

A Appendix

A.1 Pseudocode of SELU

algorithm 1 SELU

Input: critic MLLM M_c , actor MLLM M_a , critic fine-tuning dataset $\mathcal{D}_{\text{critic}}$, actor fine-tuning dataset $\mathcal{D}_{\text{actor}}$, maximum timestep T , initial instruction list L , success detection prompt p_d and action plan prompt p_a

Output: critic MLLM M_c , actor MLLM M_a

```

1:  $\mathcal{D}_{\text{critic}}, \mathcal{D}_{\text{actor}} \leftarrow \{\}$ 
2: function "SELU"
3:   for instruct  $I$  in  $L$  do
4:     while data collecting not done do
5:       for timestep  $t = 1$  to  $T$  do
6:         get observation  $o_t$  from env
7:          $l_{a,t} = M_a(I, p_a, o_t)$ 
8:         use  $l_{a,t}$  to interact with env
9:       end for
10:       $l_d = M_c(I, p_d, o_T)$ 
11:      if  $l_d = \text{"yes"}$  then
12:        store  $(I, p_d, o_T, l_d)$  into  $\mathcal{D}_{\text{critic}}$ 
13:        store  $(I, p_a, o_t, l_{a,t}), t = 1, \dots, T$  into  $\mathcal{D}_{\text{actor}}$ 
14:      else
15:        get  $l'_d$  through self-asking
16:        if  $l'_d = \text{"yes"}$  then
17:          store  $(I, p_d, o_T, l'_d)$  into  $\mathcal{D}_{\text{critic}}$ 
18:          store  $(I, p_a, o_t, l_{a,t}), t = 1, \dots, T$  into  $\mathcal{D}_{\text{actor}}$ 
19:        else
20:          get  $I'$  through hindsight relabeling
21:          if  $I' \neq \text{"None"}$  then
22:            store  $(I', p_d, o_T, \text{yes})$  into  $\mathcal{D}_{\text{critic}}$ 
23:            store  $(I', p_a, o_t, l_{a,t}), t = 1, \dots, T$  into  $\mathcal{D}_{\text{actor}}$ 
24:          end if
25:        end if
26:      end if
27:    end while
28:  end for
29:  optimization  $M_c$  and  $M_a$  by  $\mathcal{D}_{\text{critic}}$  and  $\mathcal{D}_{\text{actor}}$ 
30:  return critic MLLM  $M_c$ , actor MLLM  $M_a$ 
31: end function

```

A.2 Implementation Details

A.2.1 Environments

Figure 4 shows our experiment environments. Both environments restrict agents to only interact with visible items, limiting their operational range to guarantee behavior plans realistic. Therefore, the actor MLLM makes decisions based on first-person perspective input to ensure accuracy as Figure 4a and Figure 4c show. Given the limitations of the first-person view, the critic MLLM uses a third-person perspective to evaluate the trajectory, reducing hallucinations and obtaining accurate scene information as Figure 4b and Figure 4d show.

The positioning of the third-person camera is crucial, as it should accurately capture the agent’s position and the objects it interacts with. Any occlusion or interference can impair the MLLM’s understanding of the image, thereby affecting the results of critic success detection and hindsight relabeling.



Figure 4: The diagram of experimental environments. We utilize the first-person perspective for decision-making and a third-person perspective for trajectory evaluation.

A.2.2 Hyperparameters

The specific MLLMs we use are LLaVA-V1.6-Mistral-7B and Qwen-VL. We use LoRA to fine-tune them, the hyperparameters are as follows.

Table 7: Hyperparameters of LLaVA Fine-tuning by LoRA.

Hyperparameters	Value
Train_batch_size	16
Eval_batch_size	4
Gradient_accumulation_steps	1
Learning_rate_actor	2e-5
Learning_rate_critic	2e-6
Warmup_ratio	0.03
Weight_decay	0.0
Model_max_length	2048
Lr_scheduler_type	cosine
Tf32	True

Table 8: Hyperparameters of Qwen-VL Fine-tuning by LoRA.

Hyperparameters	Value
Train_batch_size	2
Eval_batch_size	1
Gradient_accumulation_steps	8
Learning_rate_actor	1e-5
Learning_rate_critic	1e-6
Warmup_ratio	0.01
Weight_decay	0.1
Adam_beta2	0.95
Model_max_length	2048
Lr_scheduler_type	cosine
Bf16	True
Lazy_preprocess	True

Both models are configured with a temperature of 0 and a maximum token length of 2048 for response generation. The maximum number of environment steps is set to 10. The multiple chains of thought for self-consistency (Wang et al., 2023a), and the multiple rounds of reflection in self-refine (Madaan et al., 2023) are set to 3, following the settings reported in their papers. For LLM-Planner (Song et al., 2023) and RAP (Kagaya et al., 2024), retrieval parameters follow the official implementation to ensure consistency. The number of retrieved trajectories is fixed at the minimum value of 3, as adopted in RAP. The list of hyperparameters is provided in Table 9.

Table 9: Hyperparameters List for Baselines.

Hyperparameters	Value
Response generation temperature	0
Maximum token length	2048
Maximum environment steps	10
Number of CoTs (Wang et al., 2023a)	3
Number of reflection rounds (Madaan et al., 2023)	3
Retrieve buffer	20
KNN retrieves (LLM Planner (Song et al., 2023))	9
Multimodal retrieves (RAP (Kagaya et al., 2024))	5
Retrieve trajectories input	3

A.2.3 Low-Level Actions of Agents

SELU is designed to enable embodied agents to learn atomic skills through MLLMs themselves. In our experiments, we use the “locobot” agent in AI2-THOR and the “female1” agent in VirtualHome. The set of low-level actions to these agents is summarized in Table 10.

Table 10: Low-Level Actions of AI2-THOR and VirtualHome.

Action	Env	Description
MoveAhead, MoveBack	AI2-THOR/VirtualHome	Move the agent forward or backward by a fixed step (0.25 meters).
RotateLeft, RotateRight	AI2-THOR/VirtualHome	Rotate the agent 90° to the left or right.
Pick	AI2-THOR/VirtualHome	Pick up a visible and graspable object. The action fails if the object is not visible.
Open	AI2-THOR/VirtualHome	Open an object that is both interactable and openable.
Break	AI2-THOR	Break an object that is both interactable and breakable.
Sit	VirtualHome	Make the agent sit on an object that is both interactable and sittable.

A.3 Prompts for SELU

The specific prompts we use for SELU are:

Actor-Interaction with Env

This is the current observation from a {agent} in a {AI2-THOR/VirtualHome} environment. Now the {agent} needs to finish the task {instruction}, you can only choose the following action to interact with the environment, which are {action_list}. If you choose {PickupObject/GrabObject}, OpenObject, {BreakObject/SitObject}, you should give a specific object name. Now the objects you can interact with are {visible_objs_str}. What's your next action to implement the command to {instruction}? You should output your action and the reasoning. The output format should be:

Action:...
Object:...
Reasoning:...

Critic-Success Detection

The image shows a third-person view from the {agent}'s perspective in a {AI2-THOR/VirtualHome} environment. Please check whether the {instruction.objects} in the image is {instruction.verb → adj.} or not? You should output yes or no, and the reasoning. The output format should be:

Result:...
Reasoning:...

Critic-Self Asking 1

The image shows a third-person view from the {agent}'s perspective in a {AI2-THOR/VirtualHome} environment. Please check the state of the {instruction.objects} in the image. You should output the state and the reasoning. The output format should be:

State:...
Reasoning:...

Critic-Self Asking 2

The image shows a third-person view from the {agent}'s perspective in a {AI2-THOR/VirtualHome} environment. The {instruction.objects} in the observation is in {objects.state} state, please determine whether the {instruction} has been completed or not. You should output yes or no, and the reasoning. The output format should be:

Result:...

Reasoning:...

Critic-hindsight relabeling 1

The image shows a third-person view from the {agent}'s perspective in a {AI2-THOR/VirtualHome} environment. Please see the image carefully. Determine whether there is any object that is {instruction.verb \rightarrow adj.} by the {agent}? You should output the object name and the reasoning. The output format should be:

Object:...

Reasoning:...

Critic-hindsight relabeling 2

The image shows a third-person view from the {agent}'s perspective in a {AI2-THOR/VirtualHome} environment. The {relabeling.object} in the observation is {instruction.verb \rightarrow adj.}, you should give a new instruction based on it. The original instruction is {instruction}, what's the new instruction? The output format should be:

New instruction:...

Reasoning:...

A.4 Visualization of Actor and Critic on LLaVA in AI2-THOR

The embodied actor MLLM is used to interact with the unknown environment, and collect trajectories from the evaluation of the critic MLLM. An example for 'pick up the lettuce' is shown in figure 5.

The embodied critic MLLM is used to perform success detection on each trajectory and use self-asking and hindsight relabeling techniques to build bootstrapped dataset. An example for 'break the mug' is shown in figure 6.

A.5 Multi-Frame Critic Study

To further justify our choice of using the final frame, we conduct the frame number parameter experiments to see its effect. As LLaVA can only input a single image, we use half of the pretraining dataset to fine-tune it first to allow multi-image input. The critic performance is evaluated using the DG baseline setting, with detection accuracy of task completion. The results are shown in Table 11. We conduct the parameter experiments with three configurations: the final frame (DG-1), the last 3 frames (DG-3), and the last 5 frames (DG-5).

Table 11: Multi-frame Evaluation for Critic Module.

	Pick Up	Open	Break	Avg.
DG-1	80.67%	36.50%	50.50%	55.89%
DG-3	76.67%	37.50%	51.00%	55.06%
DG-5	73.67%	37.50%	53.50%	54.89%

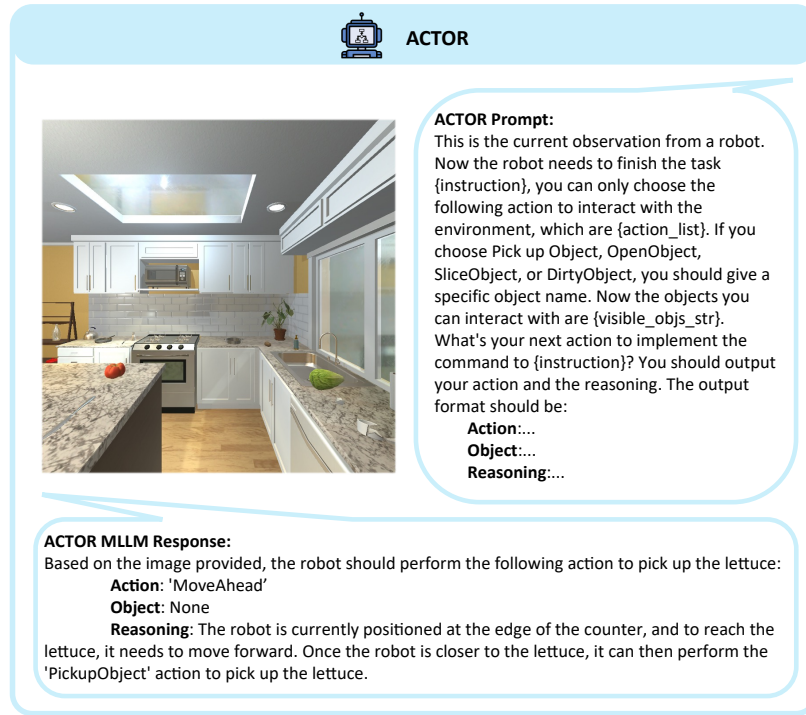


Figure 5: A visualization of the actor MLLM interacting with the AI2-THOR environment. The agent is instructed to pick up the lettuce. As the lettuce is far away, the agent needs to move closer before attempting to pick it up.



Figure 6: A visualization of the critic MLLM conducting success detection on a trajectory. Since the trajectory has completed the 'break the mug' task, it can be added to the fine-tuning dataset.

The evaluation with multiple images does not outperform single-image evaluation. There are two reasons to explain this result. First, most embodied simulators employ flash-like transitions (Li et al., 2025) to represent interaction processes. This shortcoming stems from the fact that different agents often require

distinct low-level control policies. To ensure reliable and generalizable execution, simulators always use flash-like transitions for the interaction process. Consequently, multi-images are not helpful to understand the task completion. Second, current MLLMs still exhibit limitations in spatial reasoning (Chen et al., 2024a). Whether provided with egocentric or third-person views, MLLMs can capture the task-related objects but can not give a clear description of the position relationship between the robot and these objects. Given these constraints, we adopt the final frame as the input for the task critic.

A.6 Computational Resource Costs

We run all experiments in 8 x A100 GPUs with 40GB memory. The details of computational resource costs can be seen in the Table 12. The training time and GPU hours reflect the training cost, while peak VRAM and inference time represent the inference cost. While SELU incurs higher training costs, its inference speed is quicker than traditional baselines as they need more time for retrieval techniques.

Table 12: Computational Resource Costs of LLaVA.

Method	Training Time (h)	GPU Hours (h)	Peak VRAM (GB)	Inference Time (s/sample)
DG	None	None	22.57	1.26
SC	None	None	22.62	4.13
LMSI	5.97	47.76	22.57	1.31
Self-Refine	None	None	22.57	5.22
LLM-Planner	None	None	23.21	2.76
RAP	None	None	23.21	5.71
Self-Ask	None	None	22.57	2.56
SELU-One	6.65	53.20	22.57	4.72 (critic) 1.26 (actor)
SELU	6.72	53.76	22.57	4.72 (critic) 1.26 (actor)

A.7 Name-Removed Prompt Experiments

As introduced in Section 1, an unknown environment refers to one in which the MLLM has neither been exposed to its visual data during pretraining, nor fine-tuned on its task-specific data to achieve grounding. The environment name does not significantly impact the MLLM’s initial perception and decision-making ability. This is demonstrated by the experimental results in Table 13 and Table 14, where we replace “a {AI2-THOR/VirtualHome} environment” in our prompts with “a household environment”. As the MLLM is unknown to the environment, the name does not notably affect its performance.

Table 13: Critic Performance of LLaVA in AI2-THOR Environment with Name-Removed Prompt

Method	Pick Up	Open	Break	Avg.
DG	78.33%	38.50%	51.50%	56.11%
Self-Ask	78.67%	43.00%	56.50%	59.39%
SELU-One	65.67%	33.50%	31.00%	43.39%
SELU	91.67%	67.00%	89.50%	82.72%

A.8 Long-Horizon Task Experiments

In this work, the self-learning MLLM is designed to learn atomic skills to finish short-horizon tasks. As embodied MLLMs have been proved to have the ability to decompose complex and long-horizon tasks into atomic skills. Several benchmarks, such as ALFRED, have been developed to evaluate this planning ability. If we can design a self-learning method to help the MLLM to autonomously acquire atomic skills, it will

Table 14: Actor Performance of LLaVA in AI2-THOR Environment with Name-Removed Prompt

Method	Pick Up	Open	Break	Avg.
DG	66.67%	66.50%	13.50%	48.89%
SC	67.33%	65.00%	14.50%	48.94%
Self-Refine	72.33%	69.50%	15.50%	52.44%
LMSI	77.67%	60.50%	11.50%	49.89%
LLM-Planner	69.67%	53.50%	13.50%	45.56%
RAP	66.67%	67.50%	14.50%	49.56%
SELU-One	89.33%	80.50%	29.50%	66.44%
SELU	93.67%	83.00%	31.00%	69.22%

significantly facilitate the collection of diverse data and improve the MLLM’s capabilities. In ALFRED, long-horizon tasks typically span around 50 timesteps and can be decomposed into 5-6 subtasks by an MLLM planner. Based on this, we set a horizon of 10 timesteps for the MLLM to learn each atomic skill.

To further evaluate the scalability of SELU, we extend the horizon length to 40 timesteps and use the same exploring prompt to initialize the task list. The objects are increased to 8-10 (e.g., 10 for "pick up" and "break" tasks, and 8 for "open" tasks). The corresponding experimental results are shown in Table 15 and Table 16. SELU still performs well in learning long-horizon atomic skills. However, in practice, we often decompose these skills into shorter segments to reduce the complexity of low-level skill execution for embodied robots.

Table 15: Critic Performance of LLaVA in AI2-THOR Environment with 40 Timesteps

Method	Pick Up	Open	Break	Avg.
DG	73.30%	31.13%	49.30%	51.24%
Self-Ask	74.20%	33.63%	51.60%	53.14%
SELU-One	51.70%	26.50%	36.20%	38.13%
SELU	89.50%	69.38%	89.10%	82.66%

Table 16: Actor Performance of LLaVA in AI2-THOR Environment with 40 Timesteps

Method	Pick Up	Open	Break	Avg.
DG	42.80%	51.38%	12.70%	35.63%
SC	45.30%	51.75%	12.10%	36.38%
Self-Refine	47.20%	55.13%	15.30%	39.21%
LMSI	51.60%	59.38%	13.70%	41.56%
LLM-Planner	44.70%	56.50%	16.30%	39.17%
RAP	45.60%	55.25%	17.90%	39.58%
SELU-One	81.30%	86.75%	31.40%	66.48%
SELU	83.50%	84.38%	32.70%	66.86%