GROW: ALIGNING GRPO WITH STATE-ACTION MODELING FOR OPEN-WORLD VLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, Group Relative Policy Optimization (GRPO) has been widely adopted to train vision-language model (VLM) agents for diverse tasks. However, the context lengths of trajectory samples are too long for training VLM agents with GRPO to execute open-world tasks, which usually consumes too many training resources to remain practical. To address this issue, we propose GROW, a reinforcement learning (RL) framework for open-world VLM agents that decomposes trajectory samples into state-action samples, whose context lengths are shorter than those of trajectory samples, thereby reducing the required training resources. After decomposition, GROW performs credit assignment with temporal discounting on state-action samples, thereby converting sparse reward signals into dense ones to guide efficient learning, and it then computes advantages among the state-action samples within the same rollout groups. We further provide a surrogate analysis indicating that GROW retains the training signals for the whole trajectory samples in the optimization objective that can reinforce the successful trajectory samples and suppress the failed ones, while avoiding direct training on trajectory samples. Experiments on more than 800 Minecraft tasks show that our method can achieve higher average success rates and fewer steps than recent state-of-the-art approaches.

1 INTRODUCTION

Recent studies on VLM (Luo et al., 2026; Li et al., 2026; Hu et al., 2026; Zhang et al., 2025b; Ye et al., 2025; Shi et al., 2025) that use GRPO (Shao et al., 2024) as their RL algorithms have shown the effectiveness in improving foundation-model policies through group-wise relative optimization. By comparing multiple sampled outputs within the same rollout group, GRPO constructs relative advantages without training an additional value model, making it especially suitable for large-scale VLM optimization where value estimation can be costly and unstable. These strengths make GRPO (Shao et al., 2024) a natural algorithmic basis for refining VLM agents through interaction with the environment. However, directly transferring GRPO (Shao et al., 2024) to open-world tasks is nontrivial. In these domains, an agent must repeatedly interpret visual states, choose actions, and adapt its behavior through multi-turn interaction with the environment. Most recent efforts improve such agents through SFT on expert demonstrations (Ouyang et al., 2026; Wang et al., 2025), enabling them to imitate task-relevant perception-action behaviors from curated demonstrations. However, supervised fine tuning (SFT) relies on large amounts of high-quality expert data, whose collection is often expensive and difficult to scale. Moreover, prior studies (Shi et al., 2025; Ye et al., 2025; Zhang et al., 2025b) have shown that SFT alone can lag behind RL-trained VLM agents in performance. These limitations motivate the need for advanced RL methods to train open-world VLM agents effectively.

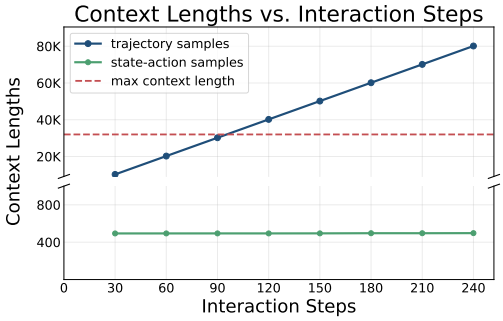


Figure 1: Context lengths of different types of samples in open-world tasks. The red line represents the max context length. The blue line represents the context lengths of trajectory samples. The green line represents the context lengths of state-action samples. The context lengths of trajectory samples often exceed the max context length as interactions accumulate, which can lead to GPU out-of-memory errors. In contrast, the context lengths of state-action samples are always below the max context length, avoiding this problem.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

When GRPO (Shao et al., 2024) is directly applied, it computes advantages across trajectory samples conditioned on the same prompts and uses trajectory samples to optimize the policies of VLM agents. As shown in Figure 1, trajectory samples of open-world tasks often exceed the maximum context length and impose substantial demands on training resources.

To address this issue, we propose GRPO for open-world VLM agents (GROW), an RL framework that adapts GRPO to open-world tasks where the context lengths of trajectory samples are often too long to train VLM agents directly with trajectory samples. GROW first applies trajectory decomposition to trajectory samples to transform them into state-action samples, and performs credit assignment with temporal discounting on these state-action samples. And then it computes relative advantages among state-action samples within the same rollout groups to optimize the policy. In addition, under simplifying assumptions, we provide a surrogate analysis showing that GROW retains trajectory training signals in its optimization objective, thereby improving the task-execution performance of VLM agents without direct training on trajectory samples.

Our main contributions are summarized as follows:

- We propose GROW, an RL framework for open-world VLM agents. To alleviate the long context lengths of trajectory samples, GROW decomposes them into state-action samples. It then performs credit assignment with temporal discounting on these samples so as to convert sparse reward signals into dense ones for efficient learning, followed by advantage computation among state-action samples within the same rollout groups.
- We provide the surrogate analysis suggesting that relative policy optimization remains effective in GROW, as its objective preserves the trajectory training signals that reinforce successful trajectory samples and suppress failed ones, without direct training on trajectory samples.
- We instantiate and test GROW mainly in Minecraft (Zheng et al., 2025; Li et al., 2025a; Bai et al., 2025). Across more than 800 Minecraft tasks, GROW achieves state-of-the-art performance in both success rate and execution efficiency, demonstrating the effectiveness of GROW for open-world tasks.

2 Related Work

2.1 Agents in the Open World

Open-world tasks provide an important testbed for developing general-purpose agents because they require agents to perceive visual observations, reason about dynamic environments, and execute actions over extended interaction sequences. Minecraft is a representative example due to its high degree of freedom and broad task diversity, spanning embodied control, resource manipulation, and precise graphical user interface interactions. Following the seminal work of VPT (Baker et al., 2022) which leverages large-scale expert demonstrations and explores RL for fine-tuning, subsequent studies have advanced agent learning in several directions. STEVE-1 (Lifshitz et al., 2023) is trained for text-to-behavior generation in Minecraft, enabling users to use text instructions to control agents for completing short-horizon and open-ended tasks relying on raw pixels and low-level controls. ROCKET-1 (Cai et al., 2025b) introduces visual-temporal context prompting to connect high-level VLM reasoning with low-level policy execution for spatially grounded interaction. ROCKET-3 (Cai et al., 2025a) further improves exploration in unseen environments through RL with cross-view reasoning. More recently, research has increasingly focused on VLM agents. JARVIS-VLA (Li et al., 2025a) adopts staged training to improve task completion in Minecraft, while similar imitation-based VLM agents have also shown effectiveness in other game environments such as *Genshin Impact* (Tan et al., 2025) and Steam games (Magne et al., 2026). Despite this progress, efficient RL methods for VLM agents remain underexplored, as most existing approaches still depend heavily on imitation learning. Our work addresses this gap by providing a scalable RL framework for training open-world VLM agents.

2.2 GRPO for Multi-Turn VLM Agents

GRPO (Shao et al., 2024) has been widely explored for RL in tasks requiring many rounds of interaction. AgentGym-RL (Xi et al., 2025) studies multi-turn RL for large language model agents and

improves long-horizon decision making with a curriculum over interaction length. InquireMobile (Ai et al., 2026) and ColorAgent (Li et al., 2025b) extend this paradigm to settings where the agent must request authorization before taking actions or incorporate human instructions during execution. AGENTRL (Zhang et al., 2025a) further investigates system-level scheduling and resource allocation for GRPO-based multi-turn training in order to improve training efficiency. However, although these works substantially broaden the study of GRPO in multi-turn settings, they mainly retain a trajectory-level or dialogue-level formulation, where each optimization sample may contain an increasingly long interaction history. When directly applied to open-world tasks, such full-trajectory samples can lead to excessive context accumulation as trajectories grow longer, which usually consumes too many training resources to remain practical. Our work addresses this issue by decomposing trajectories into state-action samples, and further provides a surrogate analysis showing that in the optimization objective, GROW retains the training signals for the whole trajectory, which can reinforce successful trajectory samples and suppress failed ones, while avoiding direct training on trajectory samples.

3 METHOD

3.1 GROW FRAMEWORK

Figure 2 provides an overview of GROW. When applying GROW to train VLM agents, the VLM agents interact with the open-world environments, producing rollout groups, and then the resulting trajectory samples are decomposed into state-action samples. After decomposition, credit assignment with temporal discounting is performed such that all state-action samples can receive a proper reward used to compute relative advantages with other samples in the same rollout groups.

We formulate the process of executing open-world tasks as a Markov decision process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$. $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The details of $\mathcal{S}$ and $\mathcal{A}$ are provided in Appendix A. P is the transition function of the environment. $\mathcal{R}$ is the episode reward function, which returns 1 when the trajectory completes the task c successfully and 0 otherwise. γ is the temporal discount factor. Given a task $c \in \mathcal{C}$, where $\mathcal{C}$ is the set of open-world tasks, the policy of the VLM agent π_θ interacts with the open-world environments, with each task c run in parallel in G environments, and each environment produces one trajectory sample τ_i with $i = 1, \ldots, G$. At the timestep t of τ_i, the policy of the VLM agent π_θ receives a state $s_{i,t}$ and the task c, and decides the optimal action $a_{i,t}$ to execute the task c. This iterative process continues until the task c is completed or the timestep reaches the limit, which yields a group of trajectory samples denoted as $\{\tau_i\}_{i=1}^{G}$. Let H_i be the length of trajectory τ_i, which is the maximum timestep in the trajectory. Then the trajectory sample τ_i is collected as a sequence of state-action samples:

$$\tau_i = \{(s_{i,1}, a_{i,1}), \ldots, (s_{i,H_i}, a_{i,H_i})\} \tag{1}$$

Then, we decompose each trajectory sample τ_i into H_i state-action samples with one sample denoted as $(s_{i,t}, a_{i,t})$ at timestep t. Because the original episode rewards are assigned to trajectory samples and the rewards for state-action samples are sparse, we propagate the sparse episodic reward backward along each trajectory sample with the temporal discount factor γ:

$$r_{i,t} = \gamma^{H_i - t} \mathcal{R}(\tau_i), \tag{2}$$

This credit assignment strategy can assign larger rewards to later state-action samples from successful trajectory samples to improve the efficiency of executing tasks, whereas those state-action samples from failed trajectory samples receive zero reward. The group of trajectory samples is therefore transformed into a group of state-action samples with rewards $\{(s_{i,t}, a_{i,t}, r_{i,t}) \mid i \in [1, G],\ t \in [1, H_i]\}$. In this way, GROW changes the training samples from trajectory samples to state-action samples, substantially reducing the context length of each training sample and thereby lowering the demands on training resources.

Unlike standard GRPO (Shao et al., 2024), where relative advantages are computed among trajectory samples in the same rollout group, GROW computes relative advantages over decomposed state-action samples in the same rollout group. As illustrated in Figure 2, the rewards are normalized among state-action samples derived from the same rollout group of trajectory samples to obtain the advantage

$$A_{i,t} = \frac{r_{i,t} - \mu}{\sigma}, \tag{3}$$

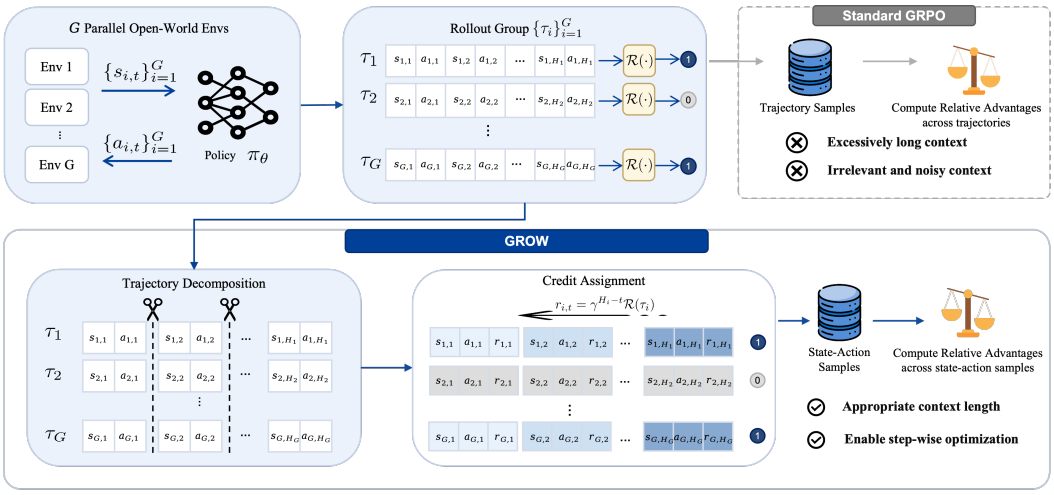


Figure 2: Overview of GROW. The VLM agents interact with the open-world environments, thereby generating rollout groups of trajectory samples that are subsequently decomposed into state-action samples. Once decomposition is completed, credit assignment with temporal discounting is performed, allowing every state-action sample to obtain an appropriate reward that is then used to compute relative advantages against other samples within the same rollout groups.

where μ and σ are the mean and standard deviation of the rewards of the state-action samples in the sample groups $\{r_{i,t} \mid i \in [1, G],\ t \in [1, H_i]\}$. This leads to the objective of GROW:

$$\mathcal{J}(\theta) = \mathop{\mathbb{E}}_{\substack{c\sim\mathcal{C} \\ a_{i,t}\sim\pi_{\theta_{\text{old}}}(\cdot|s_{i,t},c)}} \left\{ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{H_i}\sum_{t=1}^{H_i} \min\left(\rho_{i,t}(\theta)A_{i,t}, \text{clip}(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon)A_{i,t}\right) \right\} \quad (4)$$

where ϵ is the clip ratio, $\rho_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t}|s_{i,t},c)}{\pi_{\theta_{\text{old}}}(a_{i,t}|s_{i,t},c)}$ is the probability ratio and $\pi_{\theta_{\text{old}}}$ is the old policy.

3.2 Surrogate Analysis

In this section, we provide a surrogate analysis showing that computing relative advantages over decomposed state-action samples can retain the training signal for the whole trajectory samples that can reinforce successful trajectory samples and suppress failed ones. Since, within each group, the lengths of trajectory samples required to complete the same task are generally similar, and the lengths of trajectory samples are bounded by the same limit, we approximate the trajectory length for task c by a uniform constant H, i.e., $H_i \approx H$. Statistical evidence for this approximation is provided in Appendix B. Under this mild assumption, the mean reward over all decomposed state-action samples in the rollout group can be written as

$$\mu = \frac{1}{\sum_{i=1}^{G} H_i}\sum_{i=1}^{G}\sum_{t=1}^{H_i} r_{i,t} \approx \frac{1}{GH}\sum_{i=1}^{G}\mathcal{R}(\tau_i)\sum_{t=1}^{H}\gamma^{H-t}$$

We further define

$$C_\gamma = \frac{1}{H}\sum_{t=1}^{H}\gamma^{H-t} = \frac{1-\gamma^H}{H(1-\gamma)}, \qquad \gamma \in (0,1) \quad (5)$$

which is the average temporal discount coefficient within a trajectory sample. Because $\gamma \in (0,1)$, it immediately follows that $0 < C_\gamma < 1$. Let S denote the average trajectory-level return in the current rollout group. Then we have $S = \frac{1}{G}\sum_{i=1}^{G}\mathcal{R}(\tau_i)$. Then the mean reward admits the compact form $\mu \approx C_\gamma S$.

For clarity, we analyze the proxy advantages $\hat{A}_{i,t}$ only with the centered reward $r_{i,t} - \mu$, omitting the standard-deviation term σ, which acts as a shared positive scaling factor within each rollout group and

thus does not affect the preference. Under this simplification, the proxy advantages for state-action samples become

$$\hat{A}_{i,t} = r_{i,t} - \mu = \gamma^{H_i - t}\mathcal{R}(\tau_i) - C_\gamma S \tag{6}$$

Substituting Equation 6 into the objective in Equation 4 (clipping is omitted for brevity) gives

$$\begin{aligned}
\mathcal{J} &= \mathop{\mathbb{E}}_{\substack{c\sim\mathcal{C} \\ a_{i,t}\sim\pi_{\text{old}}(\cdot|s_{i,t},c)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{H_i}\sum_{t=1}^{H_i}\rho_{i,t}\left(\gamma^{H_i-t}\mathcal{R}(\tau_i) - C_\gamma S\right)\right] \\
&= \mathop{\mathbb{E}}_{\substack{c\sim\mathcal{C} \\ a_{i,t}\sim\pi_{\text{old}}(\cdot|s_{i,t},c)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{H_i}\sum_{t=1}^{H_i} C_\gamma\rho_{i,t}\big(\mathcal{R}(\tau_i) - S\big)\right] \\
&\quad + \mathop{\mathbb{E}}_{\substack{c\sim\mathcal{C} \\ a_{i,t}\sim\pi_{\text{old}}(\cdot|s_{i,t},c)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{H_i}\sum_{t=1}^{H_i}\rho_{i,t}\left(\gamma^{H_i-t} - C_\gamma\right)\mathcal{R}(\tau_i)\right]
\end{aligned} \tag{7}$$

For convenience, we define

$$\mathcal{J}_{\text{traj}} = \mathop{\mathbb{E}}_{\substack{c\sim\mathcal{C} \\ a_{i,t}\sim\pi_{\text{old}}(\cdot|s_{i,t},c)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{H_i}\sum_{t=1}^{H_i}\rho_{i,t}\big(\mathcal{R}(\tau_i) - S\big)\right] \tag{8}$$

$$\mathcal{J}_{\text{step}} = \mathbb{E}_{\substack{c\sim\mathcal{C} \\ a_{i,t}\sim\pi_{\text{old}}(\cdot|s_{i,t},c)}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{H_i}\sum_{t=1}^{H_i}\rho_{i,t}\left(\gamma^{H_i-t} - C_\gamma\right)\mathcal{R}(\tau_i)\right]. \tag{9}$$

Then Equation 7 can be rewritten as $\mathcal{J} = C_\gamma\mathcal{J}_{\text{traj}} + \mathcal{J}_{\text{step}}$. In Equation 8, $\mathcal{J}_{\text{traj}}$ compares the relative quality of different trajectory samples by weighting each trajectory sample according to the deviation of its episode reward from S, which serves as the baseline. In a manner analogous to GRPO, trajectory samples within the same rollout group are evaluated against one another, so that trajectory samples outperforming the baseline are reinforced while those falling below it are suppressed. Therefore, it reinforces successful trajectories at the trajectory level. In addition, the objective incorporates a term $\mathcal{J}_{\text{step}}$ that captures temporal-discounting effects exclusively on successful trajectory samples, thereby achieving the temporal-discounting effect in Equation 2 and improving execution efficiency. In conclusion, without training directly on trajectory samples, GROW can reinforce the successful trajectory samples while suppressing the failed ones by optimizing $\mathcal{J}_{\text{traj}}$, and can also improve the execution efficiency by optimizing $\mathcal{J}_{\text{step}}$.

4 Experiments

4.1 Experimental Setup

Environment We conduct experiments in Minecraft (Java Edition, v1.16.5). In Minecraft, the agent's observation space is strictly limited to first-person view RGB images with a resolution of $360 \times 640 \times 3$, without access to any auxiliary state information. The action space consists of discretized human-like interface commands, including mouse movements, clicks, and keyboard inputs, to simulate authentic human gameplay.

Benchmarks and Task Categorization To comprehensively assess the agent's multimodal understanding and fine-grained manipulation capabilities, we use the MCU benchmark (Zheng et al., 2025), which includes over 800 tasks. The benchmark comprises distinct task categories, each targeting specific skill sets: (1) *Embodied tasks*: The agent needs to navigate to the position of the target blocks and then use tools to mine or chop blocks. This task focuses on 3D spatial awareness and navigation. Agents must identify target textures within a complex voxel environment and utilize specific tools to excavate target blocks. (2) *GUI tasks*: The agent is requested to create the target items with a furnace or crafting table. This task tests precise 2D grid manipulation and logical reasoning to convert materials. It further introduces temporal constraints, requiring agents to manage wait times during smelting processes. (3) *Combat tasks*: The agent needs to kill the target entities. This task focuses on adversarial strategies in highly dynamic environments. Unlike static tasks, this requires the agent to

Table 1: Evaluation results of Minecraft agents on over 800 tasks. For each task category, we report two metrics: the average steps to finish tasks and the average success rate across all tasks in the category (ASR) with standard deviation. The best performance is marked in **bold**, and the second-best performance is underlined.

Model	Size	Embodied Tasks		GUI Tasks		Combat Tasks	
		Steps	ASR	Steps	ASR	Steps	ASR
VPT (Baker et al., 2022)	248M	377	$6.0^{\pm 11.4}$	398	$0.8^{\pm 3.3}$	396	$3.6^{\pm 7.7}$
STEVE-1 (Lifshitz et al., 2023)	248M	384	$8.0^{\pm 17.0}$	391	$3.2^{\pm 8.4}$	395	$3.9^{\pm 12.0}$
ROCKET-1 (Cai et al., 2025b)	72B	392	$18.9^{\pm 24.3}$	N/A	0.0	320	$27.9^{\pm 29.3}$
JARVIS-VLA (Li et al., 2025a)	7B	305	$30.0^{\pm 35.4}$	339	$25.1^{\pm 23.9}$	352	$18.5^{\pm 22.7}$
UI-TARS-1.5 (Seed, 2025)	7B	290	$42.1^{\pm 20.4}$	320	$36.7^{\pm 17.2}$	346	$31.0^{\pm 16.4}$
OpenHA (Wang et al., 2026)	7B	287	$30.1^{\pm 13.9}$	314	$32.5^{\pm 9.2}$	316	$31.9^{\pm 13.7}$
Game-TARS (Wang et al., 2025)	7B	373	$\underline{50.4}^{\pm 20.7}$	406	$\underline{39.1}^{\pm 27.5}$	372	$38.1^{\pm 4.6}$
MAIN-VLA (Zhou et al., 2026)	7B	<u>263</u>	$32.8^{\pm 15.4}$	<u>291</u>	$34.4^{\pm 14.4}$	<u>248</u>	$\underline{39.2}^{\pm 16.2}$
Ours	**7B**	**128**	$\mathbf{59.6}^{\pm 37.2}$	**248**	$\mathbf{68.4}^{\pm 29.0}$	**172**	$\mathbf{49.0}^{\pm 35.7}$

track and defeat actively hostile moving targets. Note that during evaluation, the environments are initialized with different seeds from those used in training, thereby avoiding overlap with training initial states and ensuring a fair evaluation of generalization.

Evaluation Metrics We follow (Wang et al., 2026; Zhou et al., 2026) and use two metrics to assess agent performance and efficiency: (1) *Steps*, the average number of interaction steps required to finish the tasks; and (2) *ASR*, the average success rate over all tasks in a category. Each task is evaluated over at least three independent episodes. We report ASR as mean $\pm$ standard deviation across tasks to reflect the heterogeneity of task difficulty.

4.2 Implementation Details

Given that most VLM agents in Table 1 are built upon Qwen2-VL-7B-Instruct (Wang et al., 2024), we also adopt it as the base model for our agent to ensure a fair comparison. We utilize 3M state-action samples to initialize the VLM agent using the LLaMA-Factory framework (Zheng et al., 2024) with 8 H200 GPUs. We select eight representative tasks for RL training: two navigation-and-mining tasks (*mine diorite*, *mine birch log*), four crafting/smelting tasks (*craft furnace*, *craft iron pickaxe*, *smelt cooked porkchop*, *smelt gold ingot*), and two combat tasks against ground-moving and flying targets (*kill skeleton*, *kill blaze*). Task success is automatically determined by a built-in environment verifier, which monitors game state transitions such as inventory changes and entity defeat events to produce binary rewards without human intervention. We train and evaluate in Minecraft worlds with disjoint seeds. The RL pipeline is implemented upon the open-source verl framework (Sheng et al., 2025) to facilitate efficient distributed optimization. We conduct all training procedures on a single compute node equipped with 8 H200 GPUs. The model undergoes optimization for 240 iterations with $\gamma = 0.995$. More details about the dataset for policy initialization and hyperparameters are in Appendix E and F.

4.3 Main Results

Table 1 shows that GROW achieves the best performance across all three Minecraft task categories. In embodied tasks, our agent improves the previous best ASR from 50.4% to 59.6%. This improvement suggests that GROW strengthens the agent's ability to navigate toward target objects after detection and to persistently use tools until the task is completed. In GUI tasks, our RL framework yields the largest gain, improving ASR from 39.1% to 68.4%. This gain mainly comes from improved recognition of target items and better mastery of the key interaction procedures in crafting and smelting. In combat tasks, our agent improves ASR from 39.2% to 49.0%, indicating stronger decision-making in highly dynamic environments where the agent must continuously track enemies, adjust its position, and select appropriate actions under changing visual states.

Table 2: Evaluation results on In-Distribution (ID) tasks and Out-of-Distribution (OOD) tasks. We report the ASR with standard deviation and the best performance is marked in **bold**.

	ID tasks			OOD tasks		
Model	Embodied Tasks	GUI Tasks	Combat Tasks	Embodied Tasks	GUI Tasks	Combat Tasks
Initialized Policy	$35.0^{\pm 7.1}$	$10.0^{\pm 8.2}$	$45.0^{\pm 21.2}$	$41.9^{\pm 34.7}$	$21.3^{\pm 24.8}$	$19.8^{\pm 24.4}$
Ours	$\mathbf{90.0}^{\pm 14.1}$	$\mathbf{92.5}^{\pm 5.0}$	$\mathbf{85.0}^{\pm 21.2}$	$\mathbf{59.2}^{\pm 37.4}$	$\mathbf{68.2}^{\pm 29.1}$	$\mathbf{47.5}^{\pm 35.9}$

Table 3: Ablation study of RL methods. We report the steps and ASR of all 800 Minecraft tasks for trajectory-level GRPO, PPO, GROW and the initialized policy. All RL methods are trained starting from the same initialized policy.

Method	Embodied Tasks		GUI Tasks		Combat Tasks	
	Steps	ASR	Steps	ASR	Steps	ASR
Initialized Policy	189	$41.8^{\pm 34.3}$	498	$21.2^{\pm 24.7}$	233	$20.8^{\pm 24.3}$
Trajectory-level GRPO	238	$44.1^{\pm 34.7}$	379	$13.1^{\pm 18.0}$	207	$32.6^{\pm 30.5}$
PPO	177	$58.2^{\pm 30.3}$	580	$43.9^{\pm 24.3}$	181	$45.4^{\pm 32.2}$
GROW	**128**	$\mathbf{59.6}^{\pm 37.2}$	**248**	$\mathbf{68.4}^{\pm 29.0}$	**172**	$\mathbf{49.0}^{\pm 35.7}$

Besides improving success rates, GROW also substantially reduces the number of steps required to complete tasks. Compared with the previous best step counts, our agent reduces the average steps from 263 to 128 on embodied tasks, 291 to 248 on GUI tasks, and 248 to 172 on combat tasks. These results show that GROW not only teaches the agent how to accomplish diverse Minecraft tasks, but also improves execution efficiency, leading to more direct and effective task completion.

4.4 Generalization Evaluation

As shown in Table 2, GROW substantially improves the initialized policy on both In-Distribution (ID) tasks and Out-of-Distribution (OOD) tasks, which are selected and not selected for RL training respectively. Note that we use different seeds to initialize the environments for ID tasks to avoid data leakage between training and evaluation. The difficulty distributions of tasks in ID tasks and OOD tasks are similar. We detail the way to construct these two task splits and analyze their difficulty distributions in Appendix D. For ID tasks, the success rate increases from 35.0% to 90.0% on embodied tasks, from 10.0% to 92.5% on GUI tasks, and from 45.0% to 85.0% on combat tasks. More importantly, these gains are not limited to tasks selected for RL training. For OOD tasks, GROW also improves the success rate from 41.9% to 59.2% on embodied tasks, from 21.3% to 68.2% on GUI tasks, and from 19.8% to 47.5% on combat tasks. These consistent improvements provide direct evidence that GROW strengthens reusable interaction skills rather than overfitting the tasks selected during training. In conclusion, the VLM agent trained with GROW does not overfit to the ID tasks. Instead, it acquires transferable capabilities that generalize to OOD tasks across different task categories.

4.5 Ablation Study

4.5.1 Controlled Comparison with Trajectory-Level GRPO and PPO

Comparison with Trajectory-Level GRPO To isolate the effect of trajectory decomposition, we train a trajectory-level GRPO baseline using the same Qwen2-VL-7B backbone, initialization, tasks, and training resources as GROW. This baseline is trained directly with trajectory samples and uses a maximum context length of 32K tokens, which imposes a heavy burden on training resources, while the context length of the state-action samples in GROW is less than 1K tokens. Table 3 shows that GROW substantially improves upon the initialized policy and consistently outperforms trajectory-level GRPO, achieving higher ASR and fewer interaction steps across embodied, GUI, and combat tasks. We attribute the worse performance of trajectory-level GRPO to the fact that trajectory samples frequently reach the context limit before task completion during rollouts, so only a few trajectory-level samples receive episode rewards and the resulting training signal is sparse. In

Table 4: Ablation study on the discount factor γ. All experiments are trained on the same task set and from the same initialized policy to ensure a fair comparison.

γ	Embodied Tasks		GUI Tasks		Combat Tasks	
	Steps	ASR	Steps	ASR	Steps	ASR
0.9	142	$42.0^{\pm 35.2}$	292	$34.9^{\pm 34.4}$	170	$45.1^{\pm 37.8}$
0.95	**122**	$59.0^{\pm 25.9}$	309	$62.0^{\pm 20.6}$	**167**	$46.6^{\pm 34.5}$
0.995	128	$\mathbf{59.6}^{\pm 37.2}$	**248**	$\mathbf{68.4}^{\pm 29.0}$	172	$\mathbf{49.0}^{\pm 35.7}$
1.0	162	$52.1^{\pm 40.0}$	408	$45.1^{\pm 30.6}$	228	$36.3^{\pm 35.2}$

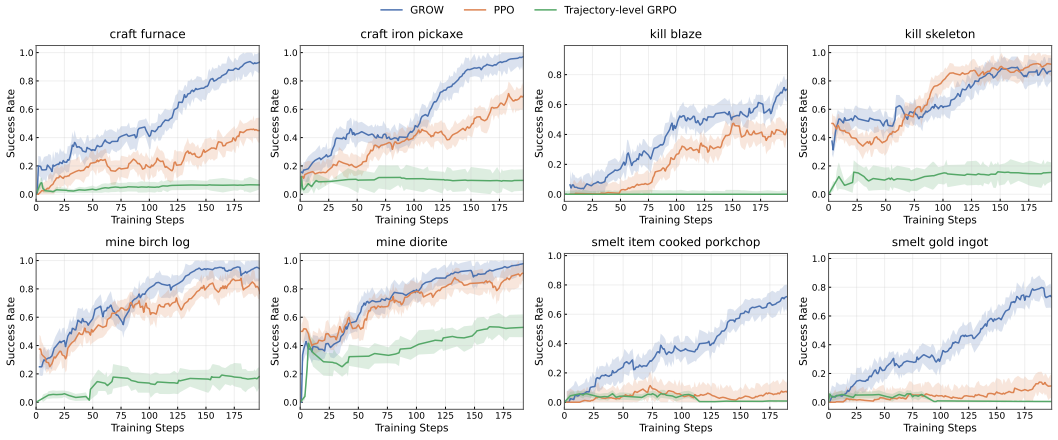


Figure 3: Training curves of GROW, trajectory-level GRPO and PPO. The curves show the success rate versus the number of training steps for each task, where the blue, green, and orange lines represent GROW, trajectory-level GRPO, and PPO, respectively. All RL methods are trained starting from the same initialized policy for fair comparison.

contrast, by decomposing trajectory samples into state-action samples with shorter context lengths, GROW is almost unaffected by the maximum limit of context lengths, allowing more timesteps for the trajectories to complete tasks and acquiring episode rewards to optimize the policy effectively.

Comparison with PPO Proximal Policy Optimization (PPO) is one of the most widely used RL algorithms and has been extensively adopted before the introduction of GRPO. To examine whether GROW provides stronger policy optimization than this classical RL baseline, we train the initialized policy with PPO and GROW under the same training budget, respectively, and compare their performance in Table 3. The results show that GROW achieves better performance than PPO.

To further analyze the training dynamics, we compare the success rate curves of GROW, trajectory-level GRPO and PPO in Figure 3. The curve of trajectory-level GRPO remains nearly flat, indicating that it fails to effectively optimize the policy, whereas the curves of PPO and GROW exhibit normal improvement, demonstrating the effectiveness of both training algorithms. Compared with PPO, GROW better improves the performances on most tasks. GROW typically reaches higher success rates in training rapidly while PPO often improves more slowly or stays in a low-success plateau for many training steps. The advantage is most evident on some GUI tasks, like *smelt cooked porkchop*, *smelt gold ingot*, which have longer trajectory lengths. For embodied tasks and combat tasks, both methods can eventually approach high success rates, but GROW generally reaches this region earlier than PPO.

4.5.2 Ablation on the Discount Factor

Our RL training framework includes a hyperparameter γ, which controls how strongly an episode reward decays when propagated backward across timesteps. Table 4 evaluates how γ affects the performance of our method in different categories of tasks. First, we validate the effectiveness of

Table 5: Results on simulated Language Table and WebShop. For simulated Language Table, we report the ASR of different methods across five task families. For WebShop, we report the success rate on the test set and the average score attained during task execution.

Method	simulated Language Table					WebShop	
	block2block	block2abs	block2rel	block2blockrel	separate	Success Rate	Score
Initialized Policy	0.02	0.03	0.02	0.02	18.80	26.2	37.5
PPO	53.11	62.53	65.62	46.91	96.93	44.8	**69.9**
Trajectory-level GRPO	4.55	2.05	6.36	2.96	33.58	4.8	5.48
GROW	**68.81**	**81.23**	**87.56**	**59.47**	**100.00**	**47.6**	68.0

credit assignment with temporally discounted rewards by comparing the results of GROW with γ of 1.0 and γ of 0.995. Across all the task categories, GROW with γ of 0.995 uses fewer steps to finish tasks than the variant with γ of 1.0 showing that appropriate γ can improve the efficiency of completing tasks, which is consistent with the original design intent. Additionally, GROW with γ of 0.995 also achieves higher ASR in all three task categories, indicating that credit assignment with temporal discounting facilitates more effective task learning. Thus, temporal discounting is beneficial for both task completion and execution efficiency. Moreover, the results for GROW with γ values of 0.9, 0.95, and 0.995 show that an appropriate γ leads to strong performance, whereas reducing γ may result in degraded performance. Comparing the variants with $\gamma = 0.95$ and $\gamma = 0.995$, although the smaller γ reduces the number of steps required to complete the embodied and combat tasks, it also causes the learning signal to decay too rapidly, resulting in less effective learning across all tasks. GROW with a γ of 0.9 employs the most aggressive decay. However, it doesn't guide the policy to complete the tasks more efficiently. In contrast, its steps to finish tasks are generally more than those of the variant with γ of 0.995 and its ASR in all three categories are consistently lower than those of the variant with γ of 0.995. Therefore, selecting an appropriate γ can facilitate improvements in both task performance and execution efficiency, but the value of γ should also be chosen carefully to avoid overly aggressive decay that prevents the policy from being optimized effectively.

4.6 Extension beyond Minecraft

We further evaluate GROW in two environments with different interaction dynamics: simulated Language Table (Lynch et al., 2022), a vision-based tabletop manipulation benchmark, and WebShop (Yao et al., 2022), a web-based decision-making benchmark. Complete experimental setups and implementation details for both environments are provided in Appendix E. The results are reported in Table 5. Across both simulated Language Table and WebShop, GROW delivers substantial gains in task completion over the initialized policy and outperforms PPO on most metrics. In particular, it achieves higher success rates on every Language Table task family and a higher test-set success rate on WebShop, despite the markedly different manipulation and web-interaction dynamics of these environments. These results show that the effectiveness of GROW generalizes beyond Minecraft to other settings while retaining a strong advantage in reliable task completion.

5 Conclusion

We propose GROW to address the challenge of excessively long context lengths of trajectory samples demanding substantial training resources when training VLM agents to execute open-world tasks with GRPO. GROW first decomposes trajectory samples into state-action samples to decrease the context lengths, and performs credit assignment with temporal discounting on these samples, thereby converting sparse reward signals into dense ones to guide efficient learning. And then relative advantages are computed among the state-action samples in the same rollout groups. We also provide a surrogate analysis showing that GROW retains the trajectory learning signals to reinforce the trajectory samples completing tasks successfully and suppress the failed ones without training on trajectory samples directly. Experiments on over 800 Minecraft tasks demonstrate that GROW achieves state-of-the-art performance across embodied, GUI, and combat tasks, outperforming existing methods and validating its effectiveness.

AI USE STATEMENT

In this work, we used generative AI tools for providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and implementing methods. We have not used generative AI tools for generating synthetic data sets, formulating mathematical claims, proposing or refining hypotheses, and designing or providing feedback on research methodology or experiments, and developing theoretical models or conceptual frameworks, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, and interpreting results are not applicable to this work. Additionally, we used generative AI tools for drafting parts of a research paper and editing the research paper to improve readability. We have reviewed all AI-assisted work. The AI-generated mathematical ingredients and proof drafts were verified and corrected by the authors through manual mathematical derivation. The AI-implemented methods were tested for correctness by the authors. All AI-drafted or AI-edited text was carefully reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed implementation information to support reproducibility. Section 4.2 describes the model initialization, RL training tasks, reward verification, computing resources, training iterations, and discount factor used in the Minecraft experiments. Appendix F reports the hyperparameter settings for the training stages. Appendix G provides the experimental setups and implementation details for simulated Language Table and WebShop, including policy initialization, RL training, and evaluation settings.

REFERENCES

Qihang Ai, Pi Bu, Yue Cao, Yingyao Wang, Jihao Gu, Jingxuan Xing, Zekun Zhu, Wei Jiang, Zhicheng Zheng, Jun Song, and Yuning Jiang. Inquiremobile: Teaching vlm-based mobile agent to request human assistance via reinforcement fine-tuning, 2026. URL https://arxiv.org/abs/2508.19679.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24639–24654. Curran Associates, Inc., 2022. doi: 10.52202/068431-1789. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c7008aff45b5d8f0973b23e1a22ada0-Paper-Conference.pdf.

Shaofei Cai, Zhancun Mu, Haiwen Xia, Bowei Zhang, Anji Liu, and Yitao Liang. Scalable multi-task reinforcement learning for generalizable spatial intelligence in visuomotor agents, 2025a. URL https://arxiv.org/abs/2507.23698.

Shaofei Cai, Zihao Wang, Kewei Lian, Zhancun Mu, Xiaojian Ma, Anji Liu, and Yitao Liang. Rocket-1: Mastering open-world interaction with visual-temporal context prompting, 2025b. URL https://arxiv.org/abs/2410.17856.

Yue Hu, Avery Xi, Qixin Xiao, Seth Isaacson, Henry X. Liu, Ram Vasudevan, and Maani Ghaffari. Longnav-r1: Horizon-adaptive multi-turn rl for long-horizon vla navigation, 2026. URL https://arxiv.org/abs/2602.12351.

LinFeng Li, Jian Zhao, Yuan Xie, Xin Tan, and Xuelong Li. Compassnav: Steering from path imitation to decision understanding in navigation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eqcDckWHik.

Muyao Li, Zihao Wang, Kaichen He, Xiaojian Ma, and Yitao Liang. JARVIS-VLA: Post-training large-scale vision language models to play visual games with keyboards and mouse. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17878–17899, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.920. URL https://aclanthology.org/2025.findings-acl.920/.

Ning Li, Qiqiang Lin, Zheng Wu, Xiaoyun Mo, Weiming Zhang, Yin Zhao, Xiangmou Qu, Jiamu Zhou, Jun Wang, Congmin Zheng, Yuanyi Song, Hongjiang Chen, Heyuan Huang, Jihong Wang, Jiaxin Yin, Jingwei Yu, Junwei Liao, Qiuying Peng, Xingyu Lou, Jun Wang, Weiwen Liu, Zhuosheng Zhang, and Weinan Zhang. Coloragent: Building a robust, personalized, and interactive os agent, 2025b. URL https://arxiv.org/abs/2510.19386.

Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 69900–69929. Curran Associates, Inc., 2023. doi: 10.52202/075280-3064. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/dd03f856fc7f2efeec8b1c796284561d-Paper-Conference.pdf.

Zhihao Luo, Wentao Yan, Jingyu Gong, Min Wang, Zhizhong Zhang, Xuhong Wang, Yuan Xie, and Xin Tan. NaviMaster: Learning a unified policy for GUI and embodied navigation tasks. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38595–38615, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1788. URL https://aclanthology.org/2026.acl-long.1788/.

Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time, 2022. URL https://arxiv.org/abs/2210.06407.

Loïc Magne, Anas Awadalla, Guanzhi Wang, Yinzhen Xu, Joshua Belofsky, Fengyuan Hu, Joohwan Kim, Ludwig Schmidt, Georgia Gkioxari, Jan Kautz, Yisong Yue, Yejin Choi, Yuke Zhu, and Linxi Fan. Nitrogen: An open foundation model for generalist gaming agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21511–21521, June 2026.

Mingyu Ouyang, Siyuan Hu, Kevin Qinghong Lin, Hwee Tou Ng, and Mike Zheng Shou. Gameworld: Towards standardized and verifiable evaluation of multimodal game agents, 2026. URL https://arxiv.org/abs/2604.07429.

ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400711961. doi: 10.1145/3689031.3696075. URL https://doi.org/10.1145/3689031.3696075.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment, 2025. URL https://arxiv.org/abs/2507.05720.

Weihao Tan, Xiangyang Li, Yunhao Fang, Heyuan Yao, Shi Yan, Hao Luo, Tenglong Ao, Huihui Li, Hongbin Ren, Bairen Yi, Yujia Qin, Bo An, Libin Liu, and Guang Shi. Lumine: An open

recipe for building generalist agents in 3d open worlds, 2025. URL https://arxiv.org/abs/2511.08892.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024. URL https://arxiv.org/abs/2409.12191.

Zihao Wang, Xujing Li, Yining Ye, Junjie Fang, Haoming Wang, Longxiang Liu, Shihao Liang, Junting Lu, Zhiyong Wu, Jiazhan Feng, Wanjun Zhong, Zili Li, Yu Wang, Yu Miao, Bo Zhou, Yuanfan Li, Hao Wang, Zhongkai Zhao, Faming Wu, Zhengxuan Jiang, Weihao Tan, Heyuan Yao, Shi Yan, Xiangyang Li, Yitao Liang, Yujia Qin, and Guang Shi. Game-tars: Pretrained foundation models for scalable generalist multimodal game agents, 2025. URL https://arxiv.org/abs/2510.23691.

Zihao Wang, Muyao Li, Kaichen He, Xiangyu Wang, Zhancun Mu, Minghao Liu, Anji Liu, and Yitao Liang. OpenHA: A series of open-source hierarchical agentic models in minecraft. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=1RQIaGQV8v.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1355. URL https://aclanthology.org/2025.acl-long.1355/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. doi: 10.52202/068431-1508. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shuhao Ye, Sitong Mao, Yuxiang Cui, Xuan Yu, Shichao Zhai, Wen Chen, Shunbo Zhou, Rong Xiong, and Yue Wang. Etp-r1: Evolving topological planning with reinforcement fine-tuning for vision-language navigation in continuous environments, 2025. URL https://arxiv.org/abs/2512.20940.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework, 2025a. URL https://arxiv.org/abs/2510.04206.

Zekai Zhang, Weiye Zhu, Hewei Pan, Xiangchen Wang, Rongtao Xu, Xing Sun, and Feng Zheng. Activevln: Towards active exploration via multi-turn rl in vision-and-language navigation, 2025b. URL https://arxiv.org/abs/2509.12618.

Xinyue Zheng, Haowei Lin, Kaichen He, Zihao Wang, QIANG FU, Haobo Fu, Zilong Zheng, and Yitao Liang. MCU: An evaluation framework for open-ended game agents. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=hrdLhNDAzp.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In Yixin Cao, Yang Feng, and Deyi Xiong (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 400–410, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.38. URL https://aclanthology.org/2024.acl-demos.38/.

Zheyuan Zhou, Liang Du, Zixun Sun, Xiaoyu Zhou, Ruimin Ye, Qihao Chen, Yinda Chen, and Lemiao Qiu. Main-vla: Modeling abstraction of intention and environment for vision-language-action models, 2026. URL https://arxiv.org/abs/2602.02212.

A Details of State Space and Action Space

State Space. Our model operates under a strict pixel-only constraint, perceiving the environment solely through raw 640×360 RGB frames without access to any internal game states, coordinates, or metadata. Following the protocol established in JARVIS-VLA (Li et al., 2025a), these observations are intended to mirror the authentic experience of a human player; as such, we do not remove or mask any standard on-screen overlays. The visual input includes all native HUD elements—such as the hotbar, health and hunger indicators, and the dynamic hand-swing animations triggered by interactions—forcing the agent to interpret the environment and its own status purely from raw sensory data. To maintain this human-centric perspective, we employ a standard 70° Field of View and a GUI scale of 2, ensuring the visual distribution remains consistent with typical gameplay.

Action Space. The action space is designed to be both atomic and expressive, mapping directly to the fundamental keyboard and mouse inputs available to a human. Rather than employing high-level macro-actions or simplified APIs, we decompose player behavior into granular primitives. This includes binary operations for movement and interaction (e.g., sprinting, jumping, and attacking) alongside continuous controls for camera orientation (pitch and yaw). By relying on these basic building blocks, the agent is required to learn the composition of complex, multi-step strategies from the ground up. The full set of these operations is summarized in Table 6. In inference, we request agents to predict an action chunk with 4 interaction steps according to their current observation.

Table 6: Mapping binary primitives to the standard Minecraft controls.

Action	Human action	Description
forward	W key	Move forward.
back	S key	Move backward.
left	A key	Strafe left.
right	D key	Strafe right.
jump	Space key	Jump.
sneak	left Shift key	Switch to a slow walking mode.
sprint	left Ctrl key	Switch to rapid walking mode.
attack	left Button	Destroy blocks (hold down); Attack entity (click once).
use	right Button	Place the item currently held or use the block the player is looking at.
drop	Q key	Drop a single item from the stack of items the player is currently holding.
hotbar.[1-9]	keys 1-9	Switch active item to the one in a given hotbar cell.
inventory	E key	Open/Close the inventory.
yaw	move Mouse X	camera movement.
pitch	move Mouse Y	camera movement.

B Statistical Evidences

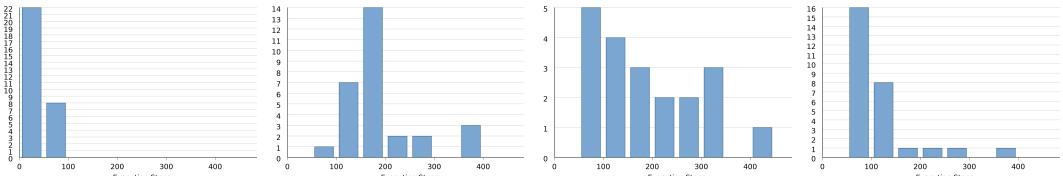
Figure 4 shows the step distribution across tasks. We observe that trajectories for the same task have similar numbers of execution steps. For failed tasks, the early termination stage also keeps trajectory lengths as close as possible to those of successful executions.

C Evaluation Variance and Repeated Evaluation

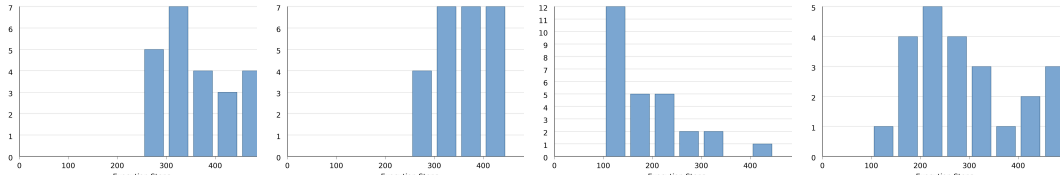
The relatively large ASR standard deviations in Table 1 primarily reflect heterogeneous task difficulty within the broad benchmark. Table 7 shows that GROW markedly reduces the proportion of low-success-rate tasks and increases the proportion of high-success-rate tasks in all three categories, while a subset of difficult tasks remains. We also repeat the evaluation in Table 8; the average ASR and their standard deviations change only marginally. Together, these results indicate that the observed variation is mainly task-dependent rather than evidence of unstable evaluation.

Table 7: Distribution (%) of per-task success rates before and after RL. Low, medium, and high denote ASR ranges of 0–33.3%, 33.3–66.7%, and 66.7–100%, respectively.

Phase	E-Low	E-Med	E-High	G-Low	G-Med	G-High	C-Low	C-Med	C-High
Before RL	49.7	22.2	28.1	78.3	13.7	8.0	68.0	26.0	6.0
After RL	29.4	19.0	51.6	16.6	19.8	63.6	42.0	24.0	34.0
Gain	-20.3	-3.2	+23.5	-61.7	+6.1	+55.6	-26.0	-2.0	+28.0



(a) Mine Diorite (b) Mine Birch Log (c) Craft Furnace (d) Craft Iron Pickaxe



(e) Smelt Cooked Porkchop (f) Smelt Gold Ingot (g) Kill Skeleton (h) Kill Blaze

Figure 4: Step-count histograms over 30 rollouts for the eight RL-trained tasks. Within a task, successful and unsuccessful rollouts have similar lengths because unsuccessful rollouts terminate at a fixed step limit. This supports the approximately equal-length assumption used in the surrogate analysis.

D Construction and Difficulty Analysis of ID and OOD Tasks

D.1 Construction of ID and OOD Tasks

ID tasks are selected for RL training, while OOD tasks are not. The ID tasks are not intended to exhaust all Minecraft tasks, but were selected to be as representative as possible of the capabilities required by the evaluated categories. As shown in the Table 9, the selected tasks cover embodied tasks involving navigation and mining, crafting and smelting tasks that require GUI interaction, and combat tasks that require weapon use and adversarial engagement.

D.2 Difficulty Matching between ID and OOD Tasks

The eight ID tasks were selected for tasks diversity and representativity rather than for extreme difficulty. The aggregate before-RL ASR is nearly identical for the ID and OOD tasks (25.0 versus 25.2), supporting broad difficulty matching. Category-level means differ because the much larger OOD tasks contains a heterogeneous mixture of task difficulties. Table 10 reports this composition explicitly. Together with the within-split before/after comparison in Table 2, the gains on OOD tasks that require related capabilities but have different objectives provide evidence of transferable category-level skills rather than simple memorization.

E Dataset Construction

In our training framework, the cold-start stage is the only phase necessitating external data collection. Conversely, the next stage, multi-turn RL with GRPO, relies solely on predefined scenarios and resource configurations for each task, eliminating the need for additional data acquisition.

We first adopted the approach from JARVIS-VLA (Li et al., 2025a) by utilizing a dataset rich in world knowledge for supervised fine-tuning. This component comprises 25k visual question answering

Table 8: Repeated evaluation of GROW on Minecraft. ASR is reported as mean ± standard deviation.

Run	Embodied Steps	Embodied ASR	GUI Steps	GUI ASR	Combat Steps	Combat ASR
Original	128	59.6 ± 37.2	248	68.4 ± 29.0	172	49.0 ± 35.7
Repeated	124	61.0 ± 38.0	258	69.4 ± 28.7	180	48.1 ± 35.5

Table 9: ID tasks and their representative capabilities.

ID Tasks	Representative Capability
mine block diorite	Navigation and mining of mineral/stone-like targets
mine birch log	Navigation and mining of vegetation/wood-like targets
craft furnace	Single-material recipe crafting
craft iron pickaxe	Multi-material recipe crafting
smelt gold ingot	Smelting mineral resources
smelt item cooked porkchop	Smelting food resources
kill skeleton	Combat against ground-moving hostile targets
kill blaze	Combat against flying hostile targets

samples, 225k grounding samples, and 9k captioning samples. Subsequently, we performed detailed processing on the open-source human player trajectory dataset from VPT (Baker et al., 2022) to construct an instruction-annotated action dataset.

Labeling and Segmentation We identified task completion based on transitions in the player's state recorded within the dataset. For instance, if the player's inventory showed the acquisition of an iron ore relative to the previous state, we labeled the preceding trajectory segment with the instruction to "mine iron ore." We defined the segment duration as the interval starting from the completion of the previous task to the moment the current state change occurred. This logic applies similarly to other interactions, such as labeling the interval preceding the defeat of a creature or the acquisition of a specific item with the corresponding objective.

Cleaning and Chunking Following the data cleaning protocol in VPT (Baker et al., 2022), we filtered the dataset to remove redundant idle sequences. Specifically, if more than three consecutive idle frames were detected, we discarded the redundant frames starting from the fourth frame. To construct action chunks, we grouped actions into sets of four. For each set, we retained the observation solely from the first frame and concatenated the actions chronologically. We applied padding with null actions for groups containing fewer than four inputs.

Balancing and Final Composition An analysis of the dataset composition revealed a severe distributional imbalance, as the task of mining stone constituted 39.45% of the total data. This skew significantly compromised the performance of the initial SFT model and hindered the subsequent RFT process. We addressed this issue by downsampling the overrepresented tasks to achieve a balanced distribution, which enabled the successful training of our supervised baseline. Through these processing and balancing steps, we ultimately collected approximately 4 million state-action samples for SFT.

F HYPERPARAMETERS

Our training framework consists of three progressive stages: (i) World Knowledge Learning, (ii) Imitation Learning, and (iii) Multi-Turn RL. The specific training parameters for all three stages are detailed in Table 11.

Table 10: Before-RL difficulty distribution for the ID and OOD tasks. Low, mid, and high use the same ASR bins as Table 7; counts are in parentheses.

Split / Category	Before-RL ASR	Low	Mid	High
ID Overall	25.0	75.0% (6)	25.0% (2)	0.0% (0)
OOD Overall	25.2	72.1% (562)	16.0% (125)	11.9% (93)
ID Embodied	35.0	50.0% (1)	50.0% (1)	0.0% (0)
OOD Embodied	41.9	49.7% (75)	21.9% (33)	28.5% (43)
ID GUI	10.0	100.0% (4)	0.0% (0)	0.0% (0)
OOD GUI	21.3	78.1% (454)	13.8% (80)	8.1% (47)
ID Combat	45.0	50.0% (1)	50.0% (1)	0.0% (0)
OOD Combat	19.8	68.8% (33)	25.0% (12)	6.3% (3)

Table 11: Hyperparameter settings across different training stages.

Hyperparameter	**World Knowledge**	**Imitation Learning**	**Multi-Turn RL**
Trainable Components	Full	Language Models	Full
LR Scheduler	Cosine	Cosine	-
Warm-up Ratio	0.1	0.1	-
Global Batch Size	32	32	8
Optimizer	AdamW	AdamW	AdamW
Learning Rate	1×10^{-5}	1×10^{-5}	1×10^{-6}
Group Size (G)	-	-	8
Clipping Parameter (ϵ)	-	-	0.2

G Details of Experiments in simulated Language Table and WebShop

G.1 simulated Language Table

To demonstrate that GROW remains effective in a markedly different embodied manipulation setting and is not limited to Minecraft, we retrain GROW on the simulated Language Table (Lynch et al., 2022), which provides a tabletop environment characterized by continuous control dynamics.

G.1.1 Experimental Setup

The policy receives an egocentric RGB view from a tabletop manipulation robot as input. Each state is represented by an image with resolution $180 \times 320 \times 3$. The action space consists of two-dimensional control values, corresponding to the displacement of the robot end effector along the x and y axes. To keep the action interface consistent with the Minecraft setting, we use μ-law encoding to discretize the x and y axes independently into 21 bins, yielding 42 action tokens in total. Each bin is mapped to a reserved token.

For multi-turn RL, we follow the task setup of the simulated Language Table (Lynch et al., 2022) and train on five task families: `block2block`, `block2abs`, `block2rel`, `block2blockrel`, and `separate`. The detailed definitions, success criteria, and the exact number of conditions for each task family are summarized in Table 12.

G.1.2 Implementation Details

First, we initialize the policy of Qwen2.5-VL-7B-Instruct (Bai et al., 2025) via SFT on 1M state-action samples. We construct the SFT dataset from the real-robot Language Table dataset by decomposing

Table 12: Definitions and configurations of the five task families in the simulated Language Table environment.

Task Family	Agent Action	Success Criterion	# Conditions
`block2block`	Pushes a source block to another target block.	Distance between the source and target blocks is below a threshold.	56 (8 src $\times$ 7 tgt)
`block2abs`	Pushes a block to an absolute board location (9 locations[1]).	Distance between the block and the target location is below a threshold.	72 (8 blk $\times$ 9 loc)
`block2rel`	Pushes a block to a relative offset location (8 directions[2]).	Distance between the block and the invisible target offset location is below a threshold.	64 (8 blk $\times$ 8 dir)
`block2blockrel`	Pushes a source block to a relative offset location of another block (8 directions[2]).	Distance between the source block and the invisible target offset of the target block is below a threshold.	448 (8 src $\times$ 7 tgt $\times$ 8 dir)
`separate`	Separates two blocks.	Distance between the two blocks exceeds a predefined threshold.	56 (8 src $\times$ 7 tgt)

[1] *9 absolute locations:* top left, top center, top right, center left, center, center right, bottom left, bottom center, and bottom right.
[2] *8 relative directions:* left, right, up, down, up-left, up-right, down-left, and down-right.

each episode into individual time steps. For each time step, the corresponding visual state and annotated control signal form one state-action sample. This procedure yields 1M state-action samples for policy initialization. We train the cold-start policy on 4 H200 GPUs for 8000 steps for about 3 days with a global batch size of 32. We use AdamW as the optimizer with a cosine learning-rate schedule and set the warm-up ratio to 0.1. During SFT, all model components are updated.

The simulated Language Table environment provides built-in task-specific verifiers for these task families. We use these verifiers to compute binary episode-level rewards according to the corresponding thresholded geometric success conditions. During RL, we train the policy with GROW for 200 training steps on 2 H200 GPUs for about 2 Days. The main hyperparameters are summarized in Table 13.

Table 13: Hyperparameter settings for policy initialization in simulated Language Table.

Hyperparameter	Value
Global Batch Size	16
Optimizer	AdamW
Learning Rate	1×10^{-6}
Group Size (G)	8
Discount Factor (γ)	0.995
Clipping Parameter (ϵ)	0.2

G.2 WebShop

To examine whether GROW transfers to language-mediated web interaction, we conduct experiments in WebShop (Yao et al., 2022), a simulated e-commerce environment containing 1.18M products and 12,087 crowd-sourced shopping instructions. An episode starts from a natural-language purchase request. At every turn, the agent observes the current text-rich webpage, including search results, product descriptions, and navigation state, and selects one environment action in the form `search[query]` or `click[value]`. A successful episode requires the agent to identify and purchase a product that satisfies the requested category, attributes, options, and price constraints.

G.2.1 Policy Initialization

We initialize Qwen2-VL-7B-Instruct using the official WebShop imitation-learning trajectories. We decompose each demonstration into transition-level state-action pairs: the input contains the shopping goal, the current page observation, and the currently valid actions, while the target is the single demonstrated `search` or `click` action. This yields 15,111 SFT examples. We fine-tune for one epoch with LLaMA-Factory, using a maximum sequence length of 4096, bf16 precision, a global batch size of 64, and AdamW with a cosine learning-rate schedule. The learning rate is 1×10^{-6} and the warm-up ratio is 0.05. Since this setting uses text-rich WebShop observations, we freeze the visual encoder and multimodal projector and update the language model parameters only.

G.2.2 Multi-turn RL and Evaluation

For RL, we use the full WebShop product catalog and human-authored goals. The training set comprises 10,587 goals with shuffled indices 1,500–12,086; the official test split comprises the disjoint first 500 goals. Each state retains at most 6,000 observation characters and exposes at most 80 valid clickable actions, in addition to the search action. The policy produces one action per turn, with a maximum of 64 turns during RL rollouts. We sample $G = 8$ rollouts per goal and train GROW for 60 training steps on four H200 GPUs. The global rollout batch size is 16, the per-GPU micro-batch size is 8, the actor learning rate is 1×10^{-6}, and the KL coefficient is 0.001. We use $\gamma = 0.995$ and apply no entropy bonus. Although WebShop supplies a graded terminal score, RL uses a strict success signal: scores below 1 are set to zero, so only fully satisfied purchases receive a nonzero optimization reward.

At evaluation, we decode greedily and allow up to 100 environment actions per episode. We report *Success Rate*, the fraction of episodes whose native WebShop reward equals 1, and *Score*, the mean native reward rescaled to $[0, 100]$. Thus, Score preserves partial credit for category, attribute, option, and price matches, whereas Success Rate measures complete task fulfillment. Table 14 summarizes the principal optimization settings.

Table 14: Hyperparameter settings for WebShop experiments.

Hyperparameter	**Value**
SFT examples	15,111
SFT epochs	1
SFT global batch size	64
SFT learning rate	1×10^{-6}
SFT warm-up ratio	0.05
RL training goals	10,587
RL training steps	60
Rollouts per goal (G)	8
RL global batch size	16
RL learning rate	1×10^{-6}
KL coefficient	0.001
Discount factor (γ)	0.995
Maximum RL turns	64
Maximum evaluation actions	100

H Case Study in Minecraft

To visualize the agent's decision-making, we overlay its action space on the left of each case study frame. We use a color-coded telemetry system to distinguish intent: yellow denotes inactive control primitives, while red signifies the specific actions selected by the model at that time step. This provides a direct, interpretable trace of the agent's behavioral primitives—such as movement keys or camera adjustments—relative to the visual context.

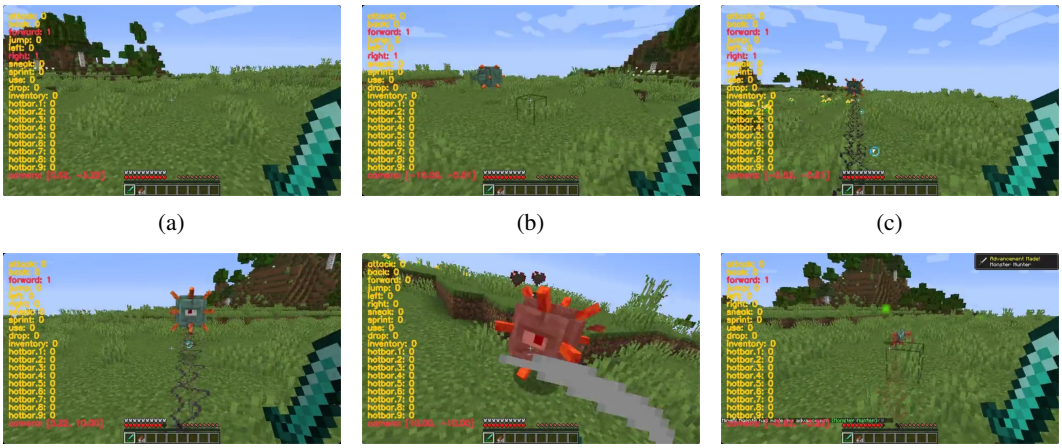
H.1 Case Study: Kill Guardian

Task Dynamics This case study highlights the model's ability to handle complex adversarial dynamics in Minecraft. The target, a Guardian, presents a unique challenge: it is a ranged attacker that maintains a tactical distance, retreating when the player approaches too closely while staying within its own offensive range. Since the agent is equipped only with a sword (a melee weapon), it must master a sophisticated behavioral loop involving active search, tactical approach, and precise engagement under evasive conditions.

Target Search (Figure 5a,5b): When the Guardian moves out of the field of view (FOV), the agent does not wander aimlessly. Instead, it performs a systematic search, utilizing camera yaw and pitch to scan the environment until the target is re-acquired.

Evasion and Approach (Figure 5c,5d): The agent identifies the Guardian's ranged beam attack and initiates a direct approach. It successfully closes the gap despite the Guardian's attempts to maintain distance.

Melee Engagement (Figure 5e,5f): Once within striking distance, the agent executes precise attack primitives while simultaneously tracking the target's movements, eventually neutralizing the threat through melee strikes.



(a) (b) (c)

(d) (e) (f)

Figure 5: Behavioral analysis of the agent engaging a Guardian in combat.

H.2 Case Study: Craft Stone Shovel

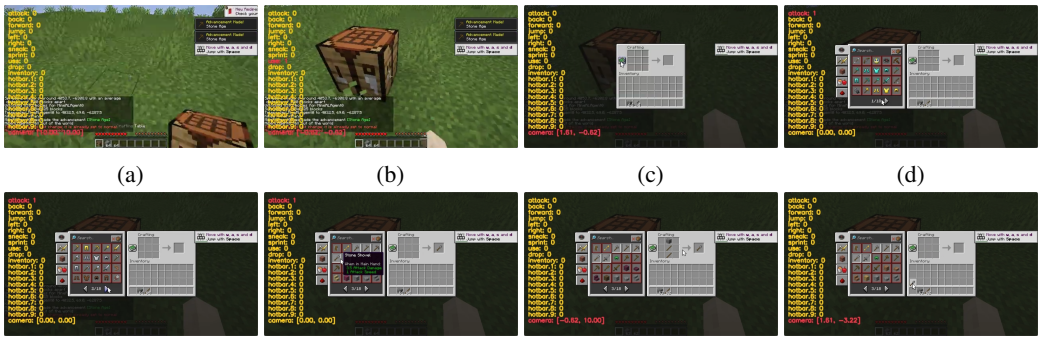
Synthesizing advanced tools in Minecraft requires the agent to transition from spatial navigation to abstract GUI manipulation. In this case study, the agent must perform a multi-stage sequence: locating and interacting with a Crafting Table, navigating a multi-page recipe book to identify a specific target (Stone Shovel), and executing the final synthesis. This task tests the model's ability to maintain long-term goal coherence across different visual modalities (3D world vs. 2D interface).

Real-time Action Telemetry As visualized in Figure 6, we use a color-coded telemetry overlay to trace the agent's intent: yellow denotes inactive controls, while red highlights active primitives. This allows us to observe the shift from movement-based exploration to precise UI-based clicking.

Environment Interaction (Figure 6a,6b): The agent approaches the Crafting Table and triggers the Use primitive (red) to open the synthesis interface.

GUI Navigation (Figure 6c,6d,6e): Once the interface is active, the agent's focus shifts to the recipe book. The telemetry reveals a sequence of precise Attack (simulated as "click") and Camera (mouse cursor movement) primitives as the agent scrolls through multiple pages to locate the Stone Shovel recipe.

Target Synthesis (Figure 6f,6g,6h): Upon selecting the correct recipe, the agent executes the crafting command. The successful completion is confirmed by the appearance of the item in the output slot and the subsequent "Stone Age" advancement notification.



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

(e) (f) (g) (h)

Figure 6: Case study of the agent synthesizing a Stone Shovel.

I PROMPT IN EXPERIMENTS

We show our prompt in Figure 7, which is used in our experiment.

Prompt used in experiment

<|im_start|>system
You are an experienced Minecraft player. Based on the current game screen, plan the next 200ms of actions, composed of 4 steps, each spaced 50ms apart. Each action begins at its execution time and lasts for 50ms until the next step starts.

Output Format
<|reserved_special_token_178|>...<|reserved_special_token_179|><|reserved_special_token_178|>...<|reserved_special_token_179|><|reserved_special_token_178|>...<|reserved_special_token_179|><|reserved_special_token_178|>...<|reserved_special_token_179|>

Instructions
1. The output consists of exactly 4 action steps. Each step is marked by a start token <|reserved_special_token_178|> and an end token <|reserved_special_token_179|>, with the specific action command placed between them.
2. Output ONLY the plain string in the exact format above. Do not add line breaks, quotes, or any additional text. Your current task is: [Task Description Here]<|im_end|>
<|im_start|>user
Your observation: [IMAGE_DATA]<|im_end|>
<|im_start|>assistant

Figure 7