ATTACCA: GOAL-DIRECTED CONTROL UNDER STATE CONTINUITY FOR LONG-HORIZON EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A central capability of embodied agents is to accomplish complex objectives through sequences of interdependent tasks. Yet existing visual goal-conditioned policies underlying these agents are typically evaluated on isolated interactions where the target is already visible, and thus do not capture the conditions that arise during continuous long-horizon task execution. In such settings, each task begins from the state left by the previous one: the agent may end at a different position and orientation, the world may have been modified, and the next interaction target may lie outside the current field of view. As a result, agents relying on such policies may struggle to proceed to the next task when they cannot ground their target in the current observation. To address this challenge, we propose Attacca, a new approach that trains visual goal-conditioned policies on complete exploration-to-interaction trajectories using goal images decoupled from the execution environment. Attacca uses context-decoupled goal sampling to pair each demonstration with a class-compatible masked goal image from another world, removing direct scene and pose correspondence. It learns dense current-view grounding through a target-mask prediction head, providing auxiliary supervision beyond action imitation. We further introduce behavioral-phase conditioning that teaches the policy to distinguish Search, Approach, and Interact stages and adapt its control as execution progresses. We evaluate Attacca on multiple short- and long-horizon embodied tasks in Minecraft. Our method achieves 39.0–47.5% clean success, improving over the strongest baseline by 1.7–2.4×. On long-horizon tasks, it attains 54%, 30%, and 28% completion, yielding up to a 7× improvement.

1 INTRODUCTION

Completing long-horizon tasks is a central challenge of embodied intelligence, as each behavior changes the conditions for subsequent ones (Ichter et al., 2023; Huang et al., 2023; Wang et al., 2024a). This capability is essential for embodied agents operating in the real world, where accomplishing complex objectives often requires a sequence of interdependent behaviors. For example, a household robot may need to navigate to a cabinet, retrieve an object, carry it elsewhere, and then locate another object, with each step beginning from the state left by the preceding one.

One line of work trains a single policy end-to-end: vision-language-action models map instructions and observations directly to low-level actions. While these methods offer a simple and unified control framework (Zitkovich et al., 2023; Kim et al., 2025; Octo Model Team et al., 2024), executing a long sequence of interdependent tasks remains challenging for a single policy. Another line of work adopts hierarchical architectures that pair a high-level reasoner with a low-level policy: the reasoner decomposes a complex objective into subtasks, and the policy executes them sequentially (Wang et al., 2023; Li et al., 2024). However, hierarchical decomposition does not by itself resolve the continuity of execution across subtasks. Even with a successful plan, the policy must still carry each task forward from the state produced by the previous one. We refer to this difficulty of carrying execution from one task to the next as the *skill hand-off gap*.

This skill hand-off gap becomes particularly apparent in open-world visuomotor control. We study it in Minecraft, where long-horizon objectives require agents to locate and interact with multiple targets in sequence. Recent visuomotor policies, including ROCKET-2 (Cai et al., 2025a), STEVE-1 (Lifshitz et al., 2023), and GROOT (Cai et al., 2024b), have demonstrated strong control capabilities

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.



Figure 1: **Goal-directed Control under State Continuity.** The agent executes tasks sequentially, with each task starting from the state left by the previous one. If the target is initially out of view, the agent must search for it, approach it once found, and interact with it to achieve the task.

on diverse tasks. However, these methods are largely developed and evaluated on short-horizon tasks, where the agent's field of view changes only modestly. As a result, robust execution of successive tasks remains underexplored. In particular, completing one task can move the agent into a new state where the next target is not visible, requiring it to locate the target before proceeding.

To address these challenges, we introduce *Attacca*[1], a novel approach for learning visuomotor policies that can reliably complete long-horizon tasks through successive task transitions. Attacca trains a target-conditioned policy on exploration-to-interaction trajectories that span the full process of searching for a target when absent, approaching it once discovered, and interacting with it when within reach. To support continuous and robust target-directed execution, Attacca introduces three essential design elements: (1) **context-decoupled goal sampling**, where each trajectory is paired with a masked target reference from a different environment, allowing the goal to specify what to seek without relying on scene or pose correspondence; (2) **goal-conditioned target grounding**, where a mask head predicts the target's visible support over visual patches in the current observation, providing spatial supervision both before and after the target enters the field of view; and (3) **behavioral-phase conditioning**, where a phase head distinguishes among *Search*, *Approach*, and *Interact* from the current execution state, encouraging the policy to adapt its control as it progresses from target acquisition to interaction. We jointly optimize action imitation with auxiliary objectives for target grounding and behavioral-phase prediction.

We evaluate Attacca across unseen environment configurations, held-out target classes, and target-switching chains of increasing horizon. These chains preserve the state left by each completed task and include transitions where the next target is initially out of view. On the short-horizon embodied tasks Mine, Hunt, and Place, Attacca achieves clean success rates of 39.0%, 47.5%, and 47.0%, respectively, outperforming the strongest baseline by 1.7×, 2.4×, and 2.1×. On the long-horizon embodied tasks, it attains 54.0%, 30.0%, and 28.0%, with up to a 7× gain over the strongest baseline. These results show that Attacca remains effective under state continuity across diverse inherited states, previously unseen environments and targets, and longer task horizons, suggesting that jointly learning target acquisition and interaction from environment-decoupled goal images can improve robustness and generalization across targets and task horizons.

2 Related Work

Target specification in goal-conditioned policies. Earlier works defined fixed Minecraft objectives such as obtaining a diamond and trained agents for them (Lin et al., 2022; Hafner et al., 2025), and

[1]The name Attacca is inspired by *attacca*, the musical direction to continue directly into the next movement without pause.

VPT (Baker et al., 2022) pre-trained a behavioral prior without a goal input. In goal-conditioned low-level policies, the target can be specified in several modalities. STEVE-1 (Lifshitz et al., 2023) conditions on MineCLIP embeddings (Fan et al., 2022) of text or of 16-frame behavior clips, and JARVIS-VLA (Li et al., 2025a) post-trains a vision-language model for instruction-conditioned control. A text goal can name any target but cannot show what an unfamiliar target looks like, and reported success on atomic tasks remains low (Wang et al., 2024b; Zheng et al., 2025). GROOT (Cai et al., 2024b) follows reference gameplay videos, which entangle the target with the demonstrator's behavior and the surrounding layout. ROCKET-1 (Cai et al., 2025b) and ROCKET-2 (Cai et al., 2025a) specify the target with a masked goal image. In their pipelines, a VLM points at the target in the current observation and a segmentation model turns the point into the goal mask (Clark et al., 2026; Ravi et al., 2025). This construction presumes the target is visible, so it is unavailable while the policy is still searching for the target. During training, these controllers also take the goal from the imitated trajectory itself, as in hindsight relabeling (Andrychowicz et al., 2017; Lynch et al., 2020), so the goal shares its scene with the observations. We keep the interface of a masked goal image, which clearly designates the target, and decouple it from the execution world. The policy is trained to read what the masked target is from a goal image taken in another world and to find a matching instance in its own surroundings, so the goal remains well-defined even before the target enters the view.

Hierarchical execution and the skill hand-off gap. When separately learned skills are composed, one skill can end in a state from which the next cannot succeed, a failure studied in skill chaining (Konidaris & Barto, 2009; Lee et al., 2019; 2022; Chen et al., 2023) and reported as a hand-off problem in mobile manipulation (Szot et al., 2021). Following language-model planners for embodied agents (Huang et al., 2022; Ichter et al., 2023; Huang et al., 2023), hierarchical Minecraft agents such as DEPS (Wang et al., 2023), JARVIS-1 (Wang et al., 2025), Optimus-1 (Li et al., 2024), and Optimus-2 (Li et al., 2025b) dispatch language subgoals from a planner to low-level policies, and DEPS, JARVIS-1, and Optimus-1 identify the controller as a main source of failure on harder tasks. Even with the next subgoal specified, the controller must still acquire the next target from the state left by the preceding skill before it can execute the requested interaction. Some agents acquire targets through simulator state, such as the block queries and pathfinding primitives in Voyager's code skills (Wang et al., 2024a) and the lidar observations in Plan4MC's finding skills (Yuan et al., 2023). These systems target long-horizon objectives as a whole, while we study the hand-off within the low-level policy itself, on task chains that preserve the state left by each completed task and require the policy to acquire and interact with the next target from that inherited state.

3 Attacca: Goal-Directed Embodied Control under State Continuity

We introduce Attacca, a new approach for training visual goal-conditioned policies to find and interact with targets specified by goal images. We first formalize goal-directed embodied control under state continuity and describe the policy architecture in Sec. 3.1. We then present the three key components for robust target-directed execution and the joint training objective (Secs. 3.2 to 3.4).

3.1 Problem Statement

We consider a long-horizon objective specified as an ordered sequence of tasks indexed by k, executed by a low-level policy. At the start of each task, the agent receives a visual goal $g_k = \left(o_g^k,\, m_g^k\right)$ and an interaction type c_k. Here, $o_g^k \in \mathbb{R}^{H\times W\times 3}$ indicates an RGB goal image and $m_g^k \in \{0,1\}^{H\times W\times 1}$ is the goal mask, a binary mask identifying the target in the goal image. The goal image serves as a visual reference for what the agent should interact with, showing the target object within a surrounding scene, typically near the center of the image. Because the world encountered at execution time is unknown before exploration, we consider a general setting in which the goal image is sampled in advance from a different world.

We consider two execution regimes: (1) In the single-task setting, the agent receives one visual goal and interaction type, while the corresponding target may initially lie outside its field of view. Success therefore requires the policy to perform the full progression from target search to approach and interaction; (2) In the long-horizon setting, multiple such tasks are executed successively. Each task begins from the state left by the previous one, preserving the agent's pose, camera orientation, and any

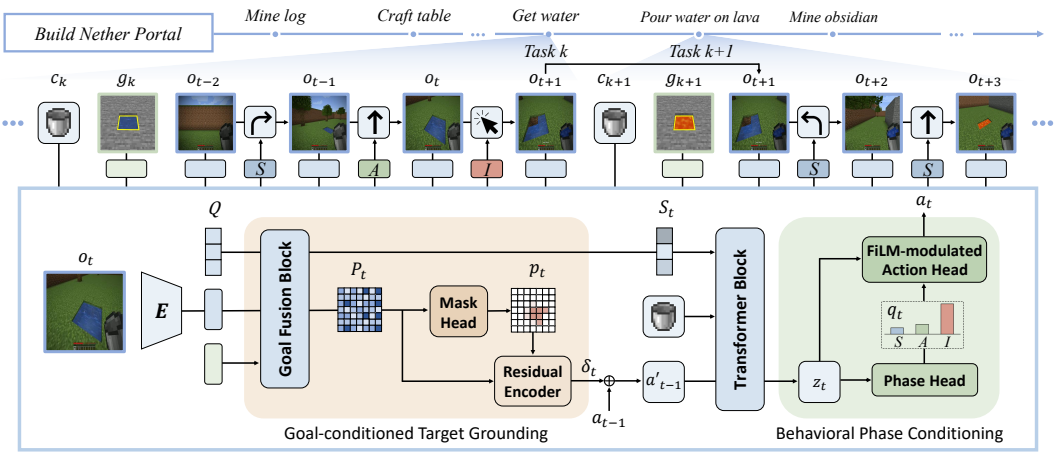


Figure 2: Policy architecture of Attacca.

changes to the world. As the embodied agent executes a task, it interacts with the environment through low-level actions and receives an egocentric RGB observation $o_t \in \mathbb{R}^{H\times W\times 3}$ at each timestep. The agent must locate the target, approach it, and execute the requested interaction solely from the visual observation stream and the goal image, without target coordinates, environment state, or external grounding and navigation modules. For each task, we model the policy at timestep t as

$$a_t \sim \pi_\theta(\cdot \mid o_{1:t},\, g_k,\, c_k)\,. \tag{1}$$

At each timestep, the policy selects a low-level action a_t based on the observation history $o_{1:t}$, the current visual goal g_k, and the interaction type c_k. Our goal is to learn a single policy that handles search, approach, and interaction from diverse inherited agent and world states, without relying on scene correspondence between the goal and execution environment.

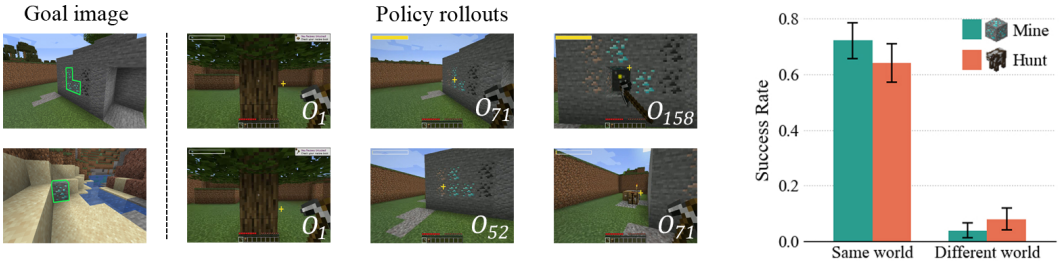
Policy architecture. Attacca is built on a neural policy comprising a visual encoder, a goal-observation fusion module, and a causal Transformer. A frozen pretrained visual encoder encodes the goal image once per task and the current observation at each timestep. The goal-fusion module then combines the goal and mask representations with the observation features to emphasize target-relevant visual evidence. The causal Transformer models the fused features over time to predict actions. With query tokens Q appended to summarize the fused visual context, the module computes

$$[P_t,\ S_t] = \mathrm{GoalFusion}(\hat{o}_t,\, \hat{g}_k,\, Q)\,, \tag{2}$$

where $\hat{o}_t$ and $\hat{g}_k$ denote the patch features of the observation and masked goal, respectively. The outputs at the observation-patch and query positions form the goal-conditioned observation features $P_t = [P_{t,1}, \cdots, P_{t,N}] \in \mathbb{R}^{N\times d}$ and summary tokens S_t. A causal Transformer processes these tokens with embeddings of the interaction type c_k and the previous action a_{t-1}, integrating temporal information across frames. We use the Transformer output at the previous-action token as the temporal readout z_t. After phase modulation (Sec. 3.4), z'_t is mapped by the action head to low-level camera motions and button presses.

3.2 Context-Decoupled Goal Sampling

In goal-directed embodied control, a reference image specifies the target to be pursued. Existing approaches typically sample this image from the same simulated world in which the demonstration trajectory is collected (Lifshitz et al., 2023; Cai et al., 2025b;a). This results in shared visual context between the reference and execution environments, enabling the policy to exploit background and contextual cues as shortcuts even when a target mask is provided. Such shortcuts can lead to overfitting and brittle generalization when the target is encountered in unfamiliar environments or visual contexts (Fig. 3). More importantly, the agent cannot assume that the reference image comes from the same world it will encounter at deployment, since that world is unknown before exploration begins. We therefore decouple goal sampling from the execution context by pairing each trajectory with a reference image of the same target class drawn from a different world.



(a) Qualitative examples (b) Goal-world robustness

Figure 3: Sensitivity to the goal world. (a) Two ROCKET-2 rollouts in the same execution world. (b) Success rates on the mine-diamond-ore and hunt-cow tasks over 50 episodes per condition.

For each demonstration trajectory τ, let $w(\tau)$ denote the world in which it was collected and $y(\tau)$ its target class, such as an *oak log* or *cow*. We randomly sample another demonstration of the same target class from a different world, select a frame where the target is visible, and use the corresponding RGB image and target mask as the goal. This ensures

$$w(g) \neq w(\tau), \qquad y(g) = y(\tau). \tag{3}$$

Since a new goal image is drawn each time a trajectory is sampled, the goal images paired with a demonstration vary in scene but always show the same target class. This design removes direct scene and pose correspondence between the goal image and the demonstration, encouraging the policy to identify matching targets by visual appearance rather than shared scene context. Sec. A illustrates the contrast with conventional goal construction and shows consistent gains across goal backgrounds.

3.3 Goal-Conditioned Target Grounding

Without shared scene context, the policy has to locate the target in its own observation. Existing goal-conditioned embodied policies are typically trained via behavior cloning (Pomerleau, 1988) on demonstration trajectories. While behavior cloning may implicitly encourage target grounding through action prediction, it provides no direct supervision for whether the target is visible or where it appears in the current observation. To provide such grounding supervision, we introduce a mask head (Fig. 2 middle) that predicts a dense target mask over the goal-conditioned observation patches, indicating the target's visible support in the current observation,

$$p_{t,i} = \sigma(h_{\text{mask}}(P_{t,i})), \qquad i = 1, \ldots, N, \tag{4}$$

where h_{mask} maps each patch feature to a scalar logit, σ denotes the sigmoid function, and $p_{t,i}$ is the predicted target-presence probability for patch i. We derive patch-level supervision from the pixel-level target mask by averaging its values over the region corresponding to each patch,

$$\bar{m}_{t,i} = \frac{1}{|R_i|} \sum_{(u,v)\in R_i} M_t(u, v), \tag{5}$$

where M_t denotes the ground-truth target mask of the current observation and R_i the pixel region corresponding to the i-th patch. At each frame, the target loss combines a per-patch binary cross-entropy with a soft Dice loss, where the former provides local patch-wise supervision and the latter encourages spatial overlap between the predicted and ground-truth masks:

$$\mathcal{L}_{\text{target}} = \frac{1}{N} \sum_{i=1}^{N} \text{BCE}(p_{t,i}, \bar{m}_{t,i}) + \text{Dice}(p_t, \bar{m}_t)\,. \tag{6}$$

Frames in which the target is not yet visible receive an all-zero target mask, training the head to capture target visibility as well as localization. To let the policy act on the predicted mask, a residual encoder aggregates the goal-conditioned features P_t and predicted mask p_t into a single vector δ_t. This vector is added to the readout token before the causal Transformer, so actions can depend on where the target is predicted to be and whether it is visible. We detach the residual encoder's inputs from the computation graph, so the mask head is trained solely by the target supervision (see Sec. 4.4 for a comparison of alternative feedback pathways).

3.4 BEHAVIORAL-PHASE CONDITIONING

Each task unfolds through three behavioral stages: searching until the target becomes visible, approaching the visible target, and interacting with it at close range. We explicitly model this progression with a behavioral-phase head h_ϕ (Fig. 2 right), which predicts the current behavioral phase from the temporal readout z_t. The predicted phase then modulates the readout before action prediction. For supervision, we assign a behavioral-phase label $y_t \in \{\text{Search}, \text{Approach}, \text{Interact}\}$ to each frame based on target visibility and annotated interaction events (Sec. B). The predicted phase distribution conditions the temporal readout through FiLM (Perez et al., 2018):

$$\begin{aligned} q_t &= \mathrm{softmax}(h_\phi(z_t)), \\ z'_t &= z_t + W_\beta\, \mathrm{sg}(q_t) + z_t \odot W_\gamma\, \mathrm{sg}(q_t), \end{aligned} \tag{7}$$

where h_ϕ is a linear head on the readout, $\mathrm{sg}(\cdot)$ denotes stop-gradient, $\odot$ indicates element-wise multiplication, and W_β and W_γ parameterize learned linear maps. FiLM allows the low-dimensional phase distribution to modulate the readout through feature-wise scaling and shifting, and the resulting z'_t is passed to the action head in place of z_t. We stop gradients through the phase distribution before FiLM modulation, ensuring that the phase head is trained solely by the phase supervision. For a sequence of T frames, the behavioral-phase loss and the overall training objective are given by

$$\mathcal{L}_\phi = -\frac{1}{T}\sum_{t=1}^{T} \log q_{t,\,y_t}, \qquad \mathcal{L} = \mathcal{L}_{\mathrm{BC}} + \lambda_{\mathrm{target}}\mathcal{L}_{\mathrm{target}} + \lambda_\phi \mathcal{L}_\phi. \tag{8}$$

We jointly optimize action imitation, target grounding, and behavioral-phase prediction, as given in Eq. (8), where $\lambda_{\mathrm{target}}$ and λ_ϕ weight the two auxiliary losses. We report their values and a sensitivity sweep in Sec. C. We use ground-truth target masks and phase labels of the current view only for supervision during training, while the policy conditions on its own grounding and phase predictions during both training and inference. At inference time, the policy operates solely on standard inputs without requiring ground-truth annotations or external target information.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Environment. We evaluate Attacca on goal-directed embodied control in Minecraft. All experiments run in MineStudio (Cai et al., 2024a). The policy receives egocentric RGB observations and acts in the mouse-and-keyboard action space of VPT (Baker et al., 2022). Each task specifies a target class and an interaction type. Attacca receives the target as a masked goal image taken in a world other than the execution world, and each baseline receives it in its own goal format (Fig. 14).

Training Dataset Construction. We collect demonstrations recorded by experienced Minecraft players in 1,160 different worlds for the seven training classes of each task. In each demonstration, the player searches for the target, approaches it, and interacts with it, and every frame is labeled with a target mask and a behavioral phase (Sec. B). Attacca is trained on these demonstrations, and every baseline other than the released ROCKET-2 checkpoint is trained on the same dataset from its released weights with its native recipe (Sec. F).

Task Design. *(1) Short-horizon tasks.* Mine, Hunt, and Place each require the agent to find a single target that is initially out of view and to complete the requested interaction, mining it, hunting it, or placing blocks at it, while objects of other classes are present. Each task covers ten target classes, seven seen in training (ID) and three held out (OOD), with 200 episodes per method. We report *clean success*, the fraction of episodes that achieve the goal with no wrong-class interaction, and *interaction precision*, the fraction of an episode's interactions that involve the target class (Sec. D). *(2) Long-horizon task chains.* The three chains are Diamond Pickaxe (oak log→diamond ore→crafting table), Wolf Feeding (coal→cow→furnace→wolf), and Nether Portal (water→lava→obsidian→portal→ignition), and each switches to the next goal after every completed stage. The world, position, and camera carry over between stages, so each target must be acquired from the state left by the previous stage. No method is trained on these chains, a shared GUI macro handles cooking and crafting interfaces, and each chain is run with 50 paired policy seeds per method (Sec. E).

Table 1: Results on short-horizon tasks. ID, OOD, and Avg. report clean success, and Prec. reports interaction precision. All baselines are trained on our demonstrations. †The released ROCKET-2 checkpoint, not trained on our data.

Policy	Goal modality	Mine				Hunt				Place			
		ID	OOD	Avg.	Prec.	ID	OOD	Avg.	Prec.	ID	OOD	Avg.	Prec.
STEVE-1	Text	0.007	0.000	0.005	0.091	0.079	0.000	0.055	0.180	0.000	0.000	0.000	0.042
JARVIS-VLA		0.114	0.000	0.080	0.727	0.200	0.083	0.165	0.468	0.121	0.000	0.085	0.521
STEVE-1	Video	0.029	0.000	0.020	0.210	0.057	0.000	0.040	0.158	0.000	0.000	0.000	0.000
GROOT		0.014	0.000	0.010	0.118	0.050	0.017	0.040	0.127	0.000	0.000	0.000	0.002
ROCKET-1	Current view	0.021	0.017	0.020	0.237	0.186	0.167	0.180	0.324	0.057	0.017	0.045	0.081
ROCKET-2	+ Molmo + SAM	0.057	0.033	0.050	0.580	0.157	0.300	0.200	0.443	0.021	0.000	0.015	0.035
ROCKET-2†	Different world image	0.007	0.000	0.005	0.333	0.093	0.117	0.100	0.232	0.000	0.000	0.000	0.216
ROCKET-2	+ mask	0.264	0.167	0.235	0.739	0.193	0.167	0.185	0.558	0.286	0.083	0.225	0.567
Attacca (Ours)		**0.450**	**0.250**	**0.390**	**0.778**	**0.479**	**0.467**	**0.475**	**0.725**	**0.550**	**0.283**	**0.470**	**0.579**

Baselines. STEVE-1 (Lifshitz et al., 2023) and JARVIS-VLA (Li et al., 2025a) receive text goals; STEVE-1 and GROOT (Cai et al., 2024b) accept video as goals; ROCKET-1 (Cai et al., 2025b) and ROCKET-2 (Cai et al., 2025a) take a current-view goal, a mask built online on the agent's current observation by pointing at the target with Molmo2 (Clark et al., 2026) and segmenting it with SAM 2 (Ravi et al., 2025). We also evaluate ROCKET-2 with the masked goal images from other worlds used by Attacca, both as the released checkpoint and after fine-tuning on our demonstrations. See Fig. 14 for examples of each method's goal specification, and Sec. F for implementation details.

4.2 Short-Horizon Task Results

We first evaluate existing embodied-agent baselines on single tasks across diverse environments, covering both seen and held-out classes. As shown in Tab. 1, STEVE-1 accepts both text and video goals, mapping them into a shared MineCLIP latent goal space. Yet, it is less effective due to limited object-level grounding across environments. JARVIS-VLA receives only a text goal, but benefits from Minecraft-specific visual-language and action post-training, including explicit spatial-grounding supervision, leading to substantially better performance than STEVE-1 and GROOT, particularly on seen classes. ROCKET-1 and ROCKET-2 with a current-view goal obtain a valid goal only when Molmo finds the target in the current observation. While the target is out of view, Molmo returns no point or points at wrong locations (Sec. G), so these pipelines stay at or below 5% on Mine and Place. On Hunt, animals wander into view on their own, and their success rises to 18–20%.

Fine-tuning ROCKET-2 on our demonstrations raises its success well above the released checkpoint on every task, which we attribute to learning from the full progression from search to interaction (Sec. B). With the same training data, this masked-goal-image policy also outperforms the text- and video-conditioned baselines on every task, suggesting that a masked goal image is a well-suited interface for goal-directed embodied control. Still, Attacca, which uses the same data, initialization, and goals as the fine-tuned ROCKET-2, consistently achieves the highest success rates on both seen and held-out classes. This gap highlights the importance of the proposed context-decoupled goal sampling, target-grounding supervision, and behavioral-phase conditioning.

4.3 Long-Horizon Task Results

We next extend our evaluation to long-horizon tasks, where multiple goals must be completed in sequence and each task starts from the state left by the previous one. Attacca completes 14–27 of 50 episodes per chain, while no baseline exceeds 4 (Tab. 2). ROCKET-2 with a current-view goal, for example, clears the first stage in up to 20% of the episodes but almost never continues, since after a goal switch the new target is often out of view and no goal can be constructed. The gap grows beyond the first stage, suggesting that Attacca's advantage accumulates across goal switches: it achieves higher next-stage completion. The largest gap occurs after mining the oak log, where Attacca proceeds to the diamond ores in nearly all successful episodes, compared with only one in seven for fine-tuned ROCKET-2 (Fig. 4). Attacca maintains a high and stable next-stage completion rate throughout each chain, reaching 85–93% at most transitions. Overall, it completes 5–7 times as many chains as the strongest baseline, suggesting that it substantially narrows the skill hand-off gap.

Table 2: Long-horizon task chains: per-stage unconditional success, so the last column of each chain is its chain success rate. †The released ROCKET-2 checkpoint, not trained on our data.

Policy	Goal modality		→	→		→	→	→		→	→	→	→
STEVE-1	Text	0.16	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
JARVIS-VLA		0.02	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
STEVE-1	Video	0.58	0.02	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
GROOT		0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00
ROCKET-1	Current view	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00
ROCKET-2	+ Molmo + SAM	0.06	0.04	0.00	0.20	0.04	0.00	0.00	0.18	0.02	0.02	0.00	0.00
ROCKET-2†	Different world image	0.06	0.00	0.00	0.06	0.02	0.00	0.00	0.06	0.00	0.00	0.00	0.00
ROCKET-2	+ mask	0.56	0.08	0.08	0.54	0.18	0.10	0.06	0.46	0.32	0.22	0.08	0.04
Attacca (Ours)		**0.84**	**0.78**	**0.54**	**0.66**	**0.40**	**0.34**	**0.30**	**0.74**	**0.64**	**0.56**	**0.32**	**0.28**

Table 3: Component ablation. Success with residual feedback and phase conditioning toggled separately (± standard error, n=50).

Residual	Phase			
–	–	0.50 ±.07	0.12 ±.05	0.18 ±.05
✓	–	**0.54** ±.07	0.14 ±.05	0.08 ±.04
–	✓	0.42 ±.07	0.28 ±.06	0.06 ±.03
✓	✓	**0.54** ±.07	**0.30** ±.06	**0.28** ±.06

Table 4: Feedback-pathway ablation. Success with each pathway from the predicted mask to the policy (± standard error, n=50).

Feedback pathway			
Cross-attention	0.36 ±.07	0.08 ±.04	0.18 ±.05
Layerwise FiLM	0.42 ±.07	0.10 ±.04	0.18 ±.05
ControlNet	0.16 ±.05	0.00 ±.00	0.08 ±.04
Residual (**Ours**)	**0.54** ±.07	**0.30** ±.06	**0.28** ±.06

4.4 Ablation Study

Components. Tab. 3 ablates the core components of Attacca on the three long-horizon chains. We begin with dense target grounding alone, where the mask head is supervised but its prediction is not fed back to the policy, and independently add residual mask feedback (Sec. 3.3) and behavioral-phase conditioning (Sec. 3.4). Neither component helps consistently on its own. Residual feedback alone barely changes Diamond Pickaxe and Wolf Feeding and lowers Nether Portal from 0.18 to 0.08, while phase conditioning alone raises Wolf Feeding from 0.12 to 0.28 but lowers the other two chains. Combining both improves all three chains by 0.04, 0.18, and 0.10 over the first row. This suggests that phase conditioning indicates when to transition from search to approach, while residual feedback provides the target location needed to act on that transition.

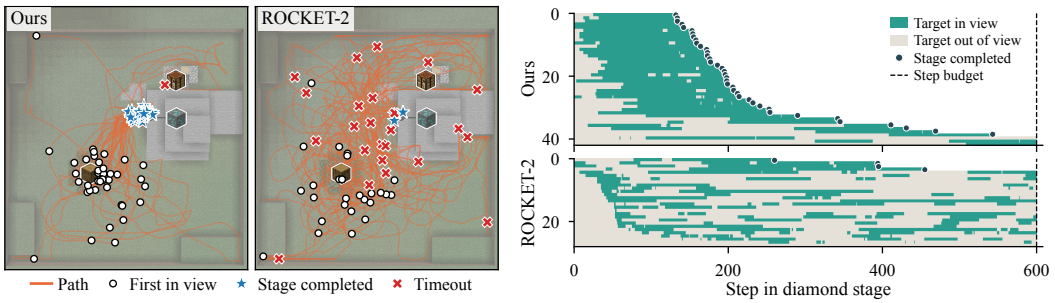
Feedback pathway. With target supervision and phase conditioning fixed, we compare four mask-injection pathways, all using the same mask head and supervision. The residual pathway adds a single grounding vector to the readout token before the causal Transformer. Cross-attention lets the readout hidden state after the first layer attend to the 14×14 patch features, with attention biased toward the predicted mask. Layerwise FiLM modulates all tokens before every layer and ControlNet runs a trainable Transformer copy in parallel with the frozen original. As shown in Tab. 4, the residual pathway performs best on all three chains. We hypothesize that a single residual input lets the policy absorb the large variation in mask presence across behavioral phases. Layerwise FiLM and ControlNet instead inject these variations into the Transformer computation, which may make the policy overly sensitive to phase-dependent changes in mask presence and scale. Cross-attention, which gives the readout direct access to mask-weighted patch features, also remains below the residual, suggesting that the target's presence and location, summarized in a single vector at the input, are sufficient for acting on the mask prediction, and that finer patch-level features add no benefit.

4.5 Qualitative Analysis

We first examine every episode that reaches the diamond stage of the Diamond Pickaxe chain in Tab. 2, comparing Attacca with the fine-tuned ROCKET-2 (Fig. 4). Once the diamond ore enters Attacca's view, it stays in view almost continuously until the stage

Table 5: Success by goal image, pooled over both scenarios (± standard error, n=100).

Policy	Same world (close)	Same world (far)	Different world
ROCKET-2	0.75 ±0.04	0.24 ±0.04	0.30 ±0.05
Ours	**0.89** ±0.03	**0.89** ±0.03	**0.81** ±0.04



(a) Trajectories (b) Target visibility over time

Figure 4: Diamond stage of the Diamond Pickaxe chain for all episodes that reach it (Attacca 42, ROCKET-2 28). (a) Trajectories. (b) Whether the diamond ore stays in view after it first appears.

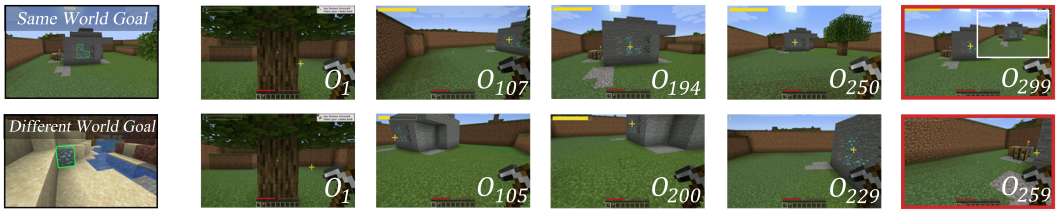


Figure 5: Fine-tuned ROCKET-2 in the diamond scenario with a goal image taken far from the target in the execution world (top) or in a different world (bottom). Frames overlay its predicted target point and visibility, and red borders mark the final frame of a failed episode.

is completed (Fig. 4b), and Attacca's paths converge on it. The diamond ore also enters ROCKET-2's view in 27 of its 28 episodes, but it repeatedly leaves the view afterward, and ROCKET-2's paths spread over the arena until the 600-step budget runs out. It completes the stage in only 4 of these 27 episodes, against 39 of 42 for Attacca, which indicates weaker target-directed control after the target is found rather than a failure to find it. To see why, Fig. 5 follows ROCKET-2 in single rollouts of the diamond scenario with two fixed goal images, and Tab. 5 reports success by goal image. Sec. G adds the other methods and the oak scenario.

We deliberately capture the same-world goal in the top row of Fig. 5 far from the target, so that reproducing its view and reaching the target lead to different places. Grounding works, and the predicted point is correct at o_{107}, but after nearing the target at o_{194} the agent backs away and settles at o_{299}, where its view reproduces the goal view, instead of interacting. With the different-world goal in the bottom row, grounding starts out correct at o_{200}, but visibility collapses at o_{229} and the agent walks past the diamond, as in Fig. 4. This suggests that ROCKET-2, trained with goals from the same world as its demonstrations, acts by reproducing the goal view rather than by pursuing the masked target. Reproducing the goal view works only for close same-world goals, whose view already places the agent within reach. With these goals ROCKET-2 reaches 0.75 success, but only 0.24 with far goals and 0.30 with different-world goals, whereas Attacca stays at 0.81–0.89 for all three (Tab. 5). Attacca is trained with goal images from other worlds, so reproducing the goal view cannot lead it to the target, and it must pursue the masked target instead. Given the goal image that ROCKET-2 receives in the bottom row of Fig. 5, Attacca switches from Search to Approach when its predicted mask first activates and mines all three diamond ores (Sec. G).

5 Conclusion

We presented Attacca, a visual goal-conditioned policy for goal-directed embodied control under state continuity, in which each task starts from the state left by the previous one and the goal image comes from a different world. Context-decoupled goal sampling, goal-conditioned target grounding, and behavioral-phase conditioning let a single policy search for, approach, and interact with targets from such inherited states. Attacca achieves the highest success on all short-horizon tasks for both seen and held-out classes and completes 5–7 times as many long-horizon chains as the strongest baseline. Extending Attacca to other embodied domains and to learned planners is left for future work, and Sec. H discusses the limitations of our evaluation.

AI USE STATEMENT

In this work, we used generative AI tools to aid in correcting grammatical errors and improving wording. We have not used generative AI tools for data collection, research conceptualization, experiments, or analysis. We have reviewed all AI-assisted text, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, 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, Kaichen He, Bowei Zhang, Xinyue Zheng, Anji Liu, and Yitao Liang. Minestudio: A streamlined package for minecraft ai agent development. *arXiv preprint arXiv:2412.18293*, 2024a.

Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot: Learning to follow instructions by watching gameplay videos. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 5523–5554, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/16986b69068fbe6acf64eb6566519c74-Paper-Conference.pdf.

Shaofei Cai, Zhancun Mu, Anji Liu, and Yitao Liang. Rocket-2: Steering visuomotor policy via cross-view goal alignment, 2025a. URL https://arxiv.org/abs/2503.02505.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12122–12131, June 2025b.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660, October 2021.

Yuanpei Chen, Chen Wang, Li Fei-Fei, and Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3809–3829. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/chen23e.html.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Rohun Tripathi, Sangho Lee, Mohammadreza Salehi, Jason Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28652–28668, June 2026.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL https://aclanthology.org/P19-1285/.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9118–9147. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/huang22a.html.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/huang23c.html.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/e0cf1f47118daebc5b16269099ad7347-Paper.pdf.

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International conference on learning representations*, 2019.

Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 406–416. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/lee22a.html.

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/.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *Advances in neural information processing systems*, 37:49881–49913, 2024.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-2: Multimodal minecraft agent with goal-observation-action conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9039–9049, June 2025b.

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.

Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, and Wei Yang. Juewu-mc: Playing minecraft with sample-efficient hierarchical reinforcement learning. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 3257–3263. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/452. URL https://doi.org/10.24963/ijcai.2022/452. Main Track.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/lynch20a.html.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988. URL https://proceedings.neurips.cc/paper_files/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu

(eds.), *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/45c1f6a8cbf2da59ebf2c802b4f742cd-Paper-Conference.pdf.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimír Vondruš, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 251–266. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/021bbc7ee20b71134d53e20206bd6feb-Paper.pdf.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian (Shawn) Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with llms enables open-world multi-task agents. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34153–34189. Curran Associates, Inc., 2023. doi: 10.52202/075280-1480. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6b8dfb8c0c12e6fafc6c256cb08a5ca7-Paper-Conference.pdf.

Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Ma, and Yitao Liang. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 73278–73308. Curran Associates, Inc., 2024b. doi: 10.52202/079017-2331. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/85f1225db986e629289f402c46eff1a4-Paper-Conference.pdf.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(3):1894–1907, 2025. doi: 10.1109/TPAMI.2024.3511593.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Plan4MC: Skill reinforcement learning and planning for open-world Minecraft tasks. *arXiv preprint arXiv:2303.16563*, 2023.

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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78221–78259. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zheng25j.html.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*,

volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Example of Context-Decoupled Goal Sampling

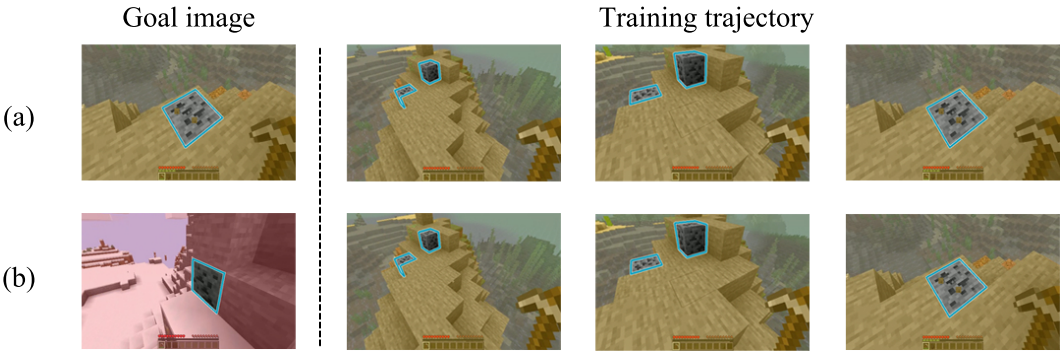


Figure 6: (a) Context-correlated sampling draws goal images from the same world as the training trajectory. (b) Our context-decoupled sampling draws goal images of the same target class from a different world.

Figure 6 pairs one training trajectory with two goal images. Under context-correlated sampling (a), the goal image comes from the same world as the trajectory and shows the same surrounding terrain, so matching the scene can substitute for locating the target. Under context-decoupled sampling (b), the goal image shows the same target class in another world with different terrain. It shares no scene content with the observations, and the appearance of the target is the only usable cue.

Robustness to the goal background. We evaluate Attacca and the fine-tuned ROCKET-2 on the mine diamond and hunt cow tasks with four goal images of different backgrounds, over 50 episodes per goal (Fig. 7). Attacca succeeds in 0.68–0.96 of the episodes for every goal image, whereas ROCKET-2 stays at 0.08–0.28, so the gain of Attacca holds for every goal background rather than depending on a particular goal image.

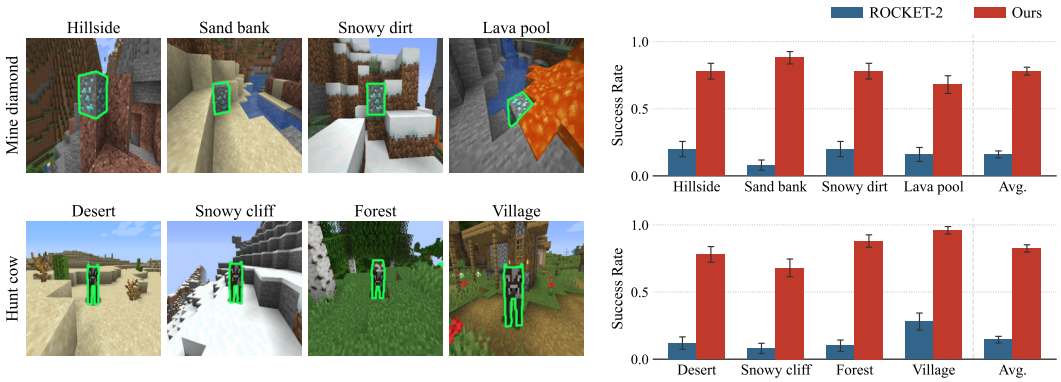


Figure 7: Success with goal images of four different backgrounds, 50 episodes per goal.

B SEARCH-TO-INTERACT DATASET

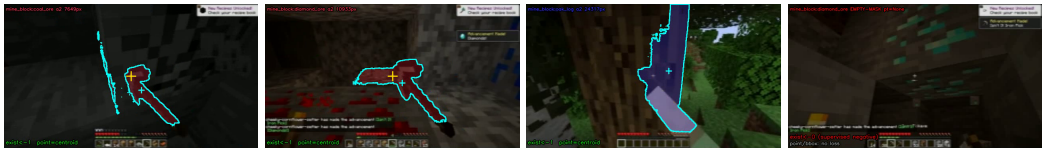


Figure 8: Errors of hindsight labels. In the two left frames, the center prompt selects the held tool, which is then labeled as *coal ore* and *diamond ore* while the ore in view stays unlabeled. In the third frame, the oak-log mask spreads onto the agent's body and shifts the mask centroid. In the right frame, diamond ore fills the view, but the frame is labeled as target-absent.

Limitations of hindsight labeling. Prior goal-conditioned agents are trained with labels produced in hindsight (Cai et al., 2025a). An interaction event is detected in gameplay, a segmentation model is prompted at the center of the frame just before the event, and the resulting mask is tracked backward in time. Figure 8 shows typical errors of this process, including masks on the held tool, masks that spread onto the player's arm, and empty masks while the target is still visible. Visibility and grounding supervision from such labels is noisy, and search frames cannot be identified reliably.

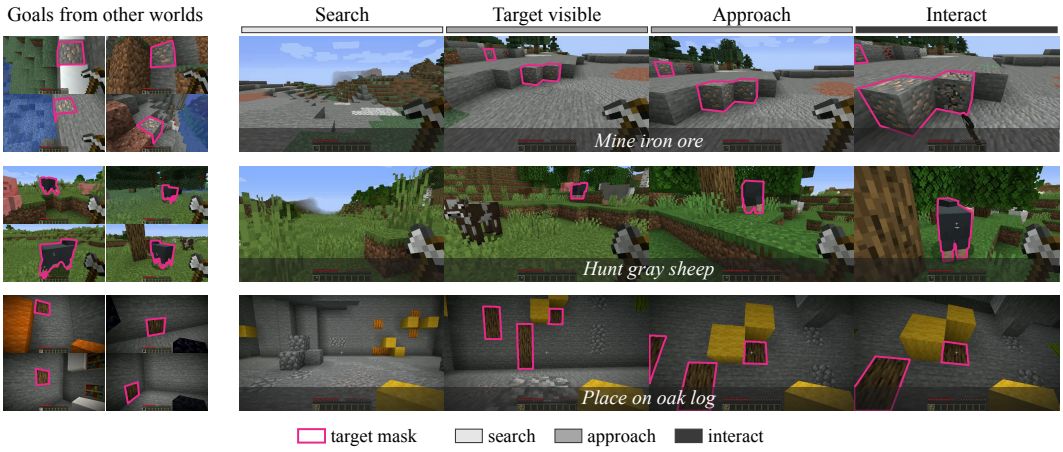


Figure 9: Search-to-interact demonstrations. For each task, one human demonstration is shown at four moments, from the start of the episode through the first view of the target and the approach to the interaction, with its target mask (outline) and behavioral phase (bar above each frame). On the left are example goal images of the same target class, each taken from a demonstration recorded in a different world and shown with its target mask.

The HumanPlay dataset. We collect 1,160 demonstrations with 284,961 frames in total, recorded by experienced Minecraft players, each in a different world. A demonstration starts at a random spawn point in a new world and covers the full progression from search to approach and interaction for one of the seven training classes of its task (Figure 9). The demonstrator continues until a task-specific interaction quota is reached, so a single episode contains several search-to-interact cycles. Every scene contains objects of other classes along with the target instances, and since the demonstrator interacts only with the requested target, the demonstrations also show target selection among distractors. The dataset contains Mine (822 episodes, 209,593 frames), Hunt (198 episodes, 47,991 frames), and Place (140 episodes, 27,377 frames), and these counts include the 10% of episodes held out for validation. The environment is built on MineStudio (Cai et al., 2024a).

Labels. Per-frame target masks come from the renderer, which back-projects the depth buffer and assigns visible pixels to target instances. The mask covers all visible surfaces of the instances of the requested target class. Frames with no visible target, 35% of all frames, receive an all-zero mask and

thus supervise target absence. Each frame is also labeled with a behavioral phase. A frame is Interact if it contains an attack or use event on the selected target, Approach if the target is visible or has been seen and is temporarily out of view, and Search otherwise. Interact covers 33% of the frames, Approach 47%, and Search 20%. These labels are used only as training targets and are never given to the policy at inference.

C Implementation Details

Table 6 lists the architecture and training settings of Attacca. The policy is initialized from the released ROCKET-2 checkpoint (Cai et al., 2025a). Goal images are sampled as described in Sec. 3.2, using only target-visible frames, which are resized to 224×224. A new reference demonstration and goal frame are drawn each time a training sample is loaded.

$\mathcal{L}_{\mathrm{BC}}$ is summed over the 128 steps of a window and $\mathcal{L}_\phi$ is averaged over frames. $\mathcal{L}_{\mathrm{target}}$ is the mean of its averages over frames with and without a visible target. Varying each loss weight over a $16\times$ range with the other held at its default changes success on Mine by at most 4.5 points for $\lambda_{\mathrm{target}}$ and 6.0 points for λ_ϕ (Table 7), within the $\pm$6.7-point 95% binomial confidence interval at n=200.

Table 6: Architecture and training settings of Attacca.

Component	**Setting**
Observation and goal image	224×224 RGB
Visual encoder	DINO ViT-B/16 (Caron et al., 2021), frozen, 196 patch tokens
Goal-mask encoder	ViT-Tiny/16, ImageNet-21k initialization, trainable
Goal-fusion module	Transformer encoder, 4 layers, 8 heads, 9 query tokens
Causal Transformer	Transformer-XL (Dai et al., 2019), 4 layers, width 1024
Mask head h_{mask}	per-patch MLP over the 14×14 patch grid
Residual encoder inputs	mask-weighted pooled P_t, p_t, and the mean, maximum, and entropy of p_t
Residual projection	1024×1024, zero-initialized
Phase head h_ϕ	linear, with zero-initialized W_β and W_γ
Action space	VPT mouse and keyboard (Baker et al., 2022)
Initialization	released ROCKET-2 checkpoint (Cai et al., 2025a)
Training data	1,160 episodes, 284,961 frames, 10% held out for validation
Window length	128 frames
Batch size	12 windows (3 GPUs $\times$ 4)
Optimization steps	1,350, with 200 warmup steps
Learning rate	2×10^{-5}
Weight decay	10^{-3}
Loss weights	$\lambda_{\mathrm{target}} = 0.5$, $\lambda_\phi = 0.1$
Training seed	2026

Table 7: Loss-weight sensitivity on Mine (n=200 per cell). Each weight is varied with the other held at its default; bold marks the defaults used in all main results.

$\lambda_{\mathrm{target}}$	0.125	0.25	**0.5**	1.0	2.0
Mine	0.355	0.345	0.390	0.370	0.380
λ_ϕ	0.025	0.05	**0.1**	0.2	0.4
Mine	0.330	0.355	0.390	0.375	0.370

D SHORT-HORIZON TASK DETAILS



Figure 10: Short-horizon tasks. Each row shows four frames of one successful Attacca episode, from the first frame through target discovery and approach to the interaction. For each task, the upper row shows a class seen in training and the lower row a held-out class.

In each short-horizon task, a single target starts out of view among objects of other classes, and the goal image comes from another world (Figure 10). Each task has ten target classes (Tables 8–10), seven seen in training and three held out (OOD), and all methods are evaluated on the same scenes and action budgets with 200 episodes per task, 140 of them on seen classes.

Mine. Mine uses seven ores seen in training and three held-out coral blocks on common terrain, each evaluated in ten worlds with two policy seeds. The staged target is swapped between classes while the terrain stays fixed, and held-out classes never appear as distractors for other classes.

Hunt and Place. Hunt uses seven seen mobs, including four sheep colors, and three held-out mobs, each in ten layouts. Place uses seven seen and three held-out target blocks, each in ten worlds, and the agent places its held block onto the target. Outcomes and off-target interactions are scored from the world state, which the scorer can query but the policy cannot observe.

Clean success. An episode is a clean success if it achieves the requested outcome without any wrong-class interaction, which means breaking the target block without breaking blocks of other classes in Mine, killing the target without killing a bystander in Hunt, and filling every target cell without off-target placements in Place. The criterion does not change episode termination or action budgets, so an episode that reaches the goal after a wrong-class interaction counts as a failure.

Table 8: Mine, per-class clean success and interaction precision (share of an episode's interactions on the target class). Each class is tested 20 times per method.

Policy	**Goal modality**	ID							OOD			**avg**	**prec**
STEVE-1	Text	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.005	0.091
JARVIS-VLA		0.15	0.00	0.10	0.15	0.05	0.15	0.20	0.00	0.00	0.00	0.080	0.727
STEVE-1	Video	0.05	0.05	0.05	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.020	0.210
GROOT		0.00	0.00	0.05	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.010	0.118
ROCKET-1	Current view	0.00	0.00	0.00	0.05	0.05	0.00	0.05	0.00	0.05	0.00	0.020	0.237
ROCKET-2	+ Molmo + SAM	0.00	0.05	0.15	0.10	0.05	0.00	0.05	0.00	0.10	0.00	0.050	0.580
ROCKET-2†	Different world image	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.005	0.333
ROCKET-2	+ mask	0.25	0.40	0.15	0.35	0.20	0.25	0.25	0.10	0.20	**0.20**	0.235	0.739
Ours		**0.45**	**0.45**	**0.20**	**0.60**	**0.45**	**0.30**	**0.70**	**0.30**	**0.25**	**0.20**	**0.390**	**0.778**

Table 9: Hunt, per-class clean success and interaction precision (share of an episode's interactions on the target class). Each class is tested 20 times per method.

Policy	**Goal modality**	ID							OOD			**avg**	**prec**
STEVE-1	Text	0.00	0.00	0.05	0.30	0.15	0.05	0.00	0.00	0.00	0.00	0.055	0.180
JARVIS-VLA		0.30	0.25	0.15	0.20	0.25	0.00	0.25	0.00	0.05	0.20	0.165	0.468
STEVE-1	Video	0.00	0.30	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.00	0.040	0.158
GROOT		0.15	0.00	0.00	0.15	0.00	0.05	0.00	0.05	0.00	0.00	0.040	0.127
ROCKET-1	Current view	0.00	0.30	**0.35**	0.15	0.05	0.15	0.30	0.10	0.20	0.20	0.180	0.324
ROCKET-2	+ Molmo + SAM	0.00	**0.45**	**0.35**	0.15	0.00	0.00	0.15	**0.30**	0.35	0.25	0.200	0.443
ROCKET-2†	Different world image	0.30	0.05	0.00	0.05	0.15	0.05	0.05	0.15	0.05	0.15	0.100	0.232
ROCKET-2	+ mask	0.25	0.30	0.10	0.15	0.30	0.15	0.10	0.10	0.15	0.25	0.185	0.558
Ours		**0.75**	0.20	**0.35**	**0.50**	**0.45**	**0.60**	**0.50**	0.15	**0.45**	**0.80**	**0.475**	**0.725**

Table 10: Place, per-class clean success and interaction precision (share of an episode's interactions on the target class). Each class is tested 20 times per method.

Policy	**Goal modality**	ID							OOD			**avg**	**prec**
STEVE-1	Text	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000	0.042
JARVIS-VLA		0.00	0.05	0.25	0.10	0.20	0.15	0.10	0.00	0.00	0.00	0.085	0.521
STEVE-1	Video	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000	0.000
GROOT		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000	0.002
ROCKET-1	Current view	0.05	0.05	0.10	0.15	0.00	0.05	0.00	0.05	0.00	0.00	0.045	0.081
ROCKET-2	+ Molmo + SAM	0.00	0.10	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.015	0.035
ROCKET-2†	Different world image	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.000	0.216
ROCKET-2	+ mask	0.10	0.35	0.40	0.35	0.35	0.35	0.10	0.15	0.05	0.05	0.225	0.567
Ours		**0.35**	**0.70**	**0.55**	**0.65**	**0.65**	**0.60**	**0.35**	**0.40**	**0.35**	**0.10**	**0.470**	**0.579**

E LONG-HORIZON TASK DETAILS

We evaluate three chains of increasing length, shown in Figures 11, 12, and 13. Diamond Pickaxe requires mining an oak log, mining three diamond ores, and opening a placed crafting table. Wolf Feeding requires mining coal, hunting a cow, opening a furnace, and feeding cooked beef to a tamed wolf. Nether Portal requires scooping water, pouring it at a marked site, mining the resulting obsidian, and completing and igniting a portal frame. Each chain uses one fixed world and 50 paired policy seeds per method, and no method is trained on these chains.



Figure 11: Diamond Pickaxe chain (oak log→diamond ore→crafting table). Each row is one stage of a single successful Attacca episode, shown from the first frame of the stage through target discovery and approach to the interaction.

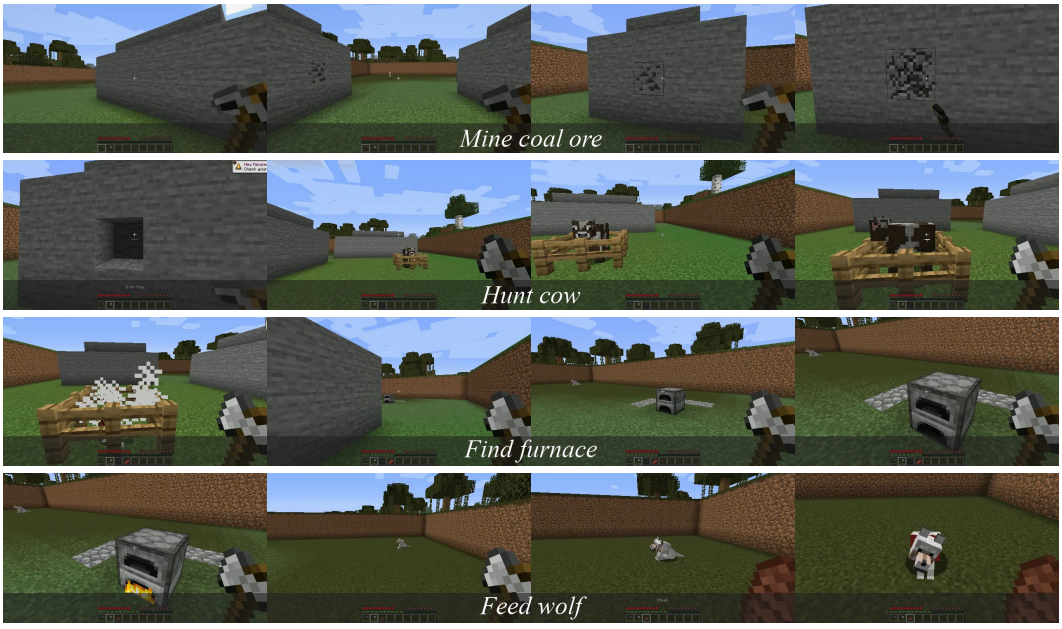


Figure 12: Wolf Feeding chain (coal→cow→furnace→wolf), shown in the same format as Figure 11.

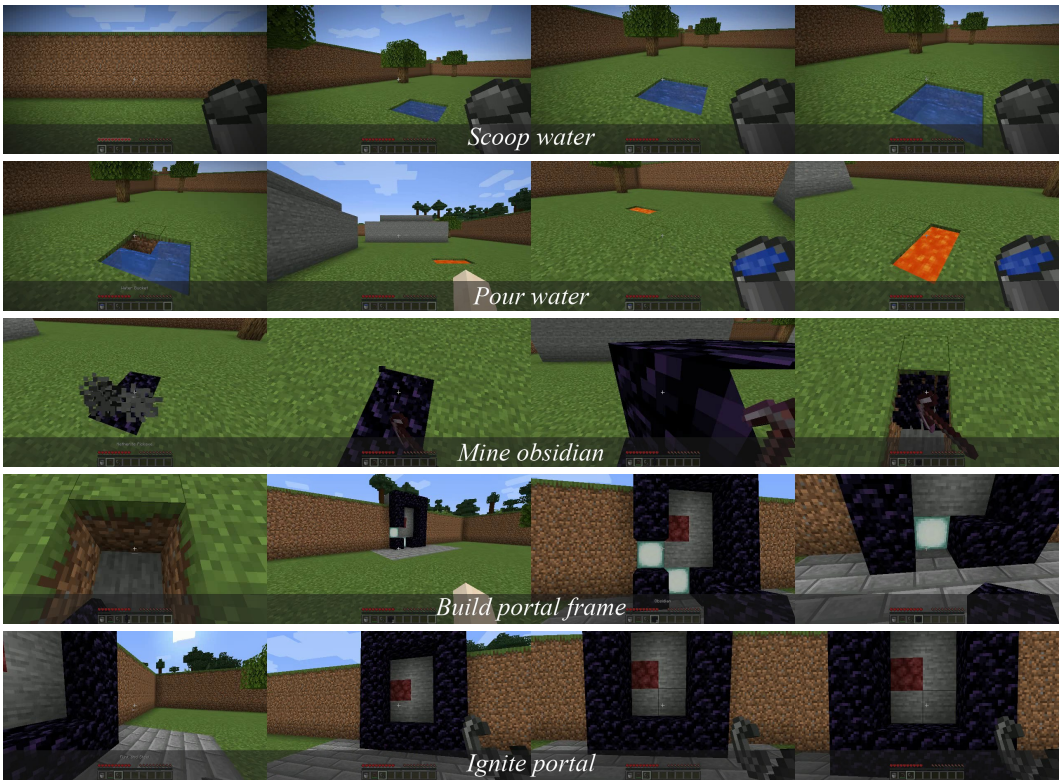


Figure 13: Nether Portal chain (water→lava→obsidian→portal→ignition), shown in the same format as Figure 11.

Stage controller and helpers. A fixed stage controller switches the goal once the current stage is completed and attributed to the policy, and no language plan is generated. The controller also supplies stage quotas such as the three diamond ores, because the goal image does not encode counts. The world state, agent position, and camera orientation carry over without teleportation or forced camera alignment. GUI macros craft sticks after the oak stage, cook beef after the policy opens the furnace, and move the cooked beef to the hotbar. When the next stage needs a different item in hand, a helper equips it right after the previous stage is completed. Diamond Pickaxe scoring ends when the policy opens the existing table.

F Baseline Implementation

All baselines except the released ROCKET-2 checkpoint are trained on our demonstrations, starting from their released weights and using their native recipes and conditioning interfaces. STEVE-1 (Lifshitz et al., 2023) is evaluated with text goals and with video goals, GROOT (Cai et al., 2024b) with reference videos, and JARVIS-VLA (Li et al., 2025a) with text instructions. ROCKET-1 (Cai et al., 2025b) and ROCKET-2 (Cai et al., 2025a) receive a current-view goal, built online by pointing at the target in the agent's observation with Molmo2-4B (Clark et al., 2026) and segmenting it with SAM 2 (Ravi et al., 2025). These models are part of the baseline pipelines and are not used by Attacca. ROCKET-2 is also evaluated with different-world goal images, both as the released checkpoint and after training on our dataset.

F.1 Goal Specification per Method

All methods are evaluated on the same task instances, and the only difference in their inputs is how the target is specified (Figure 14). Text-goal policies receive an instruction naming the target class, and STEVE-1 receives a comma-separated list of equivalent commands, which its text prior maps to a visual goal embedding. Video-goal policies receive a clip of a human completing the same task on the same class (16 frames for STEVE-1 and 128 for GROOT), taken from demonstrations not used for training. ROCKET-1 and ROCKET-2 with Molmo receive no separate goal. Every 30 steps, Molmo2-4B is asked to "Point to ⟨class⟩" in the agent's current view, SAM 2 segments the object at the returned point, and the view with this mask becomes the goal, which ROCKET-2 holds until the next query and ROCKET-1 tracks at every step. ROCKET-2†, ROCKET-2, and Attacca receive the same goal image, taken in a different world, with its target mask (Figure 15).

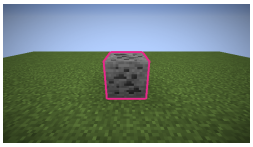
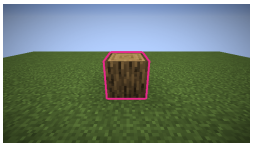
Goal modality	**Policy**	**Mine**: coal ore	**Hunt**: cow	**Place**: on oak log
Text	STEVE-1	*"mine coal ore, dig coal ore, collect coal ore, get coal ore"*	*"kill the cow, hunt the cow, attack the cow"*	*"place a block on the oak log, put a block on the oak log"*
	JARVIS-VLA	*"mine coal ore"*	*"hunt cow"*	*"place a block on the oak log"*
Video	STEVE-1			
	GROOT			
Current view o_t + Molmo + SAM	ROCKET-1 ROCKET-2			
Different world image + mask	ROCKET-2† ROCKET-2 **Attacca (Ours)**			

Figure 14: Goal given to each method for one target class per task. Video goals are shown by the first, middle, and last frame of the clip. Current-view goals show the agent's observation o_t at a Molmo query step, with the Molmo point (white dot) and the SAM 2 mask (orange outline) of the ROCKET-2 pipeline. At t=0 in Mine the target is not yet in view, and the point falls on the held pickaxe. Different-world goals show the goal image with its target mask (magenta outline).



Figure 15: Goal images of all short-horizon target classes, cropped around the target, with the target mask outlined in magenta. Mine and Place goals share one camera pose and target position.

G Additional Failure Analysis

Fig. 16 shows the diamond scenario of Sec. 4.5 for three goal specifications, and Fig. 17 shows the oak scenario in the same format. Row (a) uses the current-view goal, which Molmo and SAM 2 construct from the agent's observation every 30 steps. At o_1 the diamond is out of view, so Molmo finds no target and no goal is built. The diamond enters the view at o_{121} and the policy approaches it, but at o_{151} the mask covers the whole stone structure instead of the ore, and later points land on UI elements. Online goal construction therefore cannot solve search when the target starts out of view.

Row (b) repeats the fine-tuned ROCKET-2 rollouts of Fig. 5. Row (c) shows Attacca with the bottom goal image of (b) and no external grounding module. It switches from Search to Approach at o_{87}, when the predicted mask first activates, begins interacting at o_{182}, and mines the third diamond ore by o_{264}. The oak scenario shows the same failure modes. The online pipeline cannot build a goal while the log is out of view, ROCKET-2 aligns its view with a same-world goal instead of interacting and loses the target under a different-world goal, and Attacca completes the task.



Figure 16: Diamond scenario. (a) Current-view goal with Molmo points (purple) and SAM 2 masks (green). (b) ROCKET-2 with same-world (top) and different-world (bottom) goals, with its predicted point and visibility. (c) Attacca with the bottom goal image of (b) and its predicted phase (yellow).



Figure 17: Failure analysis on the mine-oak-log task, following the format of Figure 16.

H LIMITATIONS

General embodied intelligence would require an agent to carry out every part of a long-horizon task on its own, including item handling and task planning. Our evaluation isolates goal-directed embodied control in Minecraft and leaves these parts to the evaluation system. First, helper functions assist the agent at task transitions. For example, when the agent finishes cooking beef at the furnace, a macro moves the cooked beef to the hotbar and a helper equips it once the beef is cooked, so the agent only needs to find the wolf and feed it. Second, a scripted stage controller switches the goals and detects stage completion from the simulator state. We have not established how an LLM planner would connect to our policy at deployment, for example how its language subgoals would be turned into masked goal images and how it would decide that a stage is completed.