

Web-Gewu: A Browser-Based Interactive Playground for Robot Reinforcement Learning

Kaixuan Chen, Linqi Ye

Abstract—With the rapid development of embodied intelligence, robotics education faces a dual challenge: high computational barriers and cumbersome environment configuration. Existing centralized cloud simulation solutions incur substantial GPU and bandwidth costs that preclude large-scale deployment, while pure local computing is severely constrained by learners’ hardware limitations. To address these issues, we propose Web-Gewu, an interactive robotics education platform built on a WebRTC cloud-edge-client collaborative architecture. The system offloads all physics simulation and reinforcement learning (RL) training to the edge node, while the cloud server acts exclusively as a lightweight signaling relay, enabling extremely low-cost browser-based peer-to-peer (P2P) real-time streaming. Learners can interact with multi-form robots at low end-to-end latency directly in a web browser without any local installation, and simultaneously observe real-time visualization of multi-dimensional monitoring data, including reinforcement learning reward curves. Combined with a predefined robust command communication protocol, Web-Gewu provides a highly scalable, out-of-the-box, and barrier-free teaching infrastructure for embodied intelligence, significantly lowering the barrier to entry for cutting-edge robotics technology.

I. INTRODUCTION

With the explosive growth of embodied intelligence and robotics, complex dynamic robot control is entering the public consciousness at an unprecedented pace. However, the cutting-edge algorithms underpinning these systems—such as reinforcement learning (RL) and model predictive control—remain opaque to most students and educators. This cognitive gap creates strong demand for intuitive, accessible tools that can help non-specialist learners understand the core logic of these algorithms.

Integrating robotics into education is an effective approach, but the existing technological ecosystem erects significant barriers. On one hand, mainstream RL simulation platforms such as Isaac Lab [1], mjlab [2], Genesis, and Gewu Playground [3] impose stringent local GPU requirements or complex environment configuration, directly excluding learners with limited hardware resources and exacerbating inequality in access to cutting-edge technology. On the other hand, traditional centralized cloud simulation solutions, while bypassing local hardware limitations, incur prohibitive GPU and bandwidth costs, making large-scale deployment in universal teaching settings economically infeasible.

To break down this computational barrier and architectural dilemma, we present Web-Gewu (Figure 1): an interactive

robotics education sandbox built on a cloud-edge-client collaborative WebRTC [4] architecture. We innovatively restructure the computational topology by offloading all resource-intensive physics simulation and RL training entirely to the edge node, while the cloud server acts solely as a lightweight signaling relay to facilitate NAT traversal. Learners enjoy a truly zero-installation experience—no local GPU configuration or physical robot hardware is required—and simply open a browser to stream the simulation in real time via low-cost P2P connectivity.



Fig. 1. Main interface of Web-Gewu. It can be visited via <http://47.76.242.88:8080/receiver/index.html>.

Web-Gewu provides a purely virtual interactive environment with multiple built-in RL demonstration tasks. Through a low-latency streaming mechanism, the system transmits training and inference processes from the edge node to the client in real time, transforming the RL training process into intuitive visual feedback on the screen. Users can intervene in the behavior of bipedal, quadrupedal, and leg-wheeled robots in real time through a standard web browser with no perceptible lag.

The core contributions of this paper are as follows:

- 1) We develop Web-Gewu, a cross-platform, purely browser-based robot education platform that supports real-time control and gait verification of multi-form robots (bipedal, quadrupedal, and leg-wheeled) without any installation.
- 2) We propose a lightweight cloud-edge-client simulation architecture in which the edge node handles all heavy computation and the cloud provides only signaling relay, breaking the cost bottleneck of traditional high-fidelity cloud rendering and enabling universal low-latency remote interaction.
- 3) We construct a dynamic visualization closed loop

*This work was supported by the Science and Technology Commission of Shanghai Municipality (Grant No. 24511103304).

The authors are with the School of Future Technology, Shanghai University, Shanghai 200444, China. (Corresponding author: Linqi Ye, email: yelinqi@shu.edu.cn)

for RL algorithms by transmitting multi-dimensional monitoring data (reward curves, curriculum learning stages) in real time through the WebRTC DataChannel, providing learners with intuitive insight into the data-driven nature of reinforcement learning.

- 4) We establish a minimal scene integration contract within the layered framework: a new robot scenario can be onboarded without touching any WebRTC, signaling, or protocol code—it requires only a Unity scene runtime module, a `SceneDirector` registration, and at most one new typed command handler, making the platform straightforwardly extensible by the research community.

II. RELATED WORK

A. Interactive Educational Tools for AI Learning

The web-based architecture inherently offers zero-deployment, cross-platform, and instant-interaction advantages, making it an excellent medium for lowering AI education barriers. Lightweight interactive tools such as TensorFlow Playground [5], Reinforcement Learning Playground [6], and Interactive Deep RL Demo [7] have successfully popularized deep learning and RL concepts through intuitive 2D visualization in the browser, demonstrating the enormous potential of the web as a universal medium for AI education.

However, such systems are typically designed for low-dimensional or abstract tasks, focusing on algorithmic explanation rather than modeling real-world physical environments. The core of robot RL lies in the complex interaction between an agent and a continuous physical world, requiring a high-fidelity 3D physics engine for accurate rigid-body dynamics, collision feedback, friction, and joint constraint calculations. Traditional web tools, constrained by browser compute limitations, can only handle extremely simple tasks and cannot support the large-scale parallel sampling or high-frequency control loops required for robot RL. This creates a persistent structural contradiction between the extreme accessibility of the web and the high computational demands of robotic systems—a gap that Web-Gewu is designed to bridge.

B. Robot Simulation and RL Platforms

Debugging robot systems in a real physical environment is not only time-consuming but also incurs high trial-and-error costs and hardware damage risks. Building a high-fidelity, high-throughput simulation environment is therefore a prerequisite for converging robot RL algorithms.

Early robot simulation platforms such as Webots [8], Gazebo [9], and PyBullet advanced robot control and kinematics research significantly over the past decade. However, these platforms rely on CPU-based serial or limited-parallel physics solvers, revealing significant computational bottlenecks when large-scale environmental interaction data is required for RL training.

To overcome this, modern platforms such as Isaac Lab, mjlabs, and Genesis employ GPU-accelerated physics tensor

parallelism or highly optimized C++ back-ends to achieve massive-scale concurrent training of thousands of agents, compressing training time from weeks to hours. However, their closed rendering pipelines, stringent low-level system dependencies, and limited cross-platform compatibility make them unsuitable as universally accessible interactive tools for general education.

Gewu Playground [3], as a simulation foundation deeply integrated with the Unity ML-Agents Toolkit [10] [11], supports standard RL routines including complex whole-body imitation and omnidirectional locomotion. Researchers have successfully reproduced the Adversarial Motion Priors (AMP) [12] algorithm within Unity, demonstrating its readiness for high-difficulty robot RL. Critically for our work, Unity’s built-in WebRTC/Render Streaming pipeline enables high-fidelity simulation rendering to be streamed to a web browser with minimal overhead, while its rich ecosystem—covering sensor simulation, complex terrain generation, and URDF import for robotic systems—significantly reduces the engineering costs of environment setup. This integration of high-fidelity physics simulation with commercial-grade rendering and web streaming is the core enabler of Web-Gewu’s out-of-the-box, low-latency browser experience.

III. SYSTEM ARCHITECTURE

The Web-Gewu framework implements a low-latency cloud rendering and remote operation paradigm specifically designed for embodied intelligence education. By deeply decoupling heavy physics simulation from the client interface, the system maintains high-fidelity visual feedback while ensuring reliable bidirectional control communication across diverse network topologies.

A. Cloud-Edge-Client Topology

To optimize computational resource allocation and minimize end-to-end latency, Web-Gewu adopts a three-tier decoupled topology, as illustrated in Fig. 2.

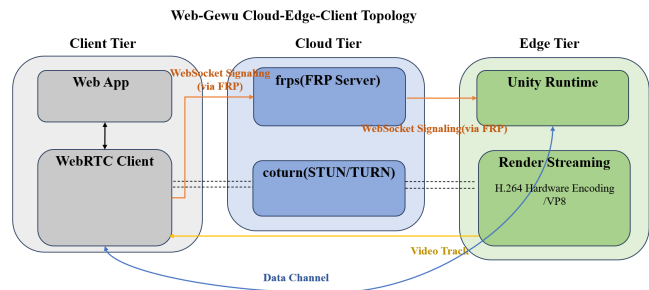


Fig. 2. The Cloud-Edge-Client topology of the Web-Gewu framework. The Edge Tier acts as the computation singularity, encapsulating high-fidelity physics simulation and H.264 hardware-accelerated render streaming. The Cloud Tier provides lightweight signaling relay (via FRP) and NAT traversal (via STUN/TURN). The Client Tier functions as a thin web interface, establishing low-latency P2P media (VideoTrack) and bidirectional control (DataChannel) pipelines.

Edge Tier (Compute Layer). The edge node—a local GPU-equipped workstation—hosts all computationally intensive tasks: the Unity runtime, high-fidelity rigid-body physics

simulation, RL training and inference logic, and hardware-accelerated WebRTC media encoding via Unity Render Streaming. No rendering or business logic is processed in the cloud.

Cloud Tier (Signaling Layer). A lightweight public cloud server is strictly restricted to signaling relay and NAT traversal support. It runs `frps` for reverse-proxy port mapping (exposing the edge-side Node.js signaling server on a public address) and `coturn` for STUN/TURN services. A standard 2-core vCPU instance with 2 GB RAM suffices for this role, sustaining CPU utilization of only 0.8%–1.1% with a single active client, demonstrating the economic viability of large-scale deployment.

Client Tier (Browser Layer). The learner accesses the system through a standard web browser with no installation required. Even a mobile device can seamlessly dispatch commands to the full edge computing stack. The browser-based WebApp handles UI interaction, command dispatch, and real-time telemetry visualization.

B. Low-Latency Render Streaming Pipeline

Web-Gewu leverages WebRTC [13] as the unified transport layer for both video streaming and bidirectional control. Unity serves as the sole rendering authority; the Render Streaming plugin captures the designated simulation camera, encodes each frame via hardware-accelerated H.264 (with VP8 as a fallback), and publishes it as a WebRTC `VideoTrack`. The browser-side client binds the decoded frames directly to an HTML5 `<video>` element for immediate display; no scene reconstruction occurs on the client side.

Connection establishment follows the standard ICE (Interactive Connectivity Establishment) procedure. Both the edge node and the browser independently gather ICE candidates and query the `coturn` STUN server to resolve their reflexive public-network addresses. The ICE agent then attempts a direct P2P UDP connection through NAT hole-punching. Under favorable network conditions, all media and control traffic flows directly between the edge and the client, producing end-to-end visual latency imperceptible to the operator and eliminating the motion-sickness-inducing lag of conventional remote-desktop approaches. When symmetric NAT or firewall policies prevent direct connectivity, the TURN relay in `coturn` transparently handles the session, maintaining 100% connection success across heterogeneous network environments while keeping the media path entirely independent of the cloud signaling infrastructure.

C. Robust Bidirectional Control via Envelope Protocol

The same WebRTC session that carries the simulation video also hosts a bidirectional `RTCDataChannel` named `input`. All control commands originating from the browser and all telemetry responses from Unity traverse this single logical channel, enabling tight synchronization between user actions and system state feedback.

To impose semantic structure and robustness on the message stream, Web-Gewu defines the *Envelope JSON* protocol.

Each message is wrapped in a typed envelope carrying a protocol version (`v`), a globally unique identifier (`id`), a semantic type string (`type`), a sender identifier (`source`), and a precise timestamp (`ts`), as shown below:

```
{ "v":1, "id":"env-m9x2...",
  "type":"scene.load", "source":"web",
  "ts":1710000000000,
  "payload":{"scene":"RoboHeTu" } }
```

The `type` field drives dispatching on both ends. Web-Gewu distinguishes two semantic command categories to maximize robustness under unreliable networks.

State-intent commands (e.g., `scene.load`, `training.set_flag`) express a desired system state rather than a strictly ordered action sequence. The Unity-side `SceneDirector` deduplicates these by ignoring requests for a scene already in the process of loading, preventing cascade failures from network jitter or repeated clicks.

Continuous control snapshots (e.g., robot movement direction and speed) represent the current instantaneous control state. Since each snapshot inherently supersedes all prior ones, the receiver discards stale packets and applies only the most recent state, naturally tolerating packet loss without retransmission overhead and eliminating control stream collapse under network backpressure.

The browser-side `RtcChannelClient` maintains an outbound buffer that retains commands issued before the `DataChannel` is open, flushing them atomically upon channel establishment. On the Unity side, all inbound WebRTC callbacks are enqueued onto a Unity main-thread processing queue and consumed sequentially in each `Update()` frame, preventing cross-thread scene mutations and ensuring command ordering.

D. Layered Engineering Architecture

Web-Gewu’s implementation is organized into five decoupled layers to achieve fault isolation, component reusability, and low-cost scenario extensibility:

- 1) **Transport Layer** (`rtc-channel.js` / `WebRtcModelCommandBridge.cs`): manages `DataChannel` lifecycle, JSON serialization, pre-connection buffering, and Unity main-thread scheduling. Scene logic is fully insulated from ICE/SDP details.
- 2) **Protocol Layer** (`envelope.js` / `Runtime/Handlers/`): defines message formats and dispatches incoming envelopes by `type`, maintaining backward compatibility with legacy command formats.
- 3) **Orchestration Layer** (`SceneDirector.cs`): manages scene loading, alias resolution, active-scene transitions, camera synchronization, deferred command replay, and in-flight deduplication.
- 4) **Scene Runtime Layer** (e.g., `GlmoeAgent`, `TinkercoinAgent`): executes physics simulation, RL training and inference, and produces typed telemetry payloads.

- 5) **UI Interaction Layer** (`scene-controller.js`): maps user gestures to semantic envelope payloads and renders incoming telemetry; it carries no Unity business logic.

This layered design deliberately minimizes the integration surface for new simulation scenarios. A new scene author need not understand ICE candidate gathering, SDP negotiation, DataChannel open timing, or browser compatibility quirks—those concerns are fully absorbed by the Transport and Protocol layers. In concrete terms, onboarding a new scenario requires satisfying only four contracts: (1) register the scene with `SceneDirector` so it can be loaded and aliased; (2) expose a bindable runtime object through which the Orchestration Layer can forward commands; (3) reuse existing `Envelope type` strings where semantics match, or add a single new typed handler if they do not; and (4) optionally define a telemetry payload for any data the scene wishes to stream back to the browser. Crucially, the typed-dispatch design means extending the protocol never requires rewriting a monolithic command parser—it is strictly additive. Scene lifecycle concerns such as deduplication of repeated load requests, deferred command replay while a scene is still initializing, and camera synchronization on transition are all handled by `SceneDirector` unconditionally, so new scenes inherit robust behavior without any additional engineering effort.

The scene routing topology is illustrated in Fig. 3. The `GlobalManager` acts as a central dispatcher: it accepts a unified instruction stream from both the browser client and the local Unity editor, decodes each command using the `Envelope` protocol, and dynamically routes it to the appropriate scene runtime—*WebRL Lab* for policy validation and model switching, *RoboHeTu* for complex terrain navigation, or *TinkerCoin* for curriculum learning demonstration. This single-entry-point design ensures that scene transitions are atomic and idempotent regardless of the command source.

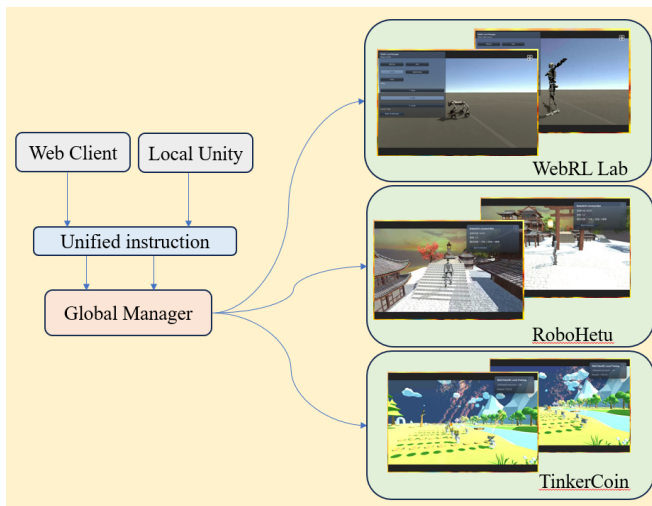


Fig. 3. Unified scene routing in Web-Gewu. The `GlobalManager` decodes a unified command stream from both the Web client and the local Unity runtime, dynamically dispatching to one of three specialized educational scenes.

IV. RESULTS

A video demo of Web-Gewu that demonstrates all functionalities is available at: <https://linqi-ye.github.io/video/web-gewu.mp4>

A. Infrastructure Overhead

A key design goal of Web-Gewu is to minimize cloud infrastructure cost by delegating all computation to the edge. To validate this, we measured the CPU utilization of the cloud signaling server (a 2-core vCPU, 2 GB RAM virtual machine running `frps` and `coturn`) during active sessions. Utilization remained stable at 0.8%–1.1% with a single active client, confirming that the cloud tier is a purely lightweight relay whose cost is independent of simulation complexity or fidelity. This contrasts sharply with conventional centralized cloud rendering, where GPU and bandwidth costs scale directly with the number of concurrent sessions and the rendering workload.

B. Educational Sandbox Showcase

Web-Gewu currently includes three built-in simulation scenarios covering representative robot morphologies and RL training paradigms. Fig. 4 illustrates the system in operation: the left panel shows the high-fidelity physics simulation running on the edge node inside the Unity editor, while the right panel shows the browser-based Web client receiving the mirrored live stream and presenting an intuitive control panel—all synchronized through the cloud relay with minimal perceptible latency.

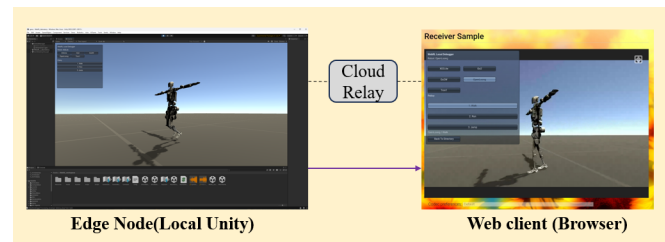


Fig. 4. Cross-platform synchronization demonstration of the Web-Gewu educational sandbox. The browser-based Web client (right) abstracts the complex RL environment into an intuitive control panel and mirrors the high-fidelity simulation running on the local edge node (left) via the cloud relay with minimal end-to-end latency.

Playground. A model inference showcase in which pre-trained RL policy weights for various robot platforms are loaded and their behaviors streamed in real time. Learners can interactively switch between trained models and observe policy differences in a controlled setting, providing an accessible entry point for understanding policy-driven behavior.

RoboHeTu. A navigation and locomotion demonstration featuring the G1 quadrupedal robot in a virtual environment inspired by the Song Dynasty painting *Along the River During the Qingming Festival*. Users can switch between walk, run, and cross-terrain locomotion modes and issue omnidirectional movement commands via keyboard input, receiving real-time visual feedback of gait adaptation.

TinkerCoin. A live RL training demonstration in which the TinkerPro bipedal robot is trained from scratch using a physically grounded curriculum learning strategy. Inspired by the assistive force formulation proposed in HoST [14], our curriculum applies a directional assistive force to the robot’s torso, directed vertically upward with a 5° forward tilt relative to the robot’s current heading—the tilt introduces a small tangential component that gently encourages forward locomotion without artificially constraining the policy.

The assistive force magnitude is governed by a scalar curriculum coefficient $\lambda \in [0, 1]$, where $\lambda = 1.0$ corresponds to an upward support force equal to 50% of the robot’s body weight ($F_{\uparrow} = \lambda \cdot 0.5 \text{ mg}$). The coefficient follows a piecewise-linear decay schedule:

$$\lambda(t) = \max\left(0, 1.0 - 0.2 \cdot \max\left(0, \left\lfloor \frac{t - 5 \times 10^5}{10^5} \right\rfloor\right)\right) \quad (1)$$

That is, λ remains at 1.0 for the first 500,000 training steps, then decreases by 0.2 for each subsequent 100,000 steps until it reaches 0.0 at step 1,000,000. Under this schedule, the robot first learns to balance and stride with substantial support, then progressively adapts to unsupported walking as the artificial aid is withdrawn.

In practice, the robot achieves stable forward locomotion in approximately 3 minutes of wall-clock time and fully unsupported continuous walking in approximately 8 minutes. The cumulative reward and episode length curves, streamed in real time to the browser and shown in Fig. 5, reveal the training dynamics clearly. The cumulative reward initially decreases as the agent begins using the full episode length and accumulates fall penalties, then rises monotonically once balance stabilizes, crosses zero near step 300,000, and reaches approximately +600 by step 550,000. The episode length curve rises from near zero to the maximum episode horizon ($\sim 1,000$ steps) by step 250,000, indicating that the robot sustains upright locomotion for full episodes well before the curriculum support is withdrawn. These real-time curves provide learners with an unambiguous, data-driven view of the exploration-to-exploitation transition central to RL.

Collectible coin assets are distributed throughout the scene; as the policy improves, the robot reaches and collects progressively more coins, providing a gamified progression that reinforces the concept of reward-driven policy improvement.

C. End-to-End System Workflow

The operational workflow is designed to minimize setup overhead for both instructors and learners. The cloud server (`frps + coturn`) runs as a persistent service requiring no per-session reconfiguration. On the edge side, the instructor launches `frpc` to expose the local signaling port on the public address, then starts the Unity runtime, which initializes to the `GlobalManager` bootstrap scene. Once these services are running, any learner can access the system by opening a single URL in a standard web browser. WebRTC ICE negotiation completes automatically, establishing either

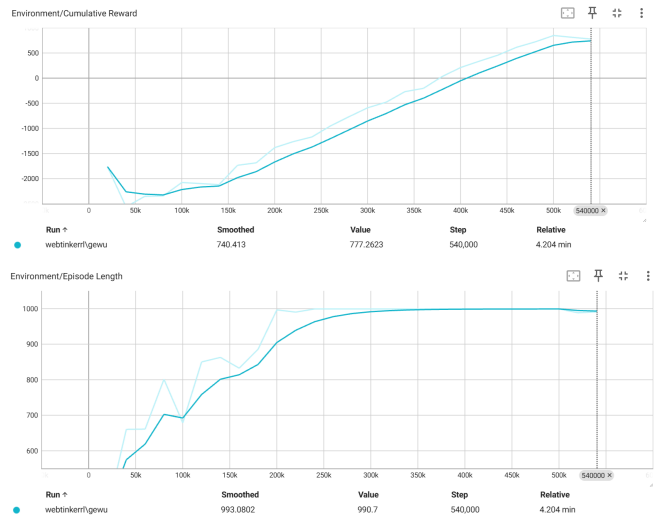


Fig. 5. TinkerCoin training curves streamed live to the browser. *Top*: cumulative reward over training steps. The reward dips initially as the agent explores with full episode length, then rises steadily as locomotion stabilizes. *Bottom*: episode length converges to the maximum horizon ($\sim 1,000$) by $\sim 250k$ steps, confirming sustained upright walking well before the curriculum support is fully withdrawn at step $\sim 100k$.

a direct P2P connection or a TURN-relayed fallback as network conditions dictate. The learner selects a scenario from the WebApp UI; `SceneDirector` handles scene transition, camera synchronization, and runtime binding without further manual intervention.

V. CONCLUSIONS

We have presented Web-Gewu, a browser-based interactive platform for robot reinforcement learning education built on a cloud-edge-client WebRTC collaborative architecture. By strictly delegating all physics simulation and RL computation to the edge node and confining the cloud tier to pure signaling relay, Web-Gewu resolves the long-standing tension between computational accessibility and high-fidelity simulation: a 2-core cloud server sustaining less than 1.1% CPU utilization suffices to support multi-user sessions, while learners interact with complex robot simulations from any device capable of running a modern browser.

The Envelope JSON protocol and five-layer engineering architecture provide a robust, extensible foundation for bidirectional control and real-time telemetry visualization. The three built-in educational scenarios demonstrate the platform’s ability to support both policy inference demonstrations and live from-scratch RL training, with real-time reward visualization serving as an intuitive bridge between abstract algorithm concepts and observable robot behavior. Beyond the three scenarios presented here, the framework’s minimal scene integration contract—register, bind, and optionally add one typed handler—means that any new Unity-based robot scenario can be incorporated without modifying the transport, signaling, or protocol layers, positioning Web-Gewu as an open, community-extensible platform rather than a fixed demonstration tool.

Future work will focus on two directions: (1) opening a custom model weight upload interface to allow instructors and learners to deploy their own trained policies directly into the platform; and (2) integrating a hardware bridge to real-world robot, enabling Sim2Real deployment demonstrations in which policies trained in Web-Gewu are transferred directly to physical hardware, closing the loop between simulation-based education and real-world embodied intelligence.

REFERENCES

- [1] M. Mittal et al., “Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning,” *arXiv preprint arXiv:2511.04831*, 2025.
- [2] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel, “mjlab: A Lightweight Framework for GPU-Accelerated Robot Learning,” *arXiv preprint arXiv:2601.22074*, 2026.
- [3] L. Ye, B. Xing, B. Liang, L. Jiang, and Y. Peng, “Gewu Playground: An Open-Source Robot Simulation Platform for Embodied Intelligence Research,” *Sci. China Technol. Sci.*, 2026, doi: 10.1007/s11431-025-3253-2.
- [4] A. Gouaillard and L. Roux, “Real-time communication testing evolution with WebRTC 1.0,” in *Proc. 2017 Princ., Syst. Appl. IP Telecommun. (IPTComm)*, 2017, pp. 1–8.
- [5] D. Smilkov, S. Carter, D. Sculley, F. B. Viégas, and M. Wattenberg, “Direct-Manipulation Visualization of Deep Networks,” *CoRR*, vol. abs/1708.03788, 2017.
- [6] A. Lazareva, “Reinforcement Learning Playground,” [Online]. Available: https://alazareva.github.io/rl_playground/, accessed Apr. 18, 2026.
- [7] P. Germon, C. Romac, R. Portelas, and P.-Y. Oudeyer, “Interactive Deep Reinforcement Learning Demo,” [Online]. Available: https://developmentalsystems.org/Interactive_DeepRL_Demo/, accessed 2021.
- [8] O. Michel, “Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation,” *Int. J. Adv. Robot. Syst.*, vol. 1, no. 1, pp. 39–42, 2004.
- [9] N. Koenig and A. Howard, “Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2004, pp. 2149–2154.
- [10] A. Nandy and M. Biswas, “Unity ml-agents,” in *Neural Networks in Unity: C# Programming for Windows 10*, Springer, 2018, pp. 27–67.
- [11] L. Almón-Manzano, R. Pastor-Vargas, and J. M. C. Troncoso, “Deep reinforcement learning in agents’ training: Unity ML-agents,” in *Proc. Int. Work-Conf. Interplay Natural Artif. Comput.*, Springer, 2022, pp. 391–400.
- [12] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, “AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control,” *ACM Trans. Graph.*, vol. 40, no. 4, Jul. 2021.
- [13] A. Bergkvist, D. Burnett, C. Jennings, A. Narayanan, B. Aboba, T. Brandstetter, and J. I. Bruaroey, “WebRTC 1.0: Real-Time Communication Between Browsers,” W3C Recommendation, Jan. 2021.
- [14] T. Huang et al., “Learning Humanoid Standing-up Control across Diverse Postures,” *arXiv preprint arXiv:2502.08378*, 2025.