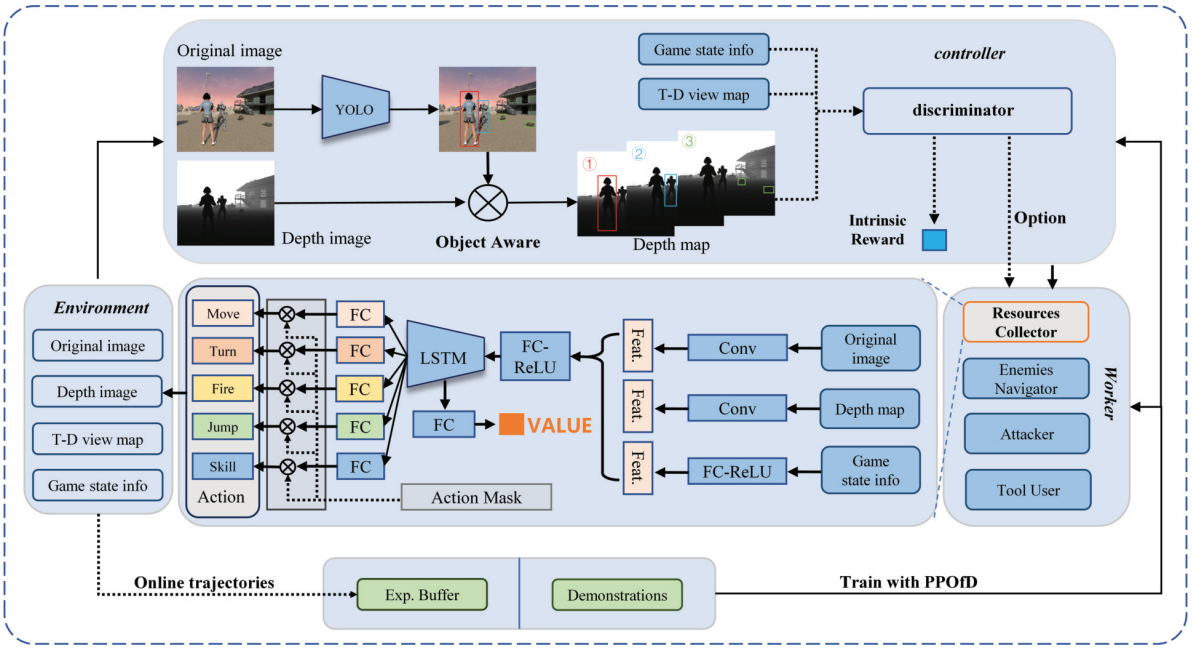


Fig. 3. Our proposed OaH-PPO architecture. The controller employs on-the-fly object detection to generate a depth mask, enabling complete object perception. It then makes "options" based on game information and calculates the corresponding intrinsic rewards. The selected worker interacts with the environment by choosing a set of orthogonal actions through the network. The online trajectories are deposited into the experience buffer and trained with the demonstration data using PPO from demonstrations (PPOfDs).

for the simultaneous execution of multiple actions. Furthermore, the environmental depth information is introduced to facilitate exploration, and a unique intrinsic reward based on potential function is designed for each worker. Last, a margin-clip PPO algorithm is proposed for the incorporation of human demonstrations, which ensures the convergence of training and the efficient utilization of demonstrations with labels (deterministic actions). The details of our network are provided in the remaining paragraphs.

A. Object-Aware Hierarchical Net

In our two-level hierarchical network, the controller functions as a high-level option-maker, learning a policy that maps the continuous state space $\mathcal{S}$ to the discrete option space $\mathcal{O} = (0, 1, \ldots, n-1)$. Low-level workers learn policies that are capable of achieving the objectives of each option, observe the environment, and output actions until termination.

Negative log-likelihood (NLL) loss was used for pretraining to enable the controller to accurately select workers at an early stage

$$\min_{\theta} \sum_{i=0}^{n-1} \mathop{\mathbb{E}}_{(s,o^i)\in\mathcal{B}_{D_i}} \left[-\log \pi_\theta^c \left(o^i|s\right)\right] \tag{2}$$

where (s, o^i) is state-option pair sampled from the demonstrations $\mathcal{B}_{D_i}$ and θ represents parameters of the controller's policy π^c. We design four types of workers: *resource collector*, *enemy navigator*, *attacker*, and *tool user*. Each worker has the same network structure and outputs the original game action after decoupling and action masking [7].

Object-Aware Controller: Inspired by the previous research [52], we employ on-the-fly object detection YOLOv7 [53] to augment the ability of workers. Unlike 2-D scenes, exploration in 3-D environments requires a sense of distance, so depth information are necessary to guide the agents. Various components are segmented from the depth image to create a depth map. We utilize the camera pose to project the component depth information into the top-down view map, allowing for improved intrinsic reward computation. It is worth noting that OaH-PPO only calls the top-down view map auxiliary calculation when inferring intrinsic rewards, and it is no longer needed during testing. The introduction of depth maps and intrinsic rewards can significantly improve the exploration efficiency of workers and reduce the costs associated with understanding and extracting high-level information.

B. Intrinsic Reward Shaping

Combining intrinsic reward shaping according to workers' utility with a hierarchical structure is an effective method for solving the problem of reward delay and sparse reward [13]. Ng [54] demonstrated that the additional shaping-reward functions $R_I: \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ take the form of a difference of potentials (3) are the only F that will guarantee consistency with the optimal policy

$$R_I\left(s, a, s'\right) = \gamma\Phi\left(s'\right) - \Phi(s) \tag{3}$$

where $\Phi: \mathcal{S} \to \mathbb{R}$ is a real-valued function. Taking the *attacker* as an example, we introduce the potential-based intrinsic reward shaping method. For more information on the intrinsic rewards of workers, please refer to the Appendix.

Attacker is responsible for aiming and shooting. The reward for kills R_E^k is provided by extrinsic rewards, and W_E^k is its weight(W_E^k for kills, 0 otherwise). Consequently, its intrinsic rewards consist of a positive reward for the proximity of the crosshair to the enemy and a negative reward for the enemy

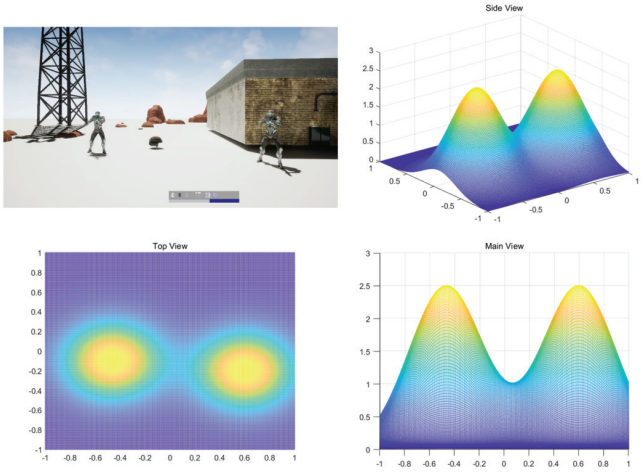


Fig. 4. Intrinsic reward potential function example.

disappearing from view. We introduce the Gaussian function as the potential function

$$\Phi^{\text{att}}(s) = \begin{cases} \max_{i=0}^{n_e} W_E^k \lambda_{\text{att}} e^{-\alpha_{\text{att}} d_{\text{att},i}^2}, & n_e > 0 \\ 0, & n_e = 0 \end{cases} \tag{4}$$

where $\lambda_{\text{att}} < 1$ and α_{att} are the hyperparameters of Gaussian functions and $d_{\text{att}} = (d_{\text{att},1}, \ldots, d_{\text{att},n_e})$ is the distance between enemies and crosshair. n_e denotes the number of enemies in the detection map. Fig. 4 visualizes the intrinsic potential function of an attacker. The figure indicates that the potential energy rises as the crosshair nears an enemy, suggesting a positive intrinsic reward. At the point where the contour lines intersect between two enemies, advancing toward either enemy proves advantageous.

Specifically, in light of the potential for nonstationary in the potential function due to factors such as partial observability, certain constraints are imposed on the intrinsic reward to ensure its efficacy

$$R_I^{\text{att}}(s, a, s') = \begin{cases} 0, & n_e \neq n_e',\ R_E^k = 0 \\ \gamma^{\text{att}} \Phi^{\text{att}}(s') - \Phi^{\text{att}}(s), & \text{otherwise} \end{cases} \tag{5}$$

where γ^{att} can be set to 1 to ensure that the potential energy of the intrinsic reward is always within the interval of $[0, \lambda_{\text{att}} W_E^k]$. As the crosshair gradually approaches and overlaps with the enemy, the potential energy continues to increase to $\lambda_{\text{att}} W_E^k$. In this process, the external reward remains constant at 0 until the enemy is hit and disappears, after which the potential energy falls to 0. The intrinsic reward value is $-\lambda_{\text{att}} W_E^k$, while the extrinsic reward value is W_E^k, and the overall reward $R^{\text{att}} = R_E^k + R_I^{\text{att}}$ will be $(1 - \lambda_{\text{att}}) W_E^k > 0$, still a positive reward for the attacker.

C. PPO From Demonstrations

To expedite the learning process for workers in solving subtasks, we employ the IL method [55]. The demonstration consists of structured records of successful gameplay by human players, which have undergone a thorough filtering process.

Specifically, in addition to the agent's current policy experience Buffer $\mathcal{B}_\pi$, we maintain a demonstration buffer $\mathcal{B}_D$. $\mathcal{B}_D$ is initialized with expert demonstrations and remains unchanged. We employ the scheme of Kapturowsk [56] to prioritize the experience buffer. The annealing parameter ρ determines whether to sample from $\mathcal{B}_D$.

The introduction of additional demonstration buffer updates necessitates the modification of the PPO algorithm's objective function. Take workers, for example, to incorporate the useful information contained in the demonstration buffer, we introduce some additional terms to the objective function without clipping

$$\mathcal{J}(\theta) = \underbrace{\mathop{\mathbb{E}}_{(s,a)\in\mathcal{B}_\pi}\left[r_t(\theta)\hat{A}_t(s,a)\right]}_{\text{Vanilla Term}} + \underbrace{\mathop{\mathbb{E}}_{(s,a)\in\mathcal{B}_D}\left[r_t(\theta)\hat{W}_D(s,a)\right]}_{\text{Demonstration Term}}. \tag{6}$$

For worker i, its reward function is $R_t^i = R_{I,t}^i + R_{E,t}$. As the controller runs at a slower time scale, its reward function is calculated as the sum of extrinsic rewards accumulated during the worker's period: $R_t^c = \sum_{t'=t}^{t+N} R_{E,t'}$. $\hat{W}_D$ is a heuristic weighting function [57]

$$\hat{W}_D(s,a) = \lambda_0 \lambda_1^k \max_{(s',a')\in\mathcal{B}_\pi} \hat{A}_t(s',a') \quad \forall (s,a) \in \mathcal{B}_D \tag{7}$$

where λ_0 and λ_1 are the hyperparameters and k represents the iteration counter. The decay of the weight term via λ_1^k is designed to maximize learning from demonstrations in the early stages while avoiding gradient bias in the middle stages.

In the case of demonstration, where $\pi_D \equiv 1$, it implies that $r_t(\theta) = \pi_\theta$, which is always less than 1. Consequently, the standard clip function will only exert half of its intended impact, which is to restrict the new policy from deviating excessively from the demonstration. However, the filtered demonstration nature implies that $\hat{W}_D(s,a)$ will be positive, and the new policy will gradually approach the demonstration. Therefore, we propose a margin-clip function to prevent the network from overfitting on small demonstration datasets

$$\mathcal{J}_D^{\text{CLIP}}(\theta) = \mathop{\mathbb{E}}_{(s,a)\in\mathcal{B}_D}\left[\min\left(r_t(\theta)\hat{W}_D, \text{clip}(r_t(\theta), \mu_1, \mu_2)\hat{W}_D\right)\right] \tag{8}$$

where $1 > \mu_2 > \mu_1 > 0$. The margin-clip function limits $r_t(\theta)$ to be within the range of μ_1 and μ_2, which helps to prevent the policy from getting too close to the demonstrations.

We introduce causal entropy $H(\pi_\theta)$ [58] to mitigate the risk of entrapment in local optima. Consequently, our final objective function is articulated as follows:

$$\mathcal{J}^{\text{CLIP}}(\theta) = \mathcal{J}_\pi^{\text{CLIP}}(\theta) + \mathcal{J}_D^{\text{CLIP}}(\theta) + \lambda_2 H(\pi_\theta). \tag{9}$$

VI. Experiments

A. Experimental Setup

We conduct experiments using the deathmatch mode on the desert map in UBG and the full deathmatch mode in ViZDoom. Note that UBG can support multiple combinations of different task types and maps. However, even for the same task and map combination, the gameplay experience can vary significantly due to the random generation of scene components. As such, we chose to focus on a representative task scene experiment: deathmatch mode on a medium-difficulty

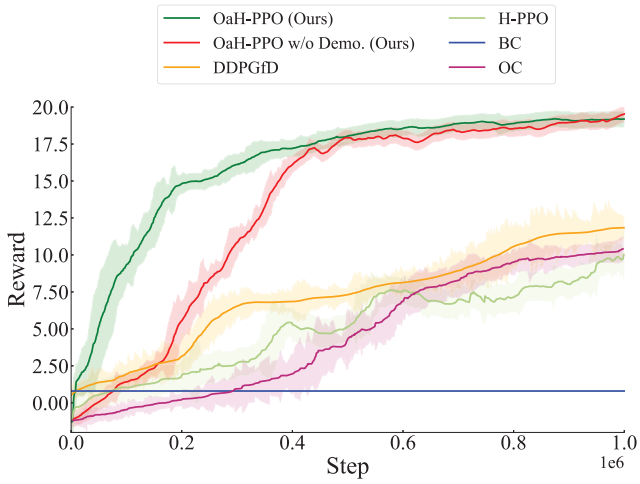


Fig. 5. Comparison of average rewards of different algorithms during training.

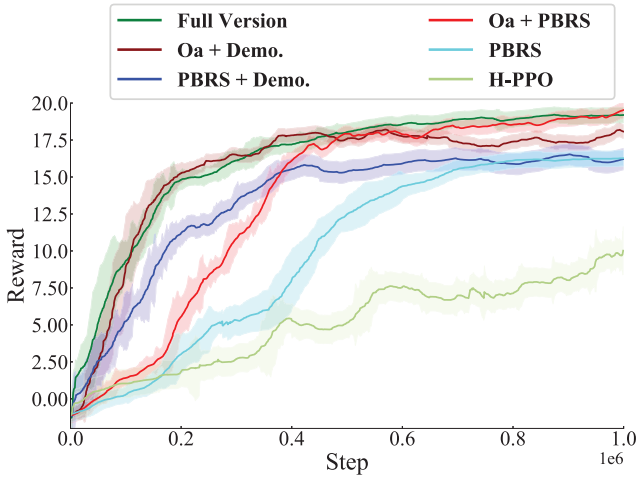


Fig. 6. Partial results of ablation experiments.

desert map. Three different random seeds were used in each experiment, and we displayed their mean value (solid line) and variance (shaded area). A smaller shaded area indicates a smaller variance, suggesting greater stability.

To ensure a fair comparison, we utilize the same resource configuration and fixed hyperparameters for all experiments. All algorithms are run on the same machine, equipped with 24 Intel[1] 4310 (2.1 GHz) CPU cores and 1 3090 GPU card. The initial learning rate for the Adam optimizer is set to $1e^{-4}$, and the reward discount factor γ is set as 0.99. In margin-clip PPO, the clipping parameters μ_1 and μ_2 were set to 0.3 and 0.6, respectively. In addition, we set $\lambda = 0.95$ in the GAE, $\lambda_0 = 0.1$ and $\lambda_1 = 0.98$ in demonstrations augmented objective function.

B. Experiment on UBG

We utilize a 2-h gameplay record of human participants as a demonstration. All experiments utilize the same extrinsic reward, which can be found in the Appendix. Our framework enables the computation of immediate extrinsic rewards during training, and the intrinsic reward is not included in the final result. This ensures the fairness of experimental comparisons.

1) Comparison With Baselines: We compare our method with several existing methods, including DDPGfD, OC, and BC. BC's score is the average score of 20 episodes. We also employ the hierarchical PPO (H-PPO) algorithm as a baseline. H-PPO is the basic version of the OaH-PPO model, but it lacks three crucial components: object-aware (Oa) module, potential-based reward shaping (PBRS), and introduction of Demo. H-PPO still adopts a hierarchical structure and directly uses depth images and sparse rewards without utilizing human demonstrations. As shown in Fig. 5, our method significantly outperforms the baselines regarding convergence speed, final score, and standard deviation. This means that our approach has improved in terms of sample efficiency, performance, and stability. Interestingly, even without demonstrations, the agent trained using our method outperforms the baselines considerably. It is worth noting that, during the middle and late stages of the experiment, the OaH-PPO without demonstrations reached the same level as the full version, which is consistent with our experimental design. By analyzing the experimental replay records, we observed that some baselines (such as DDPGfD and H-PPO) were able to learn how to find resources but were unable to handle more complex tasks, such as aiming and shooting effectively. When facing enemies with a small hit range in the UBG, these baseline agents persist in their attacks but rarely hit their targets, ultimately resulting in their defeat by the enemies.

2) Ablation Study: We perform ablation experiments to comprehend the roles of different components in our method.

In Fig. 6, we mainly analyze three components of the model network: Oa, PBRS, and Demo. The full version refers to the complete model we trained, while H-PPO, as mentioned earlier, refers to the model without the above three components.

First, we analyze the Demo. By comparing the results, we can conclude that the introduction of demonstrations can effectively guide the early actions of the agent, helping the agent achieve a "warm start" and accelerating the convergence rate. At the same time, in the middle and late stages of the experiment, the weight of the module is gradually reduced to 0, so it does not affect the final ability of the agent.

Next, we analyze the Oa module. According to a comparison between the experimental results obtained without the Oa module and those obtained with the full version, it is evident that the Oa module can significantly improve the agent's abilities. This improvement is attributed to a better understanding of enemy and resource location information, resulting in a higher hit rate for the agent. Compared with the results of the experiments conducted without Demo. and Oa modules, and combined with the conclusion that the Demo. does not affect final performance, we can be more confident in this finding.

The final part of our analysis focuses on the PBRS module. The efficiency of this module is somewhat different from our expectations. By comparing experiments using only the

[1]Registered trademark.

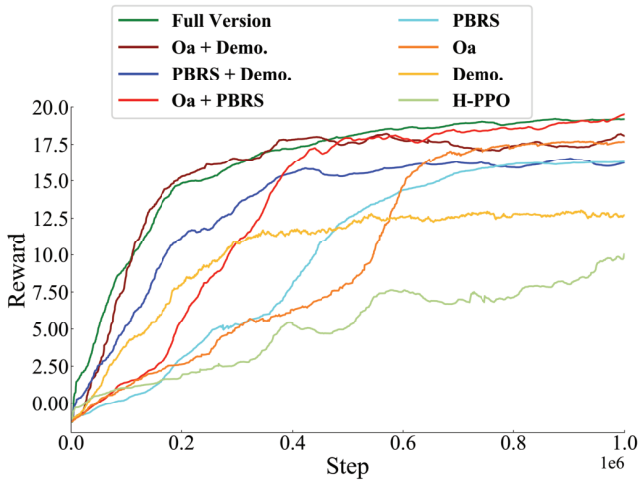


Fig. 7. Complete results of ablation experiments.

TABLE II
EXPERIMENTAL RESULTS OF CITY MAP

Method	Reward	Converge Time	Converge Steps
DDPGfD	8.6	**26h**	8.8×10^5
OC	10.6	31h	7.6×10^5
BC	1.3	-	-
H-PPO	7.2	34h	9.5×10^5
OaH-PPO	**16.2**	28h	$\mathbf{4.3}\times 10^5$

TABLE III
EXPERIMENTAL RESULTS OF FOREST MAP

Method	Reward	Converge Time	Converge Steps
DDPGfD	9.4	**21h**	8.1×10^5
OC	11.7	25h	6.9×10^5
BC	1.5	-	-
H-PPO	7.7	28h	8.9×10^5
OaH-PPO	**18.3**	23h	3.8×10^5

PBRS module to those that use H-PPO, we discover that the PBRS module can effectively accelerate model convergence. However, the performance of the experiments using the full version is not significantly different from those conducted without the PBRS module. We hypothesize that this is due to the guiding effect of demonstrations.

The complete ablation experimental results are shown in Fig. 7. When utilizing only the Oa module, the agent gradually masters the rules of the game after an extended period of exploration and quickly begins to score points. This also confirms that our other two modules can provide guidance and accelerate the learning process for the agent. When comparing the results of using only the Demo. (without Oa and PBRS) versus using both the Demo. and PBRS modules (without Oa), it is found that the increased capabilities of the agent utilizing the PBRS module also contributed to improve performance.

As shown in Tables II and III, we have conducted generalization and stability tests on our algorithm framework in various map scenarios. In addition to the final reward value and the number of steps to converge, we also calculated the time required for the algorithm to converge. It can be observed that, on the same hardware setup, OaH-PPO demonstrates notable superiority in efficiency, achieving superior results in a notably shorter timeframe compared to the majority of alternative methodologies.

TABLE IV
PERFORMANCE UNDER DIFFERENT PARAMETERS OF DEMO

Demo. Size	λ_1	Converge Steps	Reward
100%	1.00	2.6×10^5	4.70
	0.98	3.4×10^5	12.1
	0.90	4.9×10^5	11.6
50%	1.00	3.1×10^5	4.30
	0.98	3.6×10^5	11.4
	0.90	5.1×10^5	11.2
10%	1.00	4.6×10^5	4.80
	0.98	6.5×10^5	10.1
	0.90	7.8×10^5	9.20
0%	–	9.5×10^5	8.60

3) Demonstration Study: The PPO algorithm is an on-policy algorithm, but the introduction of margin-clip and importance sampling makes it robust to demonstration data. We test the performance of the Demo. module under different parameters to verify its effectiveness. As shown in Table IV, the experiment only includes demonstration modules. Adequate demonstrations and appropriate parameters ensure the effectiveness of the demonstration module. A too-short decay period ($\lambda_1 = 0.9$) reduces its impact and slows down convergence, while a too-long decay period ($\lambda_1 = 1$) may result in the network no longer being affected by policy gradients.

C. Experiment on ViZDoom

We also use the ViZDoom platform to conduct our experiments. We considered *the full deathmatch* on unknown maps, adapted from VDAIC 2016. Agents were trained and tested on different maps, starting with a pistol and the ability to pickup various weapons and items, such as ammunition, medical kits, and armor. PyOblige [13] was used to generate seven maps for training and three for testing. All maps have similar difficulty and textures.

Following the rules of the VDAIC, we utilized frags (calculated as the number of killed minus the number of suicides) as a *evaluation metric* for evaluation. In addition, we reported the number of kills, suicides, deaths, and frag-to-death (F/D) ratios to further analyze the agent's abilities. Our agent was compared to Arnold [9], CLYDE [12], and human players. To verify the effectiveness of the Oa module on other platforms, we also tested H-PPO (without the Oa module).

VDAIC itself did not provide a demonstration for players, and later prohibited the crouch action. To be fair, none of the models used demonstrations. OaH-PPO and H-PPO removed the enemy navigator worker, and blocked the crouch action while Arnold retained this action. Each map was tested for 15 min, and the final performance was averaged. As shown

TABLE V
PERFORMANCE OF THE AGENTS IN THE VIZDOOM TEST MAPS

Player	Frags	F/D ratio	Kills	Suicides	Deaths
Arnold	51.0	**4.78**	53.3	2.3	**10.7**
CLYDE	45.7	2.63	47.7	**2.0**	17.3
Human	39.0	2.17	45.7	6.7	18.0
H-PPO	52.0	3.06	56.7	4.7	17.0
OaH-PPO	**59.3**	4.05	**63.7**	4.3	14.7

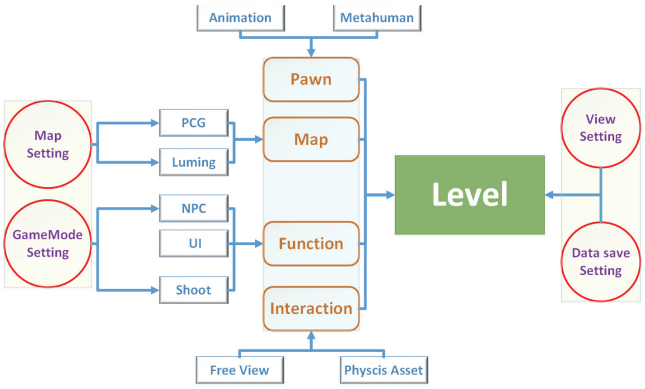


Fig. 8. Overall game design of UBG. A collection of functions and configurable interfaces.

in Table V, OaH-PPO achieved the highest frags score and exhibited a higher rate of both suicides and deaths while also achieving a higher number of kills. This may be attributed to the intrinsic reward shaping being more offensive. Without Oa module, H-PPO performance is significantly reduced, but it is more in line with the rules of VDAIC 2016.

VII. CONCLUSION

In this article, we present a novel open-world FPS benchmark, UBG, which boasts high visual fidelity and complex scenes. To train agents to play this game, we propose a hierarchical framework, OaH-PPO, which has been demonstrated to master the intricate controls of UBG effectively. Furthermore, our architecture can be generalized to other shooter games.

In terms of prospects, we aim to continue optimizing the computational costs of the UBG platform and expand it to more domains and types of games. On the other hand, our algorithm framework also has the potential to be applied in nonshooting game environments, which we will further explore in the Appendix.

APPENDIX A
MORE GAME DESIGN

We will provide a detailed introduction to the UBG game design from the aspects of game modules, game settings, API, and computational cost.

A. Game Modules

As shown in Fig. 8, UBG has four main modules: pawns, maps, functions, and interactions.

1) Pawns: Video games require smooth and dynamic character movements. To achieve this, we utilize the animation blueprint system within the UE. This system allows you to create a comprehensive state-machine transition and combine different animation sequences seamlessly. By doing so, you can provide suitable animations for the player as they perform various actions or transition between different actions.

The game also requires a character model that possesses adaptable precision, allowing the character to utilize different levels of precision. These adjustments should be responsive and depend on the developer's environment. The UE's MetaHuman resource not only speeds up the modeling process but also enables the configuration of character model precision through various levels of detail (LODs) tiers.

2) Maps: The game requires dynamic global illumination, which means that we must recreate realistic natural sky lighting and atmospheric clouds, just like in the real world. The virtual environment's sunlight goes through regular fluctuations, while changes in atmospheric clouds follow a random pattern. By combining these dynamic elements for global illumination, we can create a more realistic and diverse experience for the players.

To achieve a realistic and diverse environment, we first utilize terrain modeling to create maps of varying sizes and dimensions, sculpting the terrain accordingly. Subsequently, leveraging the PCG mechanism introduced in UE 5.2, we can generate a diverse array of randomized cover objects within maps that share similar topographical features. To further enhance the fidelity of the simulation environment, we can scan real assets (such as furniture) into a simulation environment and introduce Proctor's house layout generation method [59] to create over 10k house layouts and indoor scenes. By automatically adjusting the materials, they can adapt to desert, forest, and urban scene modes. Finally, we implement the object placement techniques from Grutopia [60] to ensure that assets are appropriately positioned within the generated layouts. Such a simulated environment will have a complexity similar to the real world, in which algorithms developed will have a higher potential for application in real-world tasks.

3) Functions: The fundamental functionalities encompass unrestricted player exploration within an open-world environment. This includes the discovery of concealed areas for exploration, retrieval of equipment, engagement in combat with adversaries, and other fundamental mechanics intrinsic to shooting games.

The game incorporates intelligent adversaries, in which the in-game enemies are nonplayable characters (NPCs) equipped with distinctive behavior trees. These opponents continuously patrol specific areas, starting their patrol again after a set period. Upon detecting the player's presence, the enemies swiftly transition into an attack mode, precisely targeting and engaging the player with gunfire. In instances where the player escapes their line of sight, the enemies revert to their routine patrol activities, encompassing the exploration of the adjacent vicinity.

The user interface (UI) offers players the capability to fine-tune their equipment choices. Within the game, each

Fig. 9. UBG multimap scene displays. (a) City map. (b) Desert map. (c) Forest map.

equipment category, such as helmets, body armor, attire, and weaponry, boasts distinct attributes. For instance, a helmet boasting heightened defense capabilities also translates to increased weight, while a more potent sniper rifle entails longer intervals between shots. Players must possess the capacity to configure their preferred equipment types through the UI, as well as discard any superfluous items.

4) Interactions: Diverse objects exhibit varying physical interaction outcomes in scenarios that require interaction within the scene. For instance, climbable walls, breakable boxes, and inert stones each demonstrate distinctive physical response characteristics. By harnessing the UE's physical assets and collision presets, it is feasible to harmonize the physical simulation effects of assorted items.

Enabling perspective switching is imperative to transition between first-person and third-person viewpoints seamlessly. The third-person perspective offers an unobstructed vantage point to glean comprehensive self-related information, while the first-person view bolsters aiming precision when equipped with a scope, enhancing targeting accuracy.

In the process of game development, it is essential to go beyond creating a functional core and provide players with a diverse range of interfaces that encourage free customization.

B. Game Settings

The content of UBG is rich and varied. Introducing some features in UBG would better guide players to customize the game content according to their needs.

1) Game Mode Setting: UI allows players to select from two distinct modes, namely, training mode and deathmatch mode.

The training mode constitutes a streamlined experience with relatively straightforward tasks. In this mode, the damage inflicted by enemies will be reduced, the enemy count will be minimal, their reaction speed will be sluggish, and their surveillance range will be limited. These adjustments collectively facilitate an environmental conducive for players to grasp fundamental movement and shooting mechanics.

The deathmatch mode requires players to arm themselves by picking up components on the ground. In this mode, the ultimate goal is to kill as many enemies as possible. The coordinates and movements of the enemies are random, so players need to constantly search for them in the scene. In comparison to the preceding mode, the deathmatch mode puts more emphasis on testing players' survival and offensive strategies.

TABLE VI
THROUGHPUT (TOTAL MEAN FPS) OF THE ENVIRONMENT WITH DIFFERENT NUMBERS OF CONCURRENT PROCESSES, AGENTS, AND ENVIRONMENTAL COMPONENT DENSITIES

Throughputs		#Process			
#Density	#Agents	1	2	3	4
0.1	1	70.6	140.2	206.2	**213.5**
	5	70.4	140.3	**203.4**	200.2
	10	70.4	140.4	**190.7**	188.5
0.2	1	70.3	139.8	205.3	**206.4**
	5	70.4	139.8	**198.4**	196.5
	10	70.3	139.6	**184.4**	178.4
0.3	1	70.2	139.4	203.2	**204.7**
	5	69.9	139.7	**192.8**	184.3
	10	70.1	139.6	**170.2**	166.4

2) Map Environmental Setting: We provide a UI that empowers players to choose from various map types while simultaneously adjusting the likelihood and concentration of dynamically generated items within the map. As shown in Fig. 9, each map captures the dynamic changes in natural lighting and the unpredictable movement of atmospheric clouds. For each distinct style of map, we have preset three resource consumption modes—low, medium, and high—to adapt to different hardware devices (corresponding to three sets of parameters in Table VI, with #density/#agents being 0.1/1, 0.2/5, and 0.3/10, respectively). By fine-tuning LOD, Lumen, Nanite, and other scene optimization techniques available in UE, it is possible to optimize performance and minimize computational costs effectively.

a) City Map: The city map encompasses dimensions of $12\,000 \times 12\,000$ unreal units and is characterized by a relatively even terrain. Across this vast expanse, there are breakable barriers scattered throughout the terrain, coexisting with static structures such as houses. The integration of these scene elements is facilitated through PCG techniques.

b) Desert Map: The desert map spans an expansive 8000×8000 unreal units area, characterized by gentle undulations in the terrain. This environment is punctuated by an abundance of shatterable obstacles and diverse vantage points for sniping. The desert terrain showcases weathered rock formations and shrubs generated through PCG methods.

c) Forest Map: The forest map spans a vast expanse measuring 6000×6000 unreal units and is characterized by

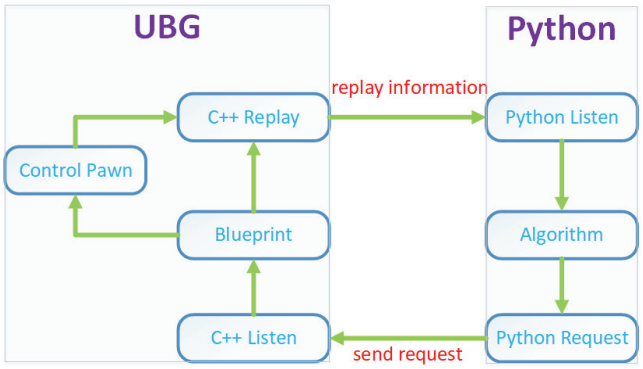


Fig. 10. Communication process diagram.

pronounced elevation variations. The terrain presents substantial fluctuations in its topography. Diverse and intensified PCG processes are applied to enrich the diverse landforms found within the forest, resulting in augmented content density.

3) Auxiliary View Setting: We provide a UI that empowers players to configure image capture settings, top-down view map resolution, and the activation of top-down view map prompts. In the training mode, the highlighting of essential items offers valuable assistance for training purposes.

4) Data Storage Mode Setting: The UI allows players to activate or deactivate the data storage function within this section. Using this feature, we recorded 2 h of human player data, approximately 90k.

5) Game-Type Expansion: Thanks to the Blueprint and map editor of UE, UBG is not limited to FPS games. For example, by modifying the map scenes and game objectives (such as arriving at a specific place), it can be turned into a maze, or even expanded into a strategic decryption game by setting up mechanisms on the map. For a more challenging experience, we can also introduce vehicle simulations such as cars and helicopters based on real-world parameters [61], providing applications for areas like autonomous driving and racing. Specific operational instructions will be provided in our documentation or UE beginner tutorial.

C. API

Upon completion of all configurations, the training of the intelligent agent can be initiated. As shown in Fig. 10, the fundamental logic governing UBG is primarily realized through blueprint structures, while communication primarily encompasses the synergy of C++ and Python frameworks. Directives originating from Python are channeled to the C++ layer, where they are subsequently decoded by the blueprint system to regulate character actions. In parallel, the blueprint system proficiently acquires the character's state information, which is then transmitted to the Python layer through the C++ interface. This bidirectional information exchange culminates in a full-communication cycle.

Considering the network-based communication between the C++ and Python components, it is essential to preconfigure the IP address and port settings in both the Python and blueprint contexts before initiating communication. These predefined configurations hold significance in enabling distributed deployment and training capabilities for both the training terminal and the UBG project.

```python
from UBG_env import UBGEnv

game = UBGEnv(record=False, image_shape=(256, 512, 1))
action_space = game.action_space # action space: MultiDiscrete((9, 11, 2, 2, 5)),
      move, rotate, fire, jump, skill
for i in range(10):  # loop over 10~episodes.
    # reset the game and get the screen buffer and game variables
    obs = game.reset()
    depth = game.img_depth
    misc = game.state
    while not misc[``game over''][0]:
        # perform a random action, here we just sample one from the action space.
        action = action_space.sample()
        next_obs, reward, done, info = game.step(action)
        next_depth = game.img_depth
        next_misc = game.state
        # do something with the observation and reward\ldots
    print(``reward: '', game.reward)
```

Below are excerpts of Python code showcasing control and configuration settings.

In our settings, movement is divided into nine directions, including standing still. The rotation direction has 11 options: $\{-30, -10, -4, -2, -1, 0, 1, 2, 4, 10, 30\}$, and there are additional actions such as firing, jumping, and four special interactive actions.

D. Computational Cost and Simulation Speed

UBG supports multiple environments in parallel, and the high-fidelity graphics produced by UBG would lead to additional computation costs. Therefore, we quantify this cost by evaluating the simulation speed across varying numbers of parallel processes, agent numbers (both self and enemy), and densities of environmental components (such as rocks, grass, and trees). The result is shown in Table VI. All results are tested on a computational device equipped with 24 Intel[1] 4310 (2.1 GHz) CPU cores and 1 NVIDIA 3090 GPU card, and the maximum throughput for each number of agents is highlighted in bold.

The results indicate that in our experimental setup, it is suggested to have three or four parallel processes when the number of NPC agents and component density are low. However, caution should be exercised when these two parameters are set at high levels. In such cases, it is advisable to not exceed three parallel processes to prevent overloading, which has the potential to negatively impact the overall experimental performance.

Appendix B
More Algorithms Design

A. Notation

For modeling the action decision process in our context, a standard Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ is considered, where $\mathcal{S}$ and $\mathcal{A}$ denote the space of feasible states and actions, respectively, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{P}(s'|s, a)$ represent the transition probability and $\gamma \in (0, 1]$ is the discount factor. A stochastic policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$ maps state into action distribution. A trajectory ξ is given by the sequence of state-action pairs $\{(s_0, a_0), (s_1, a_1), \ldots\}$. Reinforcement learning from demonstrations (RLfD) enhances traditional RL by introducing an additional demonstration trajectories buffer $\mathcal{B}_D = \{\xi_0, \xi_1, \ldots\}$.

The objective of RL is to maximize the cumulative expected (discounted) return along the whole decision procedure $\eta(\pi) = E_\pi\left[\sum_t^\infty \gamma^t r(s_t, a_t)\right]$, given current action policy π. While RLfD enhances RL by providing a set of demonstration trajectories $D = \{\xi_0, \ldots\}$ drawn from a referred expert with policy π_E as an extra guidance other than reward. Such expert data can be useful notably when environmental feedback is sparse or delayed, in which the agent may suffer from ineffective explorations since positive feedback could rarely occur. The RL agent aims to find an optimal strategy π_o to maximize the cumulative expected return, i.e., the objective $\mathbb{E}_{(s,a)\sim\pi}\left[\sum_t^\infty \gamma^t r(s_t, a_t)\right]$.

The semi-Markov decision process (Semi-MDP) offers a theoretical framework for an option-based approach in which the duration between actions (options) is uncertain. Formally, the controller's decision process can be conceptualized as a semi-MDP $M_c = (\mathcal{S}, \mathcal{O}, \mathcal{P}_c, \mathcal{R}_E, \gamma, \mathcal{F})$, based on multiple worker-based MDP processes $M_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_E + \mathcal{R}_i, \gamma)$, where $\mathcal{F}(t|s, o)$ represents the probability that the transition time is t, when option o is executed in state s, which is the termination condition. The controller selects workers i according to its policy p_c and assigns corresponding intrinsic rewards R_I^i.

B. Action Spaces Decoupling

To account for the orthogonality between action spaces, we divide $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_M$, where $\mathcal{A}$ represents the entire action space. $\mathcal{A}_i$ represents an action subspace in which actions are mutually exclusive (e.g., moving forward and backward), actions between each $\mathcal{A}_i$ are orthogonal (e.g., movement and turning), and M is the number of subspace partitions. Suppose each action $a = (a^0, \ldots, a^{M-1})$, then the workers' PPO objective without clipping after decoupling is

$$\max_\theta \mathop{\mathbb{E}}_{(s,a)\in\mathcal{B}_\pi}\left[\left(\prod_{i=0}^{M-1}\frac{\pi_\theta(a^{(i)}|s)}{\pi_{\theta_{\text{old}}}(a^{(i)}|s)}\right)\hat{A}_t(s,a)\right]. \tag{10}$$

C. More Intrinsic Reward Design

Resources Collector collects resources such as guns, ammo, and medicine by walking around the map. Extrinsic rewards provide the reward weight for successful collection W_E^r, so its intrinsic rewards should include a positive reward for finding and getting close to the item and a negative reward for staying away from the resources. The top-down view map provides easy access to the number of nearby resources n_r and the distance from each resource $d_r = (d_{r,1}, \ldots, d_{r,n_r})$

$$\Phi^{\text{res}}(s) = \begin{cases}\max_{i=1}^{n_r} W_E^r \lambda_{\text{res}} e^{-\alpha_{\text{res}} d_{r,i}^2}, & n_s > 0\\ 0, & n_r = 0\end{cases}$$

where $\lambda_{\text{res}} < 1$ and α_{res} are the hyperparameters of Gaussian functions. Therefore, the intrinsic reward function of the resources collector is

$$R_I^{\text{res}}(s, a, s') = \gamma^{\text{res}}\Phi^{\text{res}}(s') - \Phi^{\text{res}}(s).$$

Upon successful collection of a resource, the intrinsic reward will become negative, while the addition of the extrinsic reward will result in positive feedback.

Algorithm 1 OaH-PPO

Initialize experience replay buffer $\{\mathcal{B}_\pi^c, \mathcal{B}_\pi^w, \mathcal{B}_D^c, \mathcal{B}_D^w\}$ and sample probability $\rho = 1$.
Initialize controller policy π_θ^c and worker policies $\{\pi_\theta^i, i = 0, 1 \ldots, n-1\}$. {$n$ is the number of workers}
for Episode number $j = 0, 1, \ldots, E-1$ **do**
  Initialize game and get start state s
  $\hat{o} \leftarrow \arg\max \pi_\theta^c(o|s)$ {Take an option}
  Get state s_o that contains depth detection map and intrinsic reward $R_I^{\hat{o}}$ according to $\hat{o}$ {Object-aware}
  $R_c \leftarrow 0,\ s_1 \leftarrow s$
  while s is **not** terminal **do**
    while not (s is terminal **or** terminal condition $\mathcal{F}$ is satisfied) **do**
      $a \leftarrow sample(\pi_\theta^{\hat{o}}(a|s_o))$ {sample an action by worker $\hat{o}$}
      Execute a and obtain next state s' and extrinsic reward R_E from environment
      Get state s_o' that contains depth detection map and intrinsic reward $R_I^{\hat{o}}$
      Store transition $(s_o, a, R_E + R_I^i, s_o', \hat{o})$ to $\mathcal{B}_\pi^w$
      $R_c \leftarrow R_c + R_E, s_o \leftarrow s_o'$
    end while
    if Terminal condition $\mathcal{F}$ is satisfied **then**
      Store transition $(s_1, \hat{o}, R_c, s')$ to $\mathcal{B}_\pi^c$
      $\hat{o} \leftarrow \arg\max \pi_\theta^c(o|s)$
      $R_c \leftarrow 0,\ s_1 \leftarrow s'$
    end if
  end while
  Sample demonstrations with probability ρ. And update the controller and workers with Eq 7.
  Anneal ρ
end for

Enemies Navigator is responsible for finding enemies, so a positive reward for finding an enemy

$$\Phi^{\text{ene}}(s) = \begin{cases}1, & n_e > 1\\ 0, & n_e = 0.\end{cases}$$

Tools user is exclusively authorized to utilize tools such as medicine and receives an additional positive reward for successful tool usage, as well as a negative reward for being attacked while using the tool

$$\Phi^{\text{too}}(s) = \begin{cases}1, & \text{use successful}\\ -1, & R_E^h < 0\\ 0, & \text{else.}\end{cases}$$

Algorithm 1 shows the overall flow of our algorithm. We can see that although OaH-PPO is an algorithm designed for FPS games, it can be applied to various 3-D open-world games or real-world scenarios. For example, in puzzle games, the Oa module can make the agent more sensitive to key items and mechanisms, which we know are crucial for solving puzzles. Based on a progressive puzzle-solving process, we can still design intrinsic rewards for workers at different levels under a hierarchical structure to break down main tasks. Needless to say, the demonstration module is helpful for any game when used effectively.

REFERENCES

[1] OpenAI et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[2] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[3] Z. Lin, J. Li, J. Shi, D. Ye, Q. Fu, and W. Yang, "JueWu-MC: Playing minecraft with sample-efficient hierarchical reinforcement learning," 2021, *arXiv:2112.04907*.

[4] M. Johnson, K. Hofmann, T. J. Hutton, and D. E. Bignell, "The malmo platform for artificial intelligence experimentation," in *Proc. IJCAI*, Jul. 2016, pp. 4246–4247.

[5] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," 2016, *arXiv:1612.04340*.

[6] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr, "Playing doom with SLAM-augmented deep reinforcement learning," 2016, *arXiv:1612.00380*.

[7] D. Ye et al., "Mastering complex control in MOBA games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 6672–6679.

[8] S. Huang, H. Su, J. Zhu, and T. Chen, "Combo-action: Training agent for FPS game with auxiliary tasks," in *Proc. AAAI*, Jul. 2019, vol. 33, no. 1, pp. 954–961.

[9] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Proc. AAAI*, Feb. 2017, vol. 31, no. 1, pp. 1–23.

[10] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "ViZDoom: A doom-based AI research platform for visual reinforcement learning," in *Proc. IEEE CIG*, Sep. 2016, pp. 1–8.

[11] M. Wydmuch, M. Kempka, and W. Jaskowski, "ViZDoom competitions: Playing doom from pixels," *IEEE Trans. Games*, vol. 11, no. 3, pp. 248–259, Sep. 2019.

[12] D. Ratcliffe, S. Devlin, U. Kruschwitz, and L. Citi, "Clyde: A deep reinforcement learning doom playing agent," in *Proc. AAAI Workshops*, 2017, pp. 1–21.

[13] S. Song, J. Weng, H. Su, D. Yan, H. Zou, and J. Zhu, "Playing FPS games with environment-aware hierarchical reinforcement learning," in *Proc. Twenty-Eighth Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3475–3482.

[14] M. Dawes and R. Hall, "Towards using first-person shooter computer games as an artificial intelligence testbed," in *Proc. KES*, Jan. 2005, pp. 276–282.

[15] T. Pearce and J. Zhu, "Counter-strike deathmatch with large-scale behavioural cloning," in *Proc. IEEE Conf. Games (CoG)*, Aug. 2022, pp. 104–111.

[16] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Proc. Mach. Intell.* , Jan. 2000, pp. 103–129.

[17] C. Beattie et al., "DeepMind lab," 2016, *arXiv:1612.03801*.

[18] T. Le Paine et al., "Making efficient use of demonstrations to solve hard exploration problems," 2019, *arXiv:1909.01387*.

[19] Y. W. Teh et al., "Distral: Robust multitask reinforcement learning," *NeurIPS*, vol. 30, pp. 4499–4509, Dec. 2017.

[20] X. Chen, T. Shi, Q. Zhao, Y. Sun, Y. Gao, and X. Wang, "WILD-SCAV: Benchmarking FPS gaming AI on Unity3D-based environments," 2022, *arXiv:2210.09026*.

[21] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, Jun. 2013.

[22] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[23] S. Li et al., "The fittest wins: A multi-stage framework achieving new SOTA in ViZDoom competition," *IEEE Trans. Games*, vol. 16, no. 1, pp. 225–234, Mar. 2024.

[24] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Trans. Games*, vol. 12, no. 1, pp. 1–20, Mar. 2020.

[25] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Perez, and L. Pack Kaelbling, "Learning neuro-symbolic skills for bilevel planning," 2022, *arXiv:2206.10680*.

[26] N. v. Hoorn, J. Togelius, and J. Schmidhuber, "Hierarchical controller learning in a first-person shooter," in *Proc. IEEE Symp. Comput. Intell. Games*, Sep. 2009, pp. 294–301.

[27] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Proc. NIPS*, vol. 5, Nov. 1992, pp. 271–278.

[28] A. S. Vezhnevets et al., "Feudal networks for hierarchical reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3540–3549.

[29] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, Aug. 1999.

[30] P. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. AAAI*, Feb. 2017, vol. 31, no. 1, pp. 1–12.

[31] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. NIPS*, vol. 29, Dec. 2016, pp. 3682–3690.

[32] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, Nov. 2000.

[33] A. G. Barto, "Intrinsic motivation and reinforcement learning," in *Proc. Intrinsically Motivated Learn. Natural Artif. Syst.*, Nov. 2012, pp. 17–47.

[34] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," 2017, *arXiv:1712.00948*.

[35] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. NeurIPS*, May 2018, pp. 1–23.

[36] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[37] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, vol. 1, 2000, pp. 1–23.

[38] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," 2019, *arXiv:1910.11956*.

[39] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6292–6299.

[40] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 2469–2478.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[42] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*.

[43] T. Hester et al., "Deep Q-learning from demonstrations," in *Proc. AAAI*, Jan. 2017, pp. 1–11.

[44] M. Vecerik et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, *arXiv:1707.08817*.

[45] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, 2015, pp. 1889–1897.

[46] B. Luo, Z. Wu, F. Zhou, and B.-C. Wang, "Human-in-the-Loop reinforcement learning in continuous-action space," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 11, pp. 15735–15744, Nov. 2023.

[47] B. A. Wallace and J. Si, "Continuous-time reinforcement learning control: A review of theoretical results, insights on performance, and needs for new designs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 8, pp. 10199–10219, Aug. 2023.

[48] W. Qiu et al., "UnrealCV: Virtual worlds for computer vision," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1221–1224.

[49] Y. Bengio, J. Louradour, and R. Collobert, "Curriculum learning," in *Proc. Int. Conf. Mach. Learn.*, Aug. 2009, pp. 41–48.

[50] L. Niu and J. Wan, "D2AH-PPO: Playing ViZDoom with object-aware hierarchical reinforcement learning," in *Proc. 7th Int. Symp. Auto. Syst. (ISAS)*, May 2024, pp. 1–6.

[51] H. Le et al., "Hierarchical imitation and reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2917–2926.

[52] F. G. Glavin and M. G. Madden, "Adaptive shooting for bots in first person shooter games using reinforcement learning," *IEEE Trans. Comput. Intell. AI Games*, vol. 7, no. 2, pp. 180–192, Jun. 2015.

[53] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, *arXiv:2207.02696*.

[54] AY. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, San Francisco, CA, USA, 1999, pp. 278–287.

[55] R. F. Prudencio, M. R. O. A. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 8, pp. 10237–10257, Aug. 2023.

[56] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement Learning.," in *Proc. ICLR*, Sep. 2018, pp. 1–21.

[57] A. Rajeswaran et al., "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," 2017, *arXiv:1709.10087*.
[58] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. 23rd AAAI Conf. Artif. Intell.*, Chicago, IL, USA, vol. 8, Jul. 2008, pp. 1433–1438.
[59] M. Deitke et al., "ProcTHOR: Large-scale embodied AI using procedural generation," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 5982–5994.
[60] H. Wang et al., "GRUtopia: Dream general robots in a city at scale," 2024, *arXiv:2407.10943*.
[61] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Proc. Field Service Robot., Results 11th Int. Conf.* Cham, Switzerland: Springer, 2018, pp. 621–635.

Longyu Niu received the B.E. degree from the University of Chinese Academy of Science, Beijing, China, in 2021, and the master's degree from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, in 2024.

His research interests include reinforcement learning and embodied intelligence.

Baihui Li received the bachelor's degree from Henan University of Science and Technology, Luoyang, China, in 2019. He is currently pursuing the master's degree with the University of Chinese Academy of Sciences, Beijing, China.

His research interests include the procedural generation of virtual simulation environments and reinforcement learning.

Xingjian Fan received the bachelor's degree in engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2024. He is currently pursuing the master's degree with the University of Chinese Academy of Sciences, Nanjing.

His research interests lie in the fields of computer vision and virtual digital humans.

Hao Fang received the master's degree from the School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing, China in 2024.

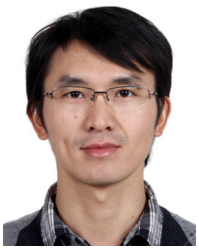
He has published some papers at top international AI conferences and journals such as IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY/IJCAI/ICASSP. His research interests include pattern recognition and multimodal large models.

Jun Li received the B.E. degree from Harbin Institute of Technology (HIT), Harbin, China, in 2021, and the master's degree from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, in 2024.

He has published some papers at the top international AI conferences and journals such as CVPR/AAAI/TIP. His research interests include long-tailed learning, foundation model, and vision language model.

Junliang Xing (Senior Member, IEEE) received the dual B.E. degree in computer science and applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2007, and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012.

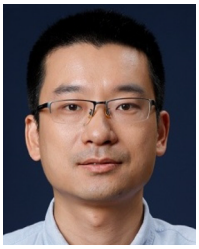
He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University. He has published over 120 peer-reviewed conference papers like IJCAI, AAAI, ICCV, CVPR, and journal papers likeIEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), *International Journal of Computer Vision* (IJCV), and *Artificial Intelligence* (AIJ), which received over 21000 citations from Google Scholar. His main research areas lie in computer vision and computer gaming, with a current focus on human-computer interactive learning in complex decision-making scenarios.



Jun Wan (Senior Member, IEEE) received the B.S. degree from China University of Geosciences, Beijing, China, in 2008, and the Ph.D. degree from the Institute of Information Science, Beijing Jiaotong University, Beijing, in 2015.

Since January 2015, he has been a Faculty Member with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, where he currently serves as an Associate Professor.

Dr. Wan is an Associate Editor of the *IET Biometrics*. He has served as a Co-Editor for the special issues in IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), *Machine Vision and Applications* (MVP), and *Entropy*.



Zhen Lei (Fellow, IEEE) received the B.S. degree in automation from the University of Science and Technology of China, Hefei, China, in 2005, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2010.

He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences. He has published over 100 papers in international journals and conferences. His research interests are in computer vision, pattern recognition, image processing, and face recognition in particular.

Dr. Lei served as the Area Chair for the International Joint Conference on Biometrics in 2014, the IAPR/IEEE International Conference on Biometrics in 2015, 2016, and 2018, and the IEEE International Conference on Automatic Face and Gesture Recognition in 2015.