Open-World Skill Discovery from Unsegmented Demonstration Videos

Jingwen Deng[1*] Zihao Wang[1*] Shaofei Cai[1] Anji Liu[2] Yitao Liang[1†]
[1]Peking University [2]University of California, Los Angeles
{dengjingwen,zhwang,caishaofei}@stu.pku.edu.cn, liuanji@cs.ucla.edu, yitaol@pku.edu.cn

Abstract

Learning skills in open-world environments is essential for developing agents capable of handling a variety of tasks by combining basic skills. Online demonstration videos are typically long but unsegmented, making them difficult to segment and label with skill identifiers. Unlike existing methods that rely on random splitting or human labeling, we have developed a self-supervised learning-based approach to segment these long videos into a series of semantic-aware and skill-consistent segments. Drawing inspiration from human cognitive event segmentation theory, we introduce ***Skill Boundary Detection*** *(SBD), an annotation-free temporal video segmentation algorithm. SBD detects skill boundaries in a video by leveraging prediction errors from a pretrained unconditional action-prediction model. This approach is based on the assumption that a significant increase in prediction error indicates a shift in the skill being executed. We evaluated our method in Minecraft, a rich open-world simulator with extensive gameplay videos available online. The SBD-generated segments yielded relative performance improvements of 63.7% and 52.1% for conditioned policies on short-term atomic tasks, and 11.3% and 20.8% for their corresponding hierarchical agents on long-horizon tasks, compared to unsegmented baselines. Our method can leverage the diverse YouTube videos to train instruction-following agents. The project page is at https://craftjarvis.github.io/SkillDiscovery/.*

1. Introduction

Most existing LLM-based instruction-following agents adopt a two-tier architecture of a planner and a controller. The planner decomposes high-level instructions into atomic skills, which the controller then maps to actions based on current observations for interaction with the environment [8, 28, 39, 41–43]. Training such agents typically involves segmenting long trajectories into atomic skill sequences, followed by separate training of the planner and controller.

Method	Illustration	Type
Random splitting	l l $l = 128$ *or* $l \sim U(m, M)$	rule-based
Reward-driven	Rewards r_1 r_2 Option learning → stop function	rule-based
Top-down	Human labels h_1 h_2 Supervised learning → boundaries	rule-based
Bottom-up	Skill 1 Skill 2	rule-based
Ours (SBD)	Action prediction Groundtruth	learning-based

Table 1. **Comparisons between existing segmentation methods and our method SBD.** Existing methods usually rely on human-designed rules, while our method is learning-based. **Random splitting** can result in a single skill spanning different segments or multiple skills located within one segment. **Reward-driven** methods require additional reward information, which is challenging for human annotators to label. **Top-down** methods often result in limited skill diversity and high computation costs. **Bottom-up** methods struggle in visually partially observable environments. ▢ are visual observations and ○ are actions.

Learning skills from long-sequence videos is critical for building such hierarchical agents. However, real-world video data are often long-horizon, unsegmented, and with no fine-grained skill labels. In addition, the concept of a "skill" is ill-defined and varies widely across domains such as video gaming [6, 20, 39, 49], robotic control [56], and autonomous driving [40]. This ambiguity, along with the immense diver-

*Equal Contribution
†Corresponding Author

sity of skills in open worlds, makes skill learning particularly challenging, especially in partially observable settings. To enable open-world skill learning from unsegmented demonstration videos, the first challenge is to segment them into semantically meaningful, self-contained skills.

We list the existing segmentation methods in Tab. 1. The naive **random splitting** methods [6, 29] divide videos into segments of predefined lengths (e.g., fixed length, uniform distribution). However, they do not ensure that each segment contains a distinct skill. Additionally, predefined lengths may not match the actual distribution of skill length in real world (see Sec. 4.4). **Reward-driven** methods [37] discover skills through the environment's reward signal. They cannot capture skills with no associated rewards and may split one skill into multiple segments when rewards are repeatedly gained during execution. **Top-down** methods [4, 35] rely on predefined skill sets from human experts. They use manual labeling or supervised learning to segment videos. Although this method can produce reasonable results, it is expensive and limited by the narrow range of predefined skills. **Bottom-up** methods [31, 55] use algorithms such as agglomerative clustering or byte-pair encoding [16] to split action sequences. However, they struggle in visually partially observable settings where both observations and actions must be considered. All the methods above rely on human-designed rules to segment the videos, which highlights the need for a learning-based method that can adaptively segment skills from unsegmented videos in open world.

Inspired by event segmentation theory (EST) [47], which proposes that humans naturally split continuous experiences into discrete events when prediction errors in perceptual expectations rise, we introduce Skill Boundary Detection (SBD)—a method for automatically identifying potential skill boundaries in long trajectories. SBD employs a predictive model trained on a dataset of unsegmented videos to predict future actions based on past observations, effectively capturing temporal dependencies [3]. We then use this pretrained model to make predictions on unsegmented videos and compare them to the ground-truth actions. A significant increase in prediction error indicates a shift in the skill being executed (Theorem 3.4), enabling us to detect boundaries between different skills. SBD relies on self-supervised learning, removing the need for additional human labeling. This allows SBD to utilize a wide range of YouTube videos to train instruction-following agents.

We evaluate SBD in Minecraft [15, 19, 30], a rich open world simulator. First, we apply SBD to create a segmented Minecraft video dataset. We then re-train a video-conditioned policy [6] and a language-conditioned policy [29] on this dataset. We evaluate them on a diverse Minecraft skills benchmark [30]. The improved versions of these policies result in relative performance increases of 63.7% and 52.1% compared to the original ones, respectively. We also test their corresponding hierarchical agents that use behavioral cloning and in-context learning, achieving relative performance increases of 11.3% and 20.8%, respectively. These findings show the effectiveness of our proposed method for skill discovery from unsegmented demonstrations and its potential to advance open-world skill learning.

2. Problem Formulation

Skills in Behavioral Cloning. Behavioral cloning learns a policy $\pi : \mathcal{O} \to \mathcal{A}$ from expert demonstrations, mapping observations to actions based on pairs $(o_i, a_i)_{i=1}^{T}$. In partially observable settings, the policy typically takes the observation history $o_{1:t}$ as input. To enable goal-directed behavior, recent works [13, 23, 44] learn goal-conditioned policies $\pi(o_{1:t}, g)$, where g denotes a target goal. Such goal-conditioned policies are referred to as skills, as they encode reusable behaviors for achieving specific goals.

Hierarchical Agent Architecture. Directly learning policy $\pi(o_{1:t}, g)$ is challenging in complex open-world environments like Minecraft. Goals such as `build a house` or `mine diamond blocks` involve numerous intermediate steps, making it computationally and conceptually difficult to model an agent's behavior without breaking the goal into manageable parts. To tackle this, hierarchical agent architectures are often used [7, 12, 21, 39]. These architectures consist of a high-level planner and a low-level controller. The planner decomposes the overall goal into a series of sub-goals and decides which one the controller should address. The controller then focuses on achieving the current sub-goal by employing a specific skill. Specifically, for a certain t, the policy can be represented as:

$$\pi(a|o_{1:t}, g) = \pi_1(g_t|o_{1:t}, g)\pi_2(a|o_{1:t}, g_t) \tag{1}$$

where π_1 is the planner's policy and π_2 is the controller's policy. The planner periodically (not at every step) checks whether the current sub-goal has been finished and decides whether to change it, so g_t does not change frequently.

Identifying Implicit Skills in Trajectories. A critical challenge in training hierarchical agents, as described above, is the identification of skills within the trajectory data. To be more specific, given $\tau = (o_i, a_i)_{i=1}^{T}$, we want to split it into $\tau_1 = (o_i, a_i)_{i=1}^{t_1}, \tau_2 = (o_i, a_i)_{i=t_1}^{t_2}, \ldots, \tau_k = (o_i, a_i)_{i=t_{k-1}}^{T}$ where each sub-trajectories τ_j shows an independent skill $\pi(g_j)$. Since some controllers use self-supervised methods to learn sub-goals [6, 29], we don't have to identify g_j, but must identify t_j (*i.e.*, the moments at which the agent adopts a different skill). For clarity, in the subsequent sections, we denote the skill adopted at a certain time step t as π_t, corresponding to $\pi(g_t)$.

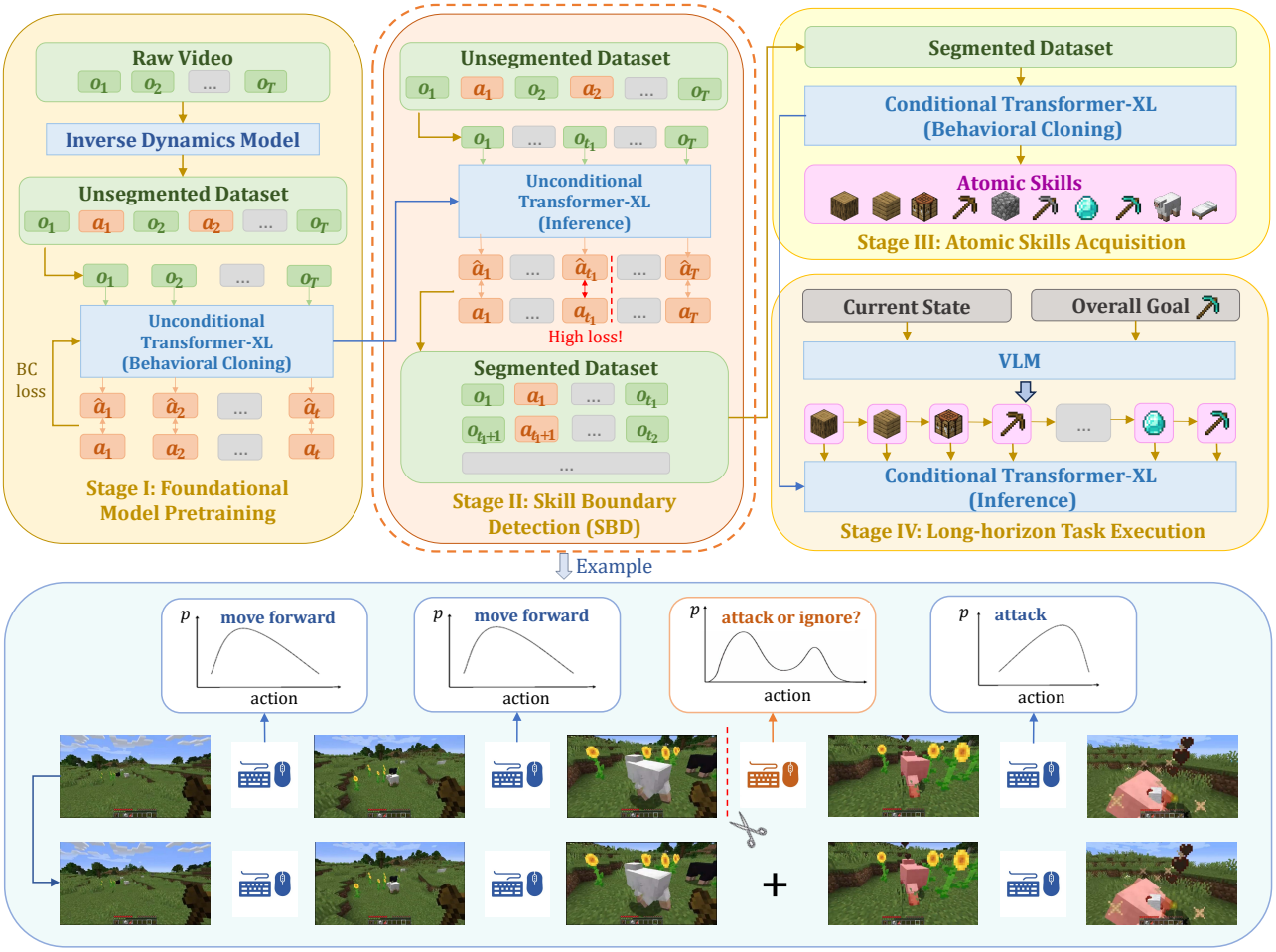


Figure 1. **Pipeline of our method SBD for discovering skills from unsegmented demonstration videos. Stage I:** An *unconditional* Transformer-XL based policy model [3, 11] is pretrained on an unsegmented dataset to predict future actions (labeled by an inverse dynamics model) based on past observations using behavioral cloning. **Stage II:** The pretrained unconditional policy will produce a high predicted action loss when encountering uncertain observations (e.g., deciding whether to kill a new sheep) in open worlds. These timesteps should be marked as skill boundaries, indicating the need for additional instructions to control behaviors. We segment the long unsegmented videos into a series of short atomic skill demonstrations. **Stage III:** We train a *conditional* Transformer-XL based policy model on the segmented dataset to master a variety of atomic skills. **Stage IV:** Finally, we use hierarchical methods (a combination of vision-language models and the conditional policy) to model the long demonstration videos and follow long-horizon instructions.

3. Method

We first introduce our overall pipeline to build policies and long-horizon hierarchical agents based on unsegmented videos in Section 3.1. Then we introduce how we split the unsegmented videos into skill sequences in Section 3.2. We finally introduce the implementation details in Section 3.3.

3.1. Pipeline

Given a long sequence of trajectories $D = (o_i, a_i)_{i=1}^{T}$, we first train an unconditional policy $\pi_{\text{unconditional}}$ using behavioral cloning to predict the actions purely based on past observations as follows:

$$\min_{\theta} \sum_{t \in [1 \ldots T]} -\log \pi_{\text{unconditional}}(a_t | o_{1:t}). \tag{2}$$

When training on action label-free videos, the action labels are generated from an Inverse Dynamics Model [3] . We then evaluate this unconditional policy $\pi_{\text{uncondtional}}$ on the entire dataset D of long sequences. Our proposed method SBD segments the complete video into a series of short segments $D_{seg} = (o_i, a_i)_{i=m}^{n}$ based on the evaluation results, where m and n is the selected segmentation timestamps. D_{seg} is then used for skill learning to obtain conditional policies [6, 29] that take additional videos or text as instructions. Finally,

Algorithm 1 Skill Boundary Detection

Input: $(o_i, a_i, e_i)_{i=1}^{T}$, a model M to predict a_t given $o_{1:t}$. e_i is the boolean external information indicating whether this step should be a boundary.
Initialize: begin $\leftarrow$ 1, loss_history $\leftarrow$ [], boundaries $\leftarrow$ []
for $t \leftarrow 1$ to T **do**
  loss $\leftarrow M(a_t \mid o_{\text{begin}:t})$
  loss_history.append(loss)
  if loss $-$ mean(loss_history) $>$ GAP **or** e_t is true **then**
    boundaries.append(t)
    begin $\leftarrow t$
    loss_history $\leftarrow$ []
  end if
end for
Return: boundaries

we integrate these skills with vision language models to build hierarchical agents, enabling them to complete long-horizon tasks [42, 43]. See the overall pipeline in Fig. 1.

3.2. Skill Boundary Detection (SBD)

SBD takes a long unsegmented trajectory and the unconditional action-prediction model as input. A sliding window is used to simulate the model's memory of past observations. At each time step t, we use the model to predict the next action and compare it with the ground truth to compute the loss. A skill change is considered likely if the loss exceeds the average loss by a hyperparameter, GAP, or if an external indicator is true. In such cases, t is marked as a boundary. The model's memory is then cleared, and the algorithm proceeds to analyze the next sub-trajectory.

We then explain the two core components of the algorithm: loss and external information.

3.2.1. Boundary with Entropy Loss

In this section, we explain the core idea of our method: why loss can indicate skill boundaries. Intuitively, without goal information, similar observations in an open world can inherently lead to different actions—even with perfect modeling. For example, upon seeing a sheep, both *ignore* and *attack* may be plausible depending on the (unknown) goal. Skill transitions are moments where such ambiguity is highest. Since continuing a skill is more likely than changing it, the policy learns to favor the former, making skill transitions statistically surprising. According to Bayes' rule, when the loss of the policy increases largely, it is likely that a skill transition has occurred.

To support this, we introduce three key assumptions about skills: skill consistency, skill confidence, and action deviance at skill transition.

The first assumption is that the skill being used does not change frequently (less than $1/K$). The idea is that the agent should consistently stick to a skill unless there is a strong reason to change to another, which would result in an increase in predictive loss.

Assumption 3.1 (Skill Consistency)**.** There exists an adequately large K, $\forall t$,

$$P(\pi_{t+1} \neq \pi_t | o_{1:t+1}) < 1/K \tag{3}$$

The second assumption states that, at any given time, the agent is confident in the action it takes. In other words, the probability of the agent choosing an action with confidence c is greater than $1 - \delta$. This assumption ensures that the policy reliably makes decisions based on its learned skills, rather than being uncertain in its actions. This helps the unconditional model make accurate predictions when the agent does not change its skill.

Assumption 3.2 (Skill Confidence)**.** There exists c and an adequately small $\delta, \forall t$,

$$P(\pi_t(a_t|o_{1:t}) > c) > 1 - \delta \tag{4}$$

The third assumption posits that when an agent changes its skill, it is likely to perform an action that is significantly less probable under the previous skill. This "surprising" action generates a clear signal, enabling us to detect the skill transition. Although the agent might change policies without performing a surprising action, such instances suggest that the agent is in a transitional, ambiguous phase between two policies, which is inherently challenging to identify. In essence, we aim to recognize the first clear skill boundary.

Assumption 3.3 (Action Deviance at Skill Transition)**.** There exists m, $\forall t$, when $\pi_1 = \pi_2 = ... = \pi_t \neq \pi_{t+1}$,

$$\frac{\pi_t(a_{t+1}|o_{1:t+1})}{(\prod_{i=1}^{t} \pi_t(a_i|o_{1:i}))^{1/t}} < \frac{1}{2}m \tag{5}$$

By the law of total probability, we have

$$\begin{aligned} P(a_{t+1}|o_{1:t+1}) &= P(\pi_{t+1} = \pi_t|o_{1:t+1})\pi_t(a_{t+1}|o_{1:t+1}) \\ &+ \sum_{\pi \neq \pi_t} P(\pi_{t+1} = \pi|o_{1:t+1})\pi(a_{t+1}|o_{1:t+1}) \end{aligned} \tag{6}$$

Intuitively, if the agent changes its skill, the predictive probability will be low. This is because, in Eq. (6), in the first term the probability of a_{t+1} under the original skill $\pi_t(a_{t+1}|o_{1:t+1})$ is low, and in the second term the probability of skill transition is low. On the contrary, if the agent does not change its skill, both terms will be high, so the predictive probability will be high. Therefore, we have the following theorem regarding the bounds of relative predictive probability under scenarios of skill transition and non-transition.

Theorem 3.4 (bounds of relative predictive probability)**.** *If* $\pi_1 = \pi_2 = ... = \pi_t = \pi_{t+1}$*, then we have*

$$P\left(\frac{P(a_{t+1}|o_{1:t+1})}{(\prod_{i=1}^{t} P(a_i|o_{1:i}))^{1/t}} > \frac{(K-1)c}{K}\right) > 1-\delta \quad (7)$$

If $\pi_1 = \pi_2 = ... = \pi_t \neq \pi_{t+1}$*, then we have*

$$P\left(\frac{P(a_{t+1}|o_{1:t+1})}{(\prod_{i=1}^{t} P(a_i|o_{1:i}))^{1/t}} < \frac{Km}{2(K-1)} + \frac{1}{c(K-1)}\right) > 1-t\delta \quad (8)$$

Proof. See Appendix B for the detailed proof. □

If $c > m$ and $(K-4)c^2 > 2$, then the lower bound of relative predictive probability under skill transition in Theorem 3.4 is greater than the upper bound of it under skill non-transition. Therefore, the theorem demonstrates that the model predicts actions less accurately when the agent changes its skill. Our action-prediction model uses the negative log-likelihood of actions $-\log P(a_t|o_{1:t})$ as loss. Consequently, at line 6 of Algorithm 1, when the loss exceeds the average loss by a predefined threshold, it is likely that a skill transition has occurred.

3.2.2. Boundary with External Information

In Sec. 3.2.1, we demonstrate that SBD can identify the first distinguishable skill boundary using only observations and actions. However, some datasets include additional external information such as privileged in-game data or VLM annotations [9, 17]. Incorporating these auxiliary signals improves the detection of skill boundaries that are otherwise difficult to recognize through loss-based detection alone. For example, in Minecraft, crafting an item does not involve abrupt changes in mouse movement, making it hard to detect boundaries purely with predictive action loss. In such cases, in-game logs of successful crafting events provide valuable hints. It is important to note that this is an optional component of SBD; as we will discuss in Sec. 4.3, it also performs well on datasets without external auxiliary information.

3.3. Implemetation Details

We take the Minecraft environment [19, 52] as the evaluation simulator. It offers a highly complex ($> 10^{10^{20}}$ states), diverse, and dynamic environment ideal for testing open-ended agents requiring *OOD generalization* [30]. See Appendix C for more information on this environment.

We now introduce the architectures and training/inference details of the policies and agents used in Sec. 3.1.

Unconditional Policy and Segmentation Details. For the unconditional policy $\pi_{\text{unconditional}}$ in Stage I, we use the pre-trained vpt-3x models [3]. This is a foundational Transformer-XL [11] based model pretrained on large-scale YouTube data and finetuned on the early-game dataset using behavioral cloning. The hyperparameter GAP in Stage II is set to 18, considering the average trajectory length and semantics. As for the external information, we use event-based information such as `mine_block:oak_log` or `use_item:torch` available in the contractor dataset [3]. To identify potential skill transitions, only the final event in a series of identical consecutive events is marked as positive, as it signifies a possible skill transition. For instance, if the agent chops a tree repeatedly and then crafts a table, the video should be split at the moment the agent mines the last block of wood. Further details are provided in Appendix D.3.

Policies for Atomic Skills. We employ both video-conditioned and language-conditioned policies to learn atomic skills from the segmented demonstration videos, as shown in Stage III. The video-conditioned policy is built upon GROOT [6], which utilizes self-supervised learning to model instructions and behaviors. In the original GROOT, a fixed-length sequence of 128 frames serves as the instruction, which is encoded into a goal embedding using a conditioned variational autoencoder (C-VAE) [24, 36]. The decoder then predicts actions based on the instruction and environmental observations in an auto-regressive manner. The language-conditioned policy is derived from STEVE-1 [29], an instruction-tuned VPT model capable of following open-ended texts or 16-frame visual instructions. This model is trained by first adapting the pretrained VPT model to follow commands in MineCLIP's [15] latent space, and then training a C-VAE model to predict latent codes from the text.

The original GROOT and STEVE-1 policies divide the training trajectories into segments of 128 frames and uniformly sample between 15 and 200 frames, respectively. We re-train these models using our SBD method on the segmented trajectories and compare the results to the original models to demonstrate the effectiveness of our approach. We retain the minimum and maximum length settings. The algorithm for pruning the length of each trajectory is detailed in D.4. Most of the original hyperparameter settings are retained as specified in the original papers. A complete list of modified hyperparameters is provided in Appendix D.2.

Long-term Instruction Following Agents. We use hierarchical agents to model long trajectories, which are widely used by [12, 41, 43]. We utilize in-context learning, as demonstrated by JARVIS-1 [42], and behavioral cloning, as shown by OmniJARVIS [43], to evaluate the long-horizon instruction-following capabilities of hierarchical agents. JARVIS-1 is a hierarchical agent that uses the text-conditioned STEVE-1 as its policy and pretrained vision-language models as planners. We replace the text-conditioned policy with our re-trained STEVE-1 based on SBD. OmniJARVIS is a vision-language action agent built on FSQ-GROOT, which encodes instructions into discrete tokens rather than continuous embeddings. It is trained on behavior trajectories encoded into unified token sequences.

Task		use furnace	hunt sheep	sleep in bed	use torch	use boat	use bow	collect stone	collect seagrass	collect wood	find and collect wood	collect dirt	collect grass	Average
VPT	*original*	11.0%	29.0%	0.0%	22.0%	7.0%	23.0%	77.0%	14.0%	73.0%	33.0%	30.0%	38.0%	-
GROOT (video conditioned)	*original*	21.0%	26.0%	100.0%	97.0%	71.0%	30.0%	21.7	34.0%	76.0%	19.0%	14.5	9.5	-
	ours	30.0%	54.0%	100.0%	88.0%	93.0%	80.0%	26.8	51.0%	90.0%	44.0%	19.7	25.4	-
	Δ	+42.9%	+107.7%	0	-9.3%	+30.1%	+166.7%	+23.3%	+50.0%	+18.4%	+131.6%	+36.1%	+166.3%	**+63.7%**
STEVE-1 (image & text conditioned	*original*	0.0%	1.0%	33.3%	33.3%	18.8%	65.6%	96.9%	18.8%	80.2%	57.3%	44.8%	46.9%	-
	ours	0.0%	3.1%	40.6%	77.1%	42.7%	71.9%	96.9%	47.9%	84.4%	67.7%	43.8%	71.9%	-
	Δ	0	–	+21.9%	+131.3%	+127.8%	+9.5%	0	+155.6%	+5.2%	+18.2%	-2.3%	+53.3%	**+52.1%**

Table 2. **Success rate of different policies on Minecraft skill benchmark.** For VPT [3], we report the results of the behavioral cloning version. For GROOT [6] and STEVE-1 [29], we report the results of the original and our re-trained models with SBD, respectively. A value with % indicates the average success rate, while a value without % indicates the average rewards. See more details in Appendix D.1.

Method	Wood	Food	Stone	Iron	Average
original	95%	44%	82%	32%	-
ours	96%	55%	90%	35%	-
Δ	+1.1%	+25.0%	+9.8%	+9.4%	**+11.3%**

(a) OmniJARVIS (behavioral cloning)

Method	Wood	Oak	Birch	Stone	Iron	Diamond	Armor	Food	Average
original	92%	89%	90%	90%	33%	8%	12%	39%	-
ours	97%	95%	94%	91%	35%	10%	19%	62%	-
Δ	+5.4%	+6.7%	+4.4%	+1.1%	+6.1%	+25.0%	+58.3%	+59.0%	**+20.8%**

(b) JARVIS-1 (in-context learning)

Table 3. **Success rate of two agents on long programmatic tasks.** The goal-conditioned policies of the agents are trained on the dataset segmented by SBD and the original random splitting method. In each group, the agent is required to obtain a certain type of item from scratch or be given an iron pickaxe. For example, the diamond group includes diamond pickaxe, diamond sword, jukebox, *etc*.

4. Experiments and Analysis

In our experiments, we answer the following questions:
• Are the short segments obtained through SBD more consistent semantically, and can a better goal-conditioned policy be trained on these segments?
• Do improvements in goal-conditioned policies enhance the ability of hierarchical agents?
• Which component of SBD is more important?

4.1. Experimental Setups

Training Dataset. We use OpenAI's contractor dataset 7.x (early game) [3] as our training dataset. It contains Minecraft offline trajectories with 68M frames with a duration of approximately 1000 hours, with at least half of the data from the first 30 minutes of the game. Our method generates a segmented dataset of 130k sub-trajectories using bc-early-game-3x [3] as the pretrained unconditional model.
Evaluation Benchmarks. For the goal-conditioned policies, we select basic skills such as `collect wood` and advanced skills like `use furnace` in Minecraft as the evaluation benchmark. We test 12 different atomic skills designed based on MCU [30]. All tasks are tested over 100 times, except for `sleep in bed` and `use bow`, which are evaluated 10 times using human rating because they cannot be automatically verified with in-game information. The evaluation metrics are success rates or rewards (for finer-grained assessment of easier tasks). See more details in Appendix D.1.

The hierarchical agents are evaluated on long-horizon programmatic tasks that require the agents to start from an empty inventory in a new world until obtaining the final required items, such as `obtain an iron pickaxe from scratch`, which is usually a chain of atomic tasks. We split the long-horizon tasks into different groups, including wooden items, food, stone items, iron *etc*. For each group, we choose the tasks from JARVIS-1 benchmarks [42] and evaluate each task over 30 times.

4.2. Main Results

In this section, we first present a human analysis of the quality of the SBD-segmented videos. Then we present the results of SBD applied to the two policies and two agents in Sec. 3.3 on the benchmarks in Sec. 4.1.
Human Analysis of Segmented Videos. We ask 10 people to analyze the SBD-segmented videos, each with 50 randomly sampled clips: On average, 54% exhibit clear independent semantics aligned with human preference, 34% are considered acceptable, and only 12% could be better segmented. This further supports our method's effectiveness.
Short-horizon Atomic Tasks. As shown in Tab. 2, the policies showed substantial improvements across most tasks, with an average relative performance enhancement of 63.7% for GROOT and 52.1% for STEVE-1.
Long-horizon Programmatic Tasks. As shown in Tab. 3, the agents have substantial improvements across most of the tasks, achieving an average relative performance enhancement of 11.3% for Omnijarvis and 20.8% for JARVIS-1.

Task	use furnace	hunt sheep	sleep in bed	use torch	use boat	use bow	collect stone	collect seagrass	collect wood	find and collect wood	collect dirt	collect grass	Average
-	21.0%	26.0%	100.0%	**97.0%**	71.0%	30.0%	21.7	34.0%	76.0%	19.0%	14.5	9.5	63.3
good dist.	38.0%	**65.0%**	90.0%	89.0%	80.0%	10.0%	24.8	43.0%	96.0%	41.0%	15.5	19.0	79.9
Info	33.0%	52.0%	100.0%	88.0%	**97.0%**	32.0%	23.7	**65.0%**	92.0%	47.0%	15.5	21.6	85.8
Loss (GAP=17.5)	**44.0%**	49.0%	100.0%	89.0%	91.0%	32.0%	**27.3**	**65.0%**	**96.0%**	38.0%	17.8	22.7	88.4
Loss (GAP=18)	**44.0%**	54.0%	100.0%	94.0%	72.0%	46.0%	25.8	63.0%	95.0%	**48.0%**	17.9	22.7	90.2
Loss (GAP=18.5)	41.0%	51.0%	100.0%	91.0%	73.0%	44.0%	26.8	57.0%	91.0%	45.0%	18.1	25.3	88.5
Both (GAP=18)	30.0%	54.0%	100.0%	88.0%	93.0%	**80.0%**	26.8	51.0%	90.0%	44.0%	**19.7**	**25.4**	**91.7**

Table 4. **Ablation studies.** We report the evaluation results of GROOT trained on datasets segmented by random splitting with fixed length (-), random splitting following the length distribution of SBD (*good* dist.), and SBD with different components and GAP values. A value with % indicates the average success rate, while a value without % indicates the average rewards. To enable a fair comparison between success rate and rewards, the average score is computed by scaling the best score to 100 and all other scores proportionally for each task.

4.3. Ablation Studies

SBD Components. As introduced in Secs. 3.2.1 and 3.2.2, the environment information and the skill boundary detection loss are important components for accurate segmentation of long videos. In this section, we explore the effectiveness of each component. We compare the full SBD with random splitting (w/o loss and info), SBD w/ info only, and SBD w/ loss only. The results are presented in Tab. 4. Our findings are as follows: (i) Using external information alone significantly improves performance, with a relative average increase of 35.5% over the original agent. (ii) Using loss alone results in a better performance (a relative average increase of 5.1%) compared to using external information alone, highlighting the effectiveness of the core component of our method. This demonstrates that our method achieves strong performance even without external information. (iii) Combining loss and external information yields the best overall performance. This suggests that each component captures unique segmentation patterns that the other misses.

We observe performance declines when employing both components in tasks `use furnace` and `collect seagrass`. This may be due to both components splitting the trajectory at the same intervals but not identical steps, leading to redundant trajectory fragments.

Hyperparameter sensitivity. GAP is an important hyperparameter of SBD. To verify the stability of our method, we also conduct an ablation study on different values of GAP. As shown in Tab. 4, our method is not sensitive to it (only 2.0% relative average performance decline).

Improved Baseline. A simple idea to improve random splitting is to follow a similar length distribution as SBD. We add this improved baseline to show that our method also catches features other than length distribution. As shown in Tab. 4, while a better length distribution can improve baseline performance (by 26.2% relatively), our method still outperforms it (by 14.8% relatively).

4.4. Visualizations

Length Distribution. Fig. 2 shows the length distribution of sub-trajectories split by different methods with or without the two components of SBD. We observe that both loss-only and info-only methods follow similar distributions, where the log length and frequency of sub-trajectories exhibit a normal distribution. Since the info-only method is intrinsically semantically meaningful, this provides indirect evidence that the loss-only method also identifies a semantically meaningful segmentation pattern as the info-only method. Furthermore, when combining loss with external information, the distribution does not significantly differ from the loss-only method. This observation highlights that the predictive loss is the key driver in learning the segmentation pattern, with the external information serving a more supplementary role.
Skill Videos. We sample one long unsegmented video segmented by our method for each skill in the datasets, such as `sleep in bed` and `collect seagrass` (see Fig. 3). More segments can be seen in Appendix E.

5. Related Works

Learning from Unsegmented Demonstrations. Prior work on learning from unsegmented demonstrations has proposed various methods to segment trajectories into sub-trajectories. For instance, some methods [25, 34, 38] utilize a variational autoencoder [24] model to generate pairs of skill types and boundaries, $(g_i, t_i)_{i=1}^{k}$, from a given trajectory. TACO [35] adopts a weakly supervised framework that identifies skill boundaries, $(t_1, \dots t_k)$, based on skill sketches, $(g_1, \dots g_k)$, by solving a sequence alignment problem. BUDS [55] constructs hierarchical task structures of demonstration sequences using a bottom-up strategy, deriving temporal segments through agglomerative clustering of the actions. FAST [31] proposes an approach based on the byte-pair encoding [16] algorithm for segmentation and tokenization of robot action trajectories.

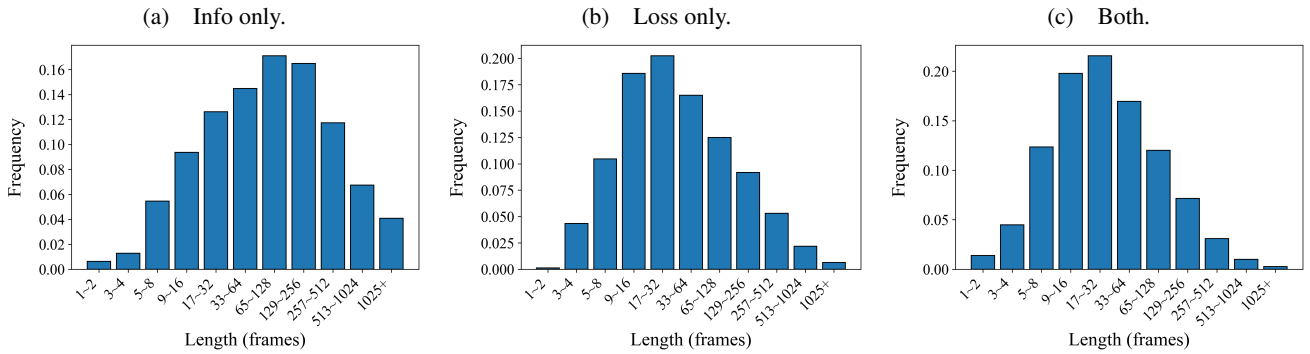


Figure 2. **The length distribution of segments, split by info and loss.** The info-only method is intrinsically semantically meaningful, suggesting that the loss-only method also identifies a semantically meaningful segmentation pattern. Furthermore, the similarity between the combined method and the loss-only method indicates that predictive loss is the primary factor in learning the segmentation pattern.

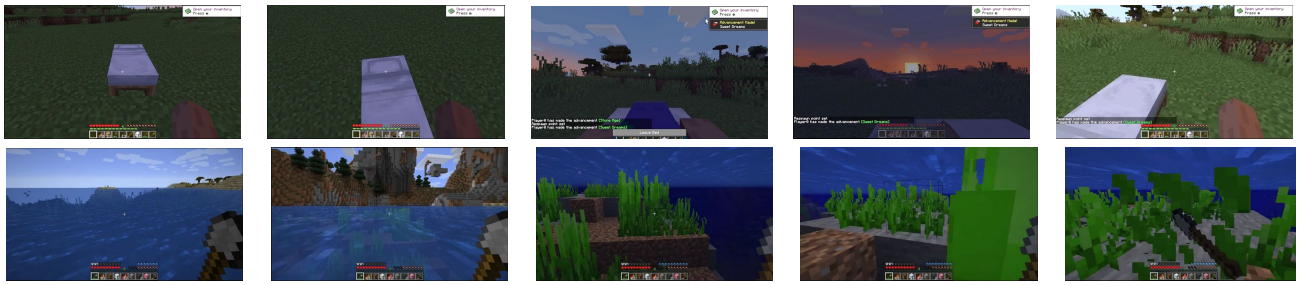


Figure 3. **Video Segment Examples. Top:** `sleep in bed`**. Bottom:** `collect grass`. Each segment is accompanied by five screenshots. The first and last screenshots represent the initial and final frames of the segment, respectively. The remaining three screenshots are manually selected to best illustrate the skill's progression. More segments can be found in Appendix E.

Option Learning. Option learning [37] is a framework to learn temporal abstractions (*i.e.*, options or skills) directly from the environment's reward signal, primarily through online reinforcement learning. Recent approaches [1, 2, 26] rely on policy gradient methods to learn the option policies. DIAYN [14] proposes a method to generate a reward function adaptively in the absence of reward signals, resulting in the unsupervised emergence of diverse skills.

Event Segmentation Theory. In neuroscience, episodic memory is the memory of everyday events consisting of short slices of experience [10]. Event Segmentation Theory (EST) [47] offers a theoretical perspective on how the neurocognitive system splits a long flow of memory into short events. According to EST, observers build event models of the current situation to generate predictions of future perceptual input. When errors in predictions arise, an event boundary is perceived, and the event model is reset and rebuilt. Our method employs a pretrained model to predict the agent's action based on past observations and uses the predictive loss as an indicator to detect skill boundaries.

Agents in Minecraft. Many agents have been developed to interact with Minecraft environments [22, 50]. For short-horizon tasks, methods typically employ behavioral cloning or reinforcement learning, as seen in works like VPT [3], which annotates a large Minecraft YouTube video dataset with actions and trains the first foundational agent in the domain, and its derivatives [5, 6, 27, 29, 46]. For long-horizon, programmatic tasks, large language models (LLMs) are used as planners combined with skill policies [28, 32, 33, 41, 42, 45, 53], and some methods [39, 42, 54] employ explicit memory mechanisms to enhance the long-horizon capabilities of LLM agents. Additionally, recent advances [43, 51] have explored using end-to-end vision-language models (VLMs) to directly follow human instructions. Finally, some research [18, 48] focuses on open-ended creative tasks such as building and decoration.

6. Conclusion

In this paper, we propose a novel temporal video segmentation algorithm to address the challenges of open-world skill discovery from unsegmented demonstration videos. To generate a segmented dataset of video clips with independent skills, our algorithm detects the potential skill boundaries based on the predictive loss of a pretrained action-prediction model. It does not require any manual annotations and can be employed directly on massive internet gameplay videos.

Acknowledgement

This work is funded in part by the National Science and Technology Major Project 2022ZD0114902. We thank a grant from CCF-Baidu Open Fund.

References

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 8

[2] Akhil Bagaria and George Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019. 8

[3] 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. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022. 2, 3, 5, 6, 8, 14

[4] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024. 2

[5] Shaofei Cai, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Open-world multi-task control through goal-aware representation learning and adaptive horizon prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13734–13744, 2023. 8

[6] Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot: Learning to follow instructions by watching gameplay videos. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 2, 3, 5, 6, 8, 14

[7] Shaofei Cai, Zihao Wang, Kewei Lian, Zhancun Mu, Xiaojian Ma, Anji Liu, and Yitao Liang. Rocket-1: Master open-world interaction with visual-temporal context prompting. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024. 2

[8] Yuheng Cheng, Ceyao Zhang, Zhengwen Zhang, Xiangrui Meng, Sirui Hong, Wenhao Li, Zihao Wang, Zekai Wang, Feng Yin, Junhua Zhao, et al. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint arXiv:2401.03428*, 2024. 1

[9] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. 5

[10] Martin A. Conway. Episodic memories. *Neuropsychologia*, 47(11):2305–2313, 2009. 8

[11] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. 3, 5

[12] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pages 8469–8488. PMLR, 2023. 2, 5

[13] Heming Du, Xin Yu, and Liang Zheng. Vtnet: Visual transformer network for object goal navigation. In *International Conference on Learning Representations*, 2021. 2

[14] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. 8

[15] 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. *Advances in Neural Information Processing Systems*, 35: 18343–18362, 2022. 2, 5

[16] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994. 2, 7

[17] Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, et al. Seed1.5-vl technical report. *arXiv preprint arXiv:2505.07062*, 2025. 5

[18] Yuxuan Guo, Shaohui Peng, Jiaming Guo, Di Huang, Xishan Zhang, Rui Zhang, Yifan Hao, Ling Li, Zikang Tian, Mingju Gao, Yutai Li, Yiming Gan, Shuai Liang, Zihao Zhang, Zidong Du, Qi Guo, Xing Hu, and Yunji Chen. Luban: Building open-ended creative agents via autonomous embodied verification. *arXiv preprint arXiv:2405.15414*, 2024. 8

[19] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela M. Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. In *International Joint Conference on Artificial Intelligence*, 2019. 2, 5, 13

[20] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. 1

[21] Shariq Iqbal, Robby Costales, and Fei Sha. Alma: Hierarchical learning for composite multi-agent tasks. *Advances in neural information processing systems*, 35:7155–7166, 2022. 2

[22] Haobin Jiang, Junpeng Yue, Hao Luo, Ziluo Ding, and Zongqing Lu. Reinforcement learning friendly vision-language model for minecraft. In *European Conference on Computer Vision*, pages 1–17. Springer, 2025. 8

[23] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14809–14818, 2021. 2

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014. 5, 7

[25] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. CompILE: Compositional imitation learning and execution. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019. 7

[26] Martin Klissarov and Doina Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34:4632–4646, 2021. 8

[27] 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. *arXiv preprint arXiv:2503.16365*, 2025. 8

[28] 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. *arXiv preprint arXiv:2408.03615*, 2024. 1, 8

[29] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36:69900–69929, 2023. 2, 3, 5, 6, 8, 14

[30] Haowei Lin, Zihao Wang, Jianzhu Ma, and Yitao Liang. Mcu: A task-centric framework for open-ended agent evaluation in minecraft. *arXiv preprint arXiv:2310.08367*, 2023. 2, 5, 6, 15

[31] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 2, 7

[32] Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. Mp5: A multi-modal open-ended embodied system in minecraft via active perception. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16307–16316. IEEE, 2024. 8

[33] ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025. 8

[34] Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020. 7

[35] Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663. PMLR, 2018. 2, 7

[36] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 5

[37] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999. 2, 8

[38] Daniel Tanneberg, Kai Ploeger, Elmar Rueckert, and Jan Peters. Skid raw: Skill discovery from raw trajectories. *IEEE robotics and automation letters*, 6(3):4696–4703, 2021. 7

[39] 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*, 2024. 1, 2, 8

[40] Letian Wang, Jie Liu, Hao Shao, Wenshuo Wang, Ruobing Chen, Yu Liu, and Steven L Waslander. Efficient reinforcement learning for autonomous driving with parameterized skills and priors. *Robotics: Science and Systems*, 2023. 1

[41] 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. *Advances in Neural Information Processing Systems*, 36, 2023. 1, 5, 8

[42] 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*, 2024. 4, 5, 6, 8, 14

[43] Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. *Advances in Neural Information Processing Systems*, 37: 73278–73308, 2025. 1, 4, 5, 8, 14

[44] Zhihui Xie, Zichuan Lin, Deheng Ye, Qiang Fu, Wei Yang, and Shuai Li. Future-conditioned unsupervised pretraining for decision transformer. In *International Conference on Machine Learning*, 2023. 2

[45] Haoqi Yuan, Chi Zhang, Hongchen Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *ArXiv*, abs/2303.16563, 2023. 8

[46] Haoqi Yuan, Zhancun Mu, Feiyang Xie, and Zongqing Lu. Pre-training goal-based models for sample-efficient reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. 8

[47] Jeffrey M Zacks, Nicole K Speer, Khena M Swallow, Todd S Braver, and Jeremy R Reynolds. Event perception: a mind-brain perspective. *Psychological bulletin*, 2007. 2, 8

[48] Chi Zhang, Penglin Cai, Yuhui Fu, Haoqi Yuan, and Zongqing Lu. Creative agents: Empowering agents with imagination for creative tasks. *arXiv preprint arXiv:2312.02519*, 2023. 8

[49] Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, et al. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17591–17599, 2024. 1

[50] Guangyu Zhao, Kewei Lian, Haowei Lin, Haobo Fu, Qiang Fu, Shaofei Cai, Zihao Wang, and Yitao Liang. Optimizing latent goal by learning from trajectory preference. *arXiv preprint arXiv:2412.02125*, 2024. 8

[51] Zhonghan Zhao, Ke Ma, Wenhao Chai, Xuan Wang, Kewei Chen, Dongxu Guo, Yanting Zhang, Hongwei Wang, and Gaoang Wang. Do we really need a complex agent system?

distill embodied agent into a single model. *arXiv preprint arXiv:2404.04619*, 2024. 8

[52] Xinyue Zheng, Haowei Lin, Kaichen He, Zihao Wang, Zilong Zheng, and Yitao Liang. Towards evaluating generalist agents: An automated benchmark in open world. *arXiv e-prints*, pages arXiv–2310, 2023. 5

[53] Enshen Zhou, Yiran Qin, Zhenfei Yin, Yuzhou Huang, Ruimao Zhang, Lu Sheng, Yu Qiao, and Jing Shao. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*, 2024. 8, 12

[54] Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyuan Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Y. Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *ArXiv*, abs/2305.17144, 2023. 8

[55] Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022. 2, 7

[56] 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 *Proceedings of The 7th Conference on Robot Learning*, pages 2165–2183. PMLR, 2023. 1

Open-World Skill Discovery from Unsegmented Demonstration Videos

Supplementary Material

A. Discussion and Future work

Computational Cost. Our method takes approximately 1 minute to process a 5-minute video on an NVIDIA RTX 3090Ti, due to the need to predict actions across the entire trajectory. Since data preprocessing is a one-time cost, spending only 1/20 of the video duration with 4 GPUs is acceptable. Also, it is more efficient than those methods relying on human or VLM annotations while ensuring quality.

Failure Case Due to Occasional Assumption Violation. Our method is occasionally unstable in action-intensive scenes like combat, where performance drops on task `combat spiders` (29% to 20%). See Appendix E for visualization of this failure case. This is likely due to a violation of Assumption 3.2 (Skill Confidence), as indicated by sharp prediction loss fluctuations. However, such issues are not observed elsewhere. Since overly short segments (< 5 frames) comprise less than 10% of all videos, overall effectiveness remains unaffected.

Choice of Loss Term. Currently our method detects changes in $P(a_t|o_{1:t})$. A possible alternative is to use $P(o_{t+1}|o_{1:t})$. We don't use it because it does not directly reflect skill changes as $P(a_t|o_{1:t})$ does, and current models [53] for predicting future observations suffer from hallucinations. We leave the attempt of this alternative to future work.

Scalability and Broader Applicability. Since our method is label-free, it has the potential to segment and train on the numerous gameplay trajectories on YouTube. We also anticipate the potential application of our method to other larger datasets and other open-world video games beyond Minecraft.

B. Proofs

In this section, we provide a detailed proof of Theorem 3.4, regarding the bounds of relative predictive probability under scenarios of skill transition and non-transition. We prove the lower bound and upper bound respectively in the following two subsections.

B.1. Proof of Upper Bound Under Skill Non-Transition

$$
\begin{aligned}
P\left(\frac{P(a_{t+1}|o_{1:t+1})}{(\prod_{i=1}^{t} P(a_i|o_{1:i}))^{1/t}} > \frac{(K-1)c}{K}\right) &> P\left(P(a_{t+1}|o_{1:t+1}) > \frac{(K-1)c}{K}\right) \\
&> P\left(P(\pi_{t+1}=\pi_t|o_{1:t+1})\pi_t(a_{t+1}|o_{1:t+1}) > \frac{(K-1)c}{K}\right) && \text{(Eq. (6))} \\
&> P(\pi_t(a_{t+1}|o_{1:t+1}) > c) && \text{(Assumption 3.1)} \\
&= P(\pi_{t+1}(a_{t+1}|o_{1:t+1}) > c) \\
&> 1-\delta && \text{(Assumption 3.2)}
\end{aligned}
$$

B.2. Proof of Lower Bound Under Skill Transition

We first magnify the likelihood ratio between the next action and the average action history,

$$\begin{aligned}
\frac{P(a_{t+1}|o_{1:t+1})}{(\prod_{i=1}^{t} P(a_i|o_{1:i}))^{1/t}} &< \frac{P(\pi_{t+1}=\pi_t|o_{1:t+1})\pi_t(a_{t+1}|o_{1:t+1}) + P(\pi_{t+1}\neq\pi|o_{1:t+1})}{(\prod_{i=1}^{t} P(a_i|o_{1:i}))^{1/t}} && \text{(Eq. (6))} \\
&< \frac{P(\pi_{t+1}=\pi_t|o_{1:t+1})\pi_t(a_{t+1}|o_{1:t+1}) + P(\pi_{t+1}\neq\pi|o_{1:i})}{\frac{K-1}{K}(\prod_{i=1}^{t} \pi_{i-1}(a_i|o_{1:i}))^{1/t}} && \text{(Eq. (6) and Assumption 3.1)} \\
&< \frac{\pi_t(a_{t+1}|o_{1:t+1}) + \frac{1}{K}}{\frac{K-1}{K}(\prod_{i=1}^{t} \pi_{i-1}(a_i|o_{1:i}))^{1/t}} && \text{(Assumption 3.1)} \\
&= \frac{K}{K-1}\frac{\pi_t(a_{t+1}|o_{1:t+1})}{\prod_{i=1}^{t} \pi_t(a_i|o_{1:i}))^{1/t}} + \frac{1}{(K-1)\prod_{i=1}^{t} \pi_i(a_i|o_{1:i}))^{1/t}} \\
&< \frac{Km}{2(K-1)} + \frac{1}{(K-1)\prod_{i=1}^{t} \pi_i(a_i|o_{1:i}))^{1/t}} && \text{(Assumption 3.3)}
\end{aligned}$$

Therefore,

$$\begin{aligned}
&P(\frac{P(a_{t+1}|o_{1:t+1})}{(\prod_{i=1}^{t} P(a_i|o_{1:i}))^{1/t}} < \frac{Km}{2(K-1)} + \frac{1}{c(K-1)}) \\
&> P(\frac{Km}{2(K-1)} + \frac{1}{(K-1)\prod_{i=1}^{t} \pi_i(a_i|o_{1:i}))^{1/t}} < \frac{Km}{2(K-1)} + \frac{1}{c(K-1)}) \\
&= P(\prod_{i=1}^{t} \pi_i(a_i|o_{1:i}))^{1/t} > c) \\
&> \prod_{i=1}^{t} P(\pi_i(a_i|o_{1:i}) > c) \\
&> (1-\delta)^t > 1 - t\delta && \text{(Assumption 3.2)}
\end{aligned}$$

C. Minecraft Environment

Minecraft is an extremely popular sandbox game that allows players to freely create and explore their world. This game has infinite freedom, allowing players to change the world and ecosystems through building, mining, planting, combating, and other methods. It is precisely because of this freedom that Minecraft becomes an excellent AI testing benchmark. In this game, AI agents need to face situations that are highly similar to the real world, making judgments and decisions to deal with various environments and problems. By using Minecraft, AI researchers can more conveniently simulate various complex and diverse environments and tasks, thereby improving the practical value and application of AI technology.

Our Minecraft environment is a hybrid between MineRL [19] and the MCP-Reborn (https://github.com/Hexeption/MCP-Reborn) Minecraft modding package. Unlike the regular Minecraft game, in which the server (or the "world") always runs at 20Hz while the client's rendering speed can typically reach 60-100Hz, the frame rate is fixed at 20 fps for the client in our experiments. The action and observation spaces in our environment are identical to what a human player can operate and observe on their device when playing the game. These details will be further explained in subsequent subsections.

C.1. Minecraft Game World Setting

We choose Minecraft version 1.16.5's survival mode as our experiment platform. In this mode, the agent may encounter situations that result in its death, such as being burned by lava or a campfire, getting killed by hostile mobs, or falling from great heights. When this happens, the agent will lose all its items and respawn at a random location near its initial spawn point within the same Minecraft world or at the last spot it attempted to sleep. Importantly, even after dying, the agent retains knowledge of its previous deaths and can adjust its actions accordingly since there is no masking of policy state upon respawn.

C.2. Observation Space

The environmental observation space consists of two parts. The first part is the raw pixels from the Minecraft game that players would see, including overlays such as the hotbar, health indicators, and animations of a moving hand in response to attack or

"use" actions. The rendering resolution of Minecraft is 640x360; however, in our experiments, we resize images to 128x128 for better computational efficiency while maintaining discernibility. The second part includes auxiliary information about the agent's current environment, such as its location and weather conditions. Human players can obtain this information by pressing F3. The specific observation details we include are shown in Tab. 5.

Sources	Shape	Description
pov (raw pixel)	(640, 360, 3) to (128,128,3)	Ego-centric RGB frames.
player_pos	(5,)	The coordinates of (x,y,z), pitch, and yaw of the agent.
location_stats	(9,)	The environmental information of the agent's current position, including `biome_id`, `sea_level`, `can_see_sky`, `is_raining` etc.
inventory	(36,)	The items in the current inventory of the agent, including the `type` and corresponding `quantity` of each item in each slot. If there is no item, it will be displayed as `air`.
equipped_items	(6,)	The current equipment of the agent, including `mainhand`, `offhand`, `chest`, `feet`, `head`, and `legs` slots. Each slot contains `type`, `damage`, and `max_damage` information.
event_info	(5,)	The events that occur in the current step of the game, including `pick_up` (picking up items), `break_item` (breaking items), `craft_item` (crafting items using a crafting table or crafting grid), `mine_block` (mining blocks by suitable tools), and `kill_entity` (killing game mobs).

Table 5. The observation space we use in Minecraft.

During the actual inference process, the controllers (VPT [3], GROOT [6], STEVE-1 [29]) only perceive the raw pixels. The agents (JARVIS-1 [42], OmniJarvis [43]) can access auxiliary information from the environment to generate the text condition of the controller.

C.3. Action Space

Our action space includes almost all actions directly available to human players, such as keypresses, mouse movements, and clicks. It consists of two parts: the mouse and the keyboard. When in-game GUIs are not open, the mouse movement is responsible for changing the player's camera perspective. When a GUI is open, it moves the cursor. The left and right buttons are responsible for attacking and using items. The keyboard is mainly responsible for controlling the agent's movement. We use the same joint hierarchical action space as VPT [3], which combines button space and camera space. Button space encodes all combinations of possible keyboard operations (excluding mutually exclusive actions such as `forward` and `back`) and a flag indicating whether the mouse is used, resulting in a total of 8461 candidate actions. The camera space discretizes the range of one mouse movement into 121 actions. Therefore, the action head of the agent is a multi-classification network with 8461 dimensions and a multi-classification network with 121 dimensions. We also filter out frames (both the observation and action) with null action as VPT [3].

In addition, we abstract the crafting and smelting actions with GUI into functional binary actions, which are the same as MineDojo (Fan et al., 2022). These advanced actions are only used by the agents (JARVIS-1 [42], OmniJarvis [43]). The detailed action space is described in Tab. 6.

Index	Action	Human Action	Description
1	Forward	key W	Move forward.
2	Back	key S	Move backward.
3	Left	key A	Strafe left.
4	Right	key D	Strafe right.
5	Jump	key Space	Jump. When swimming, keeps the player afloat.
6	Sneak	key left Shift	Slowly move in the current direction of movement.
7	Sprint	key left Ctrl	Move quickly in the direction of the current motion.
8	Attack	left Button	Destroy blocks (hold down); Attack entity (click once).
9	Use	right Button	Interact with the block that the player is currently looking at.
10	hotbar.[1-9]	keys 1 - 9	Selects the appropriate hotbar item.
11	Yaw	move Mouse X	Turning; aiming; camera movement.Ranging from -180 to +180.
12	Pitch	move Mouse Y	Turning; aiming; camera movement.Ranging from -180 to +180.
13	Equip	-	Equip the item in the main hand from the inventory.
14	Craft	-	Execute a crafting recipe to obtain a new item.
15	Smelt	-	Execute a smelting recipe to obtain a new item.

Table 6. The action space we use in Minecraft.

D. Experiment Details

In this section, we provide our experiment details, including benchmark, training details for GROOT and STEVE-1, how we use event-based information as external auxiliary information, and the length pruning algorithm for sub-trajectories.

D.1. Minecraft Skill Benchmark

MCU [30] is a diverse benchmark that can comprehensively evaluate the mastery of atomic skills by agents in Minecraft. Since our method is an improvement on dataset processing instead of a new model architecture, it cannot help agents learn new skills that they are completely incapable of acquiring previously. Therefore, we choose 10 early game skills from the original benchmark that the original agent can already grasp at a basic level. Besides, we add `use torch`, which is also a useful skill in Minecraft. We also add `find and collect wood`, an enhanced version of the skill `collect wood`, which requires the agent to start from the plains instead of the forest, aiming to test its ability to explore and find trees.

Details of the 12 skills in our early game benchmark are shown in Tab. 7. For each skill, we include the evaluation metric and a brief description of what the skill is. For skills "Sleep" and "Use bow", GROOT performs very well on the original metric, so we design a manual metric that better assesses its actual performance. STEVE-1 can take text as prompts, which are also listed in the table. For some of the skills, it is tricky to find proper text prompts, so we use visual prompts instead.

D.2. Training Details

The modified hyperparameters for GROOT and STEVE-1 are listed in Appendix D.2. We adjust parallel GPUs, gradient accumulation batches, and batch sizes to better align with our available computing resources. To speed up convergence without compromising performance, we double the learning rate for GROOT. The total number of frames for STEVE-1 is also modified, as we change the training dataset from a mixed subset of 8.x (house building from scratch), 9.x (house building from random materials), and 10.x (obtaining a diamond pickaxe) to the full 7.x dataset (early game), which better suits our benchmark tasks. [1]

All models are trained parallelly on four NVIDIA RTX 4090Ti GPUs. We follow the same policy training pipeline as the original paper, except for the modified hyperparameters mentioned above. GROOT is trained for three epochs of our dataset. STEVE-1, however, is only trained for 1.5 epochs of our dataset, because we observe that the model starts to overfit on the dataset if it is trained further.

D.3. Event-Based Information

Within the Minecraft environment, we focus on events in the categories "use item", "mine block", "craft item", and "kill entity", as they reflect the player's primary activities. Multiple events may occur simultaneously and are recorded as a set.

[1] OpenAI released five subsets of contractor data: 6.x, 7.x, 8.x, 9.x, and 10.x.

Skill	Metric	Description	Text Prompt for STEVE-1
Use furnace	Craft item cooked mutton	Given a furnace and some mutton and coal, craft a cooked mutton.	-
Hunt Sheep	Kill entity sheep	Summon some sheep before the agent, hunt the sheep.	-
Sleep in bed	**STEVE-1**: Use item white bed. **GROOT**: Sleep in bed properly.	Given a white bed, sleep on it.	Sleep in bed.
Use torch	Use item torch	Give some torches, use them to light up an area. Time is set at night.	Use a torch to light up an area.
Use boat	Use item birch boat	Given a birch boat, use it to travel on water. The biome is ocean.	-
Use bow	**STEVE-1**: Use item bow. **GROOT**: Use item bow 20%, Shoot in distance 40%, take aim 40%	Given a bow and some arrows, shoot the sheep summoned before the agent.	-
Collect stone	Mine block stone (cobblestone, iron, coal, diamond)	Given an iron pickaxe, collect stone starting from cave. Night vision is enabled.	Collect stone.
Collect seagrass	Mine block seagrass (tall seagrass, kelp)	Given an iron pickaxe, collect seagrass starting from ocean.	-
Collect wood	Mine block oak (spruce, birch, jungle, acacia) log	Given an iron pickaxe, collect wood starting from **forest**.	Chop a tree.
Find and collect wood	Mine block oak (spruce, birch, jungle, acacia) log	Given an iron pickaxe, **find** and collect wood starting from **plains**.	Chop a tree.
Collect dirt	Mine block dirt (grass block)	Given an iron pickaxe, collect dirt starting from plains.	Collect dirt.
Collect grass	Mine block grass (tall grass)	Given an iron pickaxe, collect grass starting from plains.	Collect grass.

Table 7. Details of 12 atomic skills in our Minecraft skill benchmark for testing GROOT and STEVE-1.

Hyperparameter	Value
Learning Rate	0.00004
Parallel GPUs	4
Accumulate Gradient Batches	1
Batch Size	8

Hyperparameter	Value
Parallel GPUs	4
Accumulate Gradient Batches	4
Batch Size	4
n_frames	100M

Table 8. Modified hyperparameters for training controllers. **(Left)** GROOT. **(Right)** STEVE-1.

Only the final step of a repeating sequence of sets of events is marked as positive. For example, in the sequence: ("use item iron pickaxe", "mine block iron ore"), ("use item iron pickaxe", "mine block iron ore"), ("use item iron pickaxe", "mine block diamond ore"), ("use item torch"), ("use item torch"), the second, third, fifth steps will be marked as positive. Additionally, for any step t that includes a "kill entity" event, we mark $t + 16$ as positive instead of t, because there will be a short death animation that follows the entity's death, and we want it to be included in the same segment.

D.4. Length Pruning Algorithm

STEVE-1 forces a minimum and maximum length of trajectories in its method, so we need a length pruning algorithm. In our implementation, the length of each trajectory is pruned as follows: If a trajectory is too short, it is merged with subsequent trajectories until it meets the minimum length requirement. If this results in a trajectory exceeding the maximum length, it

is truncated, and the remainder forms the beginning of the next trajectory. For instance, given a minimum and maximum length of 15 and 200, sub-trajectories of lengths 12, 12, 6, 196, and 37 are adjusted to 24, 200, and 39. Here, the first two sub-trajectories are merged, while the 6 is combined with 196 but truncated at 200, with the remaining 2 merged into the next trajectory. There might be more sophisticated strategies, but we use this straightforward algorithm for simplicity.

E. Examples of Skill Videos

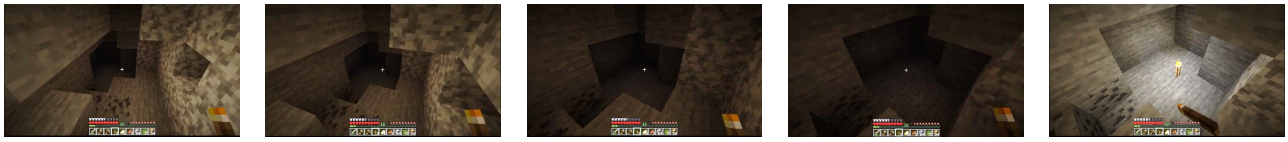
We sample one video segmented by our method for each skill in the benchmark and an extra video showing a failure case, presenting each video with five screenshots. The first and last screenshots correspond to the first and last frames of the video, while the other three are manually selected to best illustrate the progression of the skill.

- smelt food

- hunt sheep

- sleep

- use torch



- use boat

- use bow

- collect stone

- collect seagrass

- collect wood

- collect dirt

- collect grass

- combat spider (failure case: overly short)