SAGE: SEMANTIC ACTION GROUNDING FOR VLM-GUIDED OFFLINE LONG-HORIZON CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline trajectories contain reusable behaviors for long-horizon control but lack semantic skill annotations, while off-the-shelf vision-language models (VLMs) lack grounding in environment-specific control. Existing methods often assign language to pre-discovered latent skills, which may misalign high-level planning concepts with executable behaviors. In contrast, we propose Semantic Action Grounding and Execution (SAGE), a semantic-first framework that discovers semantic skills before learning their low-level realization, thereby converting reward-free offline trajectories into a planner-readable and controller-executable interface without fine-tuning the foundation model. SAGE extracts candidate skills from sliding-window trajectory segments, consolidates them through semantic merging, relevance filtering, and data-support pruning, and grounds the resulting skill library with a skill-conditioned policy trained by transition-level imitation. A memory-enhanced VLM planner then tracks task progress and selects skills in a closed loop. Across 10 long-horizon tasks, SAGE matches or outperforms the representative offline learning and hierarchical VLM-enhanced baselines on **9** tasks, with relative gains of up to **53%**.

1 INTRODUCTION

Long-horizon decision making requires agents to complete a task through many meaningful intermediate behaviors. Such tasks typically involve sparse rewards and a vast exploration space, rendering online reinforcement learning (RL) sample-inefficient and often impractical in costly or risky settings, which motivates learning from fixed offline datasets (Wang et al., 2024). Yet imitation learning and offline RL, though free of online interaction (Fujimoto et al., 2019; Levine et al., 2020), remain vulnerable to distribution shift and compounding errors over long horizons (Park et al., 2026). Hierarchical learning reduces the effective planning horizon by composing temporally extended skills (Eysenbach et al., 2019; Park et al., 2024a;b). However, most offline skill-discovery methods learn latent skills without interpretable semantics, so adapting them to new tasks still requires task-specific rewards, value functions, or additional alignment.

Large language models (LLMs) (Zhao et al., 2023) and vision-language models (VLMs) (Zhang et al., 2024) offer a promising alternative for high-level planning because they can decompose natural-language instructions into semantic subgoals. Yet many off-the-shelf foundation models are not grounded in environment-specific dynamics, action spaces, or executable behaviors, making direct control unreliable (Yin et al., 2025). Post-training LLMs (Pang et al., 2024) or training vision-language-action models on large-scale robotic data (Ma et al., 2026; Zitkovich et al., 2023) can improve grounding, but require substantial task-specific data and computation.

A more practical approach is to expose reusable low-level policies through semantic skill descriptions that a foundation model can select (Brohan et al., 2023). However, offline datasets rarely contain such annotations, and manual labeling is costly and environment-specific. Existing approaches therefore often discover latent skill codes first and assign language descriptions afterward (Fu et al., 2024; Liu et al., 2025; Zhang et al., 2025). This latent-first pipeline can produce an inconsistent language-control interface: semantically similar behaviors may be split across skills, while behaviorally distinct segments may share one code. Consequently, these methods often require online alignment (Fu et al., 2024; Zhang et al., 2025) or reward-based high-level learning (Liu et al., 2025), limiting zero-shot use in reward-free offline settings. We therefore ask:

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

How can reward-free offline trajectories be converted into a planner-readable and controller-executable skill interface for zero-shot VLM planning?

This is nontrivial because trajectory segments lack reliable semantic boundaries, VLM-generated descriptions can be noisy or weakly supported by data, and the resulting interface must support both smooth low-level execution and progress-aware high-level planning.

To address these challenges, we propose **S**emantic **A**ction **G**rounding and **E**xecution (SAGE), a semantic-first framework that converts reward-free offline trajectories into a skill interface that is both interpretable to a VLM planner and executable by a low-level controller. Concretely, SAGE applies an off-the-shelf VLM to sliding-window trajectory segments to identify candidate skills and generate their semantic descriptions from unannotated offline datasets. It then refines the resulting skill library through semantic merging, domain-relevance filtering, and data-support pruning. Next, each retained skill is grounded in environment-specific control through a skill-conditioned policy trained with two-stage transition-level imitation. During deployment, a memory-enhanced VLM planner tracks task progress and selects skills in a closed loop. Unlike latent-first methods that assign language descriptions to pre-discovered skill codes, SAGE constructs its skill library directly around semantic descriptions, thereby reducing the mismatch between the concepts used for high-level planning and the behaviors learned for low-level execution.

We evaluate SAGE on 10 long-horizon tasks across diverse domains. SAGE matches or outperforms representative offline learning and hierarchical VLM-enhanced baselines on **9** tasks, with relative improvements of up to **53%**. The results show that its semantic-first interface supports both effective long-horizon execution and adaptation to different downstream tasks. Ablations further validate the contributions of semantic skill construction and memory-enhanced planning, while experiments across foundation-model families demonstrate the scalability and robustness of SAGE.

The contributions of this paper are threefold:

- We study an important yet underexplored problem: constructing a language-readable and controller-executable skill interface from reward-free offline trajectories for zero-shot VLM-based long-horizon decision making.
- We propose SAGE, a semantic-first pipeline that annotates local trajectory segments with off-the-shelf VLMs, filters noisy candidates, grounds retained skills through transition-level imitation, and performs memory-enhanced closed-loop planning.
- Experiments on diverse long-horizon tasks show that SAGE matches or outperforms representative baselines on 9 out of 10 tasks, with relative improvements of up to 53%.

2 Related Work

Offline skill learning and hierarchical control. Hierarchical control based on skill discovery has been widely used for long-horizon decision-making through temporal abstraction (Eysenbach et al., 2019; Pertsch et al., 2021; Pateria et al., 2021). Unlike online settings, where skills can be acquired by encouraging exploration (Eysenbach et al., 2019; Park et al., 2024b; Rho et al., 2025), offline skill discovery typically extracts certain latent representations from offline trajectory datasets through unsupervised learning (Park et al., 2024a; Zhang et al., 2025; Fu et al., 2024; Liu et al., 2025). The skill spaces learned by these methods are often latent variables that are difficult for humans to interpret. Consequently, when solving downstream tasks, they usually require explicit reward signals to learn the high-level policy through online or offline RL, or to align skills via retrieval. When explicit rewards are unavailable for downstream tasks, these methods struggle to operate in a zero-shot manner. In contrast, our method constructs an explicit natural-language interface for skills, thereby leveraging the language understanding and reasoning capabilities of VLMs to enable effective zero-shot decision-making in reward-free settings.

Large models for embodied planning. Large models represented by LLMs and VLMs provide semantic priors for perception, instruction following, and planning. Prior work uses language models to generate subgoal plans (Huang et al., 2022; Song et al., 2023), grounds language in affordances or visual observations (Brohan et al., 2023; Mees et al., 2023; Myers et al., 2023), and leverages LLMs and VLMs to guide skill discovery (Rho et al., 2025; Liu et al., 2025). These methods show that, given reliable low-level policies, large models can guide decision-making processes. Unlike them,

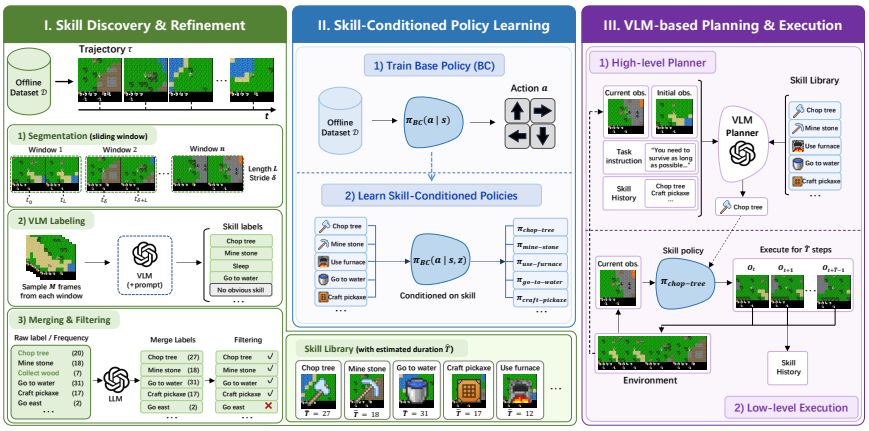


Figure 1: SAGE framework. SAGE first labels semantic skills from offline trajectories with a VLM, and semantically merges and filters the labels to obtain a compact skill list. Then, it learns a low-level skill policy library through two-stage imitation learning. Finally, a memory-feedback mechanism enables the high-level VLM planner to achieve effective zero-shot closed-loop skill planning.

SAGE explores how to directly use VLMs to construct a compact skill library with natural-language interfaces from offline trajectories, and further uses a high-level VLM planner to achieve zero-shot decision making.

3 Preliminaries

3.1 Reward-free Offline Markov Process

We consider a controlled Markov process (Park et al., 2025), $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $P(s' \mid s, a)$ is the transition dynamics, ρ_0 is the initial state distribution, and γ is the discount factor. In offline setting, we only have a static dataset $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, where each trajectory $\tau_i = (s_0, a_0, \ldots, s_{T_i})$ is collected without reward annotations. The downstream task is specified by a goal $g \in \mathcal{G}$, which may be given as a natural-language instruction rather than an explicit numerical specification, such as a reward or goal state. The objective is to learn a goal-conditioned policy $\pi(a|s, g)$ from $\mathcal{D}$ that maximizes $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t \delta_g(s_t)]$, where $\delta_g(s_t)$ is an indicator function that equals 1 if the goal g is achieved at state s_t, and 0 otherwise.

3.2 Hierarchical Control

In hierarchical control (Pateria et al., 2021; Liu et al., 2025), a high-level planner π_H selects temporally extended behaviors, or *skills*, while a low-level controller π_L realizes each selected skill through primitive actions. By planning over reusable skills rather than individual actions, this decomposition provides temporal abstraction for long-horizon control. Formally, let $\mathcal{Z}$ be the skill space, and let t_k denote the environment timestep at which the k-th high-level decision is made. At t_k, π_H samples a skill $z_k \sim \pi_H(\cdot \mid s_{t_k}, g)$ from $\mathcal{Z}$, conditioned on the current state s_{t_k} and task goal g. The low-level controller then executes primitive actions $a_t \sim \pi_L(\cdot \mid s_t, z_k)$ for the subsequent environment steps until the next high-level decision at t_{k+1}. Thus, π_H operates at a coarser temporal scale, whereas π_L grounds skills into executable control.

4 Method

As shown in Fig. 1, given an offline dataset $\mathcal{D}$ with image observations, SAGE builds a VLM-based, semantic-first hierarchical control pipeline with three stages: (i) **skill discovery & refinement**, where trajectory segments are labeled by a VLM and refined through semantic merging and filtering, yielding a compact skill set together with estimated durations for each skill; (ii) **skill-conditioned policy learning**, where a library of low-level skill policies is learned through two-stage imitation

learning; and (iii) **VLM-based planning & execution**, where we design a memory-based feedback mechanism that enables the VLM planner to execute the selected skills in a closed loop to complete downstream tasks. The full pipeline is summarized in Algorithm 1 in Appendix A.

4.1 Skill Discovery and Refinement

4.1.1 Trajectory Segmentation.

Skill boundaries in long-horizon offline trajectories are rarely annotated, and rigid boundary detection is brittle to transitional behaviors. We therefore use a sliding window to obtain candidate skill segments. Specifically, a trajectory τ of length T is decomposed as

$$\begin{aligned} S_\tau &= \left\{\tau^{(k)}\right\}_{k=1}^{K_\tau}, \;\; t_k = (k-1)\delta \\ \tau^{(k)} &= \left(s_{t_k}, a_{t_k}, s_{t_k+1}, \ldots, a_{t_k+L}\right). \end{aligned} \tag{1}$$

where L is the window length, δ is the stride, and $K_\tau = \lfloor (T-L)/\delta \rfloor + 1$ is the segment count. Overlapping windows reduce sensitivity to boundary errors and increase the chance that a coherent skill is fully covered by at least one segment.

4.1.2 VLM-based Skill Labeling.

For each segment, we infer a semantic skill label using a VLM. Instead of relying only on boundary frames (Liu et al., 2025), we uniformly sample M frames from each segment, including the boundary frames, where M is a hyperparameter. Let $\{o_j^{(k)}\}_{j=1}^{M}$ denote the sampled frames from segment $\tau^{(k)}$. The corresponding skill label is obtained as

$$z^{(k)} = f_{\text{VLM}}\big(\{o_j^{(k)}\}_{j=1}^{M}, \mathcal{Z}_k\big), \tag{2}$$

where $\mathcal{Z}_k$ denotes the set of skill labels discovered so far.

Initially, $\mathcal{Z}_k = \emptyset$. To reduce redundancy, we prompt the VLM to first search for a suitable label within $\mathcal{Z}_k$ and generate a new label only when none of the existing labels matches the segment. In addition, to suppress noise caused by uninformative or ambiguous segments, we follow (Liu et al., 2025) and provide the VLM with an explicit "*no obvious skill*" option. If this option is selected, the corresponding segment will be excluded from the subsequent skill learning.

4.2 Semantic Merging and Filtering.

Due to inevitable hallucinations of VLMs, raw skill labels contain redundancy and task-irrelevant content. To obtain a clean and learnable skill set, we apply semantic merging, domain-relevance filtering, and support-based frequency filtering, resulting in a compact and consistent skill list.

Specifically, we first use an LLM f_{LLM} to merge semantically equivalent or highly similar skill labels while preserving the action–target structure of each skill. For example, "*Move to Tree*" and "*Reach Trees*" can be merged into the same skill label. This step reduces label fragmentation and improves the consistency of skill representations, which is important for stable skill policy learning.

Next, we use f_{LLM} to assess the domain relevance of each skill label and remove skills that are irrelevant to the target domain, thereby reducing contextual contamination for the high-level planner. To avoid leaking test-time information, the relevance prompt contains only domain-level information, such as "kitchen manipulation" or "grid-world navigation", together with a pair of predefined examples of similar labels, without exposing the specific test-time task descriptions.

Finally, considering the importance of sufficient data support in the offline setting, we filter skills according to their occurrence frequencies. Skill labels associated with too few trajectory segments are removed, ensuring that each retained skill has enough trajectory support for stable and reliable training. Specifically, let $\mathcal{Z} = \{z_i\}$ denote the skill label set after semantic merging and domain-relevance filtering, and let f_i be the occurrence frequency of skill label z_i in the trajectory dataset. We use the average frequency of all labels in $\mathcal{Z}$ as a reference and retain the following subset:

$$\mathcal{Z}_{\text{final}} = \left\{ z_i \in \mathcal{Z} \;\middle|\; f_i \geq \alpha \cdot \frac{1}{|\mathcal{Z}|} \sum_{j=1}^{|\mathcal{Z}|} f_j \right\}, \tag{3}$$

where α is a hyperparameter controlling the filtering strength.

4.2.1 DURATION ESTIMATION.

Since a skill may cover multiple trajectory segments, we estimate an execution horizon $\widehat{T}_z$ for each retained skill $z \in \mathcal{Z}_{\text{final}}$. At deployment, a selected skill z_t runs the corresponding low-level controller for $\widehat{T}_{z_t}$ primitive steps before the planner is queried again. Empirically, we found that directly estimating the average segment length can be inaccurate, possibly due to noise in VLM annotations. Hence, we adopt a more relaxed estimate based on the average number of timesteps occupied by each skill within a trajectory. Formally, let $n_i(z)$ be the number of z-labeled segments in trajectory τ_i, and let $p_i(z) = n_i(z)/\sum_{z'} n_i(z')$ be the relative frequency of z in τ_i. Denoting by $\mathcal{I}(z)$ the set of trajectories that contain skill z, we set

$$\widehat{T}_z = \text{round}\left(\frac{1}{|\mathcal{I}(z)|}\sum_{i\in\mathcal{I}(z)} p_i(z)\,|\tau_i|\right), \tag{4}$$

i.e., the average number of z-occupied timesteps per trajectory where z appears, yielding an aggregate replanning horizon that we find empirically more robust to segmentation and labeling noise.

4.3 SKILL-CONDITIONED POLICY LEARNING

In the reward-free setting, the offline dataset does not provide reward signals. Therefore, we adopt behavior cloning (BC) to train low-level policies for the discovered skills. To mitigate the noise introduced by sliding-window-based skill segmentation and VLM annotation, especially around skill-switching boundaries, we perform transition-level training rather than trajectory-level training.

Meanwhile, due to the limited amount of offline data, training an independent policy solely on the subset of data associated with each skill would be insufficient. To address this issue, we adopt a two-stage imitation learning procedure. First, we pretrain a base policy π_{BC} on all offline transitions, enabling it to acquire basic environment perception and general action priors:

$$\max_{\pi_{\text{BC}}} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\log \pi_{\text{BC}}(a \mid s)\right]. \tag{5}$$

Then, for each skill z, we initialize a skill-specific policy with the weights of π_{BC} and finetune it on the transition subset associated with skill label z:

$$\max_{\pi_z} \mathbb{E}_{(s,a)\sim\mathcal{D}_z}\left[\log \pi_z(a \mid s)\right], \tag{6}$$

where $\mathcal{D}_z$ denotes the subset of offline transitions labeled with skill z. If a transition is assigned to segments with different labels, it is duplicated across the corresponding subsets $\mathcal{D}_z$, which exposes each skill policy to its boundary transitions and encourages smooth inter-skill handoffs.

For notational convenience, we denote the collection of all low-level policies by $\pi(a \mid s, z)$, where the skill label z indexes the corresponding skill-specific policy π_z.

4.4 VLM-BASED PLANNING AND EXECUTION

To enable reward-free zero-shot decision making from language instructions, we use a VLM as the high-level planner that maps visual observations and task specifications to discrete skill choices. Since each VLM query is stateless, we introduce an execution memory to preserve interaction context across planning steps, allowing the planner to condition its decisions on the history.

At each planning step, the VLM selects the next skill from the available set $\mathcal{Z}$, conditioned on the current and initial observations o_t and o_0, the textual task goal g, and the execution memory $\mathcal{H}_t$ of previously selected skills:

$$z_t = f_{\text{VLM}}(o_t, o_0, g, \mathcal{H}_t, \mathcal{Z}), \tag{7}$$

The selected skill z_t is then executed by the corresponding low-level policy $\pi(a \mid s, z_t)$ for $\widehat{T}_{z_t}$ primitive steps. Once the execution horizon is reached, z_t is appended to the memory $\mathcal{H}$, which is used in the next VLM query. Repeating this loop yields closed-loop interaction with the environment.

5 EXPERIMENTS

We evaluate SAGE on various long-horizon tasks to answer: (i) How does SAGE compare against representative offline learning and hierarchical VLM-enhanced baselines? (ii) Can SAGE adapt to different downstream tasks at test time? (iii) How does each component contribute, and how robust is SAGE to its design choices? (iv) Does SAGE discover a compact, interpretable skill library?

Table 1: Evaluation results averaged over 5 random seeds. Bold marks the best result and underline marks the second best. $\pm$ represents 1 standard deviation.

Environment	BC	CQL	IQL	RvS	LDCQ	VanTA	SAGE (Ours)
Crafter†	2.70	1.73	2.75	2.11	0.96	<u>5.46</u>	**6.52** $\pm$ 0.30
Franka Kitchen†	65.0	43.8	62.5	50.2	52.8	<u>69.2</u>	**79.6** $\pm$ 4.45
DoorKey-6x6-complete	<u>0.77</u> $\pm$ 0.05	0.53 $\pm$ 0.15	0.74 $\pm$ 0.06	0.64 $\pm$ 0.19	0.57 $\pm$ 0.11	0.70 $\pm$ 0.09	**0.84** $\pm$ 0.04
DoorKey-8x8-complete	0.26 $\pm$ 0.10	0.08 $\pm$ 0.11	0.21 $\pm$ 0.17	0.27 $\pm$ 0.17	0.25 $\pm$ 0.11	<u>0.30</u> $\pm$ 0.05	**0.46** $\pm$ 0.04
KeyCorridor-S3R3-complete	<u>0.59</u> $\pm$ 0.08	0.34 $\pm$ 0.18	0.40 $\pm$ 0.09	0.35 $\pm$ 0.12	0.15 $\pm$ 0.10	0.48 $\pm$ 0.17	**0.67** $\pm$ 0.04
RedBlueDoor-6x6-complete	**0.87** $\pm$ 0.06	**0.87** $\pm$ 0.02	<u>0.85</u> $\pm$ 0.08	0.39 $\pm$ 0.27	0.82 $\pm$ 0.06	<u>0.85</u> $\pm$ 0.04	**0.87** $\pm$ 0.05
DoorKey-6x6-mixed	0.60 $\pm$ 0.06	0.24 $\pm$ 0.19	<u>0.71</u> $\pm$ 0.07	0.70 $\pm$ 0.06	0.53 $\pm$ 0.12	0.55 $\pm$ 0.21	**0.79** $\pm$ 0.11
DoorKey-8x8-mixed	0.11 $\pm$ 0.09	0.02 $\pm$ 0.04	0.14 $\pm$ 0.11	<u>0.36</u> $\pm$ 0.07	0.24 $\pm$ 0.06	0.24 $\pm$ 0.06	**0.40** $\pm$ 0.07
KeyCorridor-S3R3-mixed	0.29 $\pm$ 0.09	0.06 $\pm$ 0.11	<u>0.33</u> $\pm$ 0.08	0.28 $\pm$ 0.14	0.19 $\pm$ 0.05	0.12 $\pm$ 0.06	**0.44** $\pm$ 0.06
RedBlueDoor-6x6-mixed	0.73 $\pm$ 0.05	0.62 $\pm$ 0.24	**0.84** $\pm$ 0.07	0.27 $\pm$ 0.35	<u>0.83</u> $\pm$ 0.04	0.70 $\pm$ 0.04	0.73 $\pm$ 0.08

† Crafter and Kitchen baselines are from VanTA's paper. Other baselines are reproduced locally.

5.1 SETUP

Environments. We evaluate on three long-horizon environments. **(i) Crafter** (Hafner, 2022) is a 2D open-world survival environment for long-horizon achievement acquisition. We use the offline data from (Liu et al., 2025). **(ii) Franka Kitchen** (Gupta et al., 2020) is a 3D Franka-Panda manipulation benchmark with multi-stage tasks. We use the D4RL dataset (Fu et al., 2020). **(iii) MiniGrid** (Chevalier-Boisvert et al., 2023) is a 2D grid-world suite with sequential subgoals. We collect offline datasets by trained PPO agents. The datasets include a *completed* setting, which contains only successful trajectories, and a *mixed* setting, which contains both successful and failed trajectories. All of them are simulated, and we discuss the potential extension to the real world in Appendix E.

Metrics. In Crafter, following (Liu et al., 2025), we adopt the Crafter score, the geometric mean of success rates across the 22 achievements, which emphasizes rare successes. In Kitchen, we report the normalized return following (Fu et al., 2020). In MiniGrid, we report per-task success rate.

Baselines. We compare against the representative offline imitation method **BC**, offline RL methods **CQL** (Kumar et al., 2020) and **IQL** (Kostrikov et al., 2022), goal-conditioned methods **RvS** (Emmons et al., 2022) and **LDCQ** (Venkatraman et al., 2024), and the SOTA hierarchical VLM-enhanced method **VanTA** (Liu et al., 2025), which obtains temporal abstraction with VQ-VAE latent skills grounded by a VLM. We implement BC, CQL, and IQL using OfflineRL-Kit (Sun, 2023), and use the official codebases for all other baselines. Results are averaged over 5 random seeds.

Implementation. We use GPT-4o (OpenAI, 2024) as the VLM and GPT-5.2 (OpenAI, 2025) as the LLM for label merging and set the temperature $\tau = 0.0$ for reproducibility. We use an MLP as the policy network. The main hyperparameters and example prompts are in Appendices B and C.

5.2 EVALUATION RESULTS

The experimental results are summarized in Table 1. The results on Crafter and Kitchen are taken from those reported in the VanTA paper (Liu et al., 2025), while the MiniGrid results are obtained from our reproduction, since we collect the offline datasets ourselves. We further report our local reproductions for Crafter and Kitchen in Appendix D. For value-based offline RL methods, we provide sparse reward signals in the dataset during training.

Overall, even without reward signals, SAGE is competitive or better on 9 of the 10 tasks: strictly best on 8, tied on RedBlueDoor-6x6-complete, and among the top on RedBlueDoor-6x6-mixed. It suggests that the VLM offers more effective decision-making guidance than reward supervision for language-specified goals. SAGE also consistently outperforms VanTA, the SOTA VLM-enhanced baseline, for two reasons: its semantics-first skill discovery yields more accurate low-level policies,

Table 2: Average return on multi-task dataset over 5 random seeds. Bold marks the best result and underline marks the second best. $\pm$ represents 1 standard deviation.

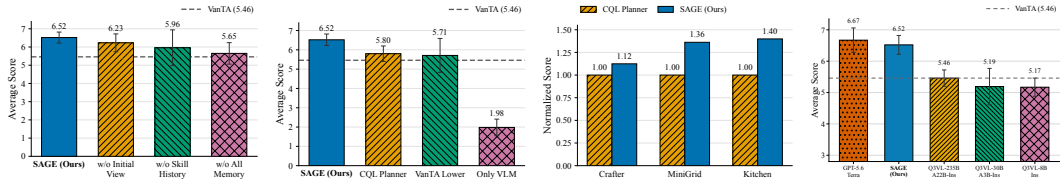
Method	Setting	Task A [R-Y-B-G]	Task B [G-B-Y-R]	Avg.
BC	Matched	0.31 ± 0.20	$\underline{0.32} \pm 0.11$	$\underline{0.32} \pm 0.15$
	Mismatched	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Mixed	0.22 ± 0.06	0.24 ± 0.07	0.23 ± 0.05
IQL	Matched	$\mathbf{0.40 \pm 0.11}$	0.25 ± 0.11	$\underline{0.32} \pm 0.13$
	Mismatched	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	Mixed	0.16 ± 0.03	0.22 ± 0.06	0.19 ± 0.04
SAGE (Ours)	Prompt	$\underline{0.37} \pm 0.12$	$\mathbf{0.35 \pm 0.13}$	$\mathbf{0.36 \pm 0.13}$

and its VLM planner selects skills by reasoning directly over instructions, avoiding the unreliable value estimation of offline RL under limited data.

Specifically, on Crafter, SAGE improves over the strongest baseline VanTA by up to **19%**, indicating that it can more effectively handle long-term dependencies under language guidance and acquire a more diverse set of achievements. On Kitchen, SAGE also achieves a **15%** improvement over the best baseline, showing that the VLM-based hierarchical control flow can more reliably support complex manipulation behaviors. On MiniGrid, SAGE obtains performance gains in most tasks, with a particularly large improvement of up to **53%** on DoorKey-8x8-complete. Even on RedBlueDoor-6x6-mixed, where its performance is slightly lower than reward-guided offline RL methods, SAGE remains among the top-performing methods and outperforms VanTA, which also adopts a VLM-enhanced hierarchical framework. These results indicate SAGE can effectively ground the prior knowledge of a frozen, off-the-shelf VLM in long-horizon decision-making tasks using reward-free offline data. We provide a concrete example in Appendix F, where the traditional baselines quickly get lost and VanTA fails midway, whereas SAGE's semantics-first interface lets it discover and execute all the required skills to solve the task.

5.3 Ablation Study

We ablate key design choices of SAGE to assess their contributions and robustness.



(a) Planner memory. (b) Architecture design. (c) Planner design. (d) VLM backbone.

Figure 2: Ablation results. Panels (a), (b) and (d) are evaluated on Crafter. Panel (c) reports the average performance across three environments, normalized to CQL-Planner (set to 1). In (d), Q3VL- means Qwen3-VL-, and -Ins means -Instruct.

Planning Memory. To validate the memory mechanism in the high-level planner, we ablate its two memory components: the skill history and the initial observation. Fig. 2a shows that the skill history contributes the most to SAGE, as it allows the VLM planner to track task progress and avoid repeatedly selecting ineffective skills. The initial observation also provides a consistent benefit, possibly because it helps the VLM identify state changes relative to the starting scene. Notably, even when all memory components are removed, SAGE still outperforms VanTA, further demonstrating the advantage of our VLM-based semantics-first pipeline.

Hierarchical Design. We examine the hierarchical design by swapping SAGE's high-level planner and low-level skill policy for VanTA's counterparts and comparing against a pure VLM controller for end-to-end decision making. In Fig. 2b, replacing either component clearly degrades performance, showing that SAGE's strength stems from the synergy between its hierarchical controllers.

Concretely, the VLM planner leverages prior knowledge and precise language-conditioned instruction following to avoid being misled by unreliable reward learning in the offline setting: SAGE still outperforms the full VanTA framework even with a VanTA-style low-level policy, and replacing the VLM planner with a CQL planner reduces performance across all tasks (Fig. 2c). Meanwhile, the semantics-first skill discovery complements it with a more accurate and interpretable control interface, so that swapping in VanTA's low-level policy likewise degrades performance. In contrast, using the VLM alone as an end-to-end controller performs poorly, showing that the advantage of SAGE comes not from the VLM alone, but from SAGE's architecture that effectively grounds VLM planning into executable low-level skills.

VLM Backbone. To study how foundation-model capability affects SAGE, we replace all its foundation models with close-source GPT-5.6-Terra (OpenAI, 2026b) and three open-source Qwen3-VL-Instruct variants (8B, 30B-A3B, and 235B-A22B) (Bai et al., 2025), keeping everything else fixed. Fig. 2d reports the results on Crafter: all the backbones score well above the strongest offline RL baseline IQL, and performance improves with backbone capacity, consistent with stronger VLMs offering higher-quality semantic abstractions and planning decisions. Its strongest configuration with GPT-5.6-Terra reaches 6.67, and even with the open-source Qwen3-VL-235B-A22B-Instruct, SAGE already matches VanTA with GPT-4o at 5.46. SAGE thus stays effective across backbones.

Temperature and Prompt Wording. We also examine how SAGE depends on the decoding temperature and the prompt wording. Table 3 reports the results on DoorKey-6x6-mixed, where the success rate stays consistently high across $\tau \in \{0.0, 0.5, 0.7\}$. Although we use $\tau = 0.0$ for reproducibility, a moderate amount of decoding diversity can further improve performance. Meanwhile, SAGE is stable under paraphrasing: rewording all prompts with GPT-5.5 (OpenAI, 2026a) while preserving their meaning still yields 0.74 ± 0.10 on this task, above the best baseline (0.71 ± 0.07).

Table 3: Sensitivity to the decoding temperature τ on DoorKey-6x6-mixed.

Temperature	$\tau = 0.0$	$\tau = 0.5$	$\tau = 0.7$
Success Rate	0.79 ± 0.11	0.85 ± 0.03	0.82 ± 0.04

5.4 Multi-task Dataset

To examine the flexibility of SAGE's high-level VLM planner across downstream tasks, we design an experiment on a multi-task dataset. Specifically, in an 8×8 grid, four landmarks with different colors are placed at the four corners: red in the upper-left, yellow in the upper-right, green in the lower-left, and blue in the lower-right. At the beginning of each episode, the agent is initialized at a random position. We consider two tasks: Task A requires the agent to visit red, yellow, blue, and green in order, while Task B requires the agent to visit green, blue, yellow, and red in order. A sparse reward is given only when the task is completed, with $r = 1 - \text{steps}/\text{MAX_STEPS}$. Otherwise, the agent receives no reward. The maximum episode length is 640 steps.

We evaluate SAGE, BC, and IQL. IQL has access to reward signals, while SAGE and BC are trained in without rewards. For the baselines, we consider three protocols: (i) *Matched*, where the model is trained on the dataset corresponding to the downstream task; (ii) *Mismatched*, where the model is trained on data from the other task; and (iii) *Mixed*, where the model is trained on the combined data from both tasks. In contrast, SAGE directly trains its low-level policies on the mixed dataset, and the downstream task goal is provided to the VLM planner only through the prompt at test time.

Table 2 shows that when the training data does not match the downstream task, the baselines perform poorly or even fail. Instead, SAGE effectively adapts to different downstream tasks by reusing skills and following instructions, highlighting the advantage of VLM-enabled zero-shot decision making.

5.5 Skill Analysis

5.5.1 Quantitative Statistics

Table 4 reports the number of trajectory segments and the number of unique labels after each refinement stage for each environment. The pipeline substantially compresses the label space while preserving domain-meaningful behaviors. On Crafter, the 43 raw labels are consolidated into 40 retained skills, for example by merging "*Make Wood Pickaxe*" and "*Craft Wood Pickaxe*" into a single skill, while the "*no obvious skill*" option discards 208 ambiguous segments (11.2% of the total).



Figure 3: Visualization of the decision sequence of SAGE in Franka Kitchen.

On Kitchen, the pipeline recovers exactly the four ground-truth subtasks and prunes low-support candidates such as "*Open the hinge cabinet*" with only one segment. Across the eight MiniGrid tasks, the retained skills consistently comprise "*Move to Key*", "*Pick up Key*", "*Unlock Door*", and "*Move to Goal*", with 30–92% of segments retained, higher for the completed datasets than the mixed datasets.

Table 4: Skill discovery statistics. We report the number of trajectory segments and the size of the label set after each refinement stage for each environment.

Environment	#Seg.	Raw	Merged	Retained
Crafter	1856	43	41	40
Kitchen	984	6	6	4
MiniGrid	927–4549	6	6	4–5

5.5.2 Qualitative Analysis

Since Kitchen encodes a fixed subtask order in its dense reward, we use it as ground truth to evaluate label quality. For each retained skill, we report its *Coverage*: the fraction of segments assigned that skill in which the skill actually occurs in the ground truth. As shown in Table 5, coverage is generally high, especially for *Open the Microwave* (95%) and *Open the Cabinet* (100%). *Pick and Move the Kettle* is relatively lower (40%), mainly because the switch is toggled with a very brief motion, so segments spanning the relatively long kettle-to-switch transition are visually ambiguous and are often labeled as the end of the kettle skill. Nevertheless, SAGE still recovers all four ground-truth subtasks with correct names and sufficient data support, and its transition-level policy training absorbs such boundary noise by producing smooth inter-skill handoffs, keeping the discovered interface effective for downstream planning.

Table 5: Per-skill label coverage on Kitchen, measured against the ground-truth subtask order.

Skill	#Seg.	Coverage
Open the Microwave	288	95%
Pick and Move the Kettle	567	40%
Rotate the Switch	86	84%
Open the Cabinet	38	100%

5.5.3 Visualization

Fig. 3 visualizes the decisions made by the VLM planner in Kitchen. Above the image trajectory are the skill labels chosen by the VLM planner, together with the number of executed primitive steps. It shows that SAGE can discover and execute semantically meaningful skills offline and complete downstream tasks. We further show that SAGE recovers more interpretable, task-aligned skills than latent-first methods on KeyCorridor-S3R3-complete (Appendix F).

6 Conclusion

In this paper, we propose SAGE, a semantic-first framework that grounds off-the-shelf VLM planning in reward-free offline trajectories. SAGE uses a VLM to summarize trajectory segments into skill labels, merges and filters them into a compact skill library, grounds each skill through two-stage imitation learning, and executes them via a memory-based closed-loop VLM planner. Across diverse long-horizon tasks, SAGE matches or outperforms representative offline learning and hierarchical VLM-enhanced baselines on 9 out of 10 tasks, with relative improvements of up to 53%, showing that semantic skill grounding is an effective way to connect VLM prior knowledge with executable long-horizon control. Future work includes finer-grained skill-duration estimation, integration with more expressive low-level policy models, and real-world deployment.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on research methodology and experiments, and assisting with translation. We have not used generative AI tools for proposing or refining hypotheses, designing research methodology or experiments, and supporting qualitative and thematic data analysis, interpreting results. Helping develop theoretical models or conceptual frameworks, providing critical ingredients for proving mathematical claims, formulating mathematical claims, assisting in the writing of proofs, generating synthetic data sets, and cleaning and reformatting dataset are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, drafting parts of a research paper, summarizing existing literature, and identifying relevant literature. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by 2 authors. LLM-generated text and LLM-suggested references were also checked manually. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pp. 287–318. PMLR, 2023.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and J K Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 73383–73394. Curran Associates, Inc., 2023.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline RL via supervised learning? In *International Conference on Learning Representations*, 2022.

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.

Haotian Fu, Pratyusha Sharma, Elias Stengel-Eskin, George Konidaris, Nicolas Le Roux, Marc-Alexandre Côté, and Xingdi Yuan. Language-guided skill learning with temporal variational inference. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 14135–14156, 2024.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1025–1037. PMLR, 30 Oct–01 Nov 2020.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Tian-Shuo Liu, Xu-Hui Liu, Ruifeng Chen, Lixuan Jin, Pengyuan Wang, Zhilong Zhang, and Yang Yu. Semantic temporal abstraction via vision-language model guidance for efficient reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2026. ISSN 2162-2388.

Oier Mees, Jessica Borja-Diaz, and Wolfram Burgard. Grounding language with visual affordances over unstructured data. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11576–11582. IEEE, 2023.

Vivek Myers, Andre Wang He, Kuan Fang, Homer Rich Walke, Philippe Hansen-Estruch, Ching-An Cheng, Mihai Jalobeanu, Andrey Kolobov, Anca Dragan, and Sergey Levine. Goal representations for instruction following: A semi-supervised language interface to control. In *Conference on Robot Learning*, pp. 3894–3908. PMLR, 2023.

OpenAI. Hello gpt-4o. `https://openai.com/index/hello-gpt-4o/`, 2024.

OpenAI. Introducing gpt-5.2. `https://openai.com/index/introducing-gpt-5-2/`, 2025.

OpenAI. Introducing gpt-5.5. `https://openai.com/index/introducing-gpt-5-5/`, 2026a.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, 2026b.

Jing-Cheng Pang, Si-Hang Yang, Kaiyuan Li, Xiong-Hui Chen, Nan Tang, and Yang Yu. Kalm: Knowledgeable agents by offline reinforcement learning from large language model rollouts. *Advances in Neural Information Processing Systems*, 37:126620–126652, 2024.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 39737–39761, 2024a.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *The Twelfth International Conference on Learning Representations*, 2024b.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025.

Seohong Park, Aditya Oberai, Pranav Atreya, and Sergey Levine. Transitive RL: Value learning via divide and conquer. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=XBTJnfbDkP`.

Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Seungeun Rho, Laura Smith, Tianyu Li, Sergey Levine, Xue Bin Peng, and Sehoon Ha. Language guided skill discovery. In *International Conference on Learning Representations*, volume 2025, pp. 87731–87752, 2025.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3009, 2023.

Yihao Sun. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. `https://github.com/yihaosun1124/OfflineRL-Kit`, 2023.

Siddarth Venkatraman, Shivesh Khaitan, Ravi Tej Akella, John Dolan, Jeff Schneider, and Glen Berseth. Reasoning with latent diffusion in offline reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 48252–48273, 2024.

Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5064–5078, 2024. doi: 10.1109/TNNLS.2022.3207346.

Haiyan Yin, Hangwei Qian, Yaxin Shi, Ivor Tsang, and Yew-Soon Ong. Grounding open-domain knowledge from llms to real-world reinforcement learning tasks: A survey. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 10797–10806. International Joint Conferences on Artificial Intelligence Organization, 8 2025. Survey Track.

Jesse Zhang, Minho Heo, Zuxin Liu, Erdem Biyik, Joseph J Lim, Yao Liu, and Rasool Fakoor. Extract: Efficient policy learning by extracting transferable robot skills from offline data. In *Conference on Robot Learning*, pp. 1148–1172. PMLR, 2025.

Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5625–5644, 2024.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

Technical Appendices

The organization of the appendix is as follows:

- Appendix A: We present the pseudocode of SAGE.
- Appendix B: We list the key hyperparameters of SAGE.
- Appendix C: We provide an example set of prompts for the full pipeline.
- Appendix D: We report our local reproduced results for Crafter and Franka Kitchen tasks.
- Appendix E: We discuss the potential extension of SAGE to real-world deployment.
- Appendix F: We qualitatively compare the execution trajectories of SAGE and other baselines, as well as the skill extraction of SAGE and latent-first methods.

A Algorithm Pseudocodes

Algorithm 1 Semantic Action Grounding and Execution (SAGE)

Require: Offline dataset $\mathcal{D}$, VLM f_{VLM}, LLM f_{LLM}
// Skill discovery and refinement
$\mathcal{Z} \leftarrow \emptyset$
Segment each $\tau \in \mathcal{D}$ into S_τ by equation 1.
for $\tau \in \mathcal{D}$ **do**
  for $k = 1$**to** K_τ **do**
    Calculate $z^{(k)}$ from f_{VLM} according to equation 2.
    Drop $\tau^{(k)}$ if $z^{(k)}$ is "no obvious skill".
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{z^{(k)}\}$
  end for
end for
Merge labels with f_{LLM}, and calculate $\mathcal{Z}_{\text{final}}$ according to equation 3.
for $z \in \mathcal{Z}_{\text{final}}$ **do**
  Estimate skill duration $\widehat{T}_z$.
end for
// Skill-conditioned policy learning
Train base policy $\pi_{\text{BC}}(a \mid s)$ on $\mathcal{D}$ by equation 5.
for $z \in \mathcal{Z}_{\text{final}}$ **do**
  $\pi_z \leftarrow \pi_{\text{BC}}$
  Finetune π_z on $\mathcal{D}_z$ by equation 6.
end for
// VLM planning and execution
Initialize execution history $\mathcal{H} \leftarrow \emptyset$, timestep $t \leftarrow 0$, and observe o_{init}.
while episode is not terminated **do**
  $z_t \leftarrow f_{\text{VLM}}(o_t, o_{\text{init}}, g, \mathcal{H}, \mathcal{Z})$
  Execute π_{z_t} for T_{z_t} primitive steps.
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{z_t\}$
  $t \leftarrow t + T_{z_t}$
end while

B Implementation Details

Table 6 summarizes the main hyperparameters of SAGE for each environment, including the sliding-window settings used in skill discovery and the shared policy-training configuration.

C Example Prompts

Taking the Kitchen environment as an example, we provide the prompts used by SAGE for the three VLM/LLM-dependent stages: VLM skill labeling (Fig. 4), LLM-based semantic merging and

Table 6: Hyperparameters for SAGE.

Hyperparameter	Crafter	Kitchen	MiniGrid
Window Length L	24	40	8 (5 for DoorKey)
Stride δ	1	3	3 (2 for DoorKey)
Frames M	2	5	5
Frequency Threshold α	0.01		
MLP Hidden Dim	256		
Batch Size	32		
Learning Rate	0.0001		
Optimizer	Adam(Kingma & Ba, 2015)		

domain-relevance filtering (Fig. 5), and closed-loop planning (Fig. 6). The prompt structures for the other environments are analogous. Note that the labeling and merging prompts only expose domain-level information (e.g., "kitchen manipulation") and never reveal the specific downstream task, so no test-time information leaks into skill construction.

System Prompt

You are an expert in understanding the behavior of a robot arm in the Franka Kitchen environment.
The environment consists of a kitchen scene with various objects that the robot can interact with.
Key objects include:
- Microwave
- Kettle (On the Stove)
- Light Switch
- Slide Cabinet
- Hinge Cabinet
- Four Burner Switches

The Microwave is on the left side of the kitchen, the stove is on the right side, and the white kettle is on the stove. The blue slide cabinet and blue hinge cabinet are above the stove and black switch panel, and the slide cabinet is right to the hinge cabinet. The four burner switches are on the black panel under the cabinets. To the left of the four burner switches is the single light switch.
Your task is to identify the specific skill or behavior exhibited by the robot arm among the provided images.
Focus on which object the robot arm is interacting with and what action it is performing.Use concise and descriptive skill labels.

User Prompt

I will provide you with five images sampled from a short trajectory segment in the Franka Kitchen environment. These images are temporally ordered and include the beginning, intermediate states, and the final state of a single skill execution.
Please analyze the difference among the visual states to determine what skill the robot arm performed.
Focus on the changes in the state of the objects.

The images are provided as:
1. First Frame
2. Second Frame
3. Third Frame
4. Fourth Frame
5. Fifth Frame

Based on these observation, analyze what happened and provide a single, concise skill label for this segment. The label should contain both the action and the object. Here are examples of possible skills already summarized:
1. Open the Cabinet.
2. Rotate the switch.
3. Pick and Move the kettle.
4. Open the microwave.

From my perspective, you can reason what the skill is according to given sequential images. If you think the task happened between two states in the summarized list, output the skill name. Remember maybe many changes happen between two images, so choose the most suitable skill name. If not in them, and you still regard it as a new skill, try your best to give only one suitable new name.
IMPORTANT: If you determine that there is no obvious skill, directly output 'No obvious skill'.
Output format: JSON with a key "skill" and the value is the label string and a key "reason" with the value being a brief explanation of how you arrived at the label.

Figure 4: Labeling prompt for Kitchen.

System Prompt

You are a helpful assistant for clustering skill labels.

User Prompt

I have extracted the following skill set from the Franka Kitchen dataset:
{labels}

Some of the skills are semantically redundant or irrelevant for high-level task planning. Please help me clean this skill list by:
Merging semantically identical or synonymous skills (e.g., "Open Microwave Door" and "Open Microwave"). Removing skills that are definitely irrelevant for downstream task planning. Keep the useful skills unchanged.
Please output the result as a JSON dictionary in the following format:

{{ "old_label": "new_label", ...}}

For skills you deem irrelevant, map them to irrelevant". Do not output anything else.

Figure 5: Merging and filtering prompt for Kitchen.

System Prompt

You are an expert robot controller in a kitchen environment.
Your task is to complete the following subtask in order:
- open the microwave
- move kettle to another stove
- turn on the toggle switch of light
- open cabinet

I will show you the initial and current views of the kitchen and the history of skills executed. You need to execute the most appropriate skill to make progress.
I will show you a list of available skills. You have to choose the most appropriate skill to execute next.
It is important to ensure the completion of each skill execution and avoid unnecessary repetitions.
You should observe the current state of the environment carefully.
CRITICAL RULES:
- When judge the completion of a subtask, you should compare the state change between the initial view and the current view.

Output a JSON object with the following structure:
{
  "reasoning": "A brief explanation of why you chose the skill next",
  "skill": "The chosen skill name (must be one of the available skills)"
}

User Prompt

Initial View: {initial_image}
Current View: {current_image}
History of Skills Executed: {skill_history}
Available Skills: {available_skills}

What skill should I do next?

Figure 6: Planner prompt for Kitchen.

D Local Reproduction of Crafter and Kitchen Baselines

For the main results in Table 1, the Crafter and Kitchen baseline numbers are taken directly from the VanTA paper (Liu et al., 2025) to keep the comparison faithful to the originally published numbers. To verify that this choice does not favor SAGE, we additionally reproduce all baselines on Crafter and Kitchen locally, using the public datasets and 5 random seeds. As shown in Table 7, SAGE remains the best method on both environments and outperforms the strongest baseline by a clear margin.

Environment	BC	CQL	IQL	RvS	LDCQ	VanTA	SAGE
Crafter	<u>3.65</u> $\pm$ 0.29	0.97 $\pm$ 0.23	2.34 $\pm$ 0.69	1.65 $\pm$ 0.37	3.05 $\pm$ 0.22	3.56 $\pm$ 1.04	**6.52** $\pm$ 0.30
Kitchen	<u>69.40</u> $\pm$ 14.02	30.10 $\pm$ 6.85	47.70 $\pm$ 3.65	36.80 $\pm$ 8.76	42.95 $\pm$ 3.68	68.25 $\pm$ 6.58	**79.60** $\pm$ 4.45

Table 7: Local reproduction of the Crafter and Kitchen baselines, averaged over 5 random seeds. Bold marks the best result and underline marks the second best. $\pm$ represents 1 standard deviation.

E Discussion on Real-World Deployment

In this section, we discuss the potential extension of SAGE to real-world deployment. In real-world deployment, it would introduce dynamics mismatch that may require low-level policy adaptation, as well as noisier observations that could degrade VLM perception and labeling. Our open-source-backbone and prompt-robustness results suggest that SAGE is relatively robust to labeling quality, and its human-readable skill interface decouples planning from control so that each component can be adapted independently. Nevertheless, real-robot validation remains an important direction for future work.

F Qualitative Comparison of Trajectories and Skills

To illustrate where the advantage of SAGE comes from, we visualize a representative episode on KeyCorridor-S3R3-complete, in which the agent must pick up the key, unlock the door, and reach the target ball.

Figure 7 compares the resulting trajectories. The traditional baselines fail early and exhaust the 270-step budget with zero return: BC stalls in front of the locked door, and CQL spins in place. VanTA, benefiting from its VLM-enhanced hierarchical architecture, makes more progress but still fails midway, never securing the key, reflecting the limitations of its architectural design. In contrast, SAGE, built on semantics-first skill discovery and a VLM planner, solves the task in only 37 steps with a return of 0.88, sequentially picking up the key, moving to and unlocking the door, and navigating to the target.

Figure 8 shows the per-frame skill labels of VanTA and SAGE on one sampled trajectory. Since VanTA does not directly produce semantic labels, we annotate each frame with the skill name that the training-time VLM calibration assigns to its latent code, while for SAGE, whose frames may fall into several overlapping segments, we use the most frequent label among them. The two decompositions differ sharply. Because the latent-first paradigm only lets the VLM guide skill discovery post hoc, VanTA's segmentation collapses into just two overloaded codes, *Move to Key* and *Move to Door*, that alternate erratically and fail to identify later phases such as unlocking the door or reaching the goal as distinct skills. In contrast, SAGE's semantics-first discovery yields contiguous, task-aligned skills, *Pick up Key*, *Move to Door*, *Unlock Door*, and *Move to Goal*, that cover every phase and give the planner a directly interpretable interface.

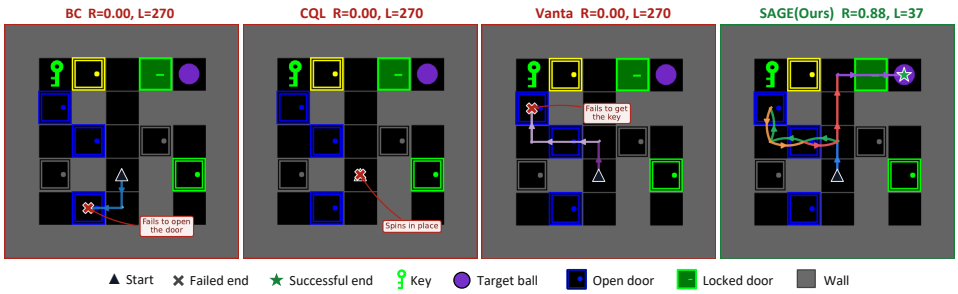


Figure 7: Trajectories of BC, CQL, VanTA, and SAGE on KeyCorridor-S3R3-complete. The three baselines exhaust the step budget without solving the task, while SAGE reaches the target in far fewer steps. R means episode return and L means episode length. Colored arrows denote the executed primitive action, latent skill, or semantic skill.

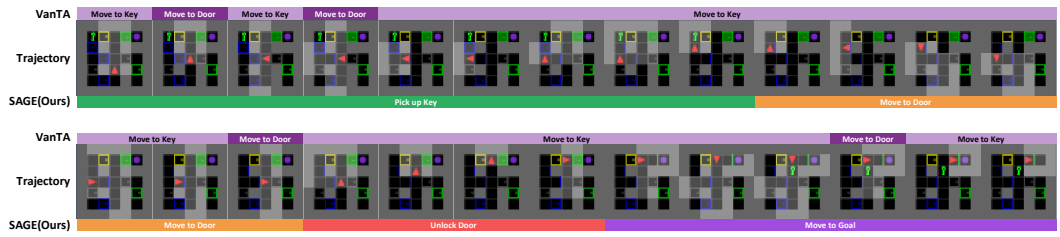


Figure 8: Per-frame skill labels of VanTA and SAGE on the same trajectory. VanTA (top) only alternates between two skills, whereas SAGE (bottom) recovers contiguous, task-aligned skills that cover every phase of the task.