

# Open-World Skill Discovery from Unsegmented Demonstrations

Jingwen Deng<sup>1†</sup>, Zihao Wang<sup>1†</sup>, Shaofei Cai<sup>1</sup>, Anji Liu<sup>2</sup> and Yitao Liang<sup>1✉</sup>

<sup>1</sup>Peking University, <sup>2</sup>University of California, Los Angeles, All authors are affiliated with Team CraftJarvis

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 sequence sampling 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. Our SBD-generated segments improved the average performance of conditioned policies by 63.7% and 52.1% on short-term atomic skill tasks, and their corresponding hierarchical agents by 11.3% and 20.8% on long-horizon tasks. Our method can leverage the diverse YouTube videos to train instruction-following agents. The project page can be found in <https://craftjarvis.github.io/SkillDiscovery/>.

## 1. Introduction

Most existing LLM-based instruction-following agents adopt a two-layer structure of planner and controller to complete tasks, which first convert the instruction into atom skills through the planner, and then use a conditioned policy to convert it into actions based on the current observation to interact with the environment (Cheng et al., 2024; Li et al., 2024; Wang et al., 2024a, 2023b, 2024b, 2025). When training these agents, we need to first convert long sequences into a series of short atom skill sequences, and then train the controller and planner separately. However, interactions in real-world videos are often long-term and unsegmented with detailed language skills.

Learning skills from long-sequence videos is critical for building such hierarchical agents. However, the concept of a “skill” is ill-defined and varies widely across domains such as video gaming (Cai et al., 2023b; Hafner et al., 2023; Wang et al., 2024a; Zhang et al., 2024), robotic control (Zitkovich et al., 2023), and autonomous driving (Wang et al., 2023a). This ambiguity,

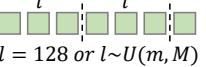
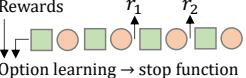
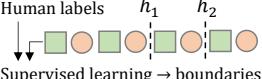
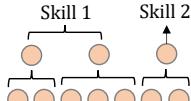
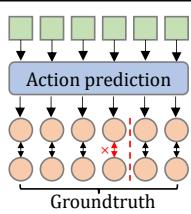
along with the immense diversity of skills in open worlds, makes skill learning particularly challenging, especially in partially observable settings. To enable open-world skill learning from unsegmented demonstrations (continuous streams of observation-action pairs without explicit labels), the first challenge is to segment these streams into semantically meaningful, self-contained skills.

We list the existing segmentation methods in Table 1, including the plain sequential sampling, reward-driven methods, top-down, and bottom-up methods. The native **sequential sampling** segmentation methods (Cai et al., 2023b; Lifshitz et al., 2023), divide videos into segments of predefined lengths (e.g., fixed length, uniform distribution). However, these methods do not ensure that each segment contains a distinct skill. Additionally, predefined lengths may not match the actual distribution of skill lengths in real-world scenarios (see Section 4.4). **Reward-driven** methods (Sutton et al., 1999) discover skills through the environment’s reward signal. It is limited by its inability to capture skills that lack associated rewards and by the risk of splitting a single skill into multi-

Corresponding author(s): Yitao Liang <yitaol@pku.edu.cn>

† indicates co-first author.

Jingwen Deng <dengjingwen@stu.pku.edu.cn>, Zihao Wang <zhwang@stu.pku.edu.cn>, Anji Liu <liuanji@cs.ucla.edu>

| Method              | Illustration  | Type           |
|---------------------|---|----------------|
| Sequential sampling |  | rule-based     |
| Reward-driven       |  | rule-based     |
| Top-down            |  | rule-based     |
| Bottom-up           |  | rule-based     |
| Ours (SBD)          |  | learning-based |

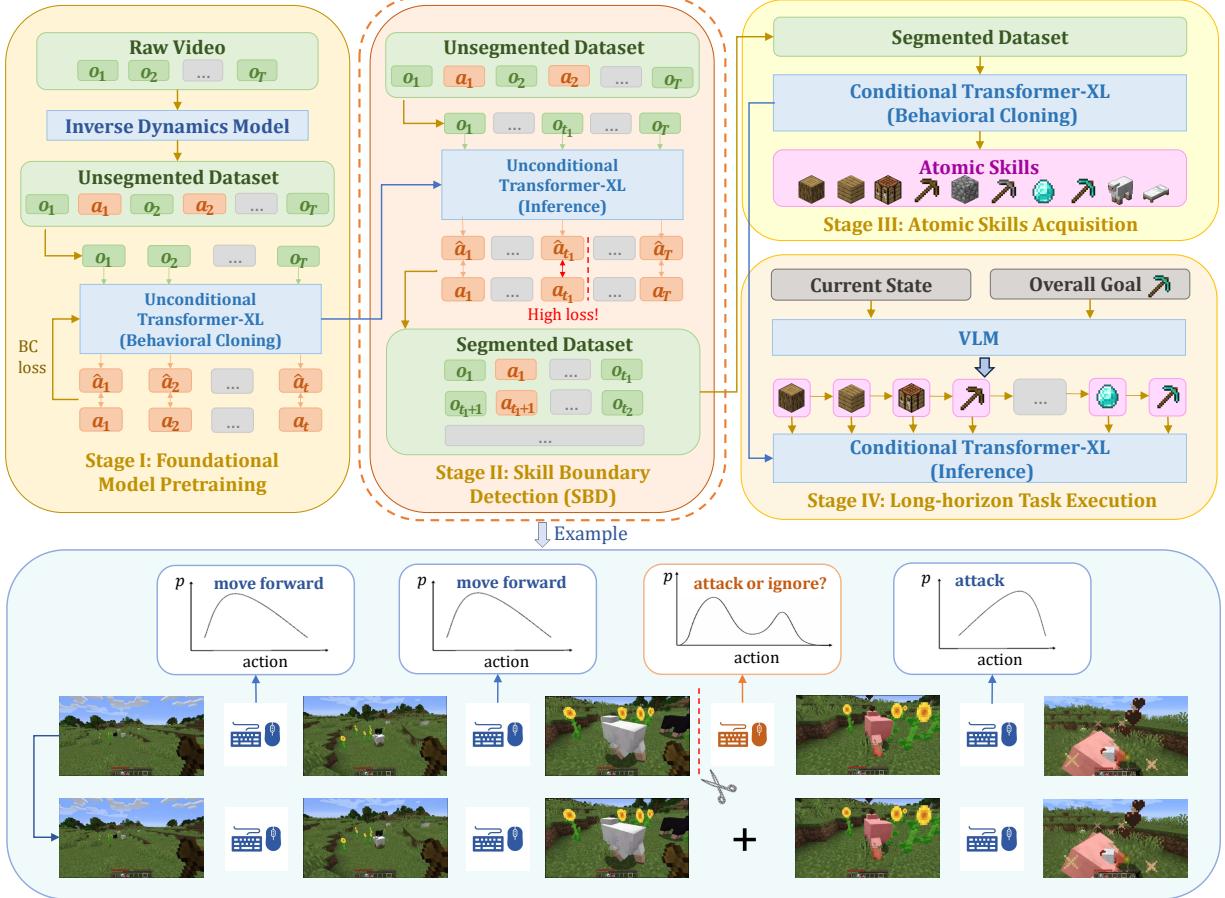
**Table 1 | Comparisons between existing segmentation methods and our method SBD.** Existing methods usually rely on human-designed rules, while our method is learning-based. **Sequential sampling** can result in a single skill spanning different segments or multiple skills located within one segment. **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 are limited in fully observable environments and hard in visually partially observable environments.  are visual observations and  are actions.

ple segments when rewards are repeatedly gained during execution. **Top-down** methods (Belkhale et al., 2024; Shiarlis et al., 2018) rely on predefined skill sets from human experts. They use manual labeling or supervised learning to segment videos. Although this approach can produce reasonable results, it is expensive and limited by the narrow range of predefined skills. **Bottom-up** methods (Pertsch et al., 2025; Zhu et al., 2022) use algorithms such as agglomerative clustering or byte-pair encoding (BPE) (Gage, 1994) to split action sequences. However, they struggle in partially observable settings where both observations and actions must be considered because these algorithms cannot deal with observations due to the

high dimensionality. All the above methods usually rely on human-designed rules to segment the unsegmented videos, which highlights the need for an effective and label-free method that can automatically segment skills from unsegmented demonstrations in open worlds.

Inspired by event segmentation theories (EST) (Zacks et al., 2007), which propose that humans naturally partition continuous experiences into discrete events when prediction errors in perceptual expectations rise, we introduce Skill Boundary Detection (SBD)—a method for autonomously identifying potential skill boundaries within extended trajectories. SBD employs a predictive model trained on a dataset of unsegmented videos to forecast future actions based on past observations, effectively capturing temporal dependencies (Baker et al., 2022). We then use this pretrained model to make predictions on an unsegmented video 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 in the video. 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 open-world Minecraft (Fan et al., 2022; Guss et al., 2019; Lin et al., 2023). First, we apply SBD to create a segmented Minecraft video dataset. We then train the video-conditioned policy (Cai et al., 2023b) and the language-conditioned policy (Lifshitz et al., 2023) on this dataset. We evaluate these policies on diverse Minecraft skills benchmark (Lin et al., 2023). The results show significant improvements compared to the existing policies (Cai et al., 2023b; Lifshitz et al., 2023), with performance increases of 63.7% and 52.1% compared to the original methods, respectively. We also test their corresponding hierarchical agents using behavior cloning and in-context learning, achieving performance increases of 11.3% and 20.8%, respectively. These findings underscore the effectiveness of our proposed method for skill discovery from unsegmented demonstrations and its potential to advance open-world skill learning.



**Figure 1 | Pipeline of our method SBD for discovering skills from unsegmented demonstration videos.** **Stage I:** An *unconditional* Transformer-XL based policy model (Baker et al., 2022; Dai et al., 2019) is pretrained on an unsegmented dataset to predict future actions (labeled by an inverse dynamics model) based on past observations. **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 and follow long-horizon instructions.

## 2. Problem Formulation

**Skills in behavioral cloning.** Behavioral cloning learns a policy  $\pi$  that maps observations  $o$  from the observation space  $O$  to actions  $a$  in the action space  $\mathcal{A}$  using supervised learning on observation-action pairs  $(o_i, a_i)_{i=1}^T$  obtained from expert demonstrations. In partially observable environments, past observations  $o_{1:t}$  are often used instead of the current observation  $o_t$  as input to the policy  $\pi$ . Additionally, to allow the agent to dynamically adjust its behavior based on its objective, recent advances (Du et al., 2021; Khandel-

wal et al., 2021; Xie et al., 2023) focus on learning goal-conditioned policies  $\pi(o_{1:t}, g)$ , where  $g$  represents the goal. This type of goal-conditioned policy can be considered a “skill”.

**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 (Cai et al., 2024a; Driess et al., 2023; Iqbal et al., 2022; Wang et al., 2024a). 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:

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

where  $\pi_1$  is the planner’s policy and  $\pi_2$  is the controller’s policy. The planner periodically (not at every step) checks whether the current sub-goal has been finished and determines whether to change policy, so  $g_t$  is relatively static and does not change frequently.

**Identifying implicit skills in offline trajectories.** A critical challenge in training hierarchical agents, as described above, is the identification of skills within the offline trajectory data. To be more specific, given  $\tau = (o_i, a_i)_{i=1}^T$ , we want to split it into  $\tau_1 = (o_i, a_i)_{i=1}^{t_1}, \tau_2 = (o_i, a_i)_{i=t_1}^{t_2}, \dots, \tau_k = (o_i, a_i)_{i=t_{k-1}}^T$  where each sub-trajectories  $\tau_j$  shows an independent skill  $\pi(g_j)$ . Since some controllers use self-supervised methods to learn sub-goals (Cai et al., 2023b; Lifshitz et al., 2023), we can ignore the language labels of goal  $g_j$ , 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  $\pi_t$ , corresponding to  $\pi(g_t)$ .

### 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 demonstration into our policy and agent learning sequences in Section 3.2. We finally introduce the implementation details on Section 3.3.

#### 3.1. Pipeline

Given a long sequence of trajectories  $D = (o_i, a_i)_{i=1}^T$  with interleaved visual observations  $o_i$

---

#### Algorithm 1 Skill Boundary Detection

---

```

1: 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.
2: Initialize:  $\text{begin} \leftarrow 1$ ,  $\text{loss\_history} \leftarrow []$ ,  $\text{boundaries} \leftarrow []$ 
3: for  $t \leftarrow 1$  to  $T$  do
4:    $\text{loss} \leftarrow M(a_t | o_{\text{begin}:t})$ 
5:    $\text{loss\_history.append}(\text{loss})$ 
6:   if  $\text{loss} - \text{mean}(\text{loss\_history}) > \text{GAP}$  or  $e_t$  is true then
7:      $\text{boundaries.append}(t)$ 
8:      $\text{begin} \leftarrow t$ 
9:      $\text{loss\_history} \leftarrow []$ 
10:   end if
11: end for
12: Return:  $\text{boundaries}$ 

```

---

and actions  $a_i$ , we first train an unconditioned policy  $\pi_{\text{uncondition}}$  using imitation learning to predict the actions as follows:

$$\min_{\theta} \sum_{t \in [1\dots T]} -\log \pi_{\text{uncondition}}(a_t|o_1, \dots, o_t). \quad (2)$$

When training on action label-free YouTube videos, the action labels are generated from Inverse Dynamics Model (Baker et al., 2022). We then evaluate this unconditioned policy  $\pi_{\text{uncondition}}$  on the entire dataset  $D$  of long sequences. We developed an automated method called SBD to segment the complete video into a series of short segments  $D_{\text{seg}} = (o_i, a_i)_{i=m}^n$  based on these evaluation results, where  $m$  and  $n$  is the selected segmentation timestamps. These short segments  $D_{\text{seg}}$  are then used for skill learning to obtain conditioned policies (Cai et al., 2023b; Lifshitz et al., 2023). Finally, we integrate these skills with vision language models to build hierarchical agents, enabling them to learn and follow long-horizon tasks in long video sequences (Wang et al., 2024b, 2025). We show the overall pipeline is illuminated in Figure 1.

#### 3.2. Skill Boundary Detection

SBD takes a long unsegmented trajectory as input and learns an unconditional model to pre-

dict the next action based on previous observations. The model does not know about implicit skill transitions. To simulate the model's memory, a sliding window is used. At each time step  $t$ , the model predicts the next action and compares it with the ground truth to compute the loss. A skill transition 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. The two core components of the algorithm are 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. 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. The parameter  $K$  is a large number that determines how rare these changes are. The idea is that the agent should consistently stick to a skill unless there is a strong reason to switch to another, which would result in an increase in predictive loss for detecting a skill transition.

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

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

The second assumption states that, at any given time, the agent is confident in the action it takes. This means the probability of the agent choosing an action with high confidence is very high. The parameter  $c$  sets the minimum confidence level, while  $\delta$  is a small value representing the frequency with which the policy might act with lower confidence. 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  $\delta$ ,  $\forall t$ ,

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

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  $\pi_1 = \pi_2 = \dots = \pi_t \neq \pi_{t+1}$ ,

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

By the law of total probability, we have

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

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

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

*Proof.* See Appendix A for the detailed proof.  $\square$

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 pre-defined threshold, it is likely that a skill transition has occurred.

### 3.2.2. Boundary with Auxiliary Information

In Section 3.2.1, we demonstrate that our algorithm 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 detections. Incorporating these auxiliary signals improves the detection of skill boundaries that are otherwise difficult to recognize through loss-based boundary detection alone. For example, in Minecraft, crafting an item does not involve abrupt changes in mouse movement, making it hard to detect boundaries purely from 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 our algorithm; as we will discuss in Section 4.3, it also performs well on datasets without external auxiliary information.

## 3.3. Implementation Details

**Unconditioned Policy and Segmentation Details.** For the unconditioned policy  $\pi_{\text{uncond}}$  in Stage I, we use the pre-trained vpt-3x models (Baker et al., 2022). This is a foundational

Transformer-XL (Dai et al., 2019) based model pretrained on large-scale YouTube data and fine-tuned 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 (Baker et al., 2022). 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 B.3.

**Policies for Atomic Skills.** We employ both language-conditioned and video-conditioned policies to learn atomic skills from the segmented demonstration videos as shown in Stage III. The video-conditioned policy is built upon GROOT (Cai et al., 2023b), 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) (Kingma and Welling, 2014; Sohn et al., 2015). 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 (Lifshitz et al., 2023), 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 (Fan et al., 2022) 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

| Method  |                 |        |         |        |         |         |         |        |         |        |         |        | Average        |
|---------|-----------------|--------|---------|--------|---------|---------|---------|--------|---------|--------|---------|--------|----------------|
| VPT     | <i>original</i> | 11.0%  | 29.0%   | 0.0%   | 22.0%   | 7.0%    | 23.0%   | 77.0%  | 14.0%   | 73.0%  | 33.0%   | 30.0%  | 38.0%          |
|         | <i>original</i> | 21.0%  | 26.0%   | 100.0% | 97.0%   | 71.0%   | 30.0%   | 21.7   | 34.0%   | 76.0%  | 19.0%   | 14.5   | 9.5            |
| GROOT   | <i>ours</i>     | 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 | <i>original</i> | 0.0%   | 1.0%    | 33.3%  | 33.3%   | 18.8%   | 65.6%   | 96.9%  | 18.8%   | 80.2%  | 57.3%   | 44.8%  | 46.9%          |
|         | <i>ours</i>     | 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 benchmarks. For VPT (Baker et al., 2022), we report the results of the behavioral cloning version. For GROOT (Cai et al., 2023b) and STEVE-1 (Lifshitz et al., 2023), we report the results of original and our re-trained with SBD, respectively. A value with % indicates the average success rate, while a value without % indicates the average rewards. The seeds for the Minecraft environment are fixed for the corresponding task to make a fair comparison between different models. Details of the tasks are provided in Appendix B.1.

the length of each trajectory is detailed in B.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 B.2.

**Long-term Instruction Following.** We use the hierarchical agents to model the long trajectories, which are widely used by (Driess et al., 2023; Wang et al., 2023b, 2025). We utilize in-context learning, as demonstrated by JARVIS-1 (Wang et al., 2024b), and imitation learning, as shown by OmniJARVIS (Wang et al., 2025), 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.

## 4. Experiments and Analysis

In our experiments, we answer the following questions:

- The short segments obtained through SBD method, are they more consistent at the semantic level, and can a better atomic skill policy be trained on these segments?
- Does using short videos segmented with new methods as skills perform better on long-horizon

tasks?

- Which component of SBD is the most important?

### 4.1. Experimental Setups

**Environmental Setups and Datasets.** We take the Minecraft environment (Guss et al., 2019; Zheng et al., 2023) as the evaluation simulator. Within the Minecraft environment, we use OpenAI’s contractor dataset 7.x (early game) (Baker et al., 2022) as trajectories. The dataset contains 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 (Baker et al., 2022) as the pretrained unconditional model.

**Evaluation Benchmarks.** For the atomic conditioned policies, we selected basic skills such as chop down trees and advanced skills like smelt items with furnace in Minecraft as evaluation benchmarks. We tested 12 different skill sets designed in MCU (Lin et al., 2023). Each task was tested over 100 times, except for sleep in bed and use bow, which were evaluated 10 times using the human Elo rating. The evaluation metrics are computed with the success rates and rewards. The hierarchical agents are evaluated on long-horizon programmatic tasks requiring 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

| Method          | Wood  | Food   | Stone | Iron  | Average | Method          | Wood  | Oak   | Birch | Stone | Iron  | Diamond | Armor  | Food   | Average |
|-----------------|-------|--------|-------|-------|---------|-----------------|-------|-------|-------|-------|-------|---------|--------|--------|---------|
| <i>original</i> | 95%   | 44%    | 82%   | 32%   | -       | <i>original</i> | 92%   | 89%   | 90%   | 90%   | 33%   | 8%      | 12%    | 39%    | -       |
| <b>ours</b>     | 96%   | 55%    | 90%   | 35%   | -       | <b>ours</b>     | 97%   | 95%   | 94%   | 91%   | 35%   | 10%     | 19%    | 62%    | -       |
| Δ               | +1.1% | +25.0% | +9.8% | +9.4% | +11.3%  | Δ               | +5.4% | +6.7% | +4.4% | +1.1% | +6.1% | +25.0%  | +58.3% | +59.0% | +20.8%  |

(a) OmniJARVIS (behavior cloning)

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

Table 3 | Success rate of two agents with their corresponding controllers trained on dataset segmented by our SBD method and the original sequential sampling (SS) method on groups of long programmatic tasks. All tasks are tested 30 times. In each group, the agent is required to obtain a certain type of items from scratch or given an iron pickaxe. For example, the diamond group includes diamond pickaxe, diamond sword, jukebox, etc.

| Task |              |              |        |              |              |              |             |              |              |              |             |             |             |  | Average |
|------|--------------|--------------|--------|--------------|--------------|--------------|-------------|--------------|--------------|--------------|-------------|-------------|-------------|--|---------|
| -    | 21.0%        | 26.0%        | 100.0% | <b>97.0%</b> | 71.0%        | 30.0%        | 21.7        | 34.0%        | 76.0%        | 19.0%        | 14.5        | 9.5         | 64.2        |  |         |
| Info | 33.0%        | 52.0%        | 100.0% | 88.0%        | <b>97.0%</b> | 32.0%        | 23.7        | <b>65.0%</b> | 92.0%        | 47.0%        | 15.5        | 21.6        | 87.4        |  |         |
| Loss | <b>44.0%</b> | <b>54.0%</b> | 100.0% | 94.0%        | 72.0%        | 46.0%        | 25.8        | 63.0%        | <b>95.0%</b> | <b>48.0%</b> | 17.9        | 22.7        | 91.8        |  |         |
| Both | 30.0%        | <b>54.0%</b> | 100.0% | 88.0%        | 93.0%        | <b>80.0%</b> | <b>26.8</b> | 51.0%        | 90.0%        | 44.0%        | <b>19.7</b> | <b>25.4</b> | <b>93.3</b> |  |         |

Table 4 | Ablation on the components within SBD. We report the evaluation results on Minecraft atomic skills from the sequential sampling (- in the table) and SBD with different components.

chain of atomic tasks. We split the long-horizon tasks into different groups, including wooden items, food, stones items, iron etc. For each group, we choose the tasks from JARVIS-1 benchmarks (Wang et al., 2024b) and evaluate each task over 10 times.

## 4.2. Main Results

In this section, we present the results of our method, SBD, applied to the two policies and two agents mentioned earlier. SBD achieves an average performance increase of **63.7%** for GROOT and **52.1%** for STEVE-1 compared to the original methods. Additionally, it enhances the performance of OmniJARVIS and JARVIS-1 by **11.3%** and **20.8%**.

**Short-horizon Atomic Tasks.** The results are shown in Table 2. Scores with % indicate success rates, while those without % represent rewards for the corresponding tasks. The controllers showed substantial improvements across most tasks, with an average performance enhancement of 63.7% for video-conditioned policy and 52.1% for language-conditioned policy.

**Long-horizon Programmatic Tasks.** To further verify that the improvements in controllers enhance the ability of hierarchical agents, we evaluate OmniJARVIS and JARVIS-1 on programmatic

tasks selected from the original papers (Wang et al., 2024b, 2025). As shown in Table 3, the agents have substantial improvements across most of the tasks, achieving an average performance enhancement of 11.3% for Omnipjarvis and 20.8% for JARVIS-1.

## 4.3. Ablation Study

As introduced in Sections 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 every component. So we compare the full SBD with sequential sampling (w/o loss and info), SBD w/ info only, and SBD w/ loss only. To evaluate the effectiveness of these configurations, we conducted an ablation study on the performance of four methods under GROOT. The results are presented in Table 4. Our findings are as follows: (i) Using external information alone significantly improves performance, with an average increase of 36.1% over the original agent. (ii) Using loss alone results in a better performance (an average increase of 5.0%) 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

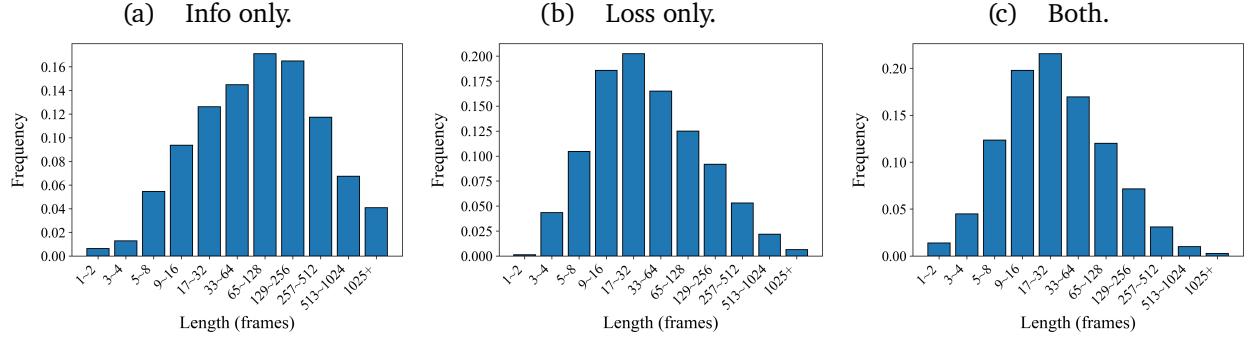


Figure 2 | The length distribution of segments, split by info and loss. The info-only method is intrinsically semantically meaningful, suggesting that the loss-only method also identifies a semantically meaningful segmentation pattern. Furthermore, the similarity between the combined method and the loss-only 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 C.

loss and external information yields the best overall performance. This combination outperforms the individual components in half of the tasks, suggesting that each component captures unique segmentation patterns that the other misses.

We observe performance declines when employ both components in tasks `smelt food` and `collect seagrass`. This may be due to both components partitioning the trajectory at the same intervals but not at identical steps, leading to redundant splits and incorrect trajectory fragments.

#### 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 ex-

hibit 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 C.

## 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 (Kipf et al., 2019; Shankar and Gupta, 2020; Tanneberg et al., 2021) utilize a variational autoencoder (Kingma and Welling, 2014) model to generate pairs of skill types and durations,  $(g_i, t_i)_{i=1}^k$ , from a given trajectory, thereby enabling segmentation. TACO (Shiarlis et al., 2018) adopts a weakly supervised framework that identifies task boundaries,  $(t_1, \dots, t_k)$  based on task sketches,  $(g_1, \dots, g_k)$ , by solving a sequence alignment problem to achieve appropriate segmentation. BUDS (Zhu et al., 2022) constructs hierarchical task structures of demonstration sequences using a bottom-up strategy, deriving temporal segments through agglomerative clustering of the actions. FAST (Pertsch et al., 2025) proposes an approach based on byte-pair encoding (Gage, 1994) algorithm for segmentation and tokenization of robot action trajectories via time-series compression.

**Option Learning.** Option learning (Sutton et al., 1999) is a framework designed to learn temporal abstractions (*i.e.*, skills or primitives) directly from the environment’s reward signal, primarily through online reinforcement learning. Recent approaches (Bacon et al., 2017; Bagaria and Konidaris, 2019; Klissarov and Precup, 2021) rely on policy gradient methods to learn the option policies. This framework requires reward signals, and DIAYN (Eysenbach et al., 2019) 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 (Conway, 2009). Event Segmentation Theory (EST) (Zacks et al., 2007) 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 pre-

dictions arise, an event boundary is perceived, causing the event model to be 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 (Jiang et al., 2025; Zhao et al., 2024a). For short-horizon tasks, methods typically employ imitation or reinforcement learning, as seen in works like VPT (Baker et al., 2022), which annotated a large YouTube Minecraft video dataset with actions and trained the first foundation agent in the domain using behavior cloning, and its derivatives (Cai et al., 2023a,b, 2024b; Lifshitz et al., 2023; Yuan et al., 2024). For long-horizon, programmatic tasks, large language models (LLMs) are used as planners combined with skill policies (Li et al., 2024; Qin et al., 2024; Wang et al., 2023b, 2024b; Yuan et al., 2023; Zhou et al., 2024), and some methods (Wang et al., 2024a,b; Zhu et al., 2023) employ explicit memory mechanisms to further enhance the long-horizon capabilities of LLM agents. Additionally, recent advances (Wang et al., 2025; Zhao et al., 2024b) have explored using end-to-end visual-language models (VLMs) to directly follow human instructions. Finally, some research (Guo et al., 2024; Zhang et al., 2023) focuses on open-ended creative tasks such as building and decoration, which often cannot be directly defined by rule-based rewards.

## 6. Conclusions

In this paper, we propose a novel temporal video segmentation algorithm to address the challenges of open-world skill discovery from unsegmented demonstrations. 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. External boundary indicators can also be incorporated in our method. The method does not require any manual annotations and can be employed directly on massive internet gameplay videos.

## 7. Limitations and Future work

The current algorithm is relatively time-consuming, requiring approximately 1 minute of processing on an NVIDIA RTX 3090Ti for every 5-minute video, due to the need to predict actions and compute loss across the entire trajectory. Since our method is label-free, it has the potential to segment and train on a large number of human trajectories on YouTube, which is future work. Additionally, we observe that the algorithm becomes unstable in action-intensive scenarios (e.g., combat situations), often generating many short segments. Future work could focus on improving the efficiency and stability of the algorithm, as well as exploring more effective evaluation methods. We also anticipate the potential application of our method to larger datasets and other open-world video games beyond Minecraft.

## 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

- P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- A. Bagaria and G. Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations*, 2019.
- B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.
- S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.
- S. Cai, Z. Wang, X. Ma, A. Liu, and Y. 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, 2023a.
- S. Cai, B. Zhang, Z. Wang, X. Ma, A. Liu, and Y. Liang. Groot: Learning to follow instructions by watching gameplay videos. In *The Twelfth International Conference on Learning Representations*, 2023b.
- S. Cai, Z. Wang, K. Lian, Z. Mu, X. Ma, A. Liu, and Y. Liang. Rocket-1: Master open-world interaction with visual-temporal context prompting. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024a.
- S. Cai, B. Zhang, Z. Wang, H. Lin, X. Ma, A. Liu, and Y. Liang. Groot-2: Weakly supervised multi-modal instruction following agents. *arXiv preprint arXiv:2412.10410*, 2024b.
- Y. Cheng, C. Zhang, Z. Zhang, X. Meng, S. Hong, W. Li, Z. Wang, Z. Wang, F. Yin, J. Zhao, et al. Exploring large language model based intelligent agents: Definitions, methods, and prospects. *arXiv preprint arXiv:2401.03428*, 2024.
- M. A. Conway. Episodic memories. *Neuropsychologia*, 47(11):2305–2313, 2009. doi: <https://doi.org/10.1016/j.neuropsychologia.2009.02.003>. URL <https://www.sciencedirect.com/science/article/pii/S0028393209000645>.
- Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Jan 2019. doi: 10.18653/v1/p19-1285. URL <http://dx.doi.org/10.18653/v1/p19-1285>.
- D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Gregor, A. Zeng, I. Mordatch, and P. Florence. PaLM-e: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/driess23a.html>.
- H. Du, X. Yu, and L. Zheng. Vtnet: Visual transformer network for object goal navigation. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=DILxQP0803B>.
- B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.
- L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

- P. Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, Feb. 1994. ISSN 0898-9788.
- Y. Guo, S. Peng, J. Guo, D. Huang, X. Zhang, R. Zhang, Y. Hao, L. Li, Z. Tian, M. Gao, Y. Li, Y. Gan, S. Liang, Z. Zhang, Z. Du, Q. Guo, X. Hu, and Y. Chen. Luban: Building open-ended creative agents via autonomous embodied verification. *arXiv preprint arXiv:2405.15414*, 2024.
- W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. M. Veloso, and R. Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations. In *International Joint Conference on Artificial Intelligence*, 2019. URL <https://api.semanticscholar.org/CorpusID:199000710>.
- D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
- S. Iqbal, R. Costales, and F. Sha. Alma: Hierarchical learning for composite multi-agent tasks. *Advances in neural information processing systems*, 35:7155–7166, 2022.
- H. Jiang, J. Yue, H. Luo, Z. Ding, and Z. Lu. Reinforcement learning friendly vision-language model for minecraft. In *European Conference on Computer Vision*, pages 1–17. Springer, 2025.
- A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: Clip embeddings for embodied ai. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14809–14818, 2021. URL <https://api.semanticscholar.org/CorpusID:244346010>.
- D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.
- T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. ComPILE: Compositional imitation learning and execution. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3418–3428. PMLR, 2019. URL <https://proceedings.mlr.press/v97/kipf19a.html>.
- M. Klissarov and D. Precup. Flexible option learning. *Advances in Neural Information Processing Systems*, 34:4632–4646, 2021.
- Z. Li, Y. Xie, R. Shao, G. Chen, D. Jiang, and L. Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. *arXiv preprint arXiv:2408.03615*, 2024.
- S. Lifshitz, K. Paster, H. Chan, J. Ba, and S. McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36:69900–69929, 2023.
- H. Lin, Z. Wang, J. Ma, and Y. Liang. Mcu: A task-centric framework for open-ended agent evaluation in minecraft. *arXiv preprint arXiv:2310.08367*, 2023.
- K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.
- Y. Qin, E. Zhou, Q. Liu, Z. Yin, L. Sheng, R. Zhang, Y. Qiao, and J. 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.
- T. Shankar and A. Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020.
- K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, pages 4654–4663. PMLR, 2018.
- K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.
- R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

- D. Tanneberg, K. Ploeger, E. Rueckert, and J. Peters. Skid raw: Skill discovery from raw trajectories. *IEEE robotics and automation letters*, 6(3):4696–4703, 2021.
- G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. 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>.
- L. Wang, J. Liu, H. Shao, W. Wang, R. Chen, Y. Liu, and S. L. Waslander. Efficient reinforcement learning for autonomous driving with parameterized skills and priors. *Robotics: Science and Systems*, 2023a.
- Z. Wang, S. Cai, G. Chen, A. Liu, X. S. Ma, and Y. Liang. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36, 2023b.
- Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang, , X. Ma, and Y. Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024b.
- Z. Wang, S. Cai, Z. Mu, H. Lin, C. Zhang, X. Liu, Q. Li, A. Liu, X. S. Ma, and Y. Liang. Omniparser: Unified vision-language-action tokenization enables open-world instruction following agents. *Advances in Neural Information Processing Systems*, 37:73278–73308, 2025.
- Z. Xie, Z. Lin, D. Ye, Q. Fu, W. Yang, and S. Li. Future-conditioned unsupervised pretraining for decision transformer. In *International Conference on Machine Learning*, 2023. URL <https://api.semanticscholar.org/CorpusID:258947476>.
- H. Yuan, C. Zhang, H. Wang, F. Xie, P. Cai, H. Dong, and Z. Lu. Plan4mc: Skill reinforcement learning and planning for open-world minecraft tasks. *ArXiv*, abs/2303.16563, 2023. URL <https://api.semanticscholar.org/CorpusID:257805102>.
- H. Yuan, Z. Mu, F. Xie, and Z. Lu. Pre-training goal-based models for sample-efficient reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.
- J. M. Zacks, N. K. Speer, K. M. Swallow, T. S. Braver, and J. R. Reynolds. Event perception: a mind-brain perspective. *Psychological bulletin*, 2007.
- C. Zhang, P. Cai, Y. Fu, H. Yuan, and Z. Lu. Creative agents: Empowering agents with imagination for creative tasks. *arXiv preprint arXiv:2312.02519*, 2023.
- C. Zhang, K. Yang, S. Hu, Z. Wang, G. Li, Y. Sun, C. Zhang, Z. Zhang, A. Liu, S.-C. Zhu, et al. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17591–17599, 2024.
- G. Zhao, K. Lian, H. Lin, H. Fu, Q. Fu, S. Cai, Z. Wang, and Y. Liang. Optimizing latent goal by learning from trajectory preference. *arXiv preprint arXiv:2412.02125*, 2024a.
- Z. Zhao, K. Ma, W. Chai, X. Wang, K. Chen, D. Guo, Y. Zhang, H. Wang, and G. Wang. Do we really need a complex agent system? distill embodied agent into a single model. *arXiv preprint arXiv:2404.04619*, 2024b.
- X. Zheng, H. Lin, K. He, Z. Wang, Z. Zheng, and Y. Liang. Towards evaluating generalist agents: An automated benchmark in open world. *arXiv e-prints*, pages arXiv–2310, 2023.
- E. Zhou, Y. Qin, Z. Yin, Y. Huang, R. Zhang, L. Sheng, Y. Qiao, and J. Shao. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*, 2024.
- X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, Y. Qiao, Z. Zhang, and J. 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. URL  
[https://api.semanticscholar.org/  
CorpusID:258959262](https://api.semanticscholar.org/CorpusID:258959262).
- Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

- B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 06–09 Nov 2023. URL [https://proceedings.mlr.  
press/v229/zitkovich23a.html](https://proceedings.mlr.press/v229/zitkovich23a.html).

## A. 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.

### A.1. Proof of Upper Bound Under Skill Non-Transition

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

### A.2. Proof of Lower Bound Under Skill Transition

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

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

Therefore,

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

## B. 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.

### B.1. Tasks in the Early Game Benchmark

Minecraft SkillForge is a diverse benchmark of 30 short atomic tasks that can comprehensively evaluate the mastery of atomic skills by agents in Minecraft. We choose 10 early game tasks from the original benchmark which the original agent can already handle at a basic level because our method is an improvement on dataset processing instead of a new model architecture, so it cannot help agents learn new skills that they are completely incapable of acquiring previously. Besides, we add the task "use torch", which is also a useful skill in Minecraft; and we add an enhanced version of the task "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 tasks in our early game benchmark are shown in Table 5. For each task, we include the evaluation metric and a brief description of what the task is. For tasks "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 tasks, it is tricky to find proper text prompts, so we use visual prompts instead.

### B.2. Training Details

The modified hyperparameters for GROOT and STEVE-1 are listed in Appendix B.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.<sup>1</sup>

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 under three epochs of our dataset. STEVE-1, however, is only trained under 1.5 epochs of our dataset, because we observe that the model starts to overfit on the dataset if it is trained further.

### B.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. Only the final step of a repeating sequence of event sets 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  $t$ ,

<sup>1</sup>OpenAI released five subsets of contractor data: 6.x, 7.x, 8.x, 9.x, and 10.x.

| Task | Metric   | Description   | Text Prompt for STEVE-1          |
|------|--|---|----------------------------------|
|      | Craft item cooked mutton   | Given a furnace and some mutton and coal, craft a cooked mutton.                  | -                                |
|      | Kill entity sheep  | Summon some sheep before the agent, hunt the sheep.                               | -                                |
|      | STEVE-1: Use item white bed.<br>GROOT: Sleep in bed properly.                          | Given a white bed, sleep on it.   | Sleep in bed.                    |
|      | 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 item birch boat  | Given a birch boat, use it to travel on water. Biome is ocean.                    | -                                |
|      | STEVE-1: Use item bow.<br>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.           | -                                |
|      | Mine block stone (cobblestone, iron, coal, diamond)                                    | Given an iron pickaxe, collect stone starting from cave. Night vision is enabled. | Collect stone.                   |
|      | Mine block seagrass (tall seagrass, kelp)  | Given an iron pickaxe, collect seagrass starting from ocean.                      | -                                |
|      | Mine block oak (spruce, birch, jungle, acacia) log                                     | Given an iron pickaxe, collect wood starting from forest.                         | Chop a tree.                     |
|      | Mine block oak (spruce, birch, jungle, acacia) log                                     | Given an iron pickaxe, find and collect wood starting from plains.                | Chop a tree.                     |
|      | Mine block dirt (grass block)  | Given an iron pickaxe, collect dirt starting from plains.                         | Collect dirt.                    |
|      | Mine block grass (tall grass)  | Given an iron pickaxe, collect grass starting from plains.                        | Collect grass.                   |

Table 5 | Details of 12 atomic tasks in our early game benchmark for testing GROOT and STEVE-1.

| Hyperparameter              | Value   | Hyperparameter              | Value |
|-----------------------------|---------|-----------------------------|-------|
| Learning Rate               | 0.00004 | Parallel GPUs               | 4     |
| Parallel GPUs               | 4       | Accumulate Gradient Batches | 4     |
| Accumulate Gradient Batches | 1       | Batch Size                  | 4     |
| Batch Size                  | 8       | n_frames                    | 100M  |

Table 6 | Modified hyperparameters for training controllers. (Left) GROOT. (Right) STEVE-1.

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.

#### B.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.

### C. Examples of Skill Videos

We sample one video segmented by our method for each skill in the benchmark, 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

